DELFlow: Dense Efficient Learning of Scene Flow for Large-Scale Point Clouds (Supplementary Material)

Chensheng Peng[1], Guangming Wang[1], Xian Wan Lo[1], Xinrui Wu[1], Chenfeng Xu[2], Masayoshi Tomizuka[2], Wei Zhan[2], and Hesheng Wang[1*]
[1]Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education, Shanghai Jiao Tong University
[2] Mechanical Systems Control Laboratory, University of California, Berkeley
{pesiter-swift,wangguangming,kaylex.lo,916806487,wanghesheng}@sjtu.edu.cn
{xuchenfeng,tomizuka,wzhan}@berkeley.edu

1. Overview

We present a comprehensive overview of our research findings in this supplementary material. Sec. 2 includes a detailed description of the network parameters and data augmentation parameters used in our experiments. In Sec. 3, we provide additional results on the dataset with only non-occluded points as input. Finally, in Sec. 4, we include more visualization results and analysis on both successful and failing cases.

2. Implementation Details

2.1. Data Processing

Since the dataset does not provide point clouds directly, we construct the point clouds and ground truth scene flow from the depth map and optical flow. We exclude points with depth exceeding $35m$ in FlyingThings3D dataset [9] and KITTI Scene Flow 2015 dataset [11]. The training dataset is augmented by adding color jitter to RGB images as well as random horizontal flipping and random cropping of the images and projected point clouds. The original image size of the FlyingThings3D dataset is 540×960. To save for computational resources, we down-sample the images and the corresponding point clouds to 270×480 with a stride of 2 and the camera intrinsic changes accordingly. The number of valid points taken as input is 56269 on average, which is much more than 8192 in previous methods. KITTI Scene Flow 2015 dataset contains 200 training samples and 200 testing samples. Due to limited training samples, the model trained on FlyingThings3D are fine-tuned on KITTI dataset. Since KITTI dataset contains images with different size, we pad the images and the projected point clouds to a uniform size of 376×1242.

*Corresponding Authors. The first two authors contributed equally.

2.2. Network Parameters

During the training process on FlyingThings3D dataset, the input shape of the projected point cloud is $270 \times 480 \times 3$. In the subsequent fine-tuning process on KITTI [11] dataset, we set the shape of the input point cloud under 2D dense representation to be $376 \times 1242 \times 3$.

Each layer in the Multi-Layer Perceptron (MLP) includes the ReLU activation function, except for the FC layer. The kernel size is set to be 1×1 with stride 1 for a 2D MLP, so that the scene flow and optical flow can be predicted from each point stored in the pixel. The detailed layer parameters including K values in K Nearest Neighbors (KNN) algorithm, the stride of each down-sampling layer, and the number of feature channels in MLP are described in Tab. 1.

In the kernel-based grouping technique, the center pixels are selected by setting a stride (s_h, s_w), which is similar to the kernel center selecting process of 2D convolution. The centers are selected at a s_h interval along the H-axis and s_w interval along the W-axis.

2.3. Training Process

FlyingThings3D dataset contains 19640 training samples. We first pre-train the model on a quarter of the dataset. The initial learning rate is 0.001, with a decay ratio of 0.8 and a decaying step of 80. Then the model is fine-tuned on the complete FlyingThings3D dataset with an initial learning rate of 0.0001 for 400 epochs. The decay ratio is 0.5 and the decaying step is 50. Finally, the model trained on complete FlyingThings3D dataset are fine-tuned on the KITTI dataset with an initial learning rate of 0.0001. Besides, for model on the KITTI dataset, we increase the size of searching kernel, since the input image is bigger than that of FlyingThings3D dataset.

Module	Layer Type		K / K_s	Stride	MLP width
Point Feature Pyramid	Set conv layer 1		16	2	[3,16,16]
	Set conv layer 2		16	2	[16,16,32]
	Set conv layer 3		16	2	[32,32,64]
	Set conv layer 4		16	2	[64,64,128]
	Set conv layer 5		16	2	[128,128,256]
Image Feature Pyramid	Residual conv block 1		3×3	2	[3,16,16]
	Residual conv block 2		3×3	2	[16,16,32]
	Residual conv block 3		3×3	2	[32,32,64]
	Residual conv block 4		3×3	2	[64,64,128]
	Residual conv block 5		3×3	2	[128,128,256]
Multi-Modal Feature Fusion	Gated fusion block 1		3×3	2	[32,32,32]
	Gated fusion block 2		1×1	2	[64,64,64]
	Gated fusion block 3		1×1	2	[128,128,128]
	Gated fusion block 4		1×1	2	[256,256,256]
Iterative warp-refinement	Flow warp refinement 1	cost volume	16	1	[32,32,32]
		Flow predictor	1 x 1	1	[96, 128, 64]
		FC layer	-	1	[64,3]
	Flow warp refinement 2	cost volume	16	1	[64,64,64]
		Flow predictor	1 x 1	1	[192, 256, 128]
		FC layer	-	1	[128,3]
	Flow warp refinement 3	cost volume	16	1	[128,128,128]
		Flow predictor	1 x 1	1	[384, 512, 256]
		FC layer	-	1	[256,3]
	Flow warp refinement 4	cost volume	16	1	[256,256,256]
		Flow predictor	1 x 1	1	[768, 1024, 512]
		FC layer	-	1	[512,3]

Table 1. **Detailed network parameters.** K points are selected using the K Nearest Neighbors (KNN) algorithm for set conv layer, set upconv layer, and attentive cost volume layer. $K s$ denotes the kernel size of convolutional block. For the set upconv layer, skip connections are used to propagate the sparse features to dense features. MLP width means the number of output channels for each layer of MLP.

2.4. KITTI Refinement

The rigidity assumption is widely used in some scene flow estimation works [14, 15, 10, 8]. Following CamLiFlow [6], we use a semantic segmentation network DDRNet [3] to determine the background and foreground of the scenes in KITTI dataset. Since most objects in the background such as buildings, we use the refinement module in RigidMask [20] to refine the background objects based on the rigid assumption.

2.5. Evaluation Metrics

Let $\hat{f}$ and f_{gt} be the predicted and ground truth flow, respectively. The evaluation metrics for scene flow predication are as follows:

EPE$_{\mathbf{3D}}$(m): the average error $\varepsilon = \|\hat{f} - f_{gt}\|_2$ per point.

$$\mathrm{EPE}_{3\mathrm{D}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{f} - f_{gt}\right\|_2. \tag{1}$$

ACC$_{\mathbf{0.05}}$(%): percentage of $\hat{f}$ whose error < 0.05m or relative error $\frac{\|\hat{f}-f_{gt}\|_2}{\|f_{gt}\|_2} < 5\%$.

ACC$_{\mathbf{0.10}}$(%): percentage of $\hat{f}$ whose error < 0.1m or relative error $< 10\%$.

Outliers3D(%): percentage of $\hat{f}$ whose error > 0.3m or relative error $> 10\%$.

EPE$_{\mathbf{2D}}$(px): 2D equivalent of EPE3D, the average error of optical flow per pixel.

$$\mathrm{EPE}_{2\mathrm{D}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\overline{of_i} - of_i\right\|_2, \tag{2}$$

where of_i stands for the ground truth optical flow and $\overline{of_i}$ stands for the predicted optical flow.

ACC$_{\mathbf{2D}}$(%): percentage of pixels whose error > 3px or relative error $\frac{\left\|\overline{of_i}-of_i\right\|_2}{\|of_i\|_2} > 5\%$.

3. Comparison with SOTA Baselines

There are two different ways to process the FlyingThings3D dataset and KITTI dataset. HPLFlowNet [2], PointPWC-Net [19] , and HALFLOW [16] only keep non-occluded points as input to the network, which simplifies the problem. However, FlowNet3D [7] keeps occluded

Dataset	Method	Data	EPE_{3D} ↓	$ACC_{0.05}$ ↑	$ACC_{0.10}$ ↑	Outliers3D ↓	EPE_{2D} ↓	ACC_{1px} ↑
$FT3D_s$	ICP [1]	No	0.4062	0.1614	0.3038	0.8796	23.2280	0.2913
	FlowNet3D [7]	Quarter	0.1136	0.4125	0.7706	0.6016	5.9740	0.5692
	SPLATFlowNet [13]	Quarter	0.1205	0.4197	0.7180	0.6187	6.9759	0.5512
	HPLFlowNet [2]	Quarter	0.0804	0.6144	0.8555	0.4287	4.6723	0.6764
	HPLFlowNet [2]	Complete	0.0696	-	-	-	-	-
	PointPWC-Net [19]	Complete	0.0588	0.7379	0.9276	0.3424	3.2390	0.7994
	HALFlow [16]	Quarter	0.0511	0.7808	0.9437	0.3093	2.8739	0.8056
	HALFlow [16]	Complete	0.0492	0.7850	0.9468	0.3083	2.7555	0.8111
	FLOT [12]	Complete	0.0520	0.7320	0.9270	0.3570	-	-
	HCRF-Flow [5]	Quarter	0.0488	0.8337	0.9507	0.2614	2.5652	0.8704
	PV-RAFT [18]	Complete	0.0461	0.8169	0.9574	0.2924	-	-
	FlowStep3D [4]	Complete	0.0455	0.8162	0.9614	0.2165	-	-
	Ours	Complete	**0.0388**	**0.8739**	**0.9637**	**0.2062**	**2.2042**	**0.8762**
$KITTI_s$	ICP [1]	No	0.5181	0.0669	0.1667	0.8712	27.6752	0.1056
	FlowNet3D [7]	Quarter	0.1767	0.3738	0.6677	0.5271	7.2141	0.5093
	SPLATFlowNet [13]	Quarter	0.1988	0.2174	0.5391	0.6575	8.2306	0.4189
	HPLFlowNet [2]	Quarter	0.1169	0.4783	0.7776	0.4103	4.8055	0.5938
	HPLFlowNet [2]	Complete	0.1113	-	-	-	-	-
	PointPWC-Net [19]	Complete	0.0694	0.7281	0.8884	0.2648	3.0062	0.7673
	HALFlow [16]	Quarter	0.0692	0.7532	0.8943	0.2529	2.8660	0.7811
	HALFlow [16]	Complete	0.0622	0.7649	0.9026	0.2492	2.5140	0.8128
	FLOT [12]	Complete	0.0560	0.7550	0.9080	0.2420	-	-
	HCRF-Flow [5]	Quarter	0.0531	**0.8631**	**0.9444**	0.1797	2.0700	0.8656
	PV-RAFT [18]	Complete	0.0560	0.8226	0.9372	0.2163	-	-
	FlowStep3D [4]	Complete	0.0546	0.8051	0.9254	**0.1492**	-	-
	Ours	Complete	**0.0501**	0.8259	0.9375	0.2225	**1.8091**	**0.8887**

Table 2. Quantitative results compared with recent methods on FlyingThings3D dataset [9] and KITTI Scene Flow dataset [11]. Only non-occluded points are taken as input. Quarter means training the model on one fourth of the data. Complete means using the complete dataset for training. ↑ denotes better performance with higher value, and ↓ is the opposite.

Evaluation Dataset	Method	Input	Sup.	EPE_{3D} ↓	$ACC_{0.05}$ ↑	$ACC_{0.10}$ ↑	Outliers3D ↓
$FT3D_o$	FlowNet3D [7]	Points	Full	0.151	0.207	0.579	0.789
	FLOT [12]	Points	Full	0.170	0.234	0.528	0.700
	FESTA [17]	Points	Full	0.111	0.431	0.744	-
	Ours	Points	Full	**0.081**	**0.695**	**0.876**	**0.401**
$KITTI_o$	Self-Point-Flow	Points	Self	0.105	0.417	0.725	0.501
	FlowNet3D [7]	Points	Full	0.173	0.276	0.609	0.649
	FLOT [12]	Points	Full	0.110	0.419	0.721	0.486
	FESTA [17]	Points	Full	0.097	0.449	0.833	-
	Ours	Points	Full	**0.083**	**0.789**	**0.870**	**0.381**

Table 3. Quantitative results compared with recent methods on FT3D dataset [9] KITTI dataset [11] where occluded points are kept.

points. FLOT [12] denotes the datasets with only non-occluded points as $FT3D_s$ and $KITTI_s$. The datasets with occluded points are denoted as $FT3D_o$ and $KITTI_o$. In the main manuscript, we provide the experimental results on $FT3D_o$ and $KITTI_o$. To compare our model with more SOTA point-based methods, we provide the experimental results on $FT3D_s$ and $KITTI_s$.

Since these point-based methods are based on only 3D learning methods, we compare our performance using only the 3D LiDAR branch with these point-based methods [7, 5, 18, 4, 19, 13, 16, 12, 17, 2]. As shown in Tab. 2, it can be observed that our model can also report excellent performance on the non-occluded dataset $FT3D_s$ even with more input point.

Our proposed method addresses the scene flow prediction problem in a 3D manner. To verify the effectiveness of our proposed 3D branch, we compare our methods with only 3D branch on $FT3D_o$ and $KITTI_o$. The experimental results in Tab. 3 demonstrate that our method outperforms other point-based methods on the occluded dataset.

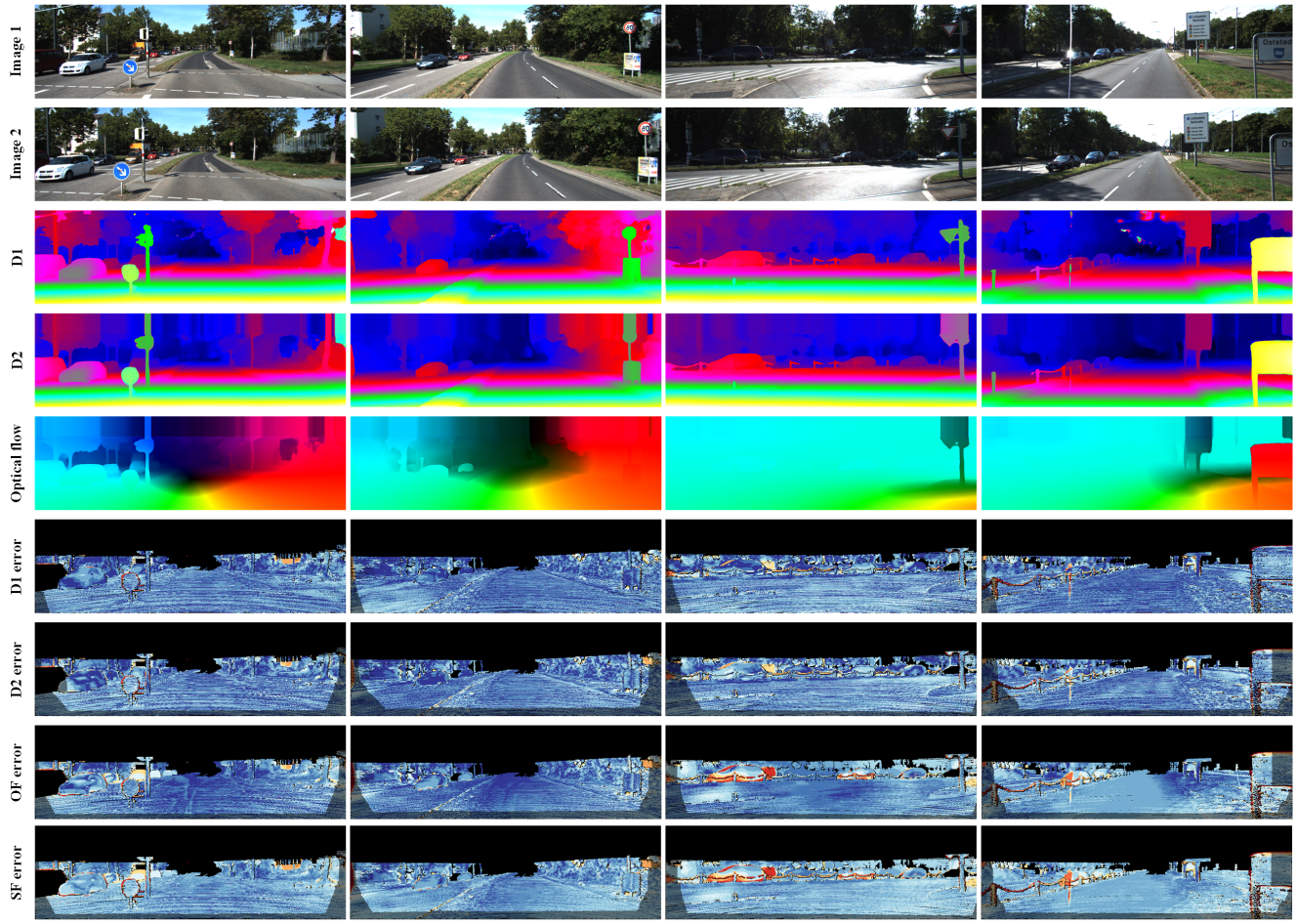

Figure 1. Qualitative visualization results of flow estimation on testing KITTI dataset. Results are obtained from the official KITTI website.

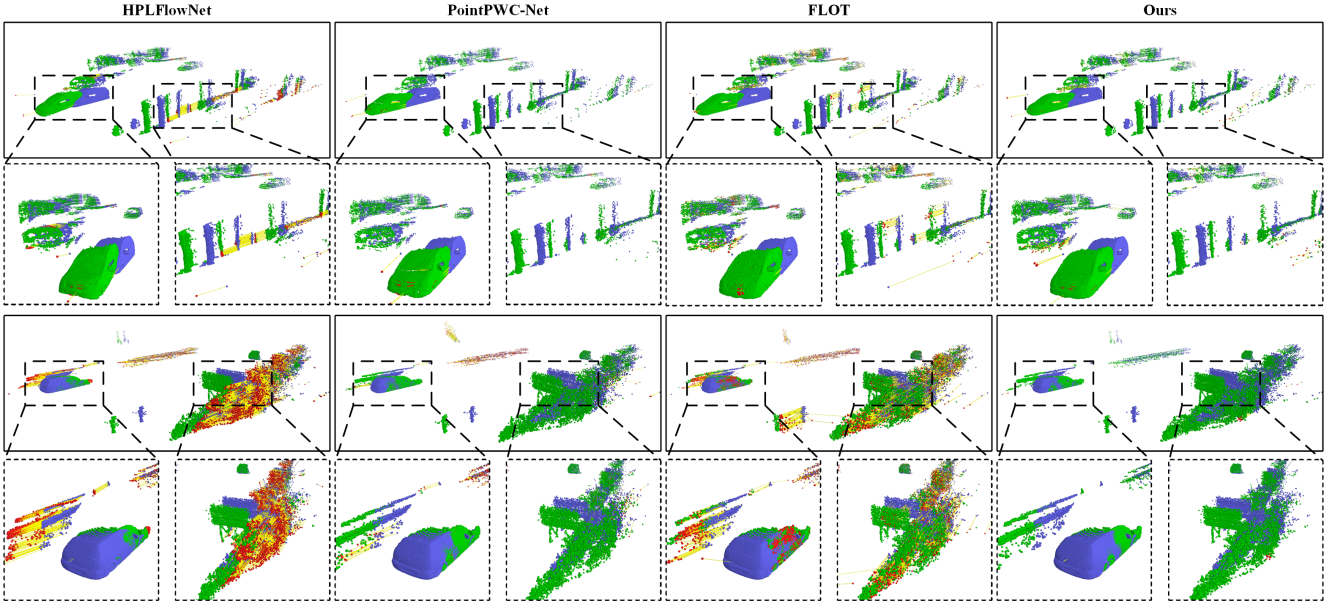

Figure 2. Qualitative visualization results of 3D scene flow estimation on KITTI dataset.

4. Visualization

Fig. 1 shows some successful and failure cases on the KITTI testing dataset. Successful cases are in the first two column and failure cases are in the last two column. We believe that the failures of prediction are mainly due to the point clouds. The point clouds that our network takes as input are pseudo point clouds obtained from depth instead of real points collected from LiDAR. Since the depth are predicted from images, over-exposure and darkness would influence the accuracy of point clouds. Therefore, the scene flow prediction could be incorrect.

In the main manuscript, we illustrate some examples from the FlyingThings3D dataset. Here we provide more 3D visualization examples from the KITTI dataset as shown in Fig. 2. It can be observed that our method achieves the best performance.

References

[1] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. 3

[2] 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. 2, 3

[3] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv preprint arXiv:2101.06085*, 2021. 2

[4] 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. 3

[5] Ruibo Li, Guosheng Lin, Tong He, Fayao Liu, and Chunhua Shen. Hcrf-flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 364–373, 2021. 3

[6] Haisong Liu, Tao Lu, Yihui Xu, Jia Liu, Wenjie Li, and Lijun Chen. Camliflow: bidirectional camera-lidar fusion for joint optical flow and scene flow estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 5791–5801, 2022. 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. 2, 3

[8] 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. 2

[9] 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, 3

[10] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 2

[11] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 1, 3

[12] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *European conference on computer vision*, pages 527–544. Springer, 2020. 3

[13] 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 conference on computer vision and pattern recognition*, pages 2530–2539, 2018. 3

[14] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, 2013. 2

[15] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115(1):1–28, 2015. 2

[16] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Hierarchical attention learning of scene flow in 3d point clouds. *IEEE Transactions on Image Processing*, 30:5168–5181, 2021. 2, 3

[17] Haiyan Wang, Jiahao Pang, Muhammad A Lodhi, Yingli Tian, and Dong Tian. Festa: Flow estimation via spatial-temporal attention for scene point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14173–14182, 2021. 3

[18] 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. 3

[19] 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. 2, 3

[20] Gengshan Yang and Deva Ramanan. Learning to segment rigid motions from two frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1266–1275, 2021. 2