Exploiting Rigidity Constraints for LiDAR Scene Flow Estimation

Guanting Dong* Yueyi Zhang* Hanlin Li Xiaoyan Sun Zhiwei Xiong†
University of Science and Technology of China

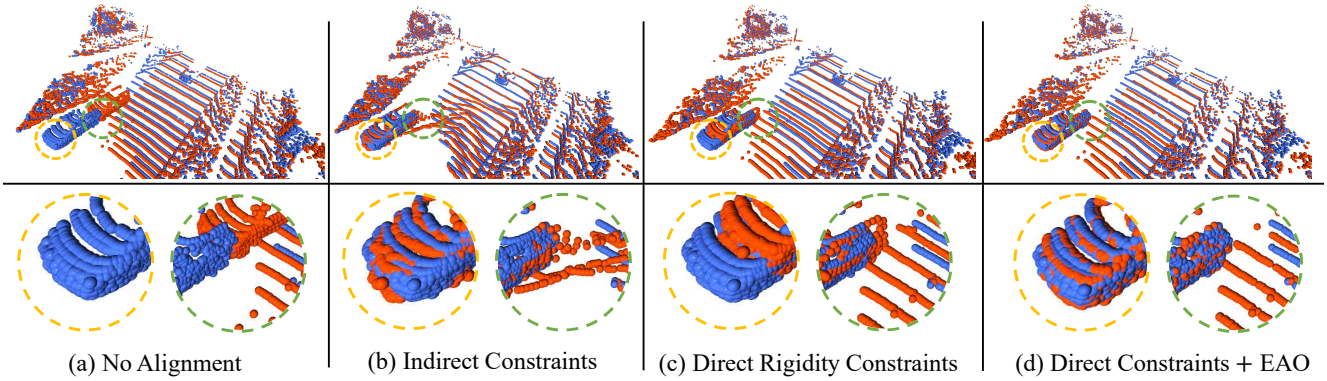

Figure 1. **The inference examples of different recurrent neural networks.** From left to right: (a) source (red) and target (blue) input point clouds from *lidarKITTI* scene flow dataset [23], (b) FlowStep3D that indirectly introduces Laplacian constraints on the optimization objective [16], (c) our method that directly introduces multi-body rigidity constraints, and (d) our method that introduces an error awarded optimization (EAO) strategy based on (c). Interesting artifacts are circled and zoomed-in at the bottom.

Abstract

Previous LiDAR scene flow estimation methods, especially recurrent neural networks, usually suffer from structure distortion in challenging cases, such as sparse reflection and motion occlusions. In this paper, we propose a novel optimization method based on a recurrent neural network to predict LiDAR scene flow in a weakly supervised manner. Specifically, our neural recurrent network exploits direct rigidity constraints to preserve the geometric structure of the warped source scene during an iterative alignment procedure. An error awarded optimization strategy is proposed to update the LiDAR scene flow by minimizing the point measurement error instead of reconstructing the cost volume multiple times. Trained on two autonomous driving datasets, our network outperforms recent state-of-the-art networks on lidarKITTI by a large margin. The code and models will be available at https://github.com/gtdong-ustc/LiDARSceneFlow.

1. Introduction

LiDAR scene flow estimation is a fundamental task in 3D scene understanding, which is vital to autonomous driving and many other computer vision applications. Compared with optical flow estimation, which focuses on feature matching for ordered RGB data, scene flow estimation poses an extra challenge as it involves aggregating unordered data from the LiDAR sensor.

To deal with unordered data, several convolutional neural networks (CNNs) for scene flow estimation have been proposed. In term of convolution operation, these CNN-based methods can be divided into two categories, 2D convolution based methods [1, 11, 32] and 3D convolution based methods [16, 19, 26, 35, 36]. Typically, the 2D convolution based methods convert 3D point clouds into 2D representations, *e.g.* depth map and bird's-eye-view map, which inevitably brings in quantification errors. With recent advances in point cloud convolution [27], methods utilizing the 3D convolution operations are proposed to directly operate unordered point cloud data and achieve success [16, 19, 34–36].

HPLFlowNet [10] and FlowNet3D [19] introduced different 3D convolutions to operate the unordered point clouds directly. They both employed an end-to-end network to learn deep hierarchical features of point clouds and correlation embeddings for scene flow estimation. For small displacements between two consecutive point clouds, these works are effective but not satisfying in large displacements. Then PointPWC-Net, utilizing a coarse-to-fine strategy, was proposed to handle large displacements [36]. The scene flow is first computed at low resolution to estimate large displacements and then gradually refined at high resolution. One drawback of PointPWC-Net lies in the error accumula-

*Equal contribution. † Corresponding author: zwxiong@ustc.edu.cn

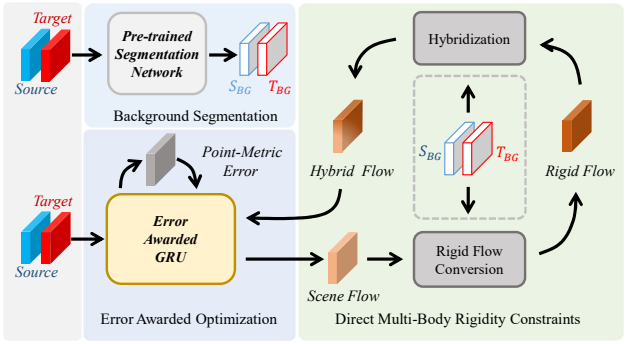

Figure 2. The overview of our scene flow estimation pipeline. The background masks, S_{BG} and T_{BG}, are obtained from a pre-trained segmentation network [8]. For implementing direct multi-body rigidity constraints, the predicted scene flow is converted to the rigid flow and mixed with the rigid flow as the hybrid flow, which is the input of next iteration. In error awarded optimization, a GRU-based recurrent neural network takes a point-metric error to guide the iterative updating of the predicted scene flow.

tion in the early step. To tackle this problem, recent methods employ a gated recurrent unit (GRU) based optimization architecture to update the predicted scene flow iteratively [19, 35]. However, this iterative mechanism suffers from structure distortion and high time consumption.

To prevent structure distortion, previous works introduced indirect constraints into iterative optimization, in the form of adding regularization terms in the objective function of the recurrent neural network, such as Laplacian regularization [16]. However, the recurrent network with indirect constraints still produces scene flow that breaks the geometric structure in some cases. Fig. 1(b) shows the structure distortion generated by FlowStep3D [16]. Different from indirect constraints that adding the regularization term, we bring in direct multi-body rigidity constraints to the recurrent neural network by converting the scene flow to rigid flow via a differentiable converter. The explicit introduction of multi-body rigidity constraints alleviates structure distortion.

Although the employment of direct rigidity constraints preserves the object structure, the performance of object aligning is not satisfying. Previous optimization-based methods tackle the misalignment by re-encoding the warped source frame to reconstruct the cost volume, which is time-consuming. Inspired by the Iterative Closest Point (ICP) algorithm [4], we propose an error awarded optimization strategy that minimizes a point-metric error to iteratively update the predicted scene flow until obtaining the perfect alignment results. Specifically, we employ a GRU-based recurrent network to adopt the nearest neighbor error [25] between the warped source frame and target frame as the optimization objective in the inference. Fig. 1(c,d) show the inference examples of our proposed framework with direct rigidity constraints and the error awarded optimization.

In Fig. 2, we show the overview of our scene flow estimation pipeline, which provides the abstraction of error awarded optimization, direct multi-body rigidity constraints and the pretrained segmentation network. Similar to [8], for training our proposed network, we don't need the ground-truth scene flow. The essential supervision is the binary background mask and ego-motion. In other words, our network is trained in a weakly supervised manner. To validate the effectiveness of the proposed network, we train the network on two datasets *SemanticKITTI* and *Waymo Open*, and test on *lidarKITTI* with and without ground points.

The main contributions are summarized as follows:

- We introduce direct multi-body rigidity constraints to a GRU-based recurrent neural network for LiDAR scene flow estimation.
- We propose an effective error awarded optimization strategy that updates the scene flow by minimizing a point-metric error instead of reconstructing cost volume multiple times.
- The proposed network is trained in a weakly supervised manner. Experimental results demonstrate that our method outperforms state-of-the-art scene flow estimation methods by a large margin.

2. Related Work

Point cloud based scene flow estimation. Recently, learning-based methods made great progresses in the field of point clouds [18,27,37], which inspires the application in point cloud based scene flow estimation. FlowNet3D [19] successfully extended FlowNet [6] by directly operating point clouds with PointNet++ [27]. HPLFlowNet [10] rearranged points into a permutohedral lattice [15] introduced in SplatNet [28] to apply bilateral convolution layers [12], which led to efficient and robust non-rigid 3D flow computation. Mittal *et al.* presented two self-supervised losses to allow the end-to-end training of the deep neural network on large-scale, unlabeled autonomous driving datasets [25]. Inspired by the classical pyramid networks [29], PointPWC-Net [36] designed a point-based cost volume over unordered point clouds layer in a coarse-to-fine manner and showed impressive results on synthetic datasets in both supervised and self-supervised manners. Different from the aforementioned methods, FLOT [26] proposed a simple correspondence-based end-to-end scene flow network and adopted the optimal transport to find correspondences.

Constraints for flow estimation. There are attempts to estimate flow by extracting physical priors, for example, multi-body rigidity and local smoothness. In the early stage, these works utilized the motion characters to separate moving objects [5, 14]. Golyanik *et al.* used rigidity constraints over segmentation of RGBD-frames [9], while Vogel *et al.* modeled scene flow using piece-wise rigidly moving pla-

nar patches, *i.e.* local smoothness [33]. Then for further exploiting the multi-body rigidity of motion [20], Ma *et al.* made use of depth and flow predictions from a stereo RGBD setup in an optimization framework to obtain the 3D motion of each instance [21]. However, this method needs instance segmentation as inputs, which is difficult to achieve even for the state-of-the-art segmentation methods. Therefore, Gojcic *et al.* proposed a weakly supervised training strategy for rigid scene flow estimation, which relaxes the requirement for dense scene flow supervision with simple binary background segmentation mask and ego-motion annotations [8]. Our proposed method expands the framework of [8] by recurrently updating the scene flow prediction, and achieves outstanding alignment with a lightweight recurrent network. In contrast, RAFT-3D [32] iteratively updated the rigid-motion embeddings during inference, but only extended RAFT [31] from optical flow estimation to scene flow estimation with an ordered RGBD as inputs. With the same strategy as RAFT-3D, Baur *et al.* [1] rebuilt RAFT [31] to perform iterative scene flow estimation and motion segmentation on the PointPillar-based [17] bird's-eye-view feature representations of point clouds.

Recurrent updating for flow estimation. Recent works utilize a point cloud convolution-based recurrent network to update the predicted scene flow through iterations [16, 35]. Inspired by RAFT [31], PV-RAFT [35] estimated the point-voxel correlation fields to integrate two types of correlations and captured all-pair relations, which made the iterative optimization of scene flow more accurate and efficient during iterations. Different from PV-RAFT, FlowStep3D [16] constructed an all-to-all correlation field at the single low resolution and updated this field at each iteration by re-coding the warped source frame. However, the point cloud convolution-based recurrent network suffers from object structure distortion, even though enforcing a strong regularization term in the loss function. This observation motivates us to exploit direct rigidity constraints for the robust iterative optimization of scene flow estimation.

3. Problem Definition

Given two consecutive point clouds, the source point cloud $X = \{x_i \in \mathbb{R}^3\}_{i=1}^{N_x}$ and the target point cloud $Y = \{y_j \in \mathbb{R}^3\}_{j=1}^{N_y}$, we define the scene flow between two point clouds as $F = \{f_i\}_{i=1}^{N_x}$, where N_x and N_y are numbers of points in the point clouds, $f_i \in \mathbb{R}^3$ represents the motion vector that aligns a point $x_i \in X$ towards its corresponding location x_i' in the other point cloud. It should be noted that the number of points in the source may differ from the number of points in the target, *i.e.* N_x and N_y are not necessarily equal. In general, the scene flow provides soft correspondences between two consecutive LiDAR point clouds. However, because of sparse reflection and motion occlusions existing in the LiDAR data, it is difficult to obtain accurate dense scene flow.

To address this problem, we assume that the motion of two consecutive point clouds, *i.e.*, scene flow F, is composed of ego-motion and multi-object motions, following [8]. Specifically, the points in the source point cloud can be clustered into $K+1$ abstractions, each of which has a point mask M_k. Points belonging to the same abstraction share a universal $SE(3)$ transformation. We define a set of $K+1$ abstraction transformations as $\mathcal{T} = \{T_k \in SE(3)\}_{k=0}^{K}$. Among these transformations, the background determines the ego-motion $T_0 \in \mathcal{T}$ and the foreground objects determine the object-level motions $T_1, T_2, ..., T_K \in \mathcal{T}$. It should be noted that not all the points in the source can be clustered in an abstraction owning a rigid transformation. For those points not having a rigid transformation, conventional scene flow needs to be computed. The final predicted scene flow is a hybrid flow, which is a mix of rigid flow and scene flow.

4. Method

We first process the source and target point clouds to generate the abstraction masks, and then estimate the scene flow with a recurrent neural network. The input of our recurrent work is the source point cloud, the target point cloud and the abstraction masks of the source point cloud. The output is the hybrid flow, which is a mix of scene flow and rigid flow. As shown in Fig. 3, our iterative framework consists of two stages, the initialization stage and the recurrent updating stage. In the initialization stage, we calculate the initial cost volume CV^0, hidden state h^0, hybrid flow HF^0 and weight, which will be used for calculation in the recurrent updating stage. In addition, the initial hybrid flow is generated for further refinement. In the recurrent updating stage, we employ an error awarded GRU to update all the variables and refine the hybrid flow. This stage will be conducted multiple times to achieve better flow estimation. The structures of these two stages are illustrated in Fig. 3. To avoid confusion, we use the iteration subscript $l = 0$ for the initialization stage and $l = 1, 2, ..., L$ for the recurrent updating stage. In the following, we will describe the generation of abstraction masks and details of important components in the two stages.

4.1. Abstraction Mask Generation

We first generate a binary background mask from a pre-trained segmentation network [8] that is trained on *SemanticKITTI*. The details about the binary background mask for LiDAR point cloud datasets are available in our supplement. Then, we apply the DBSCAN clustering algorithm [7] to the foreground points and obtain a set of binary abstraction masks, each of which refers to a foreground object. Assume that there are in total $K+1$ abstractions including the background, the set of abstraction masks can be

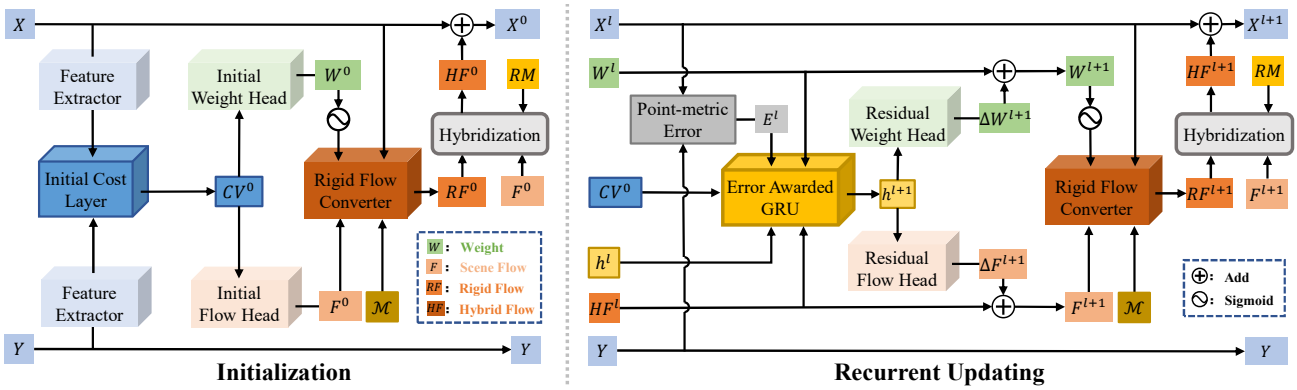

Figure 3. The architecture of our recurrent neural network. We divide the forward inference of our method into two stages, *i.e.* initialization and recurrent updating. In the initialization stage, we obtain the initial hybrid flow HF^0. While turning to recurrent updating, we introduce a point-metric error to update the initial hybrid flow without recalculating the cost volume multiple times.

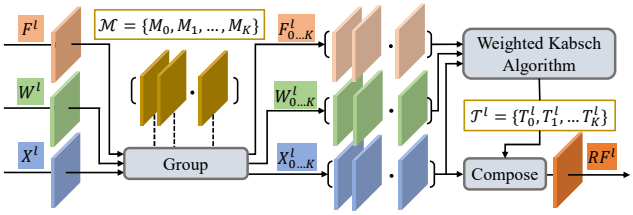

Figure 4. Rigid flow converter. 'Group' operation splits the variable into several abstraction with the guidance of abstraction masks $\mathcal{M}$. 'Compose' means recombine the obtained abstraction transformations into a complete rigid flow.

denoted as $\mathcal{M} = \{M_k\}_{k=0}^K$, which contains the background mask and the foreground object masks.

4.2. Initialization

Feature extractor. A feature extractor with shared weights is utilized to extract features from both source and target point clouds. The feature extractor is consisted of two *set_conv* layers proposed by FlowNet3D [19] and utilizes the furthest point sampling algorithm for down-sampling point clouds.

Initial cost layer. The initial cost layer calculates an all-to-all correlation field at low resolution and then aggregate the correlation field to construct an initial cost volume CV^0. The initial cost volume builds the global correlations between the source and target point clouds. Details of the feature extractor and initial cost layer are available in the supplement.

Initial heads. Based on the initial cost volume, we introduce two initial heads, a flow head and a weight head, to compute the initial scene flow and weight, respectively. The flow head comprises two one-dimensional convolutional layers and takes the initial cost volume CV^0 as input to generate the inital scene flow F^0. In practice, due to the sparse reflection and motion occlusions, not every point has its correspondence. Therefore, we employ a weight head to indicate whether the soft correspondences, denoted as F^0, are inliers or outliers. Unlike the flow head, the weight head additionally introduces the flow and its corresponding feature embedding as inputs to generate the initial weight W^0.

Rigid flow converter. In order to introduce direct rigidity constraints, we employ a non-parametric converter to preserve the object structure in the warped source scene during iterations. The architecture of our rigid flow converter is shown in Fig. 4. Specifically, we first utilize the abstraction masks to split the scene into multiple abstractions, and then employ the differentiable weighted Kabsch algorithm [13] to calculate the transformations for each abstraction, producing a set of abstraction transformations $\mathcal{T} = \{T_k\}_{k=0}^K$. Finally, the rigid flow is obtained by composing $\mathcal{T}$ with the guidance of $\mathcal{M}$. Besides, the initial hybrid scene flow HF^0 is the mix of scene flow F^0 and rigid flow RF^0.

4.3. Recurrent Updating

Unlike the initialization stage, in order to generate accurate scene flow prediction, we implement an optimizer with an error awarded gated recurrent unit (EGRU) and two residual heads. The optimizer updates the hybrid flow through iterations. At each iteration, the optimizer updates the variables as

$$\begin{aligned} &HF^l + \Delta F^{l+1} \rightarrow F^{l+1}, \\ &W^l + \Delta W^{l+1} \rightarrow W^{l+1}, \\ &\mathbf{RFC}(F^{l+1}, \sigma(W^{l+1}), \mathcal{M}) \rightarrow RF^{l+1}, \\ &RM \cdot RF^{l+1} + (1 - RM) \cdot F^{l+1} \rightarrow HF^{l+1}, \end{aligned} \tag{1}$$

where ΔF^l and ΔW^l are the intermediate residuals computed by the residual heads. The residual heads including flow and weight head, don't share parameters with the initial heads in the initialization stage. They use the new hidden state h^{l+1} as input instead of the initial cost volume CV^0. The $\sigma(\cdot)$ is the sigmoid function that normalizes the weight W^l to the range $[0, 1]$. **RFC** means the rigid flow conversion.

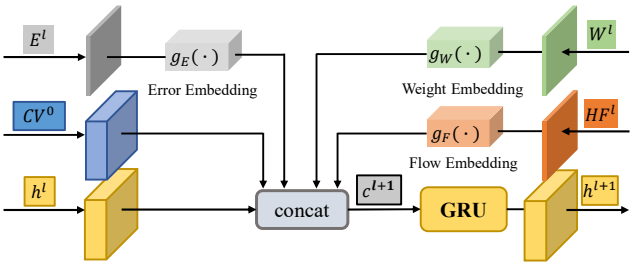

Figure 5. Structure of the error awarded gated recurrent unit.

Error awarded gated recurrent unit. Inspired by the ICP algorithm [3], our EGRU updates the previous hidden state by minimizing the point-metric error in inference. The point-metric error E^l measures the nearest neighbor distance between two consecutive point clouds [25] at $\{l+1\}$-th iteration. Given a warped source point cloud $X^l = \{x_i^l \in \mathbb{R}^3\}_{i=1}^{N_x}$, which is calculated from previous iteration, and the target point cloud $Y = \{y_j \in \mathbb{R}^3\}_{j=1}^{N_y}$, the nearest neighbor distance is computed between the warped source point x_i^l and its closest correspondences $y_j \in Y$, *i.e.* $E^l(x_i^l) = ||x_i^l - y_j||_1$, where $||\cdot||_1$ is the $L1$ norm. Additionally, the warped source point cloud is calculated as $X^l = X + HF^l$. We take the point-metric error as the optimization objective to supervise the updating of the initial hybrid flow, which makes our recurrent neural network locate the unaligned region and progressively refine the initial hybrid flow through iterations.

We show the architecture of EGRU in Fig. 5. Three feature embedding modules, *i.e.*, $g_W(\cdot)$, $g_F(\cdot)$, and $g_E(\cdot)$, are first employed to project weight W^l, hybrid flow HF^l, and point metric error E^l into high-dimensional spaces. Then, we define c^{l+1} as the concatenation of the initial cost volume CV^0, the low-dimensional variables and their corresponding high-dimensional feature representations. The low-dimensional variables include the current point-metric error E^l, the hybrid flow HF^l and the weight W^l. Finally, given previous iteration hidden state h^l, together with current iteration information c^{l+1}, our EGRU produces a new hidden state h^{l+1}. The calculations inside EGRU are as follows

$$\begin{aligned} z^{l+1} &= \sigma(set\_conv_z([h^l, c^{l+1}])), \\ r^{l+1} &= \sigma(set\_conv_r([h^l, c^{l+1}])), \\ \tilde{h}^{l+1} &= tanh(set\_conv_h([r^{l+1} \odot h^l, c^{l+1}])), \\ h^{l+1} &= (1 - z^{l+1}) \odot h^l + z^{l+1} \odot \tilde{h}^{l+1}, \end{aligned} \tag{2}$$

where $\odot$ is the Hadamard product, $[\cdot,\cdot]$ is the concatenation operation and $\sigma(\cdot)$ is the sigmoid function. We obtain the initial hidden state h^0 by passing with the features of the source point cloud X through two set_conv layers.

The recurrent updating stage of our optimization architecture starts from the initial hybrid flow HF^0 to iteratively update itself by minimizing a point-metric error and eventually achieves the final result, *i.e.*, $HF^l \rightarrow HF^*$. The final hybrid flow is a combination of final scene flow and rigid flow with the supervision of the rigidity mask RM, *i.e.*, $HF^* = RM \cdot RF^* + (1 - RM) \cdot F^*$.

4.4. Training Loss

To train our recurrent neural network, we unroll L iterations and apply a loss function for each iteration prediction. We define the overall loss function $\mathcal{L}$ as the sum of losses at each iteration in the sequence, *i.e.*, $\mathcal{L} = \sum_{l=1}^{L} \mathcal{L}^l$. Here, $\mathcal{L}^l$ means the weakly supervised loss for constraining the predicted scene flow F^l and rigid flow RF^l at l-th iteration. The loss at each iteration consists of ego-motion loss, rigidity loss and chamfer distance loss.

Ego-motion loss ($\mathcal{L}_{\text{ego}}$). We utilize an ego-motion error $\mathcal{L}_{ego}$ that measures the $L1$-discrepancy between the background points transformed with the estimated (R_{ego}, t_{ego}) and the ground-truth parameters $(\hat{R}_{\text{ego}}, \hat{t}_{\text{ego}})$. Mathematically, the ego-motion loss is formulated as

$$\mathcal{L}_{\text{ego}} = \frac{1}{N_{BG}} \sum_{j=1}^{N_{BG}} \|(R_{ego} x_j^b + t_{ego}) - (\hat{R}_{\text{ego}} x_j^b + \hat{t}_{\text{ego}})\|_1, \tag{3}$$

where N_{BG} denotes the number of background points in the source point cloud X.

Rigidity loss ($\mathcal{L}_{\text{rigid}}$). The loss $\mathcal{L}_{\text{rigid}}$ computes the average distance between warped source points by the scene flow and the rigid flow for all the foreground points in the K abstractions, which is formulated as

$$\mathcal{L}_{\text{rigid}} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{j=1}^{N_k} \|R_k x_j^k + t_k - (x_j^k + f_j^k)\|_1, \tag{4}$$

where N_k denotes the number of points in the k-th cluster of the source point cloud X.

Chamfer distance loss ($\mathcal{L}_{\text{CD}}$). Following previous works [16, 36], we introduce a two-way Chamfer Distance (CD) across all the foreground points X^f, which is formulated as

$$\mathcal{L}_{\text{CD}} = \sum_{x \in X_{warp}^f} \min_{y \in Y} \|x - y\|_2 + \sum_{y \in Y} \min_{x \in X_{warp}^f} \|x - y\|_2, \tag{5}$$

where $X_{warp}^f = X^f + HF^f$.

The overall loss at each iteration is a weighted sum of the above objectives

$$\mathcal{L}^l = \mathcal{L}_{\text{ego}} + \lambda_1 \mathcal{L}_{\text{rigid}} + \lambda_2 \mathcal{L}_{\text{CD}}, \tag{6}$$

where $\lambda_1 = 1.0$ and $\lambda_2 = 0.5$ in all our experiments.

5. Experiments

5.1. Experimental settings

Datasets and evaluation metrics. The datasets used in our experiments are *SemanticKITTI* [2], *Waymo Open*

Dataset	Method	Sup.	*lidarKITTI* (with ground)				*lidarKITTI* (w/o ground)			
			EPE3D [m] ↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	EPE3D [m] ↓	Acc3DS↑	Acc3DR↑	Outliers3D↓
FT3D	PointPWC-Net [36]	Full	0.710	0.114	0.219	0.932	0.390	0.387	0.550	0.653
	FLOT [26]	Full	0.660	0.046	0.153	0.957	0.323	0.248	0.512	0.684
	FlowStep3D [16]	Full	0.797	0.087	0.184	0.929	0.433	0.343	0.530	0.686
	PV-RAFT [35]	Full	0.658	0.139	0.267	0.895	0.434	0.292	0.516	0.673
SemanticKITTI	Gojcic *et al.* [8]	Weak	0.133	0.460	0.746	0.527	0.150	0.521	0.744	0.450
	Gojcic *et al.*++ [8]	Weak	0.102	0.686	0.819	0.410	0.094	0.784	0.885	0.314
	Ours	Weak	**0.065**	**0.857**	**0.940**	**0.290**	**0.071**	**0.824**	**0.913**	**0.295**
Waymo Open	Gojcic *et al.* [8]	Weak	0.106	0.670	0.799	0.420	0.158	0.455	0.702	0.482
	Gojcic *et al.*++ [8]	Weak	0.111	0.625	0.796	0.433	0.147	0.437	0.691	0.573
	Ours	Weak	**0.077**	**0.812**	**0.906**	**0.333**	**0.075**	**0.819**	**0.911**	**0.307**

Table 1. Quantitative comparison on *lidarKITTI*. Gojcic *et al.*++ utilizes a non-parametric test-time optimization as post-processing [8].

[30], *FlyingThings3D (FT3D)* [22], *lidarKITTI* [23, 24] and *stereoKITTI* [23, 24]. Among these datasets, *semanticKITTI* and *Waymo Open* are large-scale autonomous driving datasets with the ego-motion and binary background mask labels. In our experiment, we train our recurrent neural network on the LiDAR datasets, *SemanticKITTI* [2] and *Waymo Open* [30] with ground points. For inference, we test our trained model on *lidarKITTI* with and without ground points, respectively. Moreover, we utilize the ground truth of the background mask as inputs when training the recurrent network, and employ a pre-trained segmentation network for inference. Details of the datasets are available in the supplement.

Evaluation metrics. We use the same evaluation metrics as [10, 19, 26, 36].

- **EPE3D(m)** average end-point-error over each point.
- **Acc3DS(0.05)** percentage of points whose **EPE3D**$<0.05m$ or relative error $< 5\%$.
- **Acc3DR(0.1)** percentage of points whose **EPE3D**$<0.1m$ or relative error $< 10\%$.
- **Outliers3D** percentage of points whose **EPE3D**$>0.3m$ or relative error $> 10\%$.

Implementation details. Our training follows the mini-batch strategy and the batch size is 6 for all training data. Adam optimizer is utilized with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is initially set to 1×10^{-3} and is later down-scaled by a factor of 0.98 after every epoch till 40 epochs. For the most important hyper-parameter L, *i.e.* number of iterations, we set $L = 4$ for training and $L = 5$ for inference. In addition, the details about the generation of the binary background label for LiDAR point cloud datasets are also available in our supplement. All the models in the experiments are implemented using PyTorch 1.4 and trained with NVIDIA GeForce GTX1080Ti GPUs.

5.2. Evaluation

We quantitatively compare our recurrent neural network with recent state-of-the-art methods, as shown in Table 1. After training on *SemanticKITTI* in a weakly supervised manner, our method predicts the most accurate scene flow on *lidarKITTI*. Specifically, the EPE3D values drop to 0.065m and 0.071m on *lidarKITTI* with and without ground points. Compared with the rigid flow estimation method proposed by Gojcic *et al.* [8] on *lidarKITTI* with ground points, our recurrent neural network reduces EPE3D from 0.102m to 0.065m, a 36.3% drop. In contrast, the state-of-the-art methods, *e.g.*, PointPWC-Net [36], FlowStep3D [16] and PV-RAFT [35], trained on *FT3D* in a fully supervised manner, achieve poor accuracy in scene flow estimation because of the motion occlusions and sparse reflection in raw LiDAR data. Especially, compared with the optimization-based methods, *e.g.*, FlowStep3D [16], on the *lidarKITTI* without ground points, our method trained on *SemanticKITTI* reduces EPE3D from 0.433m to 0.071m. The *lidarKITTI* and *SemanticKITTI* are from the same data source. To investigate the generalization ability of our method, we also train the proposed network on *Waymo Open* and test on *lidarKITTI*. The quantitative results are also shown in Table 1. We observe that the domain gap caused by different sensors leads to a tiny drop in the accuracy of our method's scene flow estimation, *i. e.*, increasing EPE3D from 0.065m to 0.077m.

In Fig. 6, we visualize the scene flow estimation results produced by state-of-the-art methods and our proposed method. It can be observed that our method outputs the best alignments and preserves the structure. Compared with methods utilizing indirect constraints, *e.g.*, PointPWC-Net and FlowStep3D, our method preserves the geometric structure of the warped source scene. To be concrete, at the zooming circles around the ground area, the red and blue points are thoroughly blended for PointPWC-Net and FlowStep3D, which differs from the ground truth. Similar to us, the method proposed by Gojcic *et al.* introduces direct constraints and warps the source scene without structure distortion. However, the misalignment caused by rigidity constraints is still challenging, which can also be observed

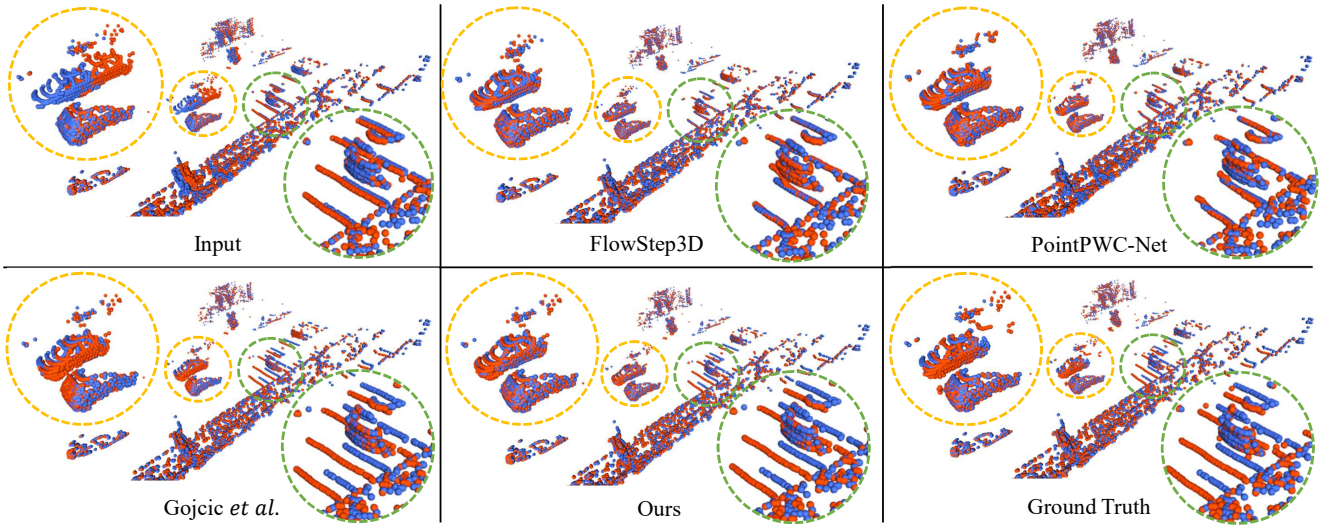

Figure 6. Qualitative comparisons with the state-of-the-art methods. Compared with other methods, our aligning result is more close to the ground truth.

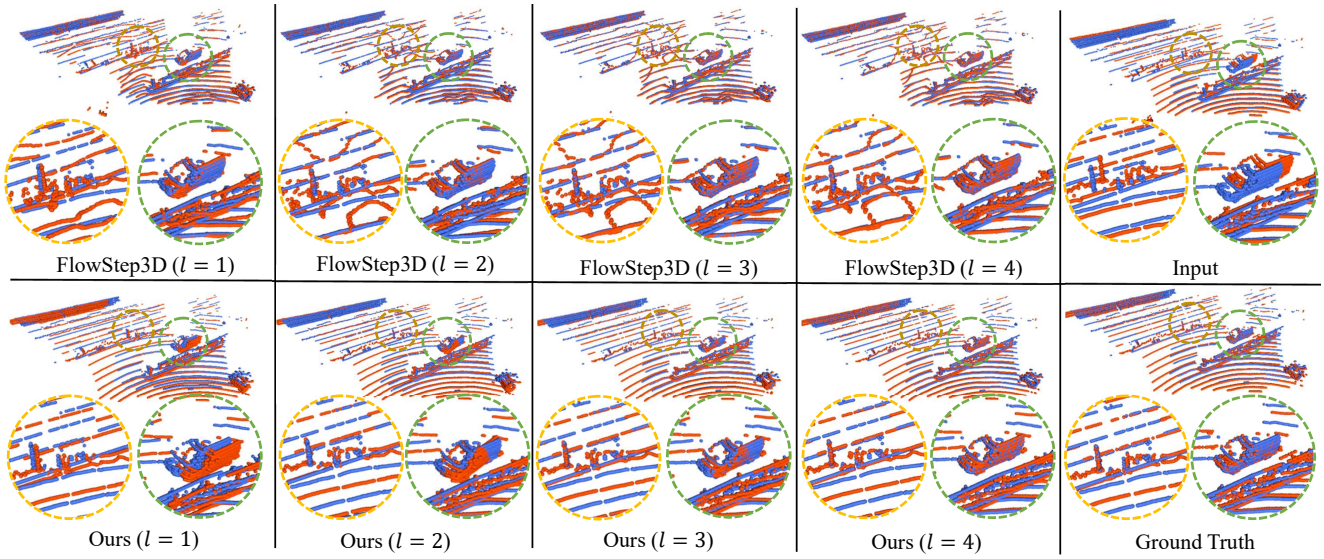

Figure 7. Qualitative comparisons between aligning results of our method and FlowStep3D [16] during iterations. The l represents the iteration index.

in the zooming circles of the two vehicles. In contrast, our method preserves the structure of the warped source scene and achieves outstanding alignment. To compare with other GRU-based optimization methods, we pick a scene from *lidarKITTI* and visualize the alignment results of ours and FlowStep3D at each iteration. It can be observed that there is severe structure distortion in the results generated by FlowStep3D during the iterative optimization, especially around the ground points. As a comparison, our method preserves structure effectively and gradually aligns source and target point clouds during iterations, as shown in Fig. 7.

5.3. Ablation Studies

Number of iterations. To investigate appropriate number of iterations, we perform an ablation experiment to look up the balance between accuracy and iteration count. We test our recurrent network with different iteration numbers on *lidarKITTI* with and without ground points, the results of which are shown in Table 2. From the table, we can see that with increase of iteration number, the performance gets improved in terms of all metrics. It should be noted that in the experiment in Sec. 5, we use 5 iterations for comparison. Table 3 presents the time consumption and parameter amount of state-of-the-art methods, as well as our network with 3 iterations and 5 iterations. All models are run in the inference mode and the time is measured at a machine equipped with NVIDIA GeForce GTX1050Ti GPU and Intel i7-8700 CPU. This table demonstrates that our method achieves excellent performance while keep a compact model size and a relatively small running time.

Dataset	Iter.	EPE3D [m] ↓	Acc3DS↑	Acc3DR↑	Outliers3D↓
	3	0.086	0.670	0.895	0.376
lidarKITTI	5	0.065	0.857	0.940	0.290
(with ground)	7	0.063	0.866	0.943	0.280
	9	**0.062**	**0.869**	**0.945**	**0.279**
	3	0.090	0.621	0.805	0.388
lidarKITTI	5	0.071	0.824	0.913	0.295
(w/o ground)	7	0.069	0.835	0.926	0.288
	9	**0.067**	**0.848**	**0.936**	**0.287**

Table 2. Ablation results of numbers of iterations for our proposed recurrent network in inference. All the models are trained on *SemanticKITTI* with 4 iterations.

Method	Recurrent	EPE3D [m]	Time [ms]	Param. [M]
PointPWC-Net [36]	No	0.710	647	7.72
Gojcic *et al.* [8]	No	0.133	**423**	8.08
Gojcic *et al.*++ [8]	No	0.102	1245	8.08
FlowStep3D [16]	Yes	0.797	2658	0.69
PV-RAFT [35]	Yes	0.658	5296	**0.19**
Ours-Iter. 3	Yes	0.086	538	1.37
Ours-Iter. 5	Yes	**0.065**	750	1.37

Table 3. Time consumption of different method on *lidarKITTI* with ground points in inference. The running time of ours contains the time consumption on segmenting background, about 170 ms.

Direct constraints and error awarded optimization. We conduct experiments to demonstrate the effectiveness of direct multi-body rigidity constraints and learning-based error awarded optimization under two scenarios: updating cost volume or not. We construct many networks, which employ different modules under different scenarios, as variants of our recurrent network. These variants of our recurrent network are trained on *SemanticKITTI* and tested on *lidarKITTI* with ground points. The quantitative results are shown in Table 4. We observe that the rigid flow converter improves the performance by a large margin. Combining these two modules together provides the best performance in terms of all metrics. Moreover, updating of the initial cost volume brings in additional computing burden and impair the time efficiency of optimization methods without performance improvement.

Direct rigidity constraints vs. Indirect rigidity constraints. We compare the performance of networks with two rigidity constraints, direct and indirect, and show the results in Table 5 (2nd and 3rd rows). Indirect rigidity constraints means introducing rigidity constraints as a regularization term in loss function. We conclude that our proposed direct rigidity constraints are better in improving performance of scene flow estimation.

Error awarded optimization vs. non-parametric optimization. Similar to the previous comparison, we compare two optimization strategies and show the results in Table 5 (1st and 3rd rows). In this experiment, the non-parametric optimization, which is from [8], is added as post-processing to our recurrent network in place of error awarded optimization. The quantitative comparison demonstrates the advantages of learning-based error awarded optimization.

Cost Volume Updating	EGRU	RFC	*lidarKITTI* EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers3D↓
No	✓	✓	**0.065**	**0.857**	**0.940**	**0.290**
	✓		0.686	0.105	0.213	0.938
		✓	0.152	0.156	0.536	0.676
Yes	✓	✓	0.069	0.843	0.925	0.301
	✓		0.652	0.008	0.036	0.975
		✓	0.132	0.252	0.660	0.573

Table 4. Ablations of proposed two modules under different updating strategies on cost volume. 'EGRU' and 'RFC' represent the error awarded GRU and the rigid flow converter respectively.

DC	IC	EAO	NPO	*lidarKITTI* EPE3D [m] ↓	Acc3DS↑	Acc3DR↑	Outliers3D↓
✓			✓	0.140	0.252	0.643	0.601
	✓	✓		0.149	0.244	0.603	0.652
✓		✓		**0.071**	**0.824**	**0.913**	**0.295**

Table 5. Direct constraints (DC) vs. indirect constraints (IC) and error awarded optimization (EAO) vs. non-parametric optimization (NPO). The results are achieved on *lidarKITTI* without ground points.

6. Conclusion and Limitations

We propose a weakly supervised optimization based method for LiDAR scene flow estimation. Concretely, we exploit the rigidity constraints by directly converting scene flow to hybrid flow on a GRU-based recurrent neural network, which preserves the object structure during iterations. We also propose an error awarded optimization strategy to refine the predicted scene flow and obtain outstanding alignment. Extensive experiments performed on *lidarKITTI* demonstrate superiority of our method.

The limitations of our work include: (1) The performance of scene flow estimation may be sensitive to the accuracy of background mask. (2) The prerequisite of our work is the rigidity assumption. For scenes with many non-rigid motions, our method may fail.

In addition, due to space limitations, we also present the results of the following experiments in our supplement, *i. e.*, influence of different loss terms and pre-segmentation methods, quantitative comparison on the *stereoKITTI* dataset and time consumption without error awarded optimization.

Acknowledgement

We acknowledge fundings from National Key R&D Program of China under Grant 2020AAA0108600, and National Natural Science Foundation of China under Grants 61901435, 62131003 and 62021001.

References

[1] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13126–13136, 2021. 1, 3

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 5, 6

[3] Rodrigo L Carceroni and Kiriakos N Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *International Journal of Computer Vision*, 49(2):175–214, 2002. 5

[4] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548. IEEE, 2002. 2

[5] Joao Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. 2

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2758–2766, 2015. 2

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996. 3

[8] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5692–5703, 2021. 2, 3, 6, 8

[9] Vladislav Golyanik, Kihwan Kim, Robert Maier, Matthias Nießner, Didier Stricker, and Jan Kautz. Multiframe scene flow with piecewise rigid motion. In *2017 International Conference on 3D Vision (3DV)*, pages 273–281. IEEE, 2017. 2

[10] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019. 1, 2, 6

[11] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7396–7405, 2020. 1

[12] Varun Jampani, Martin Kiefel, and Peter V Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016. 2

[13] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 4

[14] Ken-ichi Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, volume 2, pages 586–591. IEEE, 2001. 2

[15] Martin Kiefel, Varun Jampani, and Peter V Gehler. Permutohedral lattice cnns. *arXiv preprint arXiv:1412.6618*, 2014. 2

[16] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4114–4123, 2021. 1, 2, 3, 5, 6, 7, 8

[17] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 3

[18] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 2

[19] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 1, 2, 4, 6

[20] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *Proceedings of the European Conference on Computer Vision*, pages 468–484, 2018. 3

[21] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019. 3

[22] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 6

[23] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2:427, 2015. 1, 6

[24] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 6

[25] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11177–11185, 2020. 2, 5

[26] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 527–544. Springer, 2020. 1, 2, 6

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1, 2

[28] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 2

[29] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 2

[30] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 6

[31] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*, pages 402–419. Springer, 2020. 3

[32] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021. 1, 3

[33] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1377–1384, 2013. 3

[34] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 91–98, 2020. 1

[35] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6954–6963, 2021. 1, 2, 3, 6, 8

[36] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:1911.12408*, 2019. 1, 2, 5, 6, 8

[37] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2