Exploiting Rigidity Constraints for LiDAR Scene Flow Estimation Supplementary Material

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

This supplementary material provides additional details and results that were omitted from the main paper due to the lack of space. We describe the datasets in Sec. 1 and present the detailed architecture of the feature extractor and initial cost layer in Sec. 2. In addition, we describe the implementation details of background segmentation and foreground clustering in Sec. 3, and visualize more results of our proposed method in Sec. 4. And we provide more quantitative results and comparison with other SOTA methods in Sec. 5 and 6.

1. Datasets

The datasets we used in our experiments are, *FlyingThings3D* [8], *lidarKITTI* [9, 10], *SemanticKITTI* [1] and *Waymo Open* [13]. We use the same setting as [4] for our experiments. For each dataset, we randomly sample 8192 points from source and target point clouds, respectively.

• ***FlyingThings3D (FT3D)***: a large-scale synthetic dataset composed of stereo RGB images, disparity map, and optical flow. The point clouds are obtained by projecting the annotated disparity maps to 3D space using the optical flow and camera parameters. We also adopt the pre-processing settings of FlowNet3D [7] that remove the points with a depth larger than 35 m and the points whose disparity and optical flow are occluded. It is worth mentioning that the source and target point clouds are in one-to-one correspondence, *i.e.* $Y = X + F$, where X and Y are the source and target point cloud, and F is the ground truth scene flow. In our experiments, we only utilize this dataset to train previous methods for comparison. Our proposed network cannot be trained with FT3D, because FT3D doesn't provide the ego-motion annotations.

• ***lidarKITTI***: a real world autonomous driving dataset that contains 142 scenes captured by Velodyne 64-beam LiDAR sensors and the 3D scene flow annotations. The annotations are obtained by projecting the points of the source point cloud to the image plane using the provided calibration parameters, and associating them with the 2D scene flow annotations of the corresponding pixels from the KITTI Scene Flow benchmark [9, 10]. Different from synthetic datasets, the points from the source and target in *lidarKITTI* are not in one-to-one correspondence. Inevitably, in this dataset, there are challenging cases, *i.e.*, sparse reflection and motion occlusions, which are difficult to handle for scene flow estimation methods. In our experiments, we only utilize *lidarKITTI* as the inference dataset.

• ***SemanticKITTI***: a large-scale autonomous driving dataset that provides point-wise semantic annotations and ego-motion transformations for all sequences of the KITTI odometry benchmark [3]. Due to the difference between the fields of view, *lidarKITTI* only includes the points mapped to the image plane of the front camera, while *SemanticKITTI* contains the complete 360° LiDAR scans. Therefore, we take the pre-processing method in [4] to process *SemanticKITTI* point clouds, reducing the domain gap between *lidarKITTI* and *SemanticKITTI*. Additionally, We remove the points whose depth values are less than 1.5m or more than 35m away from the LiDAR sensor. We follow the settings proposed in [4] to merge the semantic labels and generate binary background masks indicating the background and foreground. *SemanticKITTI* contains 12 sequences with semantic mask labels, *i.e.*, sequences '00'-'11'. However, we only utilize 9 sequences as training data, except two sequences '03' and '05'.

• ***Waymo Open***: a large-scale LiDAR dataset collected by driving cars in various conditions. These cars were equipped with five LiDAR sensors that were operating in an acquisition rate of 10 Hz. We only use the points acquired by the top LiDAR sensor in our experiments. *Waymo Open* contains 473 sequences in total, of which 398 sequences are for training and 75 sequences are for validation. In our experiments, we only use the first 199 sequences in the training set to train our recurrent neural network. We transform the obtained point clouds by following the processing steps used in *SemanticKITTI* dataset.

2. Feature Extractor and Initial Cost Layer

We provide the implementation details of our feature extractor and initial cost layer, shown in Fig. 1. All convolutional layers are followed by instance normalization and the ReLU activation function. The feature extractor consists of two SetConv layers, which generate the feature representa-

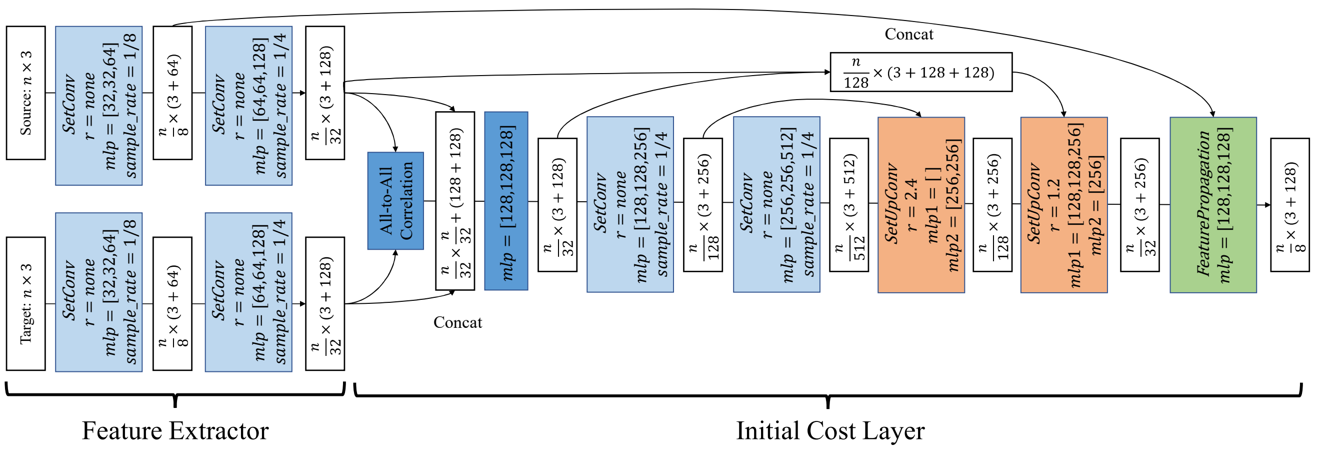

Source: $n \times 3$
SetConv r = none mlp = [32,32,64] sample_rate = 1/8
$\frac{n}{8} \times (3 + 64)$
SetConv r = none mlp = [64,64,128] sample_rate = 1/4
$\frac{n}{32} \times (3 + 128)$
Target: $n \times 3$
SetConv r = none mlp = [32,32,64] sample_rate = 1/8
$\frac{n}{8} \times (3 + 64)$
SetConv r = none mlp = [64,64,128] sample_rate = 1/4
$\frac{n}{32} \times (3 + 128)$
All-to-All Correlation
$\frac{n}{32} \times \frac{n}{32} + (128 + 128)$
Concat
mlp = [128,128,128]
$\frac{n}{32} \times (3 + 128)$
SetConv r = none mlp = [128,128,256] sample_rate = 1/4
$\frac{n}{128} \times (3 + 256)$
SetConv r = none mlp = [256,256,512] sample_rate = 1/4
$\frac{n}{512} \times (3 + 512)$
Concat
$\frac{n}{128} \times (3 + 128 + 128)$
SetUpConv r = 2.4 mlp1 = [ ] mlp2 = [256,256]
$\frac{n}{128} \times (3 + 256)$
SetUpConv r = 1.2 mlp1 = [128,128,256] mlp2 = [256]
$\frac{n}{32} \times (3 + 256)$
FeaturePropagation mlp = [128,128,128]
$\frac{n}{8} \times (3 + 128)$
Feature Extractor
Initial Cost Layer

Figure 1. The architecture of the initial cost layer.

tions of source point cloud X and target point cloud Y with shared weights. To facilitate explaining calculation of the all-to-all correlation, we name the feature extractor as $\Phi(\cdot)$. We use $\Phi(X)$ and $\Phi(Y)$ to calculate a coarse all-to-all correlation filed M. Following FlowStep3D [6], we calculate the cosine similarity between the feature representations of source and target point clouds by

$$\mathrm{sim}(i,j) = \frac{\Phi(x_i)^T \Phi(y_j)}{\|\Phi(x_i)\|_2 \, \|\Phi(y_j)\|_2}. \tag{1}$$

Then we use an exponential function to derive a soft correlation matrix:

$$M_{i,j} = \exp\left(\frac{\mathrm{sim}(i,j) - 1}{\epsilon}\right). \tag{2}$$

Thus, every entry $M_{i,j}$, describes the correlation between $\Phi(x_i)$ and $\Phi(y_j)$. The parameter ϵ is set as 0.03.

3. Abstraction Mask Generation

3.1. Background segmentation label

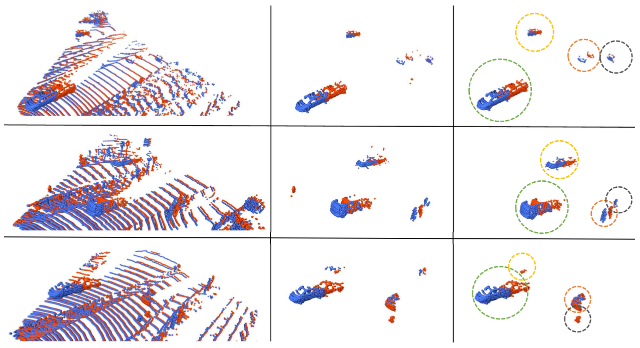
For *SemanticKITTI*, the binary background mask is generated by combining the semantic labels into the background (class labels from 40 to 249) and foreground (other class labels). Along with the ego-motion information, *Waymo Open* also provides the 3D bounding-boxes of vehicles, pedestrians, cyclists, and signs. We use the former three classes to extract the foreground and consider the remaining points as background. So according to the semantics, we consider all vehicles to belong to the foreground, whether parked or moved on the road. Particularly, we only divide the background and foreground according to semantics. All vehicles in a scene are deemed to the foreground, no matter whether they are parked or moving on the road.

3.2. Background segmentation and foreground clustering

In order to split the background and foreground points, we employ a pre-trained segmentation network trained on *SemanticKITTI* to obtain a binary background mask. The segmentation network includes a backbone and a background segmentation head. The backbone network is based on Minkowski engine [2] and follows an encoder-decoder architecture with skip connections. Its input is the source point cloud X and its outputs are per-point latent features. The same backbone with shared weights is also applied to the target point cloud. The background segmentation head includes two sparse convolutional layers, where the first one is followed by the instance normalization and the ReLU activation function [11]. It takes the latent features of the source and target point clouds as input and outputs per-point foreground probabilities.

To obtain the object masks, we utilize a simple DBSCAN clustering algorithm [5] to cluster the foreground points into objects. Specifically, we first use the inferred foreground probabilities to extract the foreground points of the source point cloud. Then we perform the DBSCAN algorithm with a scikit-learn implementation [12], on the foreground points. The *eps* is set as 0.75m and the minimum number of samples in the neighborhood is equal to 5. After clustering, we only retain clusters with at least 10 points.

Fig. 2 shows the visualization results of background removal and clustering on several point clouds from *lidarKITTI*. We visualize the source and target point clouds in Fig. 2(a), points without background points in Fig. 2(b), and the clustering results in Fig. 2(c). The different colored circles represent different clusters of the foreground points of source and target point clouds .



(a) Source and Target (b) Background Removal (c) Clustering

Figure 2. The visualization of background removal and clustering.

4. More Qualitative Results

We provide more visualization results of our method on *lidarKITTI*, *SemanticKITTI*, and *Waymo Open* in Fig. 3-6. In addition, Fig. 5 and Fig. 6 show the visualization results on the test data of *SemanticKITTI* and *Waymo Open*.

5. More Quantitative Results

5.1. Influence of different loss terms.

Compared with the ego-motion that obtained from a gyroscope, the motion of foreground objects is difficult to obtain. We believe that only providing ego-motion is more feasible for the LiDAR scene flow estimation in a weakly supervised manner. We present the ablation results on losses in Table 1.

$\mathcal{L}_{\text{ego}}$	$\mathcal{L}_{\text{CD}}$	$\mathcal{L}_{\text{rigid}}$	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers3D↓
	✓	✓	0.691	0.103	0.275	0.812
✓		✓	0.284	0.119	0.363	0.684
✓	✓		0.127	0.511	0.796	0.334
✓	✓	✓	**0.065**	**0.857**	**0.940**	**0.290**

Table 1. Ablation study of the loss function. All models are trained on *SemanticKITTI* and evaluated on *lidarKITTI* with ground points

5.2. Evaluation on the *stereoKITTI* dataset

Due to lack of the required labels on *FT3D*, we cannot provide the evaluation results on *FT3D*. Here, we provide the results of our method that trained on *SemanticKITTI* and evaluated on *stereoKITTI* as shown in Table 4, from which we can see that our method also achieves outstanding performance.

Dataset	Method	Sup.	EPE3D [m] ↓	Acc3DS↑	Acc3DR↑	Outliers3D↓
FT3D	Point-PWC [36]	Full	0.204	0.292	0.556	0.645
	FLOT [26]	Full	0.122	0.480	0.691	0.401
	FlowStep3D [16]	Full	0.109	0.577	0.765	0.391
	PV-RAFT [35]	Full	0.150	0.553	0.776	0.412
	Gojcic *et al.*(backbone) [8]	Full	0.143	0.392	0.660	0.533
SemanticKITTI	Gojcic *et al.*++ [8]	Weak	0.068	0.836	0.897	**0.263**
	Ours	Weak	**0.053**	**0.858**	**0.917**	0.269

Table 2. Evaluation results on *stereoKITTI* with ground points.

5.3. Influence of error awarded optimization

To demonstrate the effectiveness of the error reward optimization strategy, we provide the memory and time consumption of our method in Table 3.

Method	EPE3D [m]	Time [ms]	Param. [M]
Ours-Iter. 3 w/ EAO	0.086	**538**	1.37
Ours-Iter. 5 w/ EAO	**0.065**	750	1.37
Ours-Iter. 3 w/o EAO	0.131	897	**1.36**
Ours-Iter. 5 w/o EAO	0.167	2679	**1.36**

Table 3. Time consumption of different methods on *lidarKITTI* with ground points in inference. EAO is the error awarded optimization. 'w/o EAO' means the cost volume updates.

5.4. Influence of different pre-segmentation methods

The target scene of our method is autonomous driving, where non-rigid motion, although existing, but occupies a relatively small portion. Since *lidarKITTI* benchmark composes of rigid motions, we cannot quantitatively assess the impact of non-rigid motions on performance. We provide the change of accuracy with the quality of the background in Table 4.

Seg. Method	Prec. FG [m] ↑	Recall. FG [m] ↑	Prec. BG [m] ↑	Recall. BG [m] ↑	EPE3D [m] ↓
GT Seg.	1.000	1.000	1.000	1.000	0.064
Gojcic *et al.* [8]	0.726	0.885	0.991	0.980	0.065
PCT	0.797	0.398	0.955	0.992	0.092

Table 4. Evaluation results on *lidarKITTI* with different segmentation networks. PCT refers to the point cloud transformer.

6. Comparison with other methods

6.1. Differences from RAFT [32]

(1) Raft-3D [32] takes the ordered RGB-D data as inputs, different from our method which processes the unordered LiDAR point cloud directly. **(2)** Raft-3D trains the recurrent network in a fully supervised manner. We are in a weakly supervised manner. **(3)** Raft-3D does not consider the misalignment caused by the introduction of rigid constraints, and our work deploys an error awarded optimization to solve this problem.

6.2. Differences from [8]

Our method differs from [8] in the following three items. **(1)** We expand the framework of [8] by recurrently updating the scene flow prediction. We also employ a lightweight recurrent network to obtain better scene flow estimation without the geometry distortion. **(2)** The recurrent network brings in more time consumption and leads to misalignment as shown in Fig. 1(c) in the main paper. We propose an error awarded optimization method to address this problem and achieve perfect alignment. **(3)** Our method doesn't require a non-parametric optimization algorithm as post-processing, which facilitates end-to-end training of the network and hardware acceleration. Therefore, our method promises better time complexity and smaller parameter amount

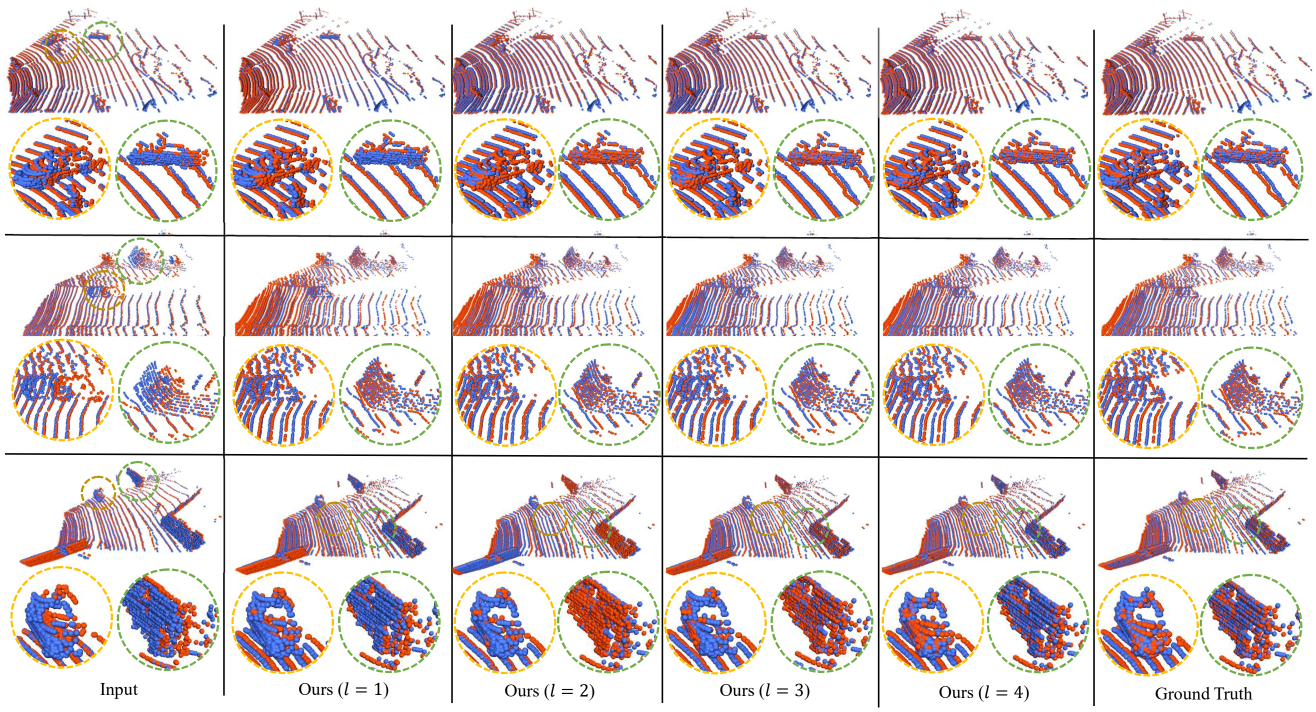


Figure 3. The visualization of our method on *lidarKITTI* with ground points.

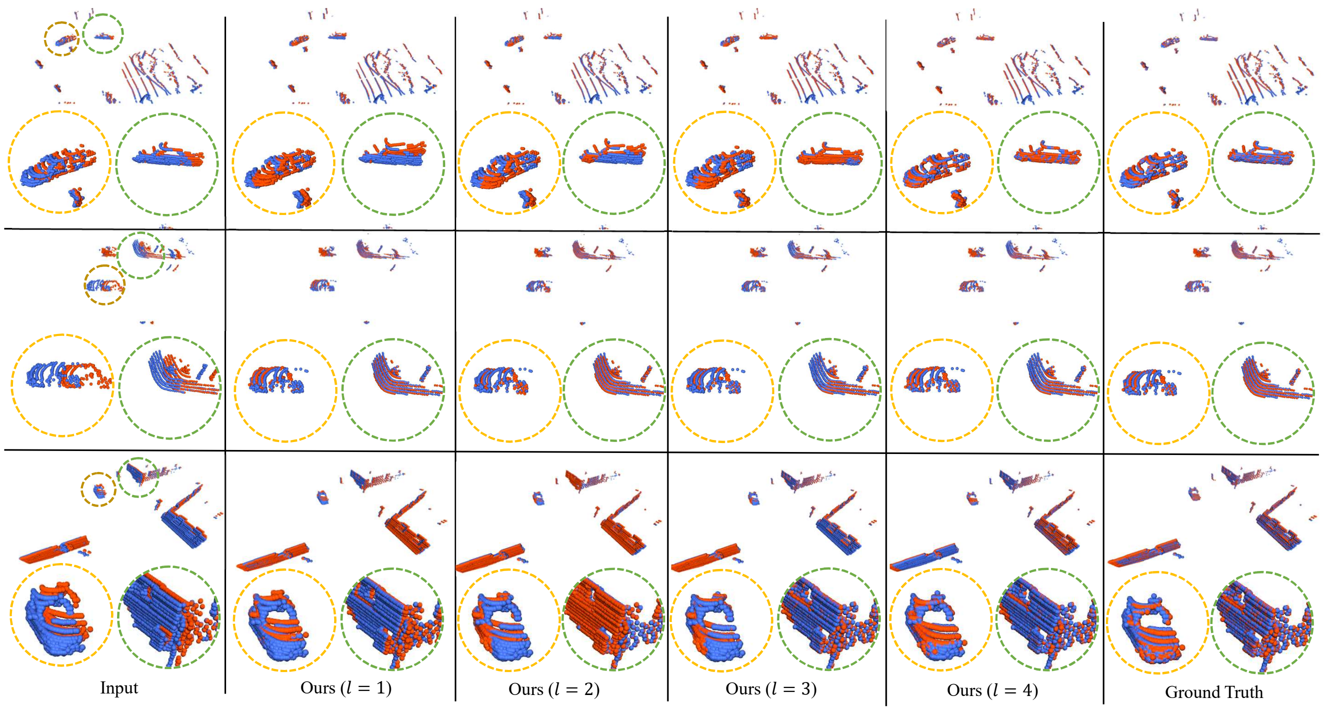


Figure 4. The visualization of our method on *lidarKITTI* without ground points.

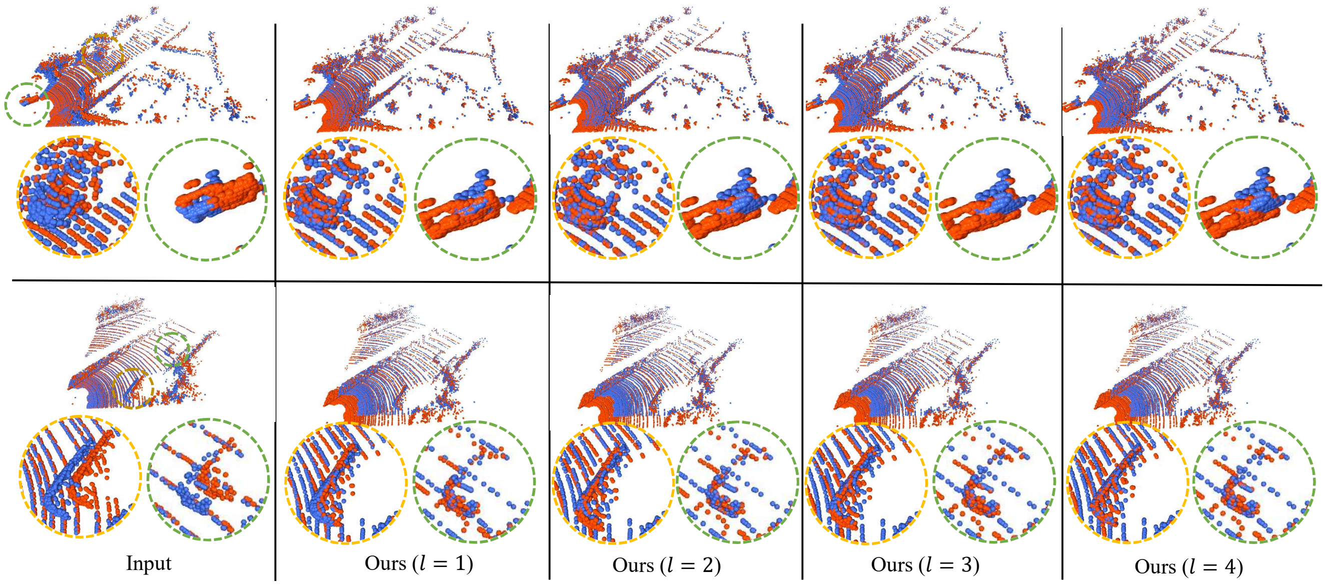


Figure 5. The visualization of our method on *SemanticKITTI* with ground points.

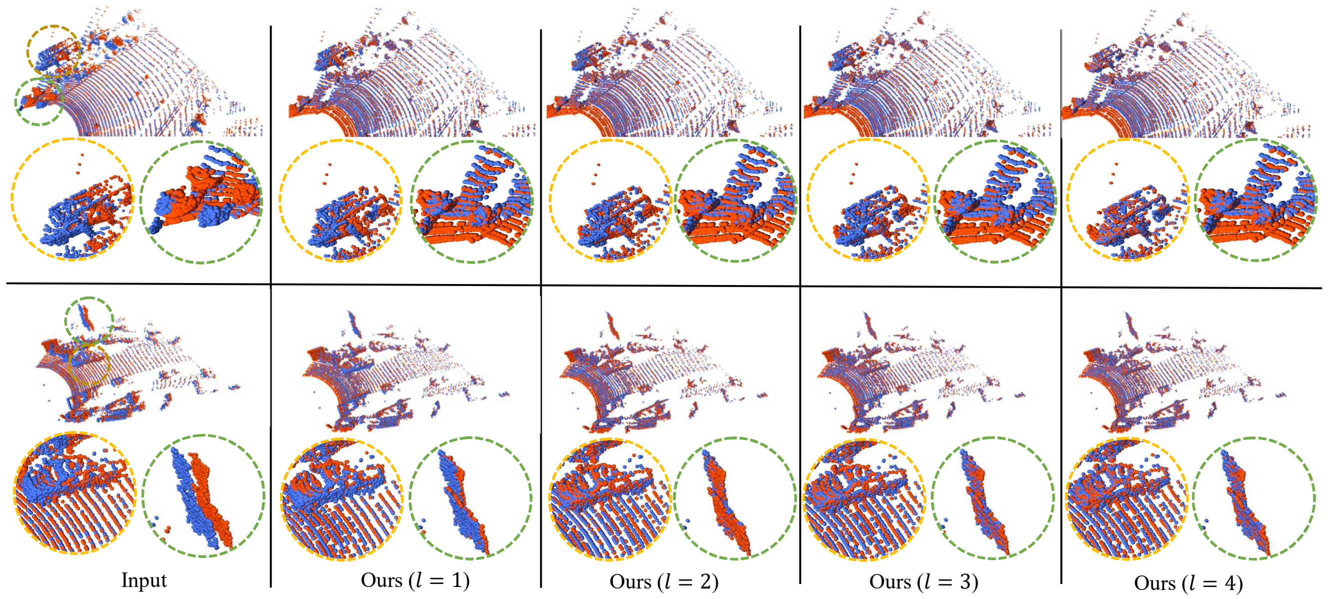


Figure 6. The visualization of our method on *Waymo Open* with ground points.

References

[1] 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. 1

[2] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 1

[4] 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. 1

[5] 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. 2

[6] 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. 2

[7] 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

[8] 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 conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 1

[9] 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

[10] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 1

[11] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 2

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 2

[13] 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. 1