2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

DeFlow: Decoder of Scene Flow Network in Autonomous Driving

Qingwen Zhang[1], Yi Yang[1,2], Heng Fang[1], Ruoyu Geng[3], Patric Jensfelt[1]

***Abstract*— Scene flow estimation determines a scene's 3D motion field, by predicting the motion of points in the scene, especially for aiding tasks in autonomous driving. Many networks with large-scale point clouds as input use voxelization to create a pseudo-image for real-time running. However, the voxelization process often results in the loss of point-specific features. This gives rise to a challenge in recovering those features for scene flow tasks. Our paper introduces DeFlow which enables a transition from voxel-based features to point features using Gated Recurrent Unit (GRU) refinement. To further enhance scene flow estimation performance, we formulate a novel loss function that accounts for the data imbalance between static and dynamic points. Evaluations on the Argoverse 2 scene flow task reveal that DeFlow achieves state-of-the-art results on large-scale point cloud data, demonstrating that our network has better performance and efficiency compared to others. The code is available at https://github.com/KTH-RPL/deflow.**

I. Introduction

Scene flow estimation, which determines the 3D motion field of a scene, is essential in the field of autonomous driving. By imitating human behavior when navigating in complex scenes using motion cues, accurate scene flow predictions empower autonomous vehicles (AVs) to interpret and navigate in dynamic environments. Such precise estimations further enhance downstream tasks in AVs, encompassing detection, segmentation, tracking, and occupancy flow.

Recent advancements [1], [2], [3] highlight the value of class-agnostic motion estimations, which are derived directly from point clouds. If satisfactory performance is guaranteed at point-level, the result of scene flow can be effortlessly integrated as a prior for subsequent tasks like prediction and detection [2]. This technique will potentially contribute to elevating the efficiency and adaptability of autonomous driving systems in dynamic scenarios.

Most methods [4], [5], [6] in object registration scene flow focus on relatively small-scale point cloud data like synthetic datasets Shapenet [7] and FlyingThing3D [8]. When they use point cloud data in autonomous driving [9], [10], the points are downsampled to a size of 8,192 points or less. These methods fail, due to memory overflow on modern driving datasets with the full number of points as input. Datasets like Argoverse2 [11] and Waymo [9] are closer to the real autonomous vehicles' sensor setup, where the number of points in one frame is around 80k-177k. Recently, some methods [12], [13] have employed Multi-Layer Perceptrons (MLPs) to optimize proposed self-supervised objective functions that can successfully run on various sizes of datasets. However, their runtimes extend from 26 to 35 seconds per frame [14]. In the field of autonomous driving, real-time capability is important. Consequently, these optimization-based methods fall short of practicality.

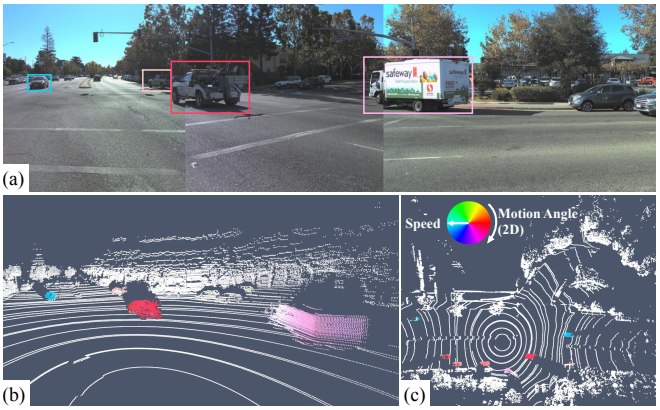

Fig. 1: LiDAR scene flow estimation using our DeFlow method on the Argoverse 2. The predicted scene flow for each point is color-coded based on direction, with the color wheel anchored in the world frame. (a) Camera view for visualization purposes only. (b)(c) Estimated LiDAR point clouds' flow. Varied colors represent different directions, with more saturated colors indicating higher velocities. (b) Front view. (c) Bird's-eye view.

Given the necessity to process and estimate scene flow on full large point cloud datasets in real-time, FastFlow3D [9] emerges as a practical solution. An essential strategy to achieve the real-time requirement is voxelization. It is a popular point cloud processing technique, particularly for detection tasks [15], [16], [17]. However, there is a distinct difference between detection and scene flow tasks: the latter necessitates point-level results. Voxelization-based methods often fail to realize the importance of the decoder design in the scene flow task, resulting in their inability to differentiate features among points contained within the same voxel. This is because all points within a given voxel inherit the same features from the convolutional network.

Addressing these challenges, we present DeFlow which conducts the Gated Recurrent Unit (GRU) refinement module to reconstruct the different features of points inside the same voxel, markedly improving the final results. We evaluated our method using the Argoverse 2 scene flow task, and it achieved state-of-the-art results on the online leaderboard, leveraging a training set of 100k labeled frames. An ex-

[1]Authors are with the Division of Robotics, Perception, and Learning (RPL), KTH Royal Institute of Technology, Stockholm 114 28, Sweden. (email: qingwen@kth.se)
[2]Authors are with Research and Development, Scania CV AB, Södertälje 151 87, Sweden.
[3]Author is with System Hub, Hong Kong University of Science and Technology, Guangzhou, China.

ample is shown in Fig. 1. Our approach is available open-source at https://github.com/KTH-RPL/deflow. In summary, our primary contributions include:

- The introduction of a novel real-time network that integrates GRU with iterative refinement in the decoder design, effectively transitioning from voxel to point features.
- The proposal of a new loss type optimized for imbalanced data distribution on static and dynamic points.
- Achieving state-of-the-art results on the large-scale point cloud dataset Argoverse 2 online leaderboard.

II. Related Work

Introduced by Vedula *et al.* [18], scene flow estimation captures the 3D point motion field of a scene. This concept evolved from the two-dimensional optical flow estimation, a classical topic that predicts apparent motion patterns in 2D images.

Existing methodologies can be broadly categorized into optimization-based and learning-based approaches. A prominent example within the optimization category is NSFP [12]. This method utilizes MLPs to refine the flow, leveraging the chamfer distance as a metric. Such methods are typically not categorized as learning-based approaches since they do not save learned weights to infer on other frames. Instead, they employ MLPs to optimize each frame. The considerable inference time of NSFP makes it impractical for real-time applications.

Recent 3D learning-based approach [5], [4], [19], [20] can be traced back to 2D approaches [21], [22], [23]. RAFT [21], a notable work in optical flow is a good representative. It constructs a multi-scale 4D correlation volume for pixel pairs by iteratively updating the flow field through a recurrent unit. Based on it, PV-RAFT [5], DPV-RAFT [24] adapt their frameworks for 3D point cloud data. However, these methods become computationally challenging for recent large-scale points due to memory overflow. The issue often happened during the construction of distance matrix [5], [24] or correlation matrix [4] that grow exponentially with the number of points.

Fortunately, there are some encoder designs for processing large point clouds in other point cloud related tasks like detection [15], [16], [17], and segmentation [25], [26]. PointPillar proposed by Lang *et al.* [17], stands out as one of the most favored encoders due to its impressive performance combined with high efficiency. The approach, which transforms points to voxels, creating pseudo-images for convolutional networks, is inspired by the 2D method: FlowNet [27], the pioneering CNN for optical flow estimation, employs a U-Net autoencoder architecture. In scene flow task, FastFlow3D [9], a network that can run on large point cloud data (80k - 177k points) in real-time, also uses PointPillar as its encoder. However, the FastFlow3D decoder struggles to differentiate features among points within the same voxel, a limitation attributed to voxelization in PointPillar. Using the PointPillar approach can improve efficiency but might reduce performance if the design of the decoder is not thought out well.

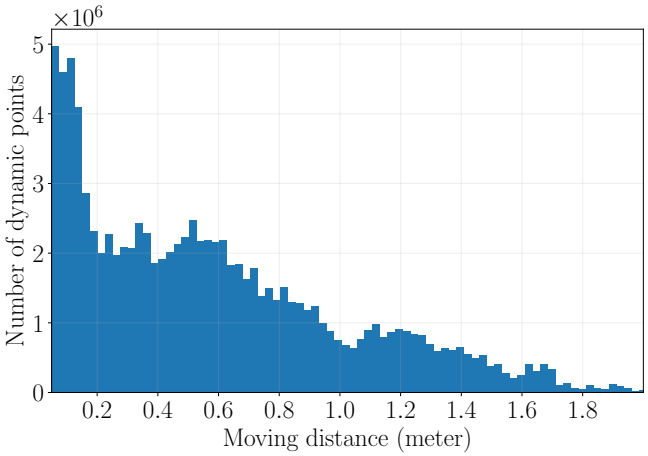

Fig. 2: Histogram of moving distances in 0.1 s for all dynamic points across all scenes in the Argoverse 2 validation dataset (10 Hz). The x-axis represents the distance in meters, ranging from 0.05 to 2.0 meters. The y-axis indicates the number of points for each distance range. The dynamic points are densely distributed within 0.2 meters.

It is applicable to adapt the decoder design developed in optical flow to the ones in the scene flow task. Previous networks used in optical flow like PWC-Net [28] employ a context network to expand the receptive field size of outputs, refining flow through dilated convolutions [29]. Moreover, IRR [30] offers iterative and residual refinement, achieving enhanced accuracy without enlarging the network. Zhang *et al.* [31] utilizes the Gated Recurrent Unit (GRU) for coarse-to-fine improvement and interpolation. Following the line of the works in optical flow, we adopt GRU in our method to emphasize the transition from voxel to point features in the 3D point scene flow task.

III. Problem Statement

Our research tackles the challenge of real-time scene flow estimation in autonomous driving. Given two sequential input point clouds, $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$, captured at times t and $t+1$ respectively, along with the ego movement as the transformation matrix $\mathbf{T}_{t,t+1}$, the objective is to predict the motion vector as flow $\hat{\mathbf{F}}_{t,t+1}(p) = (x, y, z)^T$ for each point $p \in \mathcal{P}_t$.

Knowing the frequency of our sensor data collection (10 Hz), it becomes straightforward to interpret the flow as velocity. The overarching goal is to minimize the End Point Error (EPE) which presents the discrepancy between the predicted flow and the ground truth flow, as expressed by the following equation:

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

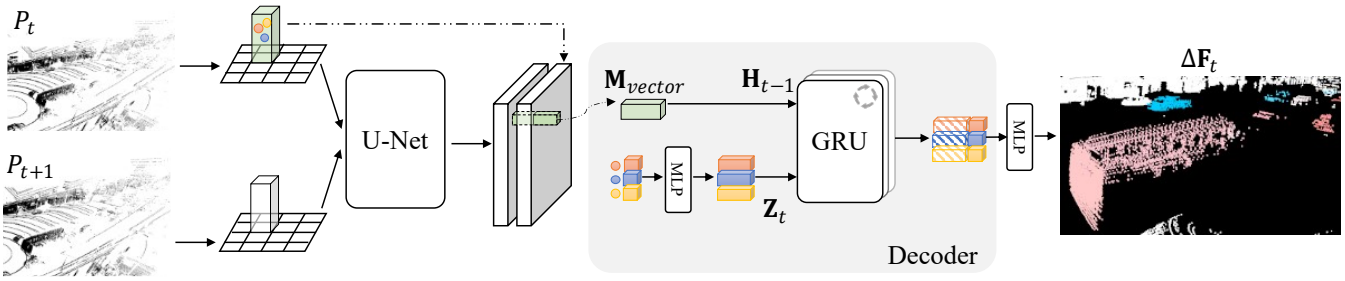

Fig. 3: **DeFlow Architecture**. The feature-extracting step, derived from PointPillars, takes two consecutive point clouds as input and transforms them into voxels. The encoder utilizes a convolutional U-Net backbone. Our novel decoder merges the encoder output with the point offset from PointPillars, employing a GRU for refinement. This process reconstructs the voxel-to-point information, ultimately producing the flow result.

IV. Approach

Our methodology builds upon the overall pipeline of FastFlow3D [9], which was designed specifically for large point cloud data. Briefly, it first voxelizes points and formats them into a bird-eye-view grid feature with a specified resolution. Through convolutional layers, the network efficiently learns features within these voxels.

Based on our analysis of Argoverse 2 (as shown in Fig. 2), we observed that the dynamic points are densely distributed within the range of 0.05 to 0.2m movement. Consequently, the choice of resolution is important, as demonstrated in Section V-B. However, a higher resolution also leads to an increase in computational demands, making it impractical for low-computational devices. If a coarse resolution is chosen, the design of the decoder becomes crucial to differentiate point features within the same voxel to achieve results on par with those of a finer resolution.

In the subsequent sections, we outline our framework as depicted in Fig. 3 and highlight the areas of focus and improvement.

A. Input and Output

Leveraging HD maps provided by most autonomous driving datasets [32], [11], or by deploying ground segmentation techniques like [33], we can easily exclude ground points from both $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$.

The flow $\hat{\mathbf{F}}$ from $\mathcal{P}_t$ to $\mathcal{P}_{t+1}$ is decomposed into two parts as following:

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

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

B. Encoder and Backbone

For point cloud rasterization, we use the dynamic voxelization technique from PointPillars [16], [17] that improves the framework's efficiency. We compute each point's offset from the pillar center and the cluster offset from the point to its cluster coordinates. Post voxelization, a linear transformation aggregates all points within a pillar.

After encoding $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ into grids, we use a 2D convolutional U-Net backbone. Both grids undergo processing through this shared-weight backbone.

C. Decoder

The process of obtaining point-wise flow in FastFlow3D is facilitated by the unpillar operation. This operation, for each point, retrieves the associated flow embedding grid cell, appends the point feature, and employs a multi-layer perceptron to deduce the flow vector. However, as previously highlighted, this approach is not appropriately designed for the reconstruction of voxel-to-point features.

As illustrated in Fig. 3, prior to the decoder's operation, we concatenate the pillar features of $\mathcal{P}_t$ with the output of the U-Net features, resulting in a format of $\mathbf{M}_{vector} \in \mathbb{R}^{N\times C}$. Here, N represents the number of points in $\mathcal{P}_t$, and C denotes the number of feature channels post concatenation. In the decoder, we observed that simply concatenating the C channels common to all points in a voxel with the 3 channels dedicated to point offset led to an imbalance. For the points in the same voxel, the majority of the channels are identical.

An intuitive solution to this imbalance is to expand the point offset features to match the dimensionality of C. However, this modification worsens the original performance, proving that a dedicated network design is necessary.

Drawing inspiration from the 2D optical flow techniques that utilize GRU [31], we propose an alternative method depicted in Fig. 3 decoder to extract different features for points inside the same voxel with multiple iterations to refine. In this approach, we designate $\mathbf{M}_{vector}$ as the first hidden states, denoted as $\mathbf{H}_0$. By employing a linear layer, we expand the point offsets and set them as input, represented as $\mathbf{x}$. The relationship between these components is captured by:

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t, \tag{3}$$

where $\mathbf{Z}_t$ serves as the update gate. It takes $\mathbf{x}$ as its input, which subsequently undergoes processing via 1D convolution layers, utilizing a Sigmoid activation function. The term $\mathbf{H}_{t-1}$ represents the previous hidden state. For the initial instance, $\mathbf{H}_0$ is set to $\mathbf{M}_{vector}$. $\tilde{\mathbf{H}}_t$ stands for the candidate hidden state, which is determined by the reset gate and the model weights.

Method		EPE 3-Way ↓	EPE FD ↓	EPE BS ↓	EPE FS ↓	Dynamic IoU ↑	Dynamic AccRelax ↑	Dynamic AccStrict ↑
FastNSF [34]		0.1657	0.3540	0.1025	0.0406	0.0924	0.3729	0.1958
NSFP w Motion Comp [12]		0.0685	0.1503	0.0248	0.0302	0.3199	<u>0.6956</u>	**0.4537**
Zeroflow [14]	Standard	0.0814	0.2109	0.0080	0.0254	0.4791	0.4363	0.1873
	XL	<u>0.0569</u>	<u>0.1440</u>	0.0089	**0.0178**	0.5224	0.6106	0.3219
FastFlow3D [9]		0.0782	0.2073	**0.0020**	0.0253	<u>0.5760</u>	0.4407	0.1965
DeFlow (Ours)		**0.0534**	**0.1340**	<u>0.0029</u>	<u>0.0232</u>	**0.6289**	**0.7213**	<u>0.4483</u>

TABLE I: Comparisons on Argoverse 2 sensor test set in the online leaderboard [35]. Our methods achieve state-of-art performance in the scene flow task. The main improvement happens in the accuracy of flow estimation on dynamic points, with larger improvements in the Dynamic IoU and Dynamic Accuracy Relaxed. We **bold** the best results and <u>underline</u> the second best results.

To maintain a small model without inflating the number of parameters, we employ multiple iterations within GRU layers. After completing these iterations, the most recent hidden state $\mathbf{H}_t$ is concatenated with the point offset features. This combined entity then proceeds through MLPs, resulting in the generation of the final delta flow, denoted as $\Delta\hat{\mathbf{F}}$ as shown in Fig. 3. A comprehensive ablation study detailing the intricacies and performance of our decoder design is presented in Section V-B.

D. Loss Function

The task of scene flow estimation in autonomous driving scenarios is inherently challenging due to the dynamic nature of the environment. A significant portion of the LiDAR points, which reflect static structures such as buildings or roads, remain stationary. This leads to a label imbalance in the dataset, with more background static points than others. To address this, the loss function incorporates a scaling function, denoted as $\sigma(p)$, to balance the contribution of each point based on its motion characteristics:

$$\mathcal{L} = \frac{1}{|\mathcal{P}_t|} \sum_{p \in \mathcal{P}_t} \sigma(p) \left\| \Delta\hat{\mathbf{F}}(p) - \Delta\mathbf{F}_{gt}(p) \right\|_2, \tag{4}$$

where $|\mathcal{P}_t|$ is the number of points in $\mathcal{P}_t$.

FastFlow3D [9], in their experiments, introduced a scaling approach based on the distinction between foreground and background points. The difference between the two is determined by whether a point is contained within the bounding box of any tracked object.

$$\sigma(p) = \begin{cases} 1 & \text{if } p \in \text{ Foreground} \\ 0.1 & \text{if } p \in \text{ Background} \end{cases} \tag{5}$$

With the advent of self-supervised learning, where labels distinguishing foreground and background are absent, ZeroFlow [14] proposed an alternative scaling function. This function scales based on the speed (flow) of the point's motion:

$$\sigma(p) = \begin{cases} 0.1 & \text{if } s(p) < 0.4 \text{ m/s} \\ 1.0 & \text{if } s(p) > 1.0 \text{ m/s} \\ 1.8s - 0.8 & \text{o.w.} \end{cases} \tag{6}$$

Building upon the insights from ZeroFlow, we propose an unweighted average loss that takes into account the distribution of dynamic and static point numbers. This approach divides $\mathcal{P}_t$ into three categories based on their motion speed $\{\mathcal{P}_{t/1}, \mathcal{P}_{t/2}, \mathcal{P}_{t/3}\}$, as defined in Eq. (6). The total loss is then the sum of the losses from these three categories:

$$\mathcal{L}_{total} = \sum_{i=1}^{3} \frac{1}{|\mathcal{P}_{t/i}|} \sum_{p \in \mathcal{P}_{t/i}} \left\| \Delta\hat{\mathbf{F}}(p) - \Delta\mathbf{F}_{gt}(p) \right\|_2. \tag{7}$$

This comprehensive loss function ensures that the model balances different types of point motion, providing a robust estimation of scene flow in diverse scenarios.

# Itr	EPE 3-Way ↓	EPE FD ↓	EPE BS ↓	EPE FS ↓
2	0.0528	0.1222	0.0055	0.0308
4	**0.0516**	0.1212	0.0047	0.0289
8	0.0517	0.1214	0.0042	0.0295
16	0.0532	0.1262	0.0038	0.0297

TABLE II: Ablation study on GRU iteration count. Analyzing the performance impact across varying GRU iteration numbers, with 4 iterations emerging as the optimal configuration for DeFlow.

V. Experiment

In this section, we outline our experimental setup, followed by a series of ablation studies to understand the contributions of our approach. We then present quantitative comparisons with state-of-the-art methods on a benchmark dataset and conclude with qualitative results.

A. Experiment Setup

Dataset: Our approach is evaluated on the large-scale autonomous driving data, Argoverse 2 [11] which encompasses a variety of sets, including *Sensor* and *LiDAR*. Given that the *LiDAR* dataset lacks imagery and any other annotations, our primary focus is on the *Sensor* dataset. The *Sensor* dataset encompasses 700 training and 150 validation scenes. Each scene is approximately 15 seconds long in 10 Hz, complete with annotations. Argoverse 2 provides an online benchmark with 150 testing scenes.

Methods: Our main comparison is with FastFlow3D [9] on the validation dataset. We reproduce this method based on ZeroFlow [14], given that FastFlow3D serves as its student network and its code is publicly available. To showcase our method's superiority, we display results of various baseline methods from the online leaderboard, including NSFP [12],

Decoder Design	Res (m)	EPE 3-Way ↓	EPE FD ↓	EPE BS ↓	EPE FS ↓	Dynamic IoU ↑	GM (MiB) ↓	FPS ↑
FastFlow3D [9]	0.4	0.1116	0.3055	0.0037	0.0254	0.4701	2114	49.71
	0.2	0.0852	0.2326	0.0025	0.0206	0.5257	2874	29.17
	0.1	0.0586	0.1463	0.0086	0.0208	0.5238	6634	11.31
Our w/o GRU	0.2	0.0916	0.2499	0.0034	0.0216	0.5224	2876	28.99
Ours	0.2	**0.0564**	0.1309	0.0045	0.0337	0.4896	2878	20.49

TABLE III: Decoder design and resolution impact on Argoverse 2 sensor validation set. The table contrasts results based on different decoder designs and voxel resolutions. 'Res' indicates voxel resolution, 'GM' denotes GPU memory consumption during model execution, and 'FPS' (frame per second) signifies the model's running speed. The findings reveal that the GRU at a 0.2 m resolution overperformance the original 0.1 m resolution decoder with reduced GPU memory requirements and faster running speed.

FastNSF [34], and ZeroFlow [14]. Specifically, NSFP and FastNSF operate on a self-supervised paradigm, and both FastFlow3D and our method employ supervised learning techniques by leveraging the *Sensor* dataset for training. Moreover, ZeroFlow adopts a semi-supervised strategy by utilizing NSFP for dataset labeling. While the standard version relies solely on the *Sensor* dataset, the XL version incorporates additional data from the *LiDAR* dataset, amounting to twice the original size. Furthermore, the XL version sets the resolution to 0.1 m and expands the model size to ten times the standard version.

Implementation Details: In Table I for the Argoverse 2 test dataset, the implementation of DeFlow is as follows: We use four GRU iterations (as shown Table II), and the model trains for a total of 50 epochs with a batch size of 80. The network optimization leverages the Adam optimizer with a learning rate set at 2×10^{-6}. The chosen loss function is detailed in Eq. (7). The resolution is set as 0.2 m, and the $[512, 512]$ grid is transformed from a 102.4 m $\times$ 102.4 m map. Other benchmarking methods in detail: FastNSF uses their official public code with config and FastFlow3D trains in 50 epochs to converge and 64 batch size [9] as same as their paper's setup. Others (ZeroFlow and NSFP) are public on the online leaderboard. For local experiments, aimed for a fair ablation study, all models are trained with a learning rate of 2×10^{-6} and the same batch size of 80. We maintain strict control over the resolution variable to see the influence of resolution, loss function, and network design on performance. All local experiments were executed on a desktop powered by an Intel i9-12900KF CPU and equipped with a GeForce RTX 3090 GPU.

Metric: The Argoverse 2 benchmark adopts a unified metric proposed by [13], which introduces the 3-way Endpoint Error (EPE). EPE, as defined in Eq. (1), measures the L2 norm of the discrepancy between the predicted and actual flow vectors, expressed in meters. The 3-way calculates the unweighted average EPE across three classifications: Foreground Dynamic (FD), Foreground Static (FS), and Background Static (BS). If the flow of a point $\hat{\mathbf{F}}(p)$ exceeds 0.05 m, the point is defined as dynamic. Given the dataset's 10 Hz collection frequency, this threshold corresponds to a speed of 0.5 m/s. The Dynamic Accuracy Relaxed (Dynamic AccRelax) metric captures the proportion of dynamic points with an EPE under 0.1 m or a relative error below 10%. In contrast, the Strict (Dynamic AccStrict) metric requires an EPE under 0.05 m or a relative error of less than 5%.

Loss Type	EPE 3-Way	EPE FD	EPE BS	EPE FS
FastFlow3D Eq. (5)	0.0852	0.2326	0.0025	0.0206
ZeroFlow Eq. (6)	0.0843	0.2112	0.0201	0.0202
Ours Eq. (7)	**0.0787**	0.2045	0.0041	0.0277

TABLE IV: Ablation study on loss types. All are trained in the same network (FastFlow3D). Our proposed loss function outperforms others, achieving the best EPE 3-Way score by addressing the imbalanced data distribution between static and dynamic points.

B. Quantitative Results

In this section, we evaluate the efficacy of our proposed DeFlow approach and compare it with alternative methods, highlighting the effects of different design choices on performance.

As referenced in Table I and briefly discussed in Section V-A, it's evident that our method achieves the state-of-art performance in the Argoverse 2 scene flow task. The most significant enhancement is observed in the boosted accuracy of dynamic point flow estimation, with only minimal errors occurring in static points.

To illustrate the efficacy of our network design and the impact of different resolution settings, we evaluated a model without the GRU iteration module, as depicted in Fig. 3. Additionally, we tested both smaller and larger resolutions in FastFlow3D. As shown in Table III, where all models use the origin loss function Eq. (5), the results without the GRU module in the network indicate that merely extending the point offset feature and grid feature channels using MLPs can adversely affect the accuracy of dynamic point flow. Conversely, our DeFlow GRU framework significantly reduces the EPE 3-Way and enhances accuracy in both relaxed and strict dynamic metrics. When comparing with FastFlow3D in 0.1 m resolution setting, with larger GPU memory consumption and slower FPS, our DeFlow GRU at 0.2 m achieves better performance with efficiency on both GPU memory and computation speed. Considering both performance and computational resources, our method is

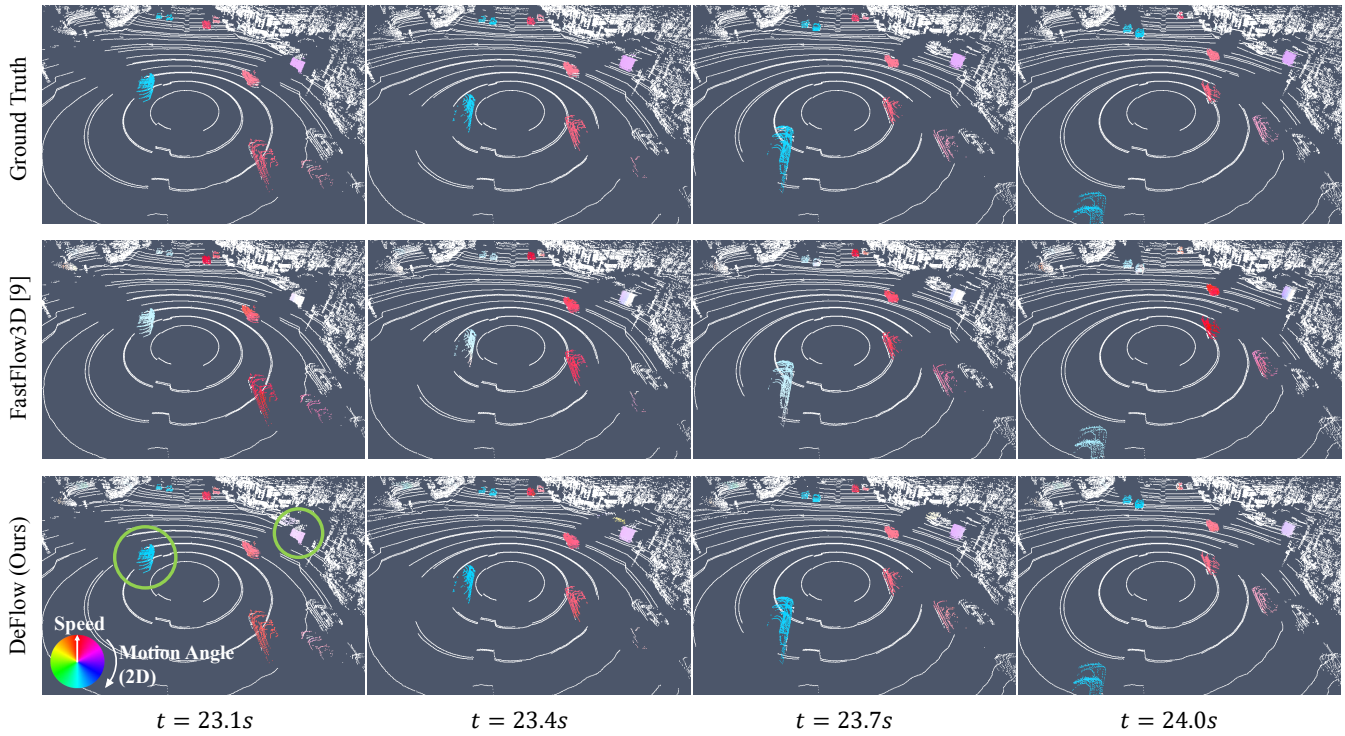

Fig. 4: Qualitative results from the validation set. The top row displays the ground truth flow, the middle row presents the FastFlow3D result, and the bottom row showcases the DeFlow outcomes. DeFlow estimates closely match the ground truth flow in both speed and angle. As highlighted in the two green circles, our DeFlow method demonstrates better performance in predicting motion angle (indicated by color variations) and speed (represented by color intensity) compared to FastFlow3D.

more suitable for lightweight on-board computation in real-time. This outcome further highlights the effectiveness of our network design.

The design of the loss function plays an important role in training the network, as discussed in Section IV-D. We conducted an ablation study using the original FastFlow3D to assess various loss types. Apart from the loss functions, all other parameters were kept consistent: a training epoch of 50, a learning rate of 2×10^{-6} using the Adam optimizer, a resolution of 0.2 m and a batch size 80. The loss proposed by us, as presented in Table IV, emerged as the superior choice, reducing the EPE FD error by 12.1% compared to FastFlow3D.

C. Qualitative Results

We showcase the qualitative outcomes of our DeFlow on the Argoverse 2 validation set with Fig. 4 serving as an example. From our observation, DeFlow demonstrates the ability to accurately capture the motion flow in most cases. When compared to FastFlow3D, it performs better in the prediction of both speed and motion angle. In certain regions, particularly those that are blocked or partially occluded, DeFlow occasionally exhibits inaccuracies, indicating potential areas for further improvement.

VI. Conclusion

In this paper, we introduce DeFlow, an efficient and high-performance method for autonomous driving in large-scale point clouds. Our primary contributions include the introduction of the DeFlow network, which enhances the extraction and reconstruction of point-voxel-point network features at the point level. Additionally, we propose a novel loss function to address the challenges of imbalanced data distribution among points. Our experimental results underscore the efficacy of our approach.

Future work could be on self-supervised exploration of DeFlow and the fusion with multi-modality sensors, like cameras and radar. The flow of dynamic objects is one that we are mainly focused on in scene flow estimation, so there is a possible solution if we can segment static and dynamic [36], [37] first, then it can hugely decrease the computation burden for neural optimization-based approaches.

Acknowledgement

Thanks to Kyle Vedder (ZeroFlow) who kindly discussed their results with us and HKUST Ramlab's member: Jin Wu who gave constructive comments on this work. We also thank the anonymous reviewers for their useful comments. 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] C. Luo, X. Yang, and A. Yuille, "Self-supervised pillar motion learning for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3183–3192.

[2] M. Najibi, J. Ji, Y. Zhou, C. R. Qi, X. Yan *et al.*, "Motion inspired unsupervised perception and prediction in autonomous driving," in *European Conference on Computer Vision*. Springer, 2022, pp. 424–443.

[3] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *arXiv preprint arXiv:2304.10049*, 2023.

[4] I. Lang, D. Aiger, F. Cole, S. Avidan, and M. Rubinstein, "Scoop: Self-supervised correspondence and optimization-based scene flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5281–5290.

[5] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds," in *CVPR*, 2021.

[6] Z. Wang, Y. Wei, Y. Rao, J. Zhou, and J. Lu, "3d point-voxel correlation fields for scene flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[8] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers *et al.*, "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*, 2016, pp. 4040–4048.

[9] P. Jund, C. Sweeney, N. Abdo, Z. Chen, and J. Shlens, "Scalable scene flow from point clouds in the real world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1589–1596, 2021.

[10] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3061–3070.

[11] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh *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.

[12] X. Li, J. Kaesemodel Pontes, and S. Lucey, "Neural scene flow prior," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7838–7851, 2021.

[13] N. Chodosh, D. Ramanan, and S. Lucey, "Re-evaluating lidar scene flow for autonomous driving," *arXiv preprint arXiv:2304.02150*, 2023.

[14] K. Vedder, N. Peri, N. Chodosh, I. Khatri, E. Eaton *et al.*, "Zeroflow: Fast zero label scene flow via distillation," *arXiv preprint arXiv:2305.10424*, 2023.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[16] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao *et al.*, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Conference on Robot Learning*. PMLR, 2020, pp. 923–932.

[17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang *et al.*, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[18] S. Vedula, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 475–480, 2005.

[19] Z. Lu and M. Cheng, "Gma3d: Local-global attention learning to estimate occluded motions of scene flow," *arXiv preprint arXiv:2210.03296*, 2022.

[20] W. Cheng and J. H. Ko, "Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation," in *European Conference on Computer Vision*. Springer, 2022, pp. 108–124.

[21] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419.

[22] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8121–8130.

[23] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu *et al.*, "Craft: Cross-attentional flow transformer for robust optical flow," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 602–17 611.

[24] Z. Wang, Y. Wei, Y. Rao, J. Zhou, and J. Lu, "3d point-voxel correlation fields for scene flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[25] Y. Hou, X. Zhu, Y. Ma, C. C. Loy, and Y. Li, "Point-to-voxel knowledge distillation for lidar semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8479–8488.

[26] J. Cen, P. Yun, S. Zhang, J. Cai, D. Luan *et al.*, "Open-world semantic segmentation for lidar point clouds," in *Computer Vision – ECCV 2022*. Cham: Springer Nature Switzerland, 2022, pp. 318–334.

[27] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[28] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.

[29] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[30] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5754–5763.

[31] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr, "Separable flow: Learning motion cost volumes for optical flow estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 807–10 817.

[32] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[33] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 560–565.

[34] X. Li, J. Zheng, F. Ferroni, J. K. Pontes, and S. Lucey, "Fast neural scene flow," *arXiv preprint arXiv:2304.09121*, 2023.

[35] A. 2, "Argoverse 2 scene flow online leaderboard," https://eval.ai/web/challenges/challenge-page/2010/leaderboard/4759, 2023 Sep 14th.

[36] Q. Zhang, D. Duberg, R. Geng, M. Jia, L. Wang *et al.*, "A dynamic points removal benchmark in point cloud maps," in *IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 608–614.

[37] D. Daniel, Q. Zhang, M. Jia, and P. Jensfelt, "Dufomap: Efficient dynamic awareness mapping," *arXiv preprint arXiv:2403.01449*, 2024.