Article | Received 6 September 2024; Accepted 19 November 2024; Published 28 November 2024
https://doi.org/10.55092/rl20240004

Optimizing scene flow with neural rigidity prior

Zhiheng Feng, Jiuming Liu and Hesheng Wang*

Department of Automation, Shanghai Jiao Tong University, Shanghai, China

* Correspondence author; E-mail: wanghesheng@sjtu.edu.cn.

Abstract: Scene flow estimation provides the 3D low-level motion understanding in dynamic scenes. In this paper, we propose an optimization-based scene flow estimation method with neural rigidity prior for the autonomous driving environment. Specifically, we utilize the rigidity prior of dynamic scenes to partition the point clouds into pillars of different resolutions. Then, the flow vector of a point is represented as the average of local rigid transformations associated with the different pillars to which it belongs. To model local rigidity, we employ the neural implicit representation for encoding the intrinsic constraints of pillars. Our method achieves state-of-the-art accuracy on three commonly-used autonomous driving datasets: Argoverse, Waymo, and nuScenes, and even surpasses previous supervised learning-based methods. Experiment results demonstrate the effectiveness of our method, particularly on sparse points in the autonomous driving scene.

Keywords: scene flow; optimization; neural rigidity prior

1. Introduction

3D scene flow represents the 3D motion of each point in a dynamic environment. It has been widely applied in scenes like autonomous driving [1], object tracking [2], *etc*. Basically, previous scene flow estimation works can be mainly categorized into two types: learning-based methods and optimization-based methods. Learning-based methods [3–7] heavily rely on large amounts of training data. Typically, they are trained on synthetic datasets and fine-tuned on real-scene datasets. However, these data-dependent approaches often suffer from poor generalization performance to the real-world data. In contrast, optimization-based methods have better generalization abilities. However, methods [8,9] that optimize point-to-point scene flows do not take full advantage of the rigidity prior in the scene, and the optimization tends to be worse when the number of points is small due to the lack of sufficient constraints. Some methods [10,11] introduce scene rigidity prior, but need to rely on strong regularization to construct constraints, which introduces some hyperparameters and does not generalize well across different scenes.

To address this issue, we propose an method that leverages the rigid priors of dynamic scenes to partition point clouds into equal-size pillars. The scene flow of each point is then represented by the average of local rigid transformations associated with the multi-resolution pillars to which it belongs. To avoid introducing strong regularization, we employ neural implicit representation to model the local rigidity prior, which can encode the intrinsic constraints of the pillars. Our method achieves the state-of-the-art accuracy on three commonly used autonomous driving datasets: Argoverse, Waymo, and nuScenes. The results of the experiment demonstrate the effectiveness of our method, particularly in the sparse point clouds in the autonomous driving scene. Specifically, our method has shown significant improvement compared to F-NSF [9] in the results of three datasets with 2048 and 8192 points.

In summary, our contributions are as follows:

- We propose a novel scene flow optimization method with neural rigidity prior for the autonomous driving environment.
- We develop a representation method for scene flow using a combination of rigid transformations of multi-resolution pillars. Local rigidity is modeled by neural implicit representation, thereby avoiding the requirement for strong regularization in our model.
- Our method achieves the state-of-the-art accuracy on three commonly used autonomous driving datasets. Particularly for sparse points, our method shows a significant improvement in accuracy.

2. Related works

2.1. Learning-based scene flow

According to whether scene flow ground truth is used during training, learning-based scene flow estimation methods can be divided into supervised scene flow and unsupervised scene flow.

2.1.1. Supervised scene flow

FlowNet3D [3] is the first precursor to learn 3D scene flow from point cloud pairs in an end-to-end manner. They construct point cloud feature pyramids and introduce a flow embedding layer. FLOT [4] incorporates optimal transport to constrain point matching based on graph matching. PointPWC-Net [12] introduces point cloud cost volume, up-sampling, and warping layers to perceive motion between consecutive frames. Wang *et al.* [5] develop an attentive cost volume considering feature space and propose a hierarchical attention learning network for scene flow estimation. PV-RAFT [13] proposes a hybrid point-voxel correlation field to capture both local and global dependencies. Bi-PointFlowNet [14] proposes a bi-directional flow embedding layer with forward and backward flow learning directions. DifFlow3D [15] designs a diffusion-based scene flow residual generation method, which refines the fine-grained flow residual from coarse to fine in a generative manner. However, due to the difficulty in obtaining ground truth for scene flow, supervised scene flow estimation models are often trained on synthetic datasets such as FlyingThings3D [16] and then fine-tune on real-scene data. Although synthetic datasets provide readily available ground truth scene flow, there is a significant domain gap between these datasets and real-scene LiDAR point cloud data. Factors such as noise and motion distortion commonly exist in real-world scenarios but are not adequately captured in the synthetic datasets. As a result, supervised methods trained on synthetic data tend to exhibit poor generalization ability in real-world scenes.

2.1.2. Unsupervised scene flow

To address the challenge of acquiring ground truth for scene flow, many works focus on unsupervised learning methods for scene flow estimation. PointPWC-Net [12] introduces three loss functions: Chamfer loss, smoothness constraint, and Laplacian regularization. These loss functions enable the model to learn scene flow without relying on ground truth flow. Inspired by PointPWC-Net [12], JGF [17] apply the concept of cycle consistency loss to enforce time consistency in the predicted scene flow. Based on the iterative closest point (ICP) algorithm, FlowStep3D [18] proposes a global correlation unit and a recurrent local update unit to iteratively refine the flow residuals. SLIM [19] and RigidFlow [20] take into account global ego-motion when predicting scene flow. SPFlow-Net [21] designs an iterative end-to-end superpoint-based flow estimation method, where superpoint can be dynamically updated by the point-to-superpoint matching construction. 3DSFLabeling [22] designs an auto-labeling method, which can generate a large number of 3D scene flow pseudo labels. SeFlow [23] integrates a dynamic classification method in formulating efficient self-supervision objectives. Self-supervised learning methods possess the advantage of adaptability to different datasets, maintaining a certain level of generalization ability. However, to achieve satisfactory estimation outcomes, these methods still require a significant amount of training data, leading to high training costs.

2.2. Optimization-based scene flow

Optimization-based methods optimize scene flow at runtime and do not rely on model training. Pontes *et al.* [24] utilizes the construction of a point cloud graph to constrain non-rigid scene flow to rigid scene flow within a specific range for optimization. However, this method is highly sensitive to the hyperparameters of the graph. A best-known work is NSFP [8], which utilizes a Multi Layer Perceptron (MLP) as an implicit regularization technique to optimize scene flow. Chodosh *et al.* [25] use NSFP [8] to re-evaluate scene flow. Building upon NSFP, F-NSF [9] further improves the optimization speed by replacing the loss function with a distance transform (DT) loss. This enhancement greatly accelerates the optimization process. RSF [10] uses a combination of global self-motion and bounding box rigid motion to represent scene flow. They introduce a learnable bounding box formulation to optimize the scene flow. Optflow [11] achieves fast convergence and improves point correspondence by using local correlation weight matrices for correspondence matches, adaptive correspondence threshold constraints for nearest-neighbour searches, and graph prior rigidity constraints. ICPFlow [26] uses ICP to estimate the scene flow utilising a rigid transformation of objects in the scene. Khatri *et al.* [27] combines a 3D object detector and a simple Kalman filter-based tracker to estimate the scene flow.

3. Methods

Let $P_{t-1} \in \mathbb{R}^{n_1 \times 3}$ and $P_t \in \mathbb{R}^{n_2 \times 3}$ represent two point clouds from consecutive frames at time t-1 and t, respectively. The number of points in these two point clouds may be different and there may not be point-to-point correspondences between two frames. The motion of a point p_i in P_{t-1} can be represented by a 3D vector f_i. The motion of all points in P_{t-1} can be represented as $F \in \mathbb{R}^{n_1 \times 3}$. The objective function of scene flow optimization is:

$$F^* = \arg\min_F \sum_{p_i \in P_{t-1}} D(p_i + f_i, P_t), \tag{1}$$

where D calculates the distance between the point $p_i + f_i$ and its closest neighbor in P_t.

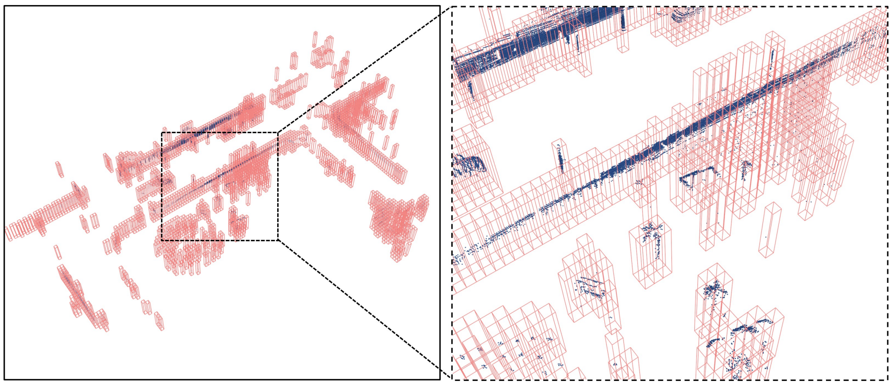

Figure 1. Visualization of point cloud pillars. We partition the whole point cloud into equally sized pillars in the vertical direction relative to the ground. We assume that the points within each pillar exhibit consistent rigid motion patterns.

3.1. Multi-resolution pillars

Local rigidity is a common prior in scene flow estimation. It refers to the assumption that neighboring regions within the scene exhibit consistent rigidity motion patterns. In the context of autonomous driving, we assume that local motion in the vertical direction (perpendicular to the ground) within certain regions is consistent. Therefore, we partition the point cloud into pillars of equal size as shown in Figure 1. We assume that the points within each pillar exhibit consistent rigid motion patterns. The advantage of using pillars to represent local rigid motion is that it does not require explicit segmentation of foreground objects and background in the scene. By partitioning the point cloud into pillars, we can capture the local motion patterns without the need for explicit object boundaries. This approach allows us to focus on modeling and estimating the motion within each pillar.

However, as shown in Figure 1, the same object may be partitioned into multiple pillars. This leads to different rigid transformations being used for points on the same object. To address this issue, we propose a multi-resolution pillar partition method, which divides the point cloud into different resolutions using different pillar sizes as shown in Figure 2. Smaller pillars exhibit higher motion consistency among points and focus more on local fine-grained point cloud motion. Larger pillars help prevent an object from being split into multiple distinct rigid motions, providing a broader scope for enforcing consistency constraints among points.

Specifically, we partition the point cloud into L layers of different sizes, and $\mathbf{T}_i^l = \{\mathbf{R}_i^l \in SO(3), \mathbf{t}_i^l \in \mathbb{R}^3\}$ is used to represent the rigid transformation of the i-th pillar in the l-th layer.

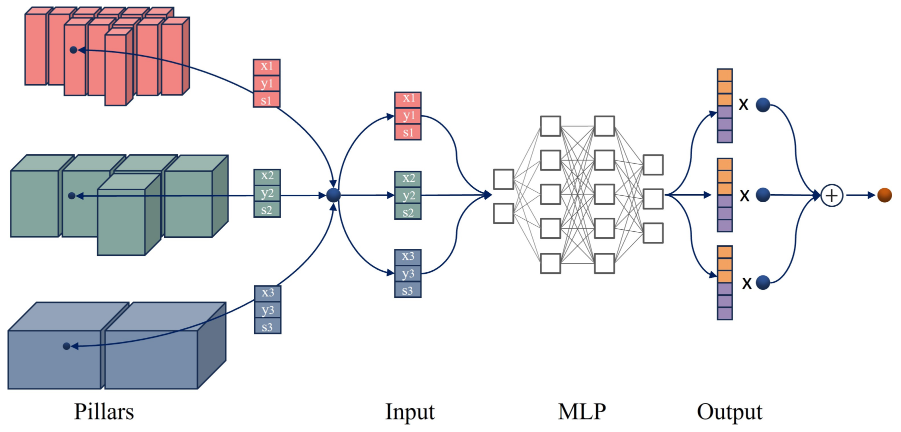

Figure 2. Pipeline for scene flow optimization. The point cloud is partitioned into pillars of different resolutions. For a point in the point cloud, we can obtain the center coordinates and size of the pillar it belongs to at different resolutions. Then, the coordinates (xy) and size (s) are used as inputs to the model. The model outputs a 6-dimensional vector for each layer, where the first three dimensions represent the rotation vector and the last three dimensions represent the translation vector. ● represents the source point, and ● represents the predicted point. $\times$ means applying the rotation and translation to the source point to get predicted point, and $\oplus$ represents taking the sum of the predicted results and then averaging them. The final predicted point is the mean of the predicted points from all layers.

Multi-resolution pillars are the result of partitioning a point cloud using different sizes. The process divides the 3D space into small cube-like regions called pillars, assigning each point to a pillar based on its location. The outputs include centroid coordinates for each pillar and the corresponding pillar index for each point.

In this case, for point p_i in the point cloud P_t, the scene flow f_i can be represented by the following equation:

$$f_i = \frac{1}{L}\sum_{l=0}^{L}(\mathbf{R}^l p_i + \mathbf{t}^l - p_i), \tag{2}$$

where $\mathbf{R}^l$ and $\mathbf{t}^l$ represent the rotation and translation of the pillar where point p_i belongs to in the l-th layer.

3.2. Neural rigidity prior

Many optimization-based methods introduce explicit regularization when modeling scene flow based on rigid priors. For example, RSF [10] optimizes object-level scene flow and initializes the size of object bounding boxes using the size of a car. Although they can optimize the size of the bounding box for objects, the ability is limited and cannot optimize the size of small objects well. To avoid explicit regularization, inspired by NSFP [8], we use an multilayer perceptrons (MLP) to model the rigid transformation of pillars. The rigid transformations of adjacent pillars may be similar, and this consistency constraint can be encoded in the model.

In detail, we use the center coordinates and the size of the pillars as the input for the model. Since the height of the center coordinates is the same, we only use the XY coordinates in our model. In practice, we use the normalized coordinates. The output of the model is a

6-dimensional vector, where the first three dimensions represent the rotation vector and the last three dimensions represent the translation vector. In fact, for a pillar, we represent the rotation using the following equation:

$$(\mathbf{r},\mathbf{t}) = MLP((x,y,s)), \tag{3}$$

where x and y represent the XY coordinates of the center point of the pillar, and s represents the size of the pillar. **r** represents a rotation vector, and **t** represents a translation vector. When computing coordinate transformations, the rotation vector is converted into a rotation matrix **R**.

As shown in Figure 2, the process of obtaining the predicted flow position for a point in the point cloud involves the following steps: First, the center coordinates and size of the corresponding pillar at different resolutions are obtained. These xy coordinates and size values are then used as inputs to the model. The model predicts the separate rigid transformations for each layer. Finally, the predicted point cloud is obtained by applying rigid transformations to the source point. We average the predicted point clouds from each layer to determine the final predicted point cloud.

3.3. Loss function

The chamfer distance [28] is utilized for our flow optimization, which is commonly leveraged in unsupervised learning scene flow and optimization-based scene flow networks. Typically, the chamfer distance loss is computed by the point distances both from the source to the target point cloud and from the target to the source point cloud. The specific calculation formula is as follows:

$$\mathscr{L}_C = \sum_{p\in P} \min_{q_j\in Q} ||p - q_j||_2^2 + \sum_{q\in Q} \min_{p_i\in P} ||p_i - q||_2^2, \tag{4}$$

where p_i represents the i-th point in the source point cloud P. q_i represents the j-th point in the target point cloud Q. And $||.||$ denotes the Euclidean distance.

From the above formula, it can be observed that calculating the distance between one point with the whole point clouds in the other frame requires searching for the nearest neighboring points. The complexity of chamfer distance computation is O(n^2). This can lead to significant increases in computation time when dealing with a large number of points. To address this issue, F-NSF [9] introduces the distance transform (DT) as an efficient loss function. The DT loss builds a DT map for the target point cloud. When calculating the distance from the source point cloud to the target point cloud, we can directly query the pre-constructed map to obtain the distance to the nearest point. This computational approach has a complexity of O(n), significantly accelerating the optimization speed.

In this paper, we choose the DT loss as our optimization loss function:

$$\mathscr{L} = \min_{p'\in P', q\in\mathscr{G}} ||p' - q||_2, \tag{5}$$

where P' refers to the predicted point cloud obtained by adding the source point cloud and the scene flow. $\mathscr{G}$ represents the constructed DT map, and q represents the regularly spaced point in a voxel.

4. Results

In this section, we evaluate the effectiveness of our method on three commonly used autonomous driving datasets. We compare our method with several state-of-the-art approaches, including supervised methods, unsupervised methods, and optimization-based methods.

Table 1. Results for 2048 and 8192 points on the Argoverse dataset.

Points	Method	Supervision	EPE3D(m) ↓	*Acc3DS*(%)↑	*Acc3DR*(%)↑	θ_ϵ(rad) ↓
2048	FlowNet3D [3]	Full	0.455	1.34	6.12	0.736
	PointPWC-Net [12]	Full	0.405	8.25	25.47	0.674
	JGF [17]	Self	0.542	8.80	20.28	0.715
	PointPWC-Net [12]	Self	0.409	9.79	29.31	0.643
	F-NSF [9]	Optim	0.239	22.44	46.47	0.472
	OptFlow [11]	Optim	<u>0.20</u>	<u>43.85</u>	<u>65.50</u>	<u>0.39</u>
	Ours	Optim	**0.112**	**52.33**	**82.46**	**0.383**
8192	FLOT [4]	Full	0.821	2.00	8.84	0.967
	R3DSF [29]	Weakly	0.417	32.52	42.52	0.551
	NSFP [8]	Optim	<u>0.113</u>	46.32	72.68	<u>0.347</u>
	F-NSF [9]	Optim	0.118	<u>69.93</u>	<u>83.55</u>	0.352
	Ours	Optim	**0.091**	**73.11**	**90.50**	**0.343**

The dataset consists of 212 test scenes. 'Full' represents supervised scene flow estimation. 'Self' represents unsupervised scene flow estimation methods. 'Weakly' represents weakly supervised scene flow estimation methods. 'Optim' represents optimization-based methods. The best results are highlighted in **bold**, and the second best results are <u>underlined</u>.

Table 2. Results for 2048 and 8192 points on the Waymo dataset.

Points	Method	Supervision	EPE3D(m) ↓	*Acc3DS*(%)↑	*Acc3DR*(%)↑	θ_ϵ(rad) ↓
2048	F-NSF[9]	Optim	<u>0.185</u>	<u>33.49</u>	<u>62.57</u>	<u>0.404</u>
	Ours	Optim	**0.121**	**65.58**	**87.68**	**0.372**
8192	FLOT [4]	Full	0.702	2.46	11.30	0.808
	R3DSF[29]	Weakly	0.414	35.47	44.96	0.527
	NSFP[8]	Optim	0.138	53.62	78.57	0.339
	F-NSF[9]	Optim	<u>0.106</u>	**77.53**	<u>88.99</u>	**0.329**
	Ours	Optim	**0.092**	<u>75.47</u>	**91.93**	<u>0.356</u>

The dataset consists of 202 test scenes. The evaluation results for previous methods on 8192 points are taken from the report of NSFP [9]. The best results are highlighted in **bold**, and the second best results are <u>underlined</u>.

4.1. Implementation detail

The experiment runs on an NVIDIA RTX 4090 GPU with PyTorch=2.0.1 and CUDA=11.7. The optimizer is Adam [30], and the initial learning rate is 0.008. We partition the point cloud using sizes 1 m, 4 m and 8 m. MLP has 8 hidden layers and 128 hidden units. We primarily conduct tests on datasets with 2048 and 8192 points. DT grid size is set to 10.

4.2. Dataset

Our proposed method primarily focuses on scene flow estimation in real-world autonomous driving scenes. Therefore, we conduct tests on three commonly used autonomous driving datasets, namely Argoverse [31], Waymo [32], and nuScenes [33]. These three datasets are all obtained from real-world scenarios using lidar sensors and contain a large number of challenging scenes. Since ground truth labels for the scenes are not provided, we utilize self-motion and 3D object tracking to create pseudo-labels. The processing methods for datasets Argoverse, Waymo, and nuScenes respectively follow approaches [8,24], [34,35], and [8,24]. In practice, we have used the preprocessed datasets provided by the approaches [8] and [9].

Table 3. Results for 2048 and 8192 points on the nuScenes dataset.

Points	Method	Supervision	EPE3D(m) $\downarrow$	$Acc3DS(\%)\uparrow$	$Acc3DR(\%)\uparrow$	θ_ε(rad) $\downarrow$
2048	FlowNet3D [3]	Full	0.505	2.12	10.81	0.620
	PointPWC-Net [12]	Full	0.442	7.64	22.32	0.497
	JGF [17]	Self	0.625	6.09	0.139	0.432
	PointPWC-Net [12]	Self	0.431	6.87	22.42	0.406
	Graph Prior[24]	Optim	0.289	20.12	43.54	0.337
	F-NSF[9]	Optim	0.322	34.69	61.13	0.293
	OptFlow[11]	Optim	**0.216**	<u>40.85</u>	<u>65.97</u>	**0.271**
	Ours	Optim	<u>0.263</u>	**53.71**	**75.22**	<u>0.291</u>
8192	F-NSF[9]	Optim	<u>0.252</u>	<u>49.77</u>	<u>74.55</u>	**0.240**
	Ours	Optim	**0.232**	**63.43**	**79.59**	<u>0.280</u>

The dataset consists of 310 test scenes. The evaluation results for previous methods on 2048 points are taken from the report of NSFP [8]. The best results are highlighted in **bold**, and the next best results are <u>underlined</u>.

4.3. Metric

We evaluated the optimization results of the scene flow using commonly used metrics such as in [3,8,12,15,24]. EPE3D indicates the average absolute distance between two point clouds. Acc3DS represents the percentage of scene flow where EPE3D ≤ 0.05 m or relative EPE3D error $\leq 5\%$. Acc3DR represents the percentage of scene flow where EPE3D ≤ 0.1 m or relative EPE3D error $\leq 10\%$. θ_ε denotes the mean angular error between the estimated scene flow and the ground truth.

4.4. Comparison with other methods

In the supervised methods, FlowNet3D [3], FLOT [4], and PointPWC-Net [12] train their models on the Flythings3D dataset. Additionally, PointPWC-Net can also be trained using an unsupervised approach. JGF [17] is an unsupervised scene flow learning method. R3DSF [29] is a weakly supervised approach that does not directly use the ground truth values of scene flow during model training. Instead, it utilizes background masks and ego-motion signals for training. Graph prior [24], NSFP [8], F-NSF [9], and Optflow [11] are optimization-based methods. In some learning-based methods, such as PointPWC-Net, the estimation of scene flow often restricts the point cloud range to 35 m. However, optimization-based methods usually do not require such restrictions when evaluating scene flow. Therefore, to ensure the fairness, we do not limit the point cloud range in evaluation. Furthermore, for some methods, the evaluation results have already been reported, so we directly use those reported results, which will be specified in the corresponding table.

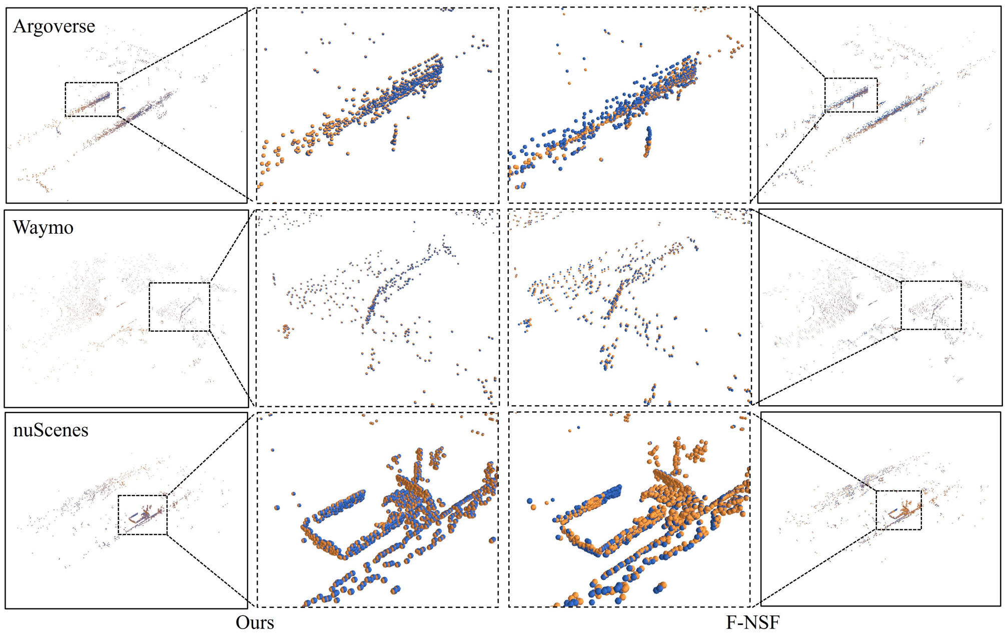

Figure 3. Visualization results for scene flow with 2048 points. From top to bottom the results are for Argoverse, Waymo, and nuScenes respectively. Our results are compared with those of F-NSF [9]. Blue points represent the ground truth, and orange points represent the predicted point cloud.

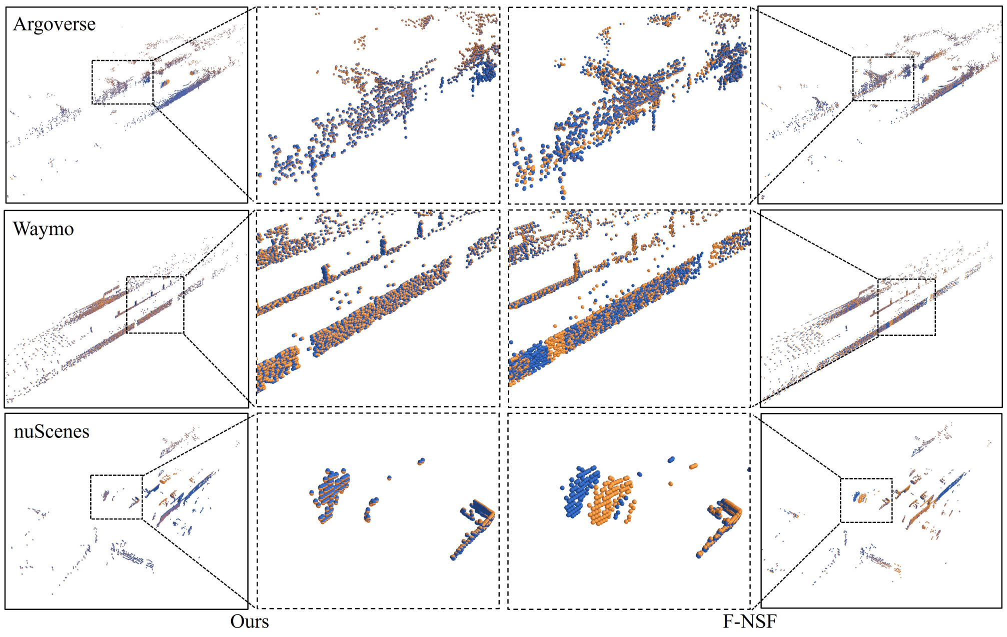

Figure 4. Visualization results for scene flow with 8192 points. From top to bottom the results are for Argoverse, Waymo, and nuScenes respectively. Our results are compared with those of F-NSF [9]. Blue points represent the ground truth, and orange points represent the predicted point cloud.

Our test results on the three datasets with 2048 points achieved state-of-the-art accuracy as shown in Tables 1–3. Specifically, our results show a significant improvement in accuracy, with Acc3DS increasing by 19% to 95% and Acc3DR increasing by 14% to 40%. In the test results with 8192 points, our results still achieve the best performance on the Argoverse dataset. Although some metrics on Waymo and nuScenes are slightly inferior to the best results, they still outperform learning-based methods. Figures 3, 4 show the visualisation results for 2048 and 8192 points, respectively. Due to the lack of rigid constraints, the estimation results of F-NSF [9], which is based on point-to-point scene flow optimization, are significantly worse than our results, especially for points belonging to the static background. Thanks to the constraints of local rigidity, our method ensures the consistency of local point cloud motion. This guarantees that the network can optimize the rigid motion of pillars even with a smaller number of points in the point cloud.

4.5. Ablation studies

To validate the effectiveness of the method, we conducted ablation experiments on the Argoverse dataset with 8192 points, as shown in Table 4. ‘W/O neural’ represents not using neural implicit representation to model the rigid transformation of pillars, but directly optimizing a rigid transformation for each pillar. The experiment results indicate that the experiment without using implicit representation yields large errors in terms of EPE3D and θ_ε. This demonstrates that the estimated rigid transformations for the majority of pillars are incorrect. This is due to the explicit representation leading to a lack of inherent constraints between different pillars. ‘W/O multi-resolution’ means not using a combination of multi-resolution pillar rigid transformations to represent scene flow. Experiment results show that using just a single layer is worse than multi-resolution. This is because some objects in the scene may be split into several pillars, and points on the same object may not be subject to the same rigidity constraints.

Table 4. Ablation experiment results for 8192 points on the Argoverse dataset.

Method	EPE3D(m) ↓	*Acc3DS* (%)↑	*Acc3DR*(%)↑	θ_ε(rad) ↓
W/O neural	0.706	16.99	18.23	1.614
W/O multi-resolution	0.093	71.02	88.52	0.356
Complete	**0.091**	**73.11**	**90.50**	**0.343**

4.6. Limitation

Compared to these learning-based methods for scene flow estimation, optimization-based methods perform optimization at runtime and are therefore more time-consuming. Learning-based methods can usually run in real-time and optimization-based methods can only run offline.

5. Conclusion

In this paper, We propose a novel scene flow optimization method with neural rigidity prior for the autonomous driving environment. The point cloud is partitioned into pillars of different resolutions. Scene flow of a point is represented as a combination of rigid transformations of multi-resolution pillars. Local rigidity is modeled in the network using implicit representation. We conduct tests on commonly used autonomous driving datasets. The experiment results show that our method significantly improves the estimation performance of scene flow on sparse point clouds compared to point-to-point optimization methods and strongly regularized optimization methods. This improvement is attributed to the introduction of rigid priors and the multi-resolution representation of scene flow. Finally, we demonstrate the feasibility of using neural networks to represent rigid priors and the effectiveness of using a combination of multi-resolution pillar rigid transforms to represent scene flows.

Conflicts of interests

The authors declare no conflict of interest.

Authors contribution

Conceptualization, Z.F.; Methodology, Z.F.; Software, Z.F.; Visualization, Z.F.; Writing—original draft preparation, Z.F. and J.L.; Investigation, J.L.; Data curation, J.L.; Supervision, H.W.; Project administration, H.W.; Writing—review and editing, H.W. All authors have read and agreed to the published version of the manuscript.

References

[1] Luo C, Yang X, Yuille A. Self-supervised pillar motion learning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, 20–25 June 2021, pp. 3183–3192.

[2] Schuster R, Wasenmüller O, Unger C, Kuschk G, Stricker D. SceneFlowFields++: Multi-frame matching, visibility prediction, and robust interpolation for scene flow estimation. *Int. J. Comput. Vis.* 2020, 128:527–546.

[3] Liu X, Qi CR, Guibas LJ. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Long Beach, USA, 15–20 June 2019, pp. 529–537.

[4] Puy G, Boulch A, Marlet R. Flot: Scene flow on point clouds guided by optimal transport. In *European conference on computer vision*, Glasgow, UK, 23–28 August 2020, pp. 527–544.

[5] Wang G, Wu X, Liu Z, Wang H. Hierarchical Attention Learning of Scene Flow in 3D Point Clouds. *IEEE Trans. Image Process.* 2021, 30:5168–5181.

[6] Gu X, Wang Y, Wu C, Lee YJ, Wang P. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Long Beach, USA, 15–20 June 2019, pp. 3254–3263.

[7] Wang G, Hu Y, Wu X, Wang H. Residual 3-D scene flow learning with context-aware feature extraction. *IEEE Trans. Instrum. Meas.* 2022, 71:1–9.

[8] Li X, Kaesemodel Pontes J, Lucey S. Neural scene flow prior. *Adv. Neural Inf. Process. Syst.* 2021, 34:7838–7851.

[9] Li X, Zheng J, Ferroni F, Pontes JK, Lucey S. Fast neural scene flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Paris, France, 1–6 October 2023, pp. 9878–9890.

[10] 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*, Waikoloa, USA, 2–7 January 2023, pp. 1277–1286.

[11] Ahuja R, Baker C, Schwarting W. OptFlow: Fast Optimization-based Scene Flow Estimation without Supervision. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, Waikoloa, USA, 3–8 January 2024, pp. 3161–3170.

[12] Wu W, Wang ZY, 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, 23–28 August 2020, pp. 88–107.

[13] Wei Y, Wang Z, Rao Y, Lu J, Zhou J. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Nashville, USA, 20–25 June 2021, pp. 6954–6963.

[14] Cheng W, Ko JH. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *European Conference on Computer Vision*, Tel Aviv, Israel, 23–27 October 2022, pp. 108–124.

[15] Liu J, Wang G, Ye W, Jiang C, Han J, *et al.* DifFlow3D: Toward Robust Uncertainty-Aware Scene Flow Estimation with Iterative Diffusion-Based Refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, 16–22 June 2024, pp. 15109–15119.

[16] Mayer N, Ilg E, Hausser P, Fischer P, Cremers D, *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*, Las Vegas, USA, 27–30 June 2016, pp. 4040–4048.

[17] 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*, Seattle, USA, 13–19 June 2020, pp. 11177–11185.

[18] Kittenplon Y, Eldar YC, Raviv D. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, 20–25 June 2021, pp. 4114–4123.

[19] Baur SA, Emmerichs DJ, Moosmann F, Pinggera P, Ommer B, *et al.* Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Montreal, Canada, 10–17 October 2021, pp. 13126–13136.

[20] Li R, Zhang C, Lin G, Wang Z, Shen C. Rigidflow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, USA, 18–24 June 2022, pp. 16959–16968.

[21] 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*, Vancouver, Canada, 17–24 June 2023, pp. 5271–5280.

[22] Jiang C, Wang G, Liu J, Wang H, Ma Z, *et al.* 3dsflabelling: Boosting 3d scene flow estimation by pseudo auto-labelling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, 16–22 June 2024, pp. 15173–15183.

[23] Zhang Q, Yang Y, Li P, Andersson O, Jensfelt P. Seflow: A self-supervised scene flow method in autonomous driving. In *European Conference on Computer Vision*, Milan, Italy, 29 September–4 October 2024, pp. 353–369.

[24] Pontes JK, Hays J, Lucey S. Scene Flow from Point Clouds with or without Learning. In *Int. Conf. 3D Vis. (3DV)*, Fukuoka, Japan, 25–28 November 2020, pp. 261–270.

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

[26] Lin Y, Caesar H. Icp-flow: Lidar scene flow estimation with icp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, 16–22 June 2024, pp. 15501–15511.

[27] Khatri I, Vedder K, Peri N, Ramanan D, Hays J. I Can't Believe It's Not Scene Flow! In *European Conference on Computer Vision*, Milan, Italy, 29 September–4 October 2024, pp. 242–257.

[28] Fan H, Su H, Guibas LJ. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, USA, 21–26 July 2017, pp. 605–613.

[29] Gojcic Z, Litany O, Wieser A, Guibas LJ, Birdal T. Weakly supervised learning of rigid 3D scene flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Nashville, USA, 20–25 June 2021, pp. 5692–5703.

[30] Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv* 2014, ArXiv:1412.6980.

[31] Chang MF, Lambert J, Sangkloy P, Singh J, Bak S, *et al.* Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Long Beach, USA, 15–20 June 2019, pp. 8748–8757.

[32] Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, *et al.* Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Seattle, USA, 13–19 June 2020, pp. 2446–2454.

[33] Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, *et al.* nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Seattle, USA, 13–19 June 2020, pp. 11621–11631.

[34] Jin Z, Lei Y, Akhtar N, Li H, Hayat M. Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, USA, 18–24 June 2022, pp. 7233–7243.

[35] Yang J, Shi S, Wang Z, Li H, Qi X. St3d: Self-training for unsupervised domain

adaptation on 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, Nashville, USA, 20–25 June 2021, pp. 10368–10378.