SeFlow: A Self-Supervised Scene Flow Method in Autonomous Driving

Qingwen Zhang[1], Yi Yang[1,2], Peizheng Li[3,4], Olov Andersson[1], and Patric Jensfelt[1]

[1] RPL, KTH Royal Institute of Technology, Stockholm, Sweden
[2] Scania CV AB, Södertälje, Sweden
[3] University of Tübingen, Tübingen, Germany
[4] Mercedes-Benz AG, Sindelfingen, Germany
{qingwen,yiya,olovand,patric}@kth.se,
carol-yi.yang@scania.com, peizheng.li@mercedes-benz.com

Abstract. Scene flow estimation predicts the 3D motion at each point in successive LiDAR scans. This detailed, point-level, information can help autonomous vehicles to accurately predict and understand dynamic changes in their surroundings. Current state-of-the-art methods require annotated data to train scene flow networks and the expense of labeling inherently limits their scalability. Self-supervised approaches can overcome the above limitations, yet face two principal challenges that hinder optimal performance: point distribution imbalance and disregard for object-level motion constraints. In this paper, we propose SeFlow, a self-supervised method that integrates efficient dynamic classification into a learning-based scene flow pipeline. We demonstrate that classifying static and dynamic points helps design targeted objective functions for different motion patterns. We also emphasize the importance of internal cluster consistency and correct object point association to refine the scene flow estimation, in particular on object details. Our real-time capable method achieves state-of-the-art performance on the self-supervised scene flow task on Argoverse 2 and Waymo datasets. The code is open-sourced at https://github.com/KTH-RPL/SeFlow.

Keywords: 3D scene flow, self-supervised, autonomous driving, large-scale point cloud

1 Introduction

Scene flow [30] captures the 3D velocity at every point in a point cloud. These detailed 3D flow estimates can enhance downstream tasks in autonomous driving, such as detection, segmentation, tracking, and occupancy flow estimation [20]. A common paradigm for addressing the scene flow problem is supervised learning by utilizing annotated LiDAR data [11, 38]. However, expensive labeling inherently limits the scalability of supervised learning methods.

Given the difficulty and expense of labeling scene flow ground truth, we instead focus on self-supervised scene flow approaches. Existing self-supervised

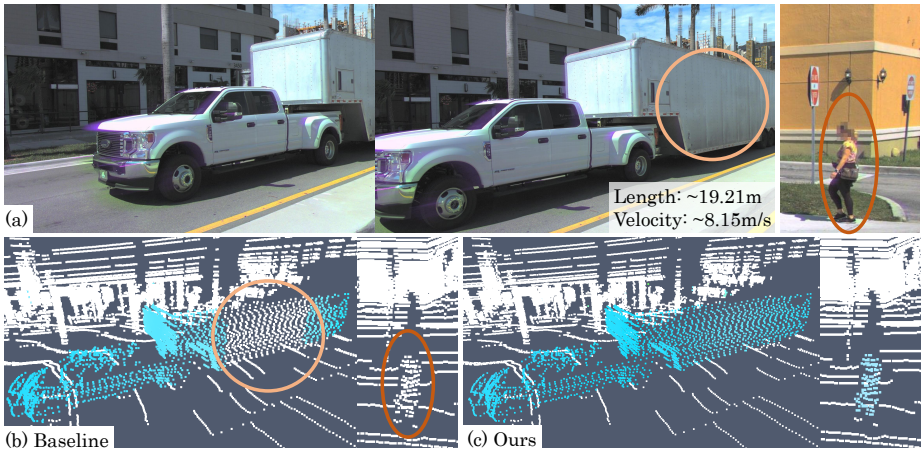


Fig. 1: LiDAR scene flow estimation using our SeFlow method on Argoverse 2. The predicted scene flow for each point is color-coded based on direction. The white indicates static points whose flow is zero. More saturated colors indicate higher velocities. (a) Camera view for visualization purposes only. (b),(c) are zoomed-in views showing the baseline from ZeroFlow [29] as well as SeFlow (ours). When predicting the flow of a large and long vehicle, the baseline predicts a portion of the flow as 0, whereas our estimates are consistent. In addition, the baseline tends to ignore small-scale objects, e.g., pedestrians, while our method can better handle such small and slow-moving objects.

methods can be divided into two categories: knowledge distillation and data exploration. Knowledge distillation methods [2,25,29] typically rely on a 'teacher' method to generate pseudo flow labels. These pseudo labels are then used to supervise student models. Data exploration methods [3, 14, 15, 19, 31], on the other hand, directly utilize the predicted flow to project the input points and find nearest neighbors in the next frame to establish constraints for training.

A key challenge for self-supervised methods is that the vast majority of the points are static (about 86% of points are background [7, 34]). This data imbalance often leads to overly conservative scene flow predictions. One common way to counter this is to use large amounts of training data as in ZeroFlow [29]. ZeroFlow experimentally shows that more data helps to improve the dynamic flow accuracy. In this paper, we investigate ways to improve data efficiency and take inspiration from supervised methods [11, 38]. We address the data imbalance by first classifying points as dynamic or static based on traditional ray casting when integrating frames [9]. This allows us to constrain the corresponding motion patterns by proposing novel loss functions.

Another shortcoming of existing self-supervised scene flow methods is that they disregard object-level motion constraints. The flow should be consistent, i.e., all points in a rigid object should have similar flows. A clear case of inconsistent flow can be seen on the big truck in Fig. 1, where ZeroFlow [29] predicts an absence of flow in certain sections. This inconsistency is caused by incorrect

object point associations which also can be observed in small-scale objects, as shown in the case of the pedestrian. To account for object-level motion constraints, we propose to cluster the dynamic points into object candidates and encourage consistent flow and correct associations for the points in each cluster. This reduces the fragmentation of the flow estimate for points inside the same object.

Overall, our method improves the estimated flow accuracy and addresses issues in previous methods as shown in Fig. 1(c). We propose an efficient and effective self-supervised scene flow method that integrates traditional classification and learning-based strategies. Our approach is open-source at https://github.com/KTH-RPL/SeFlow. Our primary contributions include:

- We propose SeFlow, a novel method that integrates a dynamic classification method in formulating efficient self-supervision objectives.
- We further construct loss functions to learn dynamic flow estimation in imbalanced data and ensure consistent object-level flow, mitigating the effects of correspondence errors.
- We show that SeFlow achieves state-of-the-art results on the self-supervised scene flow task in Argoverse 2 and Waymo datasets and even outperforms all but one supervised method on the leaderboard.

2 Related Work

In this section, we explore existing works in scene flow estimation. We also discuss current traditional frameworks capable of classifying dynamic points, highlighting their relevance and application in the context of scene flow tasks.

2.1 Scene Flow Estimation

Scene flow estimation in autonomous driving is slightly different from flow estimation in object registration. Methods in object registration [8, 13, 16, 24, 32, 33] focus on relatively small-scale point cloud data like synthetic datasets Shapenet [5] and FlyingThing3D [17]. These methods scale poorly with the number of points. When applied to point cloud data for autonomous driving, they require downsampling to 8,192 points or less [11, 18]. In recent datasets like Argoverse 2 [34] and Waymo [11], the number of points in one full frame is around 80k-177k. Methods that can handle such large-scale point cloud data as input usually use a voxel-based pipeline [11, 38]. However, expensive labeling of ground truth flow limits the scalability of these supervised methods, especially in autonomous driving where we have continuously increasing amounts of potential training data.

To train models without labeled data, recent methods propose self-supervised losses. Many commonly used losses, such as Chamfer distance, are based on the nearest neighbor distance between two point cloud inputs [3, 12, 14, 15, 19, 27, 36]. However, a major limitation of nearest neighbor based losses is that only part

of the points on dynamic objects provides good supervision due to incorrect associations. This is especially apparent when big trucks are moving fast in autonomous driving scenarios (see Fig. 1 and Fig. 3). Mittal *et al.* [19] define a self-supervised loss by tracking a patch forward and backward in time to form a cycle while penalizing the errors through cycle consistency and feature similarity. Baur *et al.* [3] propose three loss functions with k-NN loss, rigid cycle consistency inspired by Mittal, and artificial labels based on the distance between points in the original and estimated point clouds. In the following we describe NSFP [14], FastNSF [15], and ZeroFlow [29] in more detail. These are the publically available methods that perform best on the Argoverse 2 self-supervised scene flow challenge and are therefore used as our baselines.

NSFP [14] is an optimization-based method. For each pair of consecutive input point clouds, NSFP iteratively learns new weights for an MLP network to predict the flow using the Chamfer distance loss. However, thousands of iterations are needed and their runtimes extend from 26 to 35 seconds per frame, which fails to meet the real-time requirements of autonomous driving. FastNSF [15] improves the efficiency by voxelizing the point cloud first and then converting it to a distance transform map for faster neighbor calculation. This reduces the runtime to 0.5 seconds, at the expense of increased error.

ZeroFlow [29] adopts a semi-supervised strategy. Pseudo-flow labels are created offline using NSFP [14], and a FastFlow3D [11] model is used as a student for knowledge distillation. This setup allows the student model to perform real-time inference. In this case, the teacher network needs significant resources (around 3.6 GPU months [29]) to label the entire training data. The final accuracy is also influenced by the performance of the teacher.

To increase data efficiency and address the limitations of the above methods, we propose to integrate efficient ray-casting-based dynamic awareness mapping into our pipeline. The core idea is to classify which points move and cluster these points into objects for which we can estimate group-level motion statistics and define better self-supervised loss functions.

2.2 Dynamic Awareness in Mapping

In the field of Simultaneous Localization and Mapping (SLAM), there is a growing interest in dynamic awareness [9, 22, 23, 35, 37]. This interest stems from the fact that points associated with moving objects in a scene can significantly impact localization and planning performance. Existing dynamic awareness methods in mapping [9, 23, 37] often utilize ray casting techniques to binary classify points as either dynamic or static.

The scene flow task, on the other hand, aims to predict the specific 3D flow at each point, a goal that extends beyond the mere categorization of dynamic and static points. In mapping, a point is considered dynamic if it moves once within a scene (even if it becomes static later), whereas, in scene flow tasks, a point is deemed dynamic if it moves faster than a certain velocity threshold in the current frame. Despite these differences, insights from the mapping field are invaluable. By integrating information over time, these frameworks develop a

more comprehensive understanding of the entire scene. Such scene level dynamic awareness can inform and enhance our exploration of point cloud data in scene flow tasks.

3 Problem Statement

This work addresses the problem of real-time scene flow estimation in autonomous driving. Given two consecutive input point clouds, $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$, along with the ego movement $\mathbf{T}_{t,t+1} \in SE(3)$, the goal is to predict the motion vector as flow $\hat{\mathcal{F}}_{t,t+1}(p) = (x, y, z)^T$ for each point $p \in \mathcal{P}_t$.

The objective is to minimize the End Point Error (EPE) which represents the difference between the predicted flow and the ground truth flow, as expressed by the following equation:

$$\min \underbrace{\frac{1}{|\mathcal{P}_t|} \sum_{p \in \mathcal{P}_t} \left\| \hat{\mathcal{F}}(p) - \mathcal{F}_{gt}(p) \right\|_2}_{\text{EPE}}, \tag{1}$$

where $|\mathcal{P}_t|$ denotes the cardinality of (i.e., the number of points in) $\mathcal{P}_t$. For consistency, in the subsequent presentation, capital symbols correspond to sets, while the lowercase symbols represent variables of specific points.

4 Method

4.1 Input and Output

The first step to process the input data, as is commonly done, is to remove ground points from $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$. This is typically done using HD maps [26, 34] or ground segmentation techniques [10].

The estimated flow $\hat{\mathcal{F}}$ from $\mathcal{P}_t$ to $\mathcal{P}_{t+1}$ can be decomposed into two parts as follows:

$$\hat{\mathcal{F}} = \mathcal{F}_{ego} + \Delta\hat{\mathcal{F}}, \tag{2}$$

where $\mathcal{F}_{ego}$ is the flow resulting from the ego vehicle's motion which can be directly obtained from $\mathbf{T}_{t,t+1}$, and $\Delta\hat{\mathcal{F}}$ is our network output.

4.2 Model Backbone

To enable real-time computation and estimation of scene flow across a large point set, voxelization is considered a practical encoding strategy in the model backbone. However, the reduction in resolution often leads to a poorer distinction of point features within the same voxel. With this in mind, we use DeFlow [38] as the architectural basis in this paper. DeFlow integrates GRU [6] with iterative refinement in the decoder design. More specifically, the GRU module maintains voxel features as its hidden state and selectively forgets or updates

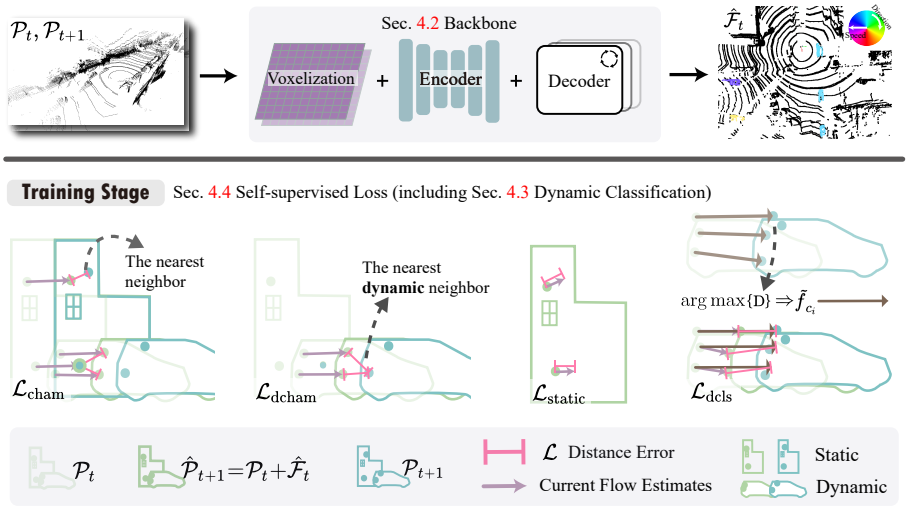


Fig. 2: SeFlow Architecture. Top: With two consecutive point clouds as inputs, our model predicts the estimated flows of all points. Bottom: Conceptual visualization of the Chamfer loss and the three proposed training losses. With the original input $\mathcal{P}_t$ (2 static points for the building, and 2 dynamic points for the car) plus the estimated flow $\hat{\mathcal{F}}_t$, we can calculate the error between estimated $\hat{\mathcal{P}}_{t+1}$ and the next frame point cloud $\mathcal{P}_{t+1}$ ($\mathcal{L}_{\text{cham}}$). The second part is $\mathcal{L}_{\text{dcham}}$ that only calculates the distance error between dynamic points. The third loss says that the estimated flows of static points should be zero. Finally, we assume that the flow at points from the same cluster should be consistent, and mitigate underestimation by using the proposed upper bound on the flow.

the information of the hidden state during each iteration according to the input point features. After multiple iterations, the optimized voxel features are concatenated with the original point features to obtain the final individual point features. Benefiting from this design, we improve the inference efficiency compared to the commonly used backbone FastFlow3D [11] without sacrificing the accuracy of scene flow estimation in coarse resolution settings.

4.3 Dynamic Classification of Points

To facilitate dynamic classification of points during the training stage, we incorporate the DUFOMap framework, a mapping-based dynamic awareness approach [9]. The key insight of this is that points observed inside a region that at one time has been observed as empty must be dynamic. Built on this insight, DUFOMap utilizes ray-casting to classify dynamic points at the sensor rate on the CPU. The result is two disjoint sets, $\mathcal{P}_{t,d}$ and $\mathcal{P}_{t,s}$, where $\mathcal{P}_{t,d}$ is the set of dynamic points that have moved once inside a scene (even if they later become static) and $\mathcal{P}_{t,s}$ is the set of static points that did not move at any time.

Note that the dynamic-static classification is separated from the inference and training, offering our method good flexibility.

4.4 Self-supervised Loss

As discussed, self-supervised learning with only Chamfer distance loss is vulnerable to problems with data imbalance and incorrect associations. We therefore construct three additional loss functions, illustrated in Fig. 2 and described in turn below, to mitigate problems with data imbalance as well as encourage consistent object-level flow.

Chamfer Distance The Chamfer distance, a common self-supervised loss in existing work [3, 14, 15, 19], has the following definition:

$$\mathcal{L}_{\text{cham}} = \frac{1}{|\hat{\mathcal{P}}_{t+1}|} \sum_{p \in \hat{\mathcal{P}}_{t+1}} \mathrm{S}(p, \mathcal{P}_{t+1}) + \frac{1}{|\mathcal{P}_{t+1}|} \sum_{p \in \mathcal{P}_{t+1}} \mathrm{S}(p, \hat{\mathcal{P}}_{t+1}) \tag{3}$$

$$\mathrm{S}(p, \mathcal{P}_o) = \min_{p_i \in \mathcal{P}_o} ||p - p_i||_2^2, \tag{4}$$

where, $\hat{\mathcal{P}}_{t+1} = \mathcal{P}_t + \hat{\mathcal{F}}_t$, $\mathrm{S}(\cdot) = \mathrm{D}(\cdot)^2$, $\mathrm{D}(p, \mathcal{P}_o)$ denotes the distance between point p and its nearest neighbor in $\mathcal{P}_o$.

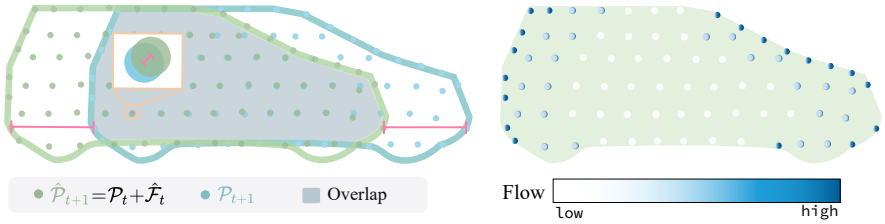
The Chamfer distance is proposed to calculate a similarity between two point clouds. However, when using it directly as a supervised signal for scene flow in autonomous driving, there are two issues we note. First, the number of static points is often much larger than that of dynamic points and averaging $\mathrm{S}(\cdot)$ over all points leads the training to favor static points, i.e., zero flow estimation. Second, due to erroneous correspondence assumptions, only part of the flows within a dynamic object can be estimated correctly as shown in Fig. 3, where the estimated flows of points in the overlap area are zero.

To solve these issues, we contribute the following constraints using the dynamic classification information prior to supervising the network.

Dynamic Chamfer Distance The imbalance in the number of dynamic and static points presents a significant challenge in scene flow estimation, as highlighted in previous works [11, 29, 38]. Supervised methods, provided with ground truth labels for point velocity or classification, can weight their loss functions to account for this imbalance. For self-supervised learning, we introduce a *dynamic* Chamfer distance. In this context, 'dynamic' points are identified based on the output from Sec. 4.3. Specifically, this loss, $\mathcal{L}_{\text{dcham}}$, is only computed on points classified as dynamic in two consecutive point clouds. By exclusively considering dynamic points, this loss captures the nuances of motion in point cloud data. This is defined as:

$$\mathcal{L}_{\text{dcham}} = \frac{1}{|\hat{\mathcal{P}}_{t+1,d}|} \sum_{p \in \hat{\mathcal{P}}_{t+1,d}} \mathrm{S}(p, \mathcal{P}_{t+1,d}) + \frac{1}{|\mathcal{P}_{t+1,d}|} \sum_{p \in \mathcal{P}_{t+1,d}} \mathrm{S}(p, \hat{\mathcal{P}}_{t+1,d}), \tag{5}$$

where $\hat{\mathcal{P}}_{t+1,d} = \mathcal{P}_{t,d} + \hat{\mathcal{F}}_{t,d}$.



(a) Chamfer distance (b) Point flow magnitude

Fig. 3: Simple visualization of the shortcomings of using Chamfer distance as a supervisory signal for flow value estimation. The denser color on points in (b) represents higher flow values and white means the point's flow is zero. (a) illustrates how to calculate loss based on Chamfer distance, and (b) shows that the flow results, based on the nearest neighbor principle, can lead to zero flow estimation for the middle of the object.

Static Flow The standard Chamfer distance rests on a very strong assumption: nearest neighbor matching can find perfect one-to-one correspondences of all points between two frames. However, due to varying observations, the number of points in the two frames differs, leading to inconsistencies and potential mismatches. To deal with the possible mismatches in the static areas, we add a constraint to encourage the model to estimate zero flow for static points. This loss is defined as follows:

$$\mathcal{L}_{\text{static}} = \frac{1}{|\mathcal{P}_{t,s}|} \sum_{p \in \mathcal{P}_{t,s}} ||\Delta\hat{\mathcal{F}}(p)||_2^2, \tag{6}$$

where $\Delta\hat{\mathcal{F}}(p)$ represents the network output flow of point p.

Dynamic Cluster Flow For dynamic fields, the inconsistencies and correspondence errors are even more complicated. For instance, we observe that nearest neighbor matching often leads to erroneous results, e.g., on large objects undergoing translation, which is very common for moving vehicles in urban street environments as shown in Fig. 1. More specifically, Figure 3 illustrates the issue on a simplified car example. In this case, the flow is parallel to the object's surface, which means that nearest neighbor matching does not provide good supervision in the interior of the surface (D $\approx$ 0 for the points in the overlap area). This mismatch results in an incorrect local optimum that remains unresolved during the training process as it minimizes the losses based on Chamfer Distance.

Therefore, we suggest that the motion of all parts within an object (a cluster) should be approximately homogeneous over a short time interval. We use the HDBSCAN [4,21] clustering algorithm to identify different moving objects. This clustering is only applied to dynamic points, thus reducing the computational overhead.

$$\mathcal{C}_{t,d} = \text{HDBSCAN}(\mathcal{P}_{t,d}). \tag{7}$$

Directly constraining the variance of the flow distribution inside a cluster would be a straightforward idea, but does not guarantee the correctness of the obtained mean value after convergence. As we know that nearest-neighbour point correspondence can considerably underestimate flow on geometrically featureless object surfaces, we here instead propose to use an upper bound on the object motion as the supervisory signal. We derive this upper bound, per object cluster, by taking the maximum inter-frame distance of all point correspondences within the cluster. Specifically, we exploit information from the original input point cloud data, i.e., $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$. After the dynamic classification and clustering process, we find the index of the point p_k in cluster $c_i \in \mathcal{C}_{t,d}$ with the largest distance to its nearest neighbor point in $\mathcal{P}_{t+1,d}$, i.e.,

$$\kappa_i = \arg\max_k \{\mathrm{D}(p_k, \mathcal{P}_{t+1,d}) | p_k \in \mathcal{P}_{c_i}\}. \tag{8}$$

We calculate the upper bound, $\tilde{f}_{c_i}$ on the flow for cluster c_i as

$$\tilde{f}_{c_i} = p'_{\kappa_i} - p_{\kappa_i}, \tag{9}$$

where p'_{κ_i} is the nearest neighbor of point p_{κ_i} in $\mathcal{P}_{t+1,d}$. We use this to drive the estimated flows of cluster c_i towards $\tilde{f}_{c_i}$ as follows:

$$\mathcal{L}_{c_i} = \sum_{p_j \in \mathcal{P}_{c_i}} ||\hat{f}_{p_j} - \tilde{f}_{c_i}||_2^2, \tag{10}$$

$$\mathcal{L}_{\mathrm{dcls}} = \frac{1}{|\mathcal{P}_{t,d}|} \sum_{c_i \in \mathcal{C}_{t,d}} \mathcal{L}_{c_i}. \tag{11}$$

In conclusion, the final SeFlow loss incorporates all four losses introduced above,

$$\mathcal{L}_{total} = \mathcal{L}_{\mathrm{cham}} + \mathcal{L}_{\mathrm{static}} + \mathcal{L}_{\mathrm{dcham}} + \mathcal{L}_{\mathrm{dcls}}. \tag{12}$$

The effect of each loss will be evaluated in the ablation study in Sec. 5.3.

5 Experiment

In this section, we first outline evaluation details and then present quantitative comparisons with state-of-the-art methods on two benchmark datasets. A series of ablation studies are presented to better evaluate the individual components of our approach. Finally, we conclude with qualitative results on Argoverse 2 and discuss limitations of the approach.

5.1 Dataset and Metric

We briefly introduce the dataset and metrics for evaluation in the following section. More implementation details, such as hyperparameters for training and dataset descriptions, can be found in the supplementary material.

Table 1: Comparisons on Argoverse 2 test set from the online leaderboard [1]. Upper groups are supervised methods, lower groups are **self-supervised** methods. Our method achieves state-of-art performance in the self-supervised scene flow task. † means methods can run in real-time (10 Hz) onboard.

Method	Run Time per frame [ms]	EPE ↓			
		3-way	FD	FS	BS
FastFlow3D† [11]	34 ± 5	0.0782	0.2072	0.0253	0.0020
DeFlow† [38]	48 ± 4	0.0534	0.1340	0.0232	0.0029
FastNSF [15]	507 ± 312	0.1657	0.3540	0.0406	0.1025
NSFP [14]	32,060 ± 10,112	0.0685	0.1503	0.0302	0.0248
ZeroFlow† [29]	34 ± 5	0.0814	0.2109	0.0254	0.0080
SeFlow (Ours)†	48 ± 4	**0.0628**	0.1525	0.0321	0.0038

Dataset We evaluate our approach on two large-scale autonomous driving datasets: Argoverse 2 [34] and Waymo [26]. Ground removal is performed using HD maps for both datasets as described in [34]. Waymo datasets contain 798 training and 202 validation scenes respectively. We focus our description here on Argoverse 2, which provides official and public scene flow challenges [1, 28], with the *Sensor* and *LiDAR* datasets. The *Sensor* dataset encompasses 700 training and 150 validation scenes. Each scene is approximately 15 seconds long in 10 Hz, with complete annotations for evaluation. The evaluation for another 150 test scenes can be accessed indirectly by submitting a solution to the leaderboard. The *LiDAR* dataset contains 20,000 scenes without any annotation and is only used as extra data in Sec. 5.3.

Metric The benchmark follows existing works [1,7,29,38] and uses the three-way End Point Error. End Point Error (EPE), as defined in Eq. (1), measures the L2 norm of the discrepancy between the predicted and actual flow vectors, expressed in meters. The EPE three-way (3-way) is defined as the unweighted average EPE across three disjoint sets of points: Foreground Dynamic (FD), Foreground Static (FS), and Background Static (BS). The definition of 'dynamic' is as follows: If the flow at a point exceeds the threshold (the public leaderboard setting is 0.05m), the point is defined as dynamic. Given the 10 Hz sensor frequency, this threshold corresponds to a speed of 0.5 m/s. All evaluations are limited to points that are within a 100m × 100m box centered on the ego vehicle.

5.2 Quantitative Results

We evaluate the performance of our method SeFlow and compare it with the currently best-performing methods on the Argoverse 2 test set and Waymo validation set. In Tab. 1 and Tab. 2, the upper group, FastFlow3D and DeFlow, are supervised methods trained with ground truth flow. Compared to the supervised

Table 2: Comparisons on Waymo Open dataset validation set. Upper groups are supervised methods, lower groups are **self-supervised** methods. Our method achieves state-of-art performance in the self-supervised scene flow task. † means methods can run in real-time (10 Hz) onboard.

Method	Run Time per frame [ms]	EPE ↓			
		3-way	FD	FS	BS
FastFlow3D† [11]	27 ± 6	0.0782	0.1954	0.0246	0.0152
DeFlow† [38]	42 ± 4	0.0446	0.0980	0.0259	0.0098
FastNSF [14]	593 ± 308	0.1579	0.3012	0.0146	0.0403
NSFP [15]	76,163 ± 32,256	0.1005	0.1712	0.1081	0.0221
ZeroFlow† [29]	27 ± 6	0.0921	0.2162	0.0153	0.0241
SeFlow (Ours)†	42 ± 4	**0.0598**	0.1506	0.0181	0.0106

methods, we note that SeFlow outperforms FastFlow3D and approaches the level of DeFlow in terms of EPE 3-way. This comparative analysis demonstrates the great potential of self-supervised learning in scene flow estimation and validates the effectiveness of our approach.

In the self-supervised category, our SeFlow method achieves state-of-the-art performance on both datasets. While FastNSF and NSFP are optimization-based methods that do not rely on pre-trained weights for subsequent estimations, their inference times are not conducive to real-time requirements. NSFP, with the second best result in Tab. 1, takes approximately 30 seconds to predict a single frame, which is impractical for real-time autonomous driving applications. FastNSF, on the other hand, enhances efficiency through voxelization for quicker Chamfer distance calculation. Although significantly faster than NSFP, the performance of FastNSFP is the worst among all methods evaluated.

ZeroFlow is capable of estimating scene flow in real-time, with an accuracy similar to NSFP (slightly better on Waymo and slightly worse on Argoverse 2). SeFlow stands out not only for achieving the best result, but also for drastically reducing the run time to 50 milliseconds, which is more than two orders of magnitude faster than NSFP while providing more accurate flow estimates. SeFlow's state-of-the-art performance demonstrated in Tab. 1 and Tab. 2 underscores the effectiveness of our novel self-supervised method in scene flow estimation.

5.3 Ablation study

In this section, we delve into two key aspects of our SeFlow pipeline. Firstly, we examine the impact of different loss terms on the accuracy of our flow prediction results. This analysis aims to demonstrate the necessity and effectiveness of the loss components we propose. Secondly, we explore how the size of the training dataset, especially in the case of limited training data, affects the outcomes of our self-supervised training process.

Table 3: Ablation study of loss terms. Results are evaluated on the Argoverse 2 validation set with 20 training epochs.

Exp. Id	$\mathcal{L}_{\text{cham}}$	$\mathcal{L}_{\text{dcham}}$	$\mathcal{L}_{\text{static}}$	$\mathcal{L}_{\text{dcls}}$	EPE ↓			
					3-way	FD	FS	BS
1	✓				0.0962	0.203	0.052	0.033
2	✓	✓			0.0916	0.181	0.059	0.035
3	✓	✓	✓		0.0779	0.220	**0.012**	**0.002**
4	✓	✓	✓	✓	**0.0643**	**0.160**	0.029	0.004

Loss Terms The advantages of each loss design are evident in Tab. 3 evaluated on the Argoverse 2 validation set with 20 training epochs. Instead of relying solely on the chamfer loss $\mathcal{L}_{\text{cham}}$, we investigate how incorporating the additional three losses ($\mathcal{L}_{\text{static}}$, $\mathcal{L}_{\text{dcham}}$, $\mathcal{L}_{\text{dcls}}$) can boost the performance.

After adding the dynamic Chamfer loss $\mathcal{L}_{\text{dcham}}$, experiment 2 shows a decrease in flow estimation error of about 0.22 (10%) for foreground dynamics (FD), while the static errors (FS, BS) remain essentially the same. Experiment 3 then incorporates $\mathcal{L}_{\text{static}}$ constraint, which significantly reduces the foreground and background static flow estimation errors, 80% for FS and 94% for BS. However, adding $\mathcal{L}_{\text{static}}$ also increases the foreground dynamic error. We attribute this to the difficulty of estimating the flow of moving, geometrically featureless, objects as mentioned previously, which would be reinforced by $\mathcal{L}_{\text{static}}$. Even so, considering the two static components in the EPE 3-way metric, the EPE 3-way would still benefit considerably from this loss (15% error reduction). Finally, in experiment 4, we incorporate the $\mathcal{L}_{\text{dcls}}$. This results in a notable decrease in overall EPE 3-way by 33% compared to solely using the chamfer loss (experiment 1). The above experiments illustrate that our method is not a simple stack of losses, but a complementary holistic design.

Training Dataset Size In the context of robotics and autonomous driving, there are situations where the number of accessible frames is limited. This section evaluates the performance of SeFlow given different amounts of training data.

In this experiment, we denote the entire Argoverse 2 *Sensor* training set as 100%. Extra unlabeled data can be retrieved from the *LiDAR* dataset for self-supervised methods. To assess the impact of dataset size, we conduct the same 50 epochs of training for methods using 10%, 20%, 50%, and 100% of the total data in Tab. 4 and Fig. 4 and evaluate the resulting models.

From Tab. 4, we can observe that with only 20% or 50% training data, our SeFlow can already outperform ZeroFlow which uses 100% or 200% data. Figure 4 illustrates even more intuitively that our SeFlow can easily outperform both existing supervised (FastFlow3D) and unsupervised (ZeroFlow) approaches with the same amount of data. We attribute the demonstrated data efficiency of our method to the well-designed loss functions, which integrate a dynamic

awareness mapping method into our framework and enable better scene-level comprehension.

Dataset	EPE ↓			
	3-way	FD	FS	BS
10%	0.094	0.234	0.040	0.006
20%	0.078	0.197	0.032	0.004
50%	0.066	0.167	0.028	0.004
100%	**0.059**	0.147	0.026	0.004
ZF 100%	0.088	0.231	0.022	0.011
ZF 200%	0.076	0.198	0.018	0.011

Table 4: Ablation study of dataset size compared with Zeroflow (ZF), another self-supervised learning method. Results are evaluated on the Argoverse 2 validation set with 50 training epochs. The total dataset (100%) is 107k frames. 200% data means all of the *Sensor* dataset plus an equal amount of the *LiDAR* dataset (214k frames).

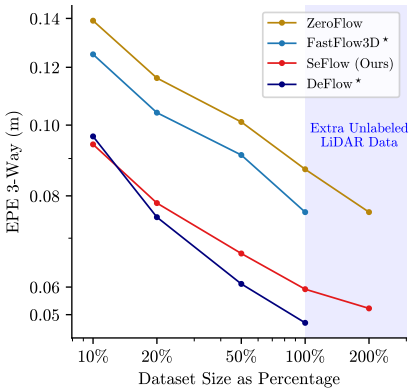


Fig. 4: The relationship between flow estimation error and training dataset size, scaled in $\log_{10}$. Methods with ⋆ are supervised by ground truth labels. SeFlow uses less data but gets comparable results compared to FastFlow3D and ZeroFlow.

5.4 Qualitative Results

Figure 5 presents a qualitative flow estimation result on a sequence of scenes in the Argoverse 2 validation set. The first three columns showcase the same scene at different timestamps, and the last column shows a zoomed-in view of the scene.

In the first column featuring a vehicle partially occluded by a traffic pole, SeFlow demonstrates superior flow estimation capabilities compared to ZeroFlow. Sometimes, SeFlow can even detect flow overlooked by the ground truth annotations. Argoverse 2 derives ground truth flow from manual instance level labels, and as a result, the flow of points outside the bounding box may be ignored. This issue is particularly pronounced in smaller objects, and an example of this can be seen in the fourth column where we zoom in on a case. In this example, a pedestrian is pushing a shopping cart across the road. By comparing the first and the third column, we can observe that both objects are moving. However, the ground truth labels zero flow (white) for the shopping cart. SeFlow, on the other hand, successfully predicts consistent flow even at this small scale. In comparison, the baseline method ZeroFlow suffers at flow prediction of small objects and regards both the pedestrian and the shopping cart as static (white).

Although our method shows superior scene flow estimation compared to other baseline methods, it also has some limitations. Based on the zoom-in view in the

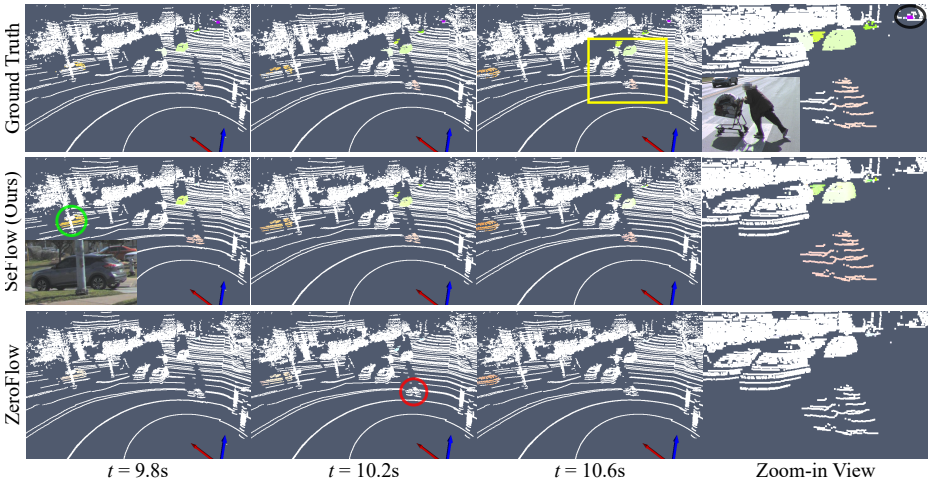


Fig. 5: Qualitative results from Argoverse 2 validation set. The top row displays the ground truth flow, the middle row presents the SeFlow result, and the bottom row showcases another self-supervised method ZeroFlow [29] result. Different color indicates different directions and more saturated color means larger flow estimation. Ego motion is compensated for a clearer view.

fourth column in Fig. 5, we can find some purple flow labels at the top right corner in the first row (Ground Truth). Both SeFlow (Ours) and ZeroFlow fail to predict any flow for these points. One possible reason is that the point cloud of distant objects is sparse and easily ignored by both the voxelization process and clustering algorithms. This makes it difficult to estimate the flow of distant objects using only two consecutive frames. Based on this, multi-modality or time-consistent networks would be a further direction for future research.

6 Conclusions

In this paper, we proposed SeFlow, an efficient and effective self-supervised training method for scene flow estimation in autonomous driving with large-scale point clouds as input. Our primary contributions include a learning method that integrates static and dynamic awareness to construct self-supervision objectives. We further identify problems with the correspondence assumptions of Chamfer-based loss functions commonly used for self-supervised learning, and mitigate these with a constraint based on the upper bound of object motion. Our experimental results underscore the efficacy of our approach.

Future research may concentrate on integrating multi-modality (e.g., cameras and radar) within the SeFlow framework for higher flow estimation accuracy or embedding temporal coherence concepts within our pipeline. Additionally, the optimization of multi-loss learning weights warrants further exploration.

Acknowledgement

Thanks to RPL member: Li Ling helps review this manuscript. Thanks to Kyle Vedder (the author of ZeroFlow), who kindly opened his code including pre-trained weights, and discussed their result with us which helped this work a lot. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and Prosense (2020-02963) funded by Vinnova. The computations were enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg Foundation, Sweden.

References

1. 2, A.: Argoverse 2 scene flow online leaderboard. `https://eval.ai/web/challenges/challenge-page/2010/leaderboard/4759` (2024 Mar 4th)
2. Aleotti, F., Poggi, M., Tosi, F., Mattoccia, S.: Learning end-to-end scene flow by distilling single tasks knowledge. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 10435–10442 (2020)
3. Baur, S.A., Emmerichs, D.J., Moosmann, F., Pinggera, P., Ommer, B., Geiger, A.: Slim: Self-supervised lidar scene flow and motion segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13126–13136 (2021)
4. Campello, R.J., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining. pp. 160–172. Springer (2013)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
6. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. Chodosh, N., Ramanan, D., Lucey, S.: Re-evaluating lidar scene flow for autonomous driving. arXiv preprint arXiv:2304.02150 (2023)
8. Deng, D., Zakhor, A.: Rsf: Optimizing rigid scene flow from 3d point clouds without labels. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1277–1286 (2023)
9. Duberg, D., Zhang, Q., Jia, M., Jensfelt, P.: DUFOMap: Efficient dynamic awareness mapping. IEEE Robotics and Automation Letters **9**(6), 5038–5045 (2024). `https://doi.org/10.1109/LRA.2024.3387658`
10. Himmelsbach, M., Hundelshausen, F.V., Wuensche, H.J.: Fast segmentation of 3d point clouds for ground vehicles. In: Intelligent Vehicles Symposium (IV), 2010 IEEE. pp. 560–565. IEEE (2010)
11. Jund, P., Sweeney, C., Abdo, N., Chen, Z., Shlens, J.: Scalable scene flow from point clouds in the real world. IEEE Robotics and Automation Letters **7**(2), 1589–1596 (2021)
12. Kittenplon, Y., Eldar, Y.C., Raviv, D.: Flowstep3d: Model unrolling for self-supervised scene flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4114–4123 (2021)

13. Lang, I., Aiger, D., Cole, F., Avidan, S., Rubinstein, M.: Scoop: Self-supervised correspondence and optimization-based scene flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5281–5290 (2023)
14. Li, X., Kaesemodel Pontes, J., Lucey, S.: Neural scene flow prior. Advances in Neural Information Processing Systems **34**, 7838–7851 (2021)
15. Li, X., Zheng, J., Ferroni, F., Pontes, J.K., Lucey, S.: Fast neural scene flow. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9878–9890 (2023)
16. Liu, J., Wang, G., Ye, W., Jiang, C., Han, J., Liu, Z., Zhang, G., Du, D., Wang, H.: Difflow3d: Toward robust uncertainty-aware scene flow estimation with diffusion model. arXiv preprint arXiv:2311.17456 (2023)
17. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4040–4048 (2016)
18. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3061–3070 (2015)
19. Mittal, H., Okorn, B., Held, D.: Just go with the flow: Self-supervised scene flow estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11177–11185 (2020)
20. Najibi, M., Ji, J., Zhou, Y., Qi, C.R., Yan, X., Ettinger, S., Anguelov, D.: Motion inspired unsupervised perception and prediction in autonomous driving. In: European Conference on Computer Vision. pp. 424–443. Springer (2022)
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
22. Pfreundschuh, P., Hendrikx, H.F., Reijgwart, V., Dubé, R., Siegwart, R., Cramariuc, A.: Dynamic object aware lidar slam based on automatic generation of training data. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 11641–11647. IEEE (2021)
23. Schmid, L., Andersson, O., Sulser, A., Pfreundschuh, P., Siegwart, R.: Dynablox: Real-time detection of diverse dynamic objects in complex environments. IEEE Robotics and Automation Letters (RA-L) **8**(10), 6259 – 6266 (2023). https://doi.org/10.1109/LRA.2023.3305239
24. Shen, Y., Hui, L., Xie, J., Yang, J.: Self-supervised 3d scene flow estimation guided by superpoints. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5271–5280 (2023)
25. Song, J., Lee, S.J.: Knowledge distillation of multi-scale dense prediction transformer for self-supervised depth estimation. Scientific Reports (18939) (2023)
26. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)
27. Tishchenko, I., Lombardi, S., Oswald, M.R., Pollefeys, M.: Self-supervised learning of non-rigid residual flow and ego-motion. In: 2020 international conference on 3D vision (3DV). pp. 150–159. IEEE (2020)
28. Vedder, K., Khatri, I., Peri, N., Chodosh, N., Liu, Y., Hays, J.: Av2 2024 scene flow challenge announcement. https://www.argoverse.org/sceneflow.html (2024 Feb 25th)

29. Vedder, K., Peri, N., Chodosh, N., Khatri, I., Eaton, E., Jayaraman, D., Ramanan, Y.L.D., Hays, J.: ZeroFlow: Fast Zero Label Scene Flow via Distillation. International Conference on Learning Representations (ICLR) (2024)
30. Vedula, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. IEEE transactions on pattern analysis and machine intelligence **27**(3), 475–480 (2005)
31. Vidanapathirana, K., Chng, S.F., Li, X., Lucey, S.: Multi-body neural scene flow. In: 2024 International Conference on 3D Vision (3DV). pp. 126–136. IEEE (2024)
32. Wang, Z., Wei, Y., Rao, Y., Zhou, J., Lu, J.: 3d point-voxel correlation fields for scene flow estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
33. Wei, Y., Wang, Z., Rao, Y., Lu, J., Zhou, J.: PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds. In: CVPR (2021)
34. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021) (2021)
35. Wu, H., Li, Y., Xu, W., Kong, F., Zhang, F.: Moving event detection from lidar point streams. Nature Communications **15**(1), 345 (2024)
36. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: 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. pp. 88–107. Springer (2020)
37. Zhang, Q., Duberg, D., Geng, R., Jia, M., Wang, L., Jensfelt, P.: A dynamic points removal benchmark in point cloud maps. In: IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). pp. 608–614 (2023)
38. Zhang, Q., Yang, Y., Fang, H., Geng, R., Jensfelt, P.: Deflow: Decoder of scene flow network in autonomous driving. arXiv preprint arXiv:2401.16122 (2024)