Neural Scene Flow Prior
Supplementary Document

Xueqian Li[*1,2] **Jhony Kaesemodel Pontes**[1] **Simon Lucey**[2]
[1]Argo AI [2]The University of Adelaide

A Method

A.1 Algorithm

We provide a simple pseudo-code to describe our method in Algorithm 1. Please see Eq. 1-4 in the main paper for reference. Note that there are typos in the main paper: in line 135, the output of g is $\mathbf{f}$, $\mathbf{f}^* = g(\mathbf{p}; \mathbf{\Theta}^*)$, and $\mathcal{F}^*$ is the collection of all estimated flows; in line 142, the optimal backward flow should be defined as $\mathbf{f}^*_{bwd} = g\left(\mathbf{p}'; \mathbf{\Theta}^*\right)$, and $\mathcal{F}^*_{bwd}$ is the collection of all estimated backward flows. Note that we use $\mathbf{f}$ to denote the flow of a single point, and $\mathcal{F}$ to denote the collection of the flows, which is the scene flow.

Algorithm 1: Neural scene flow prior

Input : $\mathcal{S}_1$ = point cloud at time t-1, $\mathcal{S}_2$ = point cloud at time t
Output : $\mathcal{F}^*$ = predicted scene flow

begin
  Initialize : Randomly initialize network parameters $\mathbf{\Theta}$, $\mathbf{\Theta}_{bwd}$
  initial loss $\mathcal{L}_0 = \texttt{inf}$
  for $j \leftarrow 1$ **to** `Max_iters` **do**
    foreach $\mathbf{p}$ *in* $\mathcal{S}_1$ **do**
      $\mathbf{f} = g\left(\mathbf{p}; \mathbf{\Theta}\right)$
      $\mathbf{p}' = \mathbf{p} + \mathbf{f}$, $\mathcal{S}'_1 = \{\mathbf{p}'_i\}_{i=1}^{|\mathcal{S}_1|}$
      $\mathbf{f}_{bwd} = g\left(\mathbf{p}'; \mathbf{\Theta}_{bwd}\right)$
    end
    $\mathcal{L}_j = \sum_{\mathbf{p}\in\mathcal{S}_1} \mathcal{D}\left(\mathbf{p} + g\left(\mathbf{p}; \mathbf{\Theta}\right), \mathcal{S}_2\right) + \sum_{\mathbf{p}'\in\mathcal{S}'_1} \mathcal{D}\left(\mathbf{p}' + g\left(\mathbf{p}'; \mathbf{\Theta}_{bwd}\right), \mathcal{S}_1\right)$
    if $\mathcal{L}_j < \mathcal{L}_{j\text{-}1}$ **then**
      $\mathcal{F}^* = \{\mathbf{f}^*\}_{i=1}^{|\mathcal{S}_1|}$
    end
  end
  return $\mathcal{F}^*$
end

A.2 Backward flow

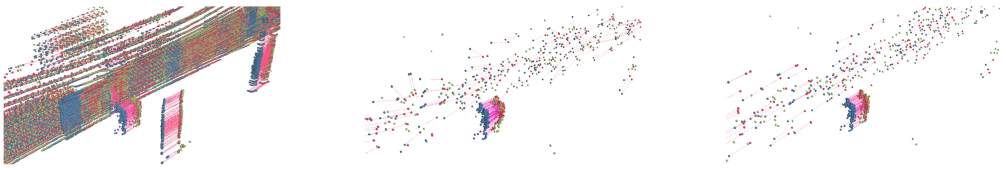
In the main paper, we propose to use the backward flow to ensure the cycle consistency between the two consecutive time frames. We find that the backward flow will further smooth the flow when the point cloud is sparse. When the point cloud becomes denser and has plenty of correspondences available, we might neglect the backward flow. Fig. 1 shows the effect of the backward flow in a zoom-in scene from the Argoverse Scene Flow dataset.

[*]Research done during internship at Argo AI. Corresponding e-mail: xueqian.li@adelaide.edu.au.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Table 1: Comparison of our method with or without backward flow on KITTI Scene Flow dataset.

	$\mathcal{E} \downarrow$ (m)	$Acc_5 \uparrow$ (%)	$Acc_{10} \uparrow$ (%)	$\theta_\epsilon \downarrow$ (rad)
Ours (w/o backward flow)	0.113	60.36	78.85	0.191
Ours (w/ backward flow)	**0.052**	**80.70**	**92.09**	**0.133**



(a) dense points w/o backward flow (b) sparse points w/o backward flow (c) sparse points w/ backward flow

Figure 1: **Backward flow helps regularization.** Here we show a zoom-in sample from the Argoverse Scene Flow dataset. All figures show a subset of the scene. The left figure shows a subset point cloud with all points included in the scene, while the middle and right figures show a subset point cloud with randomly sampled 2k points. When the point cloud is sparse (b,c), there exist scant point correspondences. Adding backward flow further helps regularization and smooths the flow. When the point cloud becomes denser, the backward flow might be unnecessary. The **green** points denote the point cloud at time t ($\mathcal{S}_2$), **blue** points denote the point cloud at time t-1 ($\mathcal{S}_1$). The **red** points represent the point cloud ($\mathcal{S}_1' = \{\mathbf{p}_i'\}_{i=1}^{|\mathcal{S}_1|}$) perturbed by the estimated scene flow $\mathcal{F}^*$ (denoted by **magenta** arrows).

We also performed a small experiment (see Table 1) on the KITTI Scene Flow dataset to show the importance of the backward flow to sparse points (2048 points).

B Experiments

B.1 Dataset

We provide additional preprocessing of the dataset we used in our main experiments.

Argoverse Scene Flow dataset Since there are no official ground truth annotations available in the Argoverse [2] dataset or nuScenes [1] dataset, we followed the preprocessing method described in [8] to create ground truth flows. For each of two consecutive point clouds $\mathcal{S}_1$ and $\mathcal{S}_2$, we first used the object information provided by Argoverse to separate rigid and non-rigid segments. Then we extracted the ground truth translation of rigid parts using the self-centered poses of autonomous vehicles and non-rigid parts using object poses, respectively. Thus we could combine these translational vectors to generate the ground truth scene flow. While given the situation that the object information and provided poses may be inaccurate, the computed scene flow can be imperfect. Moreover, we removed the ground points using the information provided by the ground height map. The same strategies were applied to the nuScenes dataset.

Removal of ground points We removed the ground points for the autonomous driving scenes. The ground is a large piece of flat geometry with little cue to predict motion. Imagine trying to find correspondences in a large flat white wall to compute optical flow. It is intractable without a very large context. We observe the same aperture problem during scene flow estimation. Also, lidar point clouds from driving scenes have a specific ground sampling pattern that resembles an arch of points every other meter. If not removed during scene flow prediction, those points will be snapped/stitched to the closest arch of points and thus biasing too much the end-point-error metric.

Scale of the dataset The four datasets we used contain various number of points. nuScenes [1], KITTI [5, 6], and Argoverse [2] are real-world data that contain large-scale scenes while FlyingThings3D [4] is a synthetic dataset with smaller point clouds. We summarize the statistics of the number of points in each dataset in Table 2. The result showed here and the performance

Table 2: Number of points in each dataset

	FlyingThings3D	nuScenes Scene Flow	KITTI Scene Flow	Argoverse Scene Flow
Average No. of points	6,971	7,069	32,033	57,526
Minimum No. of points	2,188	2,186	14,004	31,665
Maximum No. of points	8,062	15,316	68,084	78,088

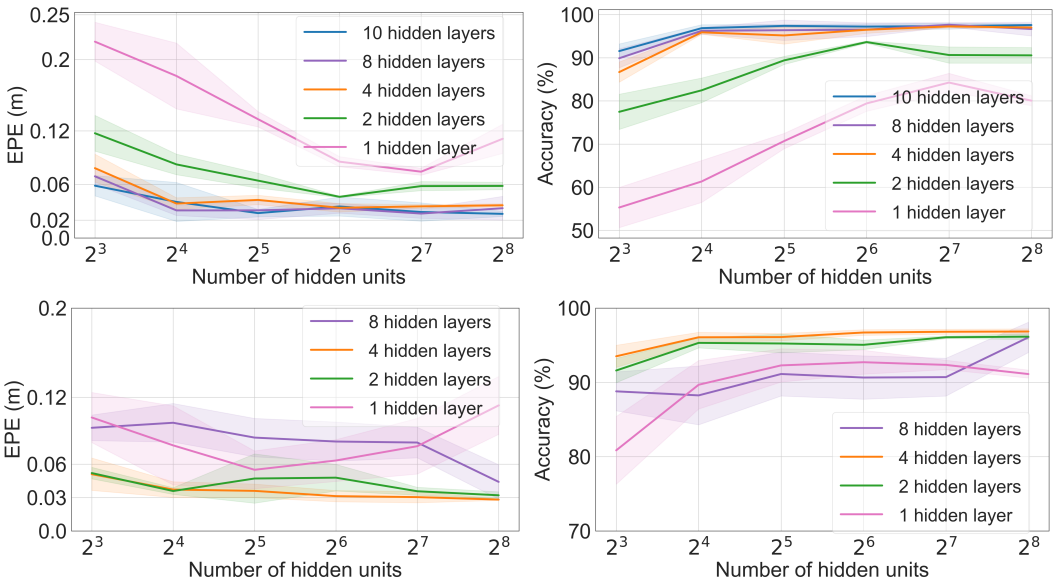


Figure 2: **Performance with different network architectures.** We choose EPE and Accuracy (Acc_{10} metric) to test the effect of using different number of hidden layers, different hidden units size, and different activation functions. The top row shows networks with ReLU activation, the bottom row shows networks with Sigmoid activation function.

we showed in the main paper clearly indicate the capability of our method to deal with large-scale real-world scenes that have a large number of points.

B.2 Network architecture

To further study the effect of using different network architectures, we tested our method on the KITTI Scene Flow dataset (all points included) with different metrics. Results are shown in Fig. 2, which is compatible with what we found in the main paper. Generally, the performance improved with deeper layers and larger hidden units. While we found that using a simple fully connected network with 8 hidden layers, each of which has 128 hidden units and a ReLU activation in the end, has relatively better performance and maintains a relatively small structure.

When the network is shallow, the Sigmoid activation function may have good performance, while with the network architecture going deeper, the Sigmoid function easily fails to maintain high accuracy. Here, we only show results of Sigmoid activation function with at most 8 hidden layers. We found that with 4 hidden layers, the performance was already saturated. With 8 or more hidden layers, the performance drops due to limitations of the Sigmoid functions, such as vanishing gradients. Instead, ReLU activation function is more suitable for deeper networks because it provides a sparse representation, prevents possible vanishing gradients, and leads to efficient computation. We only use ReLU activation function for our main experiments. The Sigmoid function is not needed in the last output layer, since we do not require an extra constraint for the output scene flow.

Note that the input to the network is purely 3D point cloud, we do not map the input to a higher dimensional space using any positional encoding [7] or random Fourier features [9]. We performed an ablation study, comparing experiments with positional encoding to those without, and found that positional encoding was not helpful in our case. The possible explanation lies in the property of the

Table 3: Memory consumption of our neural prior with all points available in the KITTI Scene Flow dataset [5, 6], and variable network architectures. For example, **4 MLP: 64** means network with 4 hidden layers and each layer has 64 hidden units.

	1 MLP: 8	1 MLP: 16	1 MLP: 32	1 MLP: 64	1 MLP: 128	1 MLP: 256
GPU memory (GiB)	0.58	0.58	0.62	0.63	0.82	1.06
	2 MLP: 8	2 MLP: 16	2 MLP: 32	2 MLP: 64	2 MLP: 128	2 MLP: 256
GPU memory (GiB)	0.58	0.60	0.64	0.76	1.03	1.45
	4 MLP: 8	4 MLP: 16	4 MLP: 32	4 MLP: 64	4 MLP: 128	4 MLP: 256
GPU memory (GiB)	0.60	0.64	0.72	0.93	1.42	2.21
	8 MLP: 8	8 MLP: 16	8 MLP: 32	8 MLP: 64	8 MLP: 128	8 MLP: 256
GPU memory (GiB)	0.67	0.71	0.85	1.29	2.21	3.75

specific problem that we want to solve. Since the neural prior acts as a regularizer to smooth the scene flow prediction and the scene flow is rigid within a certain range, we do not expect to learn 3D space representations with high-frequency variations.

B.3 Implementation details

Here we provide more implementation details. In our experiments, we employed a larger learning rate for networks with fewer hidden units and decreased the learning rate for networks with larger hidden units. It is also practical to decrease the learning rate when the network goes deeper. For example, for 1 MLP with 8 layers, lr=0.02 may be deployed; for 8 MLP with 256 layers, lr=0.001 is ideal.

In the main paper Eq. 3, we defined the distance function to be the point distance between a single point to a point set. As we claimed in the main paper, we employed this loss function to both point sets, which is equivalent to the Chamfer distance [3]. In practice, we use the truncated Chamfer loss function to eliminate the extremely large point distance. We found that it is good enough to use an empirical value $2m$ as the maximum tolerance distance. Thus, we forced $\mathcal{D}>2=0$.

B.4 Performance and memory consumption trade-offs

We provide the memory consumption of different network architectures in Table 3. Although deeper and larger network architectures consume more GPU memory, they are still relatively small compared to state-of-the-art learning-based methods which need large number of GPUs to process point clouds with large number of points. Note that the memory consumption is largely affected by the number of points in a point cloud. In this experiment, the average number of points is 32k in the KITTI Scene Flow dataset.

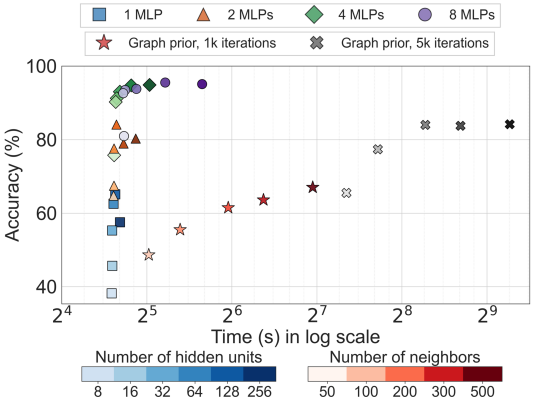


Figure 3: **Accuracy and time comparison.** We show a complete accuracy and time comparison with our neural prior and the graph prior.

We also compared the accuracy and the computation time of our method with different network architectures and graph prior method [8] with different number of neighbors. The result is shown in Fig. 3. For this experiment, we fix the number of iterations for our neural prior method to be 1k, since in most cases, our method will converge in 1k iterations. We allow the graph prior method to optimize for larger iterations (5k) since it cannot converge in 1k iterations. With a deeper network and an increasing number of hidden layers, the

Table 4: Comparison of our method with full-supervised PointPWC-Net on the KITTI dataset preprocessed by the original PointPWC-Net authors. Note that there were 200 scenes in the dataset. Each point cloud only contains points within 35 meters depth.

	$\mathcal{E} \downarrow$ (m)	$Acc_5 \uparrow$ $(\%)$	$Acc_{10} \uparrow$ $(\%)$	$\theta_\epsilon \downarrow$ (rad)
1. PointPWC-Net [10] (reported in their paper, *8,192* points)	0.0694	72.81	88.84	-
2. PointPWC-Net [10] (their pretrained model, *8,192* points)	0.0778	82.24	90.96	0.1127
3. Ours (*8,192* points)	**0.0493**	**90.38**	**95.27**	**0.1099**
4. PointPWC-Net [10] (their pretrained model, *2,048* points)	0.1298	58.14	79.85	0.1562
5. Ours (*2,048* points)	**0.0504**	**85.27**	**94.66**	**0.1209**

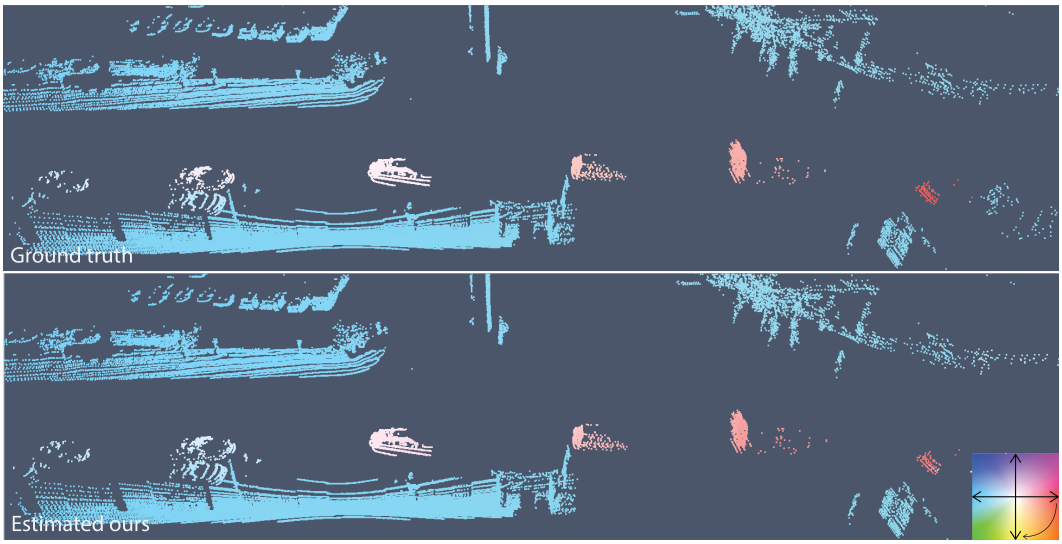


Figure 4: Qualitative example of a scene flow estimation using our proposed method on Argoverse Scene Flow. The scene flow color encodes the magnitude (color intensity) and direction (angle) of the flow vectors.

performance of our neural prior improves with only a little time compromising. The result we found here is compatible with what we found in the main paper—the neural prior with simple MLPs can produce high-fidelity flows, and is capable to deal with large-scale data in a fast manner.

B.5 Performance gap between our reported results and the original PointPWC-Net paper

There was a performance gap between our reported results to the original PointPWC-Net paper [10] in the main paper Table 1 due to two reasons: 1. our data contains point clouds with full range while they cropped point clouds to only include points within 35 meters of depth; 2. we used sparse point cloud that only contains 2,048 points while they used 8,192 points.

We performed a new experiment with the hope that it would help clarify the performance gap. We show the results in Table 4. We first tried to use a pretrained PointPWC-Net model (pretrained on FlyingThings3D) released by the authors to directly test on our KITTI dataset. However, this model is pretrained on a dataset that only has points within 35 meters of depth, when tested on our full-range point cloud data, it completely failed (we did not report the result in the table). To further test the influence of the point density, we then tested this full-supervised PointPWC-Net using the authors' pretrained model (https://github.com/DylanWusee/PointPWC/tree/master/pretrain_weights) and their preprocessed KITTI data with points only within 35 meters of depth (Table 4, line 2, 4). To make a fair comparison, we also compared the results with our method on their preprocessed dataset (Table 4, line 3, 5). Line 1 in Table 4 were shown for a complete comparison. Note that the pretrained model provided by the PointPWC-Net's authors was fine-tuned after the paper submission as mentioned in the official PointPWC-Net GitHub repository (https://github.com/DylanWusee/PointPWC.

These results reveal three facts: 1. the point density does greatly affect the performance of PointPWC-Net; 2. our method, although simple and tested on sparse point clouds, achieves better accuracy; 3. our dataset (following original FlowNet3D, Graph Prior, and other papers) contains large-range raw point clouds that are more challenging than the dataset used in the PointPWC-Net paper.

B.6 Visual results

Figs. 4, 5, 6 show additional visual results for our method on different datasets.

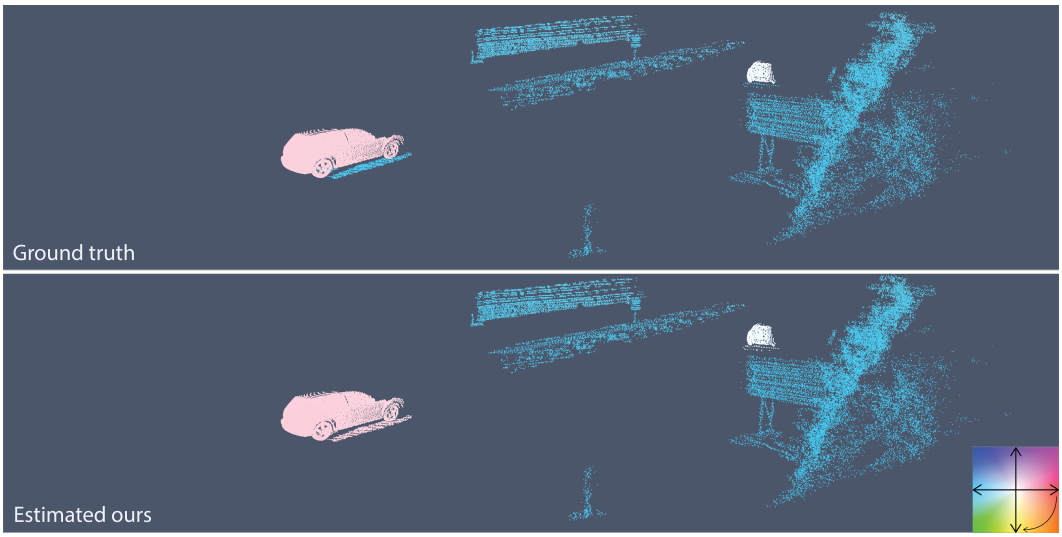


Figure 5: Qualitative example of a scene flow estimation using our method on KITTI Scene Flow.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020. 2

[2] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8748–8757, 2019. 2

[3] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 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 (CVPR)*, pages 605–613, 2017. 4

[4] 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 (CVPR)*, pages 4040–4048, 2016. 2

[5] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015. 2, 4

[6] 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. 2, 4

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 3

[8] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, 2020. 2, 4

[9] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Neural Information Processing Systems (NeurIPS)*, 2020. 3

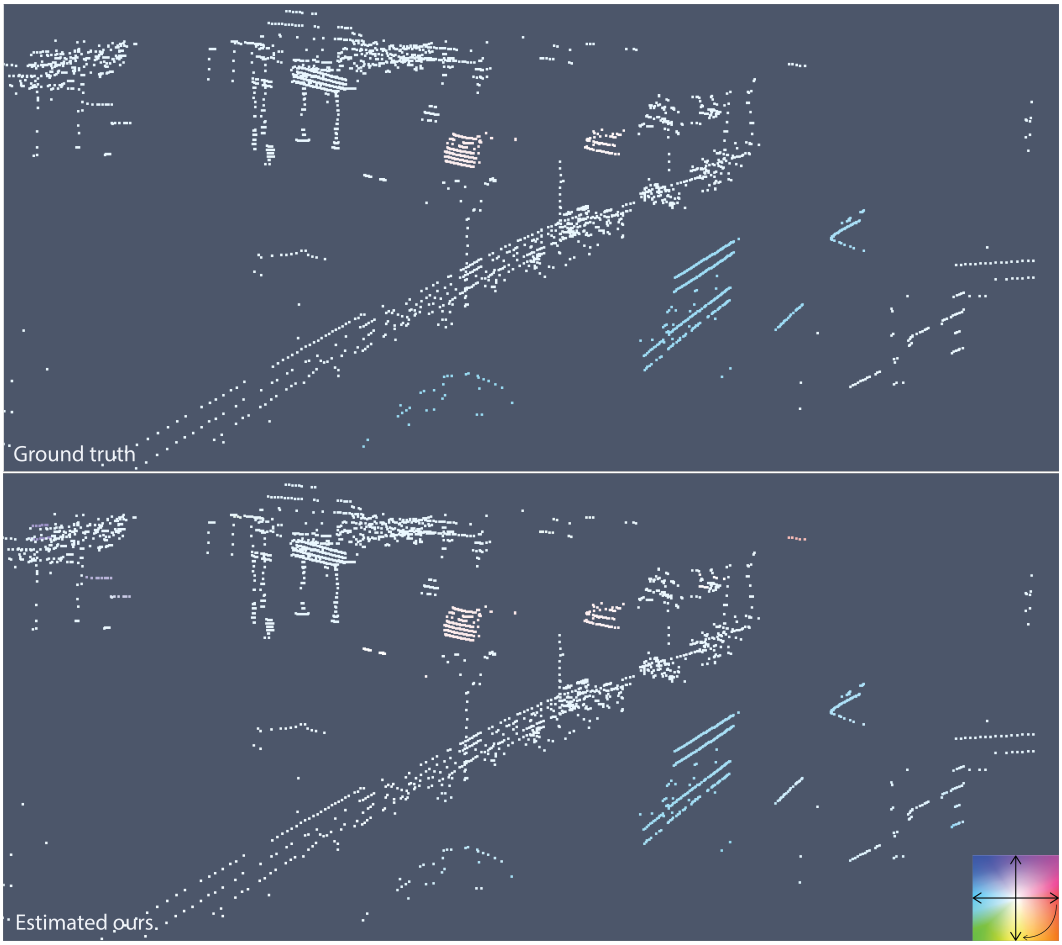


Figure 6: Qualitative example of a scene flow estimation using our proposed method on nuScenes Scene Flow. Note the sparsity of the lidar data.

[10] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 88–107. Springer, 2020. 5