Supplementary of PointConvFormer: Revenge of the Point-based Convolution

Wenxuan Wu[1,2] *, Li Fuxin[1,3], Qi Shan[3]
[1]Oregon State University, [2]CASIA, [3]Apple, Inc.
{wuwen, lif}@oregonstate.edu, {fli26,qshan}@apple.com

1. Network Structure

1.1. Network structure for semantic segmentation

As in Fig. 1, we use a U-Net structure for semantic segmentation tasks where we specify a number of base channels, and then have each downsampling stage to use successively more channels. The U-Net contains 5 resolution levels. For each resolution level, we use grid subsampling to downsample the input point clouds, then followed by several pointconvformer residual blocks. The number of layers in the blocks at each level is $[3, 2, 4, 6, 6]$ respectively for the main model. For the *Lite* model at 10cm voxel size, the number of layers is $[3, 2, 2, 2, 2]$ at each level, respectively. For the *Lite* model at 5cm voxel size, the number of layers at each level is $[3, 3, 3, 3, 3]$, respectively (Table 1). For the 9.4M model at the $2cm$ grid resolution, because it is too fine to be captured by 5 downsampling levels, we utilize a sixth block which contains 2 layers. For the 5.6M parameter model at the $2cm$ resolution, we followed PointTransformerv2 to use a set of resolution levels of $[0.02, 0.06, 0.15, 0.375, 0.9375]$, and also replaced the 3 costly initial layers with a single linear layer, which significantly reduced the size and runtime of the model. This model obtained only 73.0% without mix3D, however, mix3D boosted its performance to 74.4% which is as good as the original model that is significantly larger. For the full model, we have $C_{mid} = 16$, and for the *Lite* model we have $C_{mid} = 4$ which greatly reduced the parameter count with no performance drop at the *Lite* model scale. Latter blocks have more layers since they are cheaper to compute, similar to image convolutional models. For deconvolution, we just use PointConv as described in the main paper. And each block has a single PointConvTranspose layer, which is a PointConv layer that upsamples to locations without any features. For the encoder, we utilize the bottleneck residual architecture described in the paper. For the decoder, because the output dimensionality is always smaller than the input dimensionality, we find that having a bottleneck to $1/4$ of the output dimensionality reduced too much capacity to the model and reduced performance, hence we did not utilize the bottleneck architecture in the decoder and directly performed PointConv on the original input/output dimensionality.

*this work was done entirely at Apple Inc., Wenxuan Wu was an intern at Apple Inc. when he participated in the work

1.2. Network structure for scene flow estimation

Fig. 2 illustrates the network structure we used for scene flow estimation. Following the network structure of PointPWC-Net [10], which is a coarse-to-fine network design, the PCFPWC-Net also contains 5 modules, including the feature pyramid network, cost volume layers, upsampling layers, warping layers, and the scene flow predictors. We replace the PointConv in the Feature pyramid layers with the PointConvFormer and keep the rest of the structure the same as the original version of PointPWC-Net for fair comparison.

2. Evaluation Metrics for Scene flow estimation

Evaluation Metrics. For comparison, we use the same metrics as [10]. Let SF_Θ denote the predicted scene flow, and SF_{GT} be the ground truth scene flow. The evaluate metrics are computed as follows:

- *EPE3D(m)*: $\|SF_\Theta - SF_{GT}\|_2$ averaged over each point in meters.
- *Acc3DS*: the percentage of points with *EPE3D* $< 0.05m$ or relative error $< 5\%$.
- *Acc3DR*: the percentage of points with *EPE3D* $< 0.1m$ or relative error $< 10\%$.
- *Outliers3D*: the percentage of points with *EPE3D*$> 0.3m$ or relative error $> 10\%$.
- *EPE2D(px)*: 2D end point error obtained by projecting point clouds back to the image plane.
- *Acc2D*: the percentage of points whose *EPE2D* $< 3px$ or relative error $< 5\%$.

3. Ablation Studies

In this section, we perform thorough ablation experiments to investigate our proposed PointConvFormer. The ablation studies are conducted on the ScanNet [2] dataset. For efficiency, we downsample the input point clouds with

Input Grid Size	2cm	5cm	10cm
Downsampling levels (cm)	2, 5, 10, 20, 40, 80	5, 10, 20, 40, 80	10, 20, 40, 80, 160
Number of Layers in PointConvFormer	3, 2,4,6,6,2	3, 2,4,6,6	3,2,4,6,6
Number of Layers in PointConvFormer-*Lite*	N/A	3, 3,3,3,3	2,2,2,2,2

Table 1. Downsampling grid levels and number of layers at each level for different PointConvFormer models

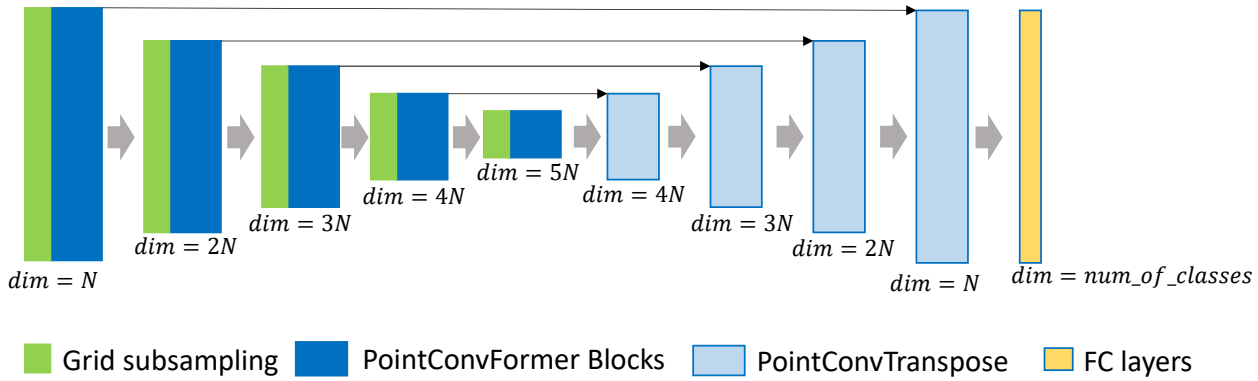

Figure 1. **The network structure of semantic segmentation.** We use a U-Net structure for semantic segmentation tasks. The U-Net contains 5 resolution levels. For each resolution level, we use grid subsampling to downsample the input point clouds, then followed by several pointconvformer residual blocks. For deconvolution, we just use PointConv as described in the main paper. We set $N = 64$ for ScanNet [2] Dataset and $N = 48$ for SemanticKitti [1] Dataset. (Best viewed in color.)

a grid-subsampling method [8] with a grid size of $10cm$ as in [7].

Number of neighbours. We first conduct experiments on the neighbourhood size k in the PointConvFormer for feature aggregation. The results are reported in Table. 2. The best result is achieved with a neighbourhood size of 16. Larger neighbourhood sizes of $32, 48$ do not introduce significant gains on the result, and 48 actually decreased the performance a bit, which may be caused by introducing excessive less relevant features in the neighbourhood [11]. We choose 16 based on similar performance to 32 and significantly smaller memory footprint and faster speed.

Table 2. **Ablation Study.** Number of neighbours in each local neighbourhood.

Nieghbourhood Size	4	8	16	32	48
mIoU(%)	64.61	69.54	**71.40**	71.19	69.84

Table 3. **Ablation Study.** Number of heads.

Number of Head	2	4	8	16
mIoU(%)	70.71	70.58	**71.40**	70.97

Number of heads in ψ. As described in the main paper, our PointConvFormer could employ the multi-head mechanism to further improve the representation capabilities of the model. We conduct ablation experiments on the number of heads in the PointConvFormer. The results are shown in Table. 3. From Table. 3, we find that PointConvFormer achieves the best result with 8 heads.

Decoder c_{mid} PointConv and PointConvFormer implementations lead to a dimensionality expansion of the network that is c_{mid} times the size of the input dimensionality, hence significantly increase the number of parameters in the subsequent linear layer W_l. One empirical contribution in semantic segmentation we made is that we found that the decoder does not really need this dimensionality expansion, which leads to significant savings in the number of parameters. In Table 4, it is shown that adding 3 million parameters by using a c_{mid} of 16 in the decoder only leads to a small improvement of 0.4%, hence in our model we choose to set $c_{mid} = 1$ in the decoder of segmentation models, since those parameters could be used better elsewhere. Parameter savings here and the bottleneck blocks allow us to use more layers yet still have a smaller model than [3].

c_{mid} in decoder	1	3	4	8	16
mIoU (%)	71.4	71.5	70.8	71.5	**71.8**
# Params (M)	**5.48**	5.90	6.11	6.96	8.64

Table 4. Different c_{mid} in the decoder. c_{mid} of 1 in the decoder did not significantly lower the performance, yet saves a significant amount of parameters, hence we choose it in the final model

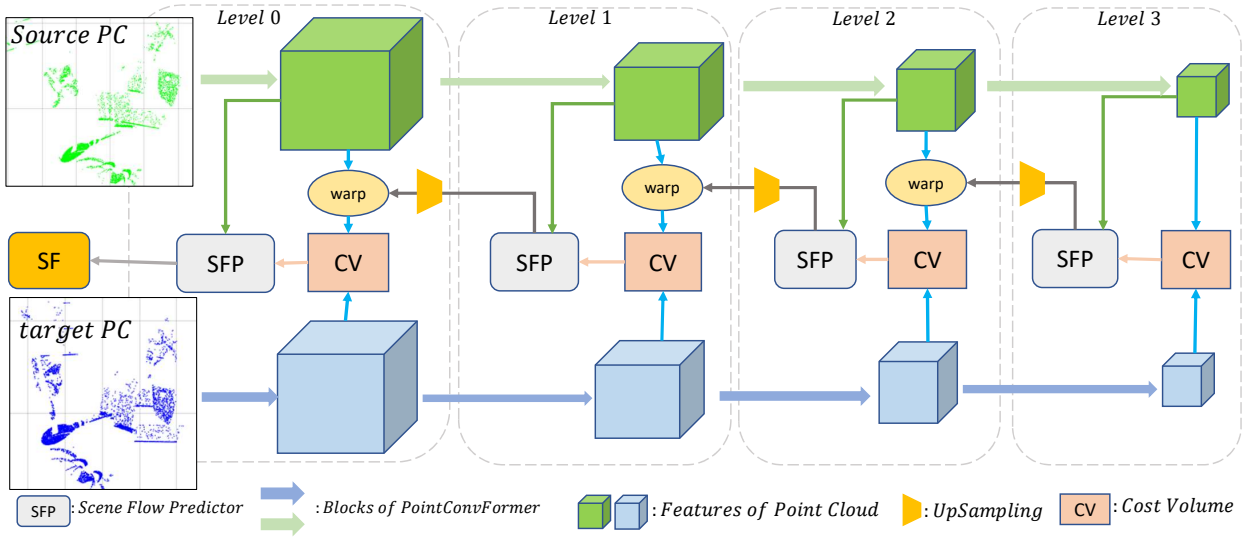

Figure 2. **The network structure of PointPWC-Net with PointConvFormer.** The feature pyramid is built with blocks of PointConvFormers. As a result, there are 4 resolution levels in the PointPWC-Net. At each level, the features of the source point cloud are warped according to the upsampled coarse flow. Then, the cost volume are computed using the warped source features and target features. Finally, the scene flow predictor predicts finer flow at the current level using a PointConv with features from the first point cloud, the cost volume, and the upsampled flow. (Best viewed in color.)

4. More Result Visualizations

Fig. 3 is the visualizations of the comparison among PointConv [9], Point Transformer [11] and PointConvFormer on the ScanNet dataset [2]. We observe that PointConvFormer is able to achieve better predictions with fine details comparing with PointConv [9] and Point Transformer [11]. Interestingly, it seems that PointConvFormer is usually able to find the better prediction out of PointConv [9] and Point Transformer [11], showing that its novel design brings the best out of both operations.

Fig. 4 illustrates the prediction of PointConvFormer on the SemanticKitti dataset [1]. Fig. 5, and Fig.6 are the comparison between the prediction of PointPWC [10] and PCFPWC-Net on the FlyingThings3D [4] and the KITTI Scene Flow 2015 dataset [6]. Please also refer to the video for better visualization.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 2, 3

[2] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1, 2, 3

[3] Xingyi Li, Wenxuan Wu, Xiaoli Z Fern, and Li Fuxin. The devils in the point clouds: Studying the robustness of point cloud convolutions. *arXiv preprint arXiv:2101.07832*, 2021. 2

[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*, pages 4040–4048, 2016. 3

[5] 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. 6

[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. 3, 6

[7] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. *arXiv preprint arXiv:2112.04702*, 2021. 2

[8] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 2

[9] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings*

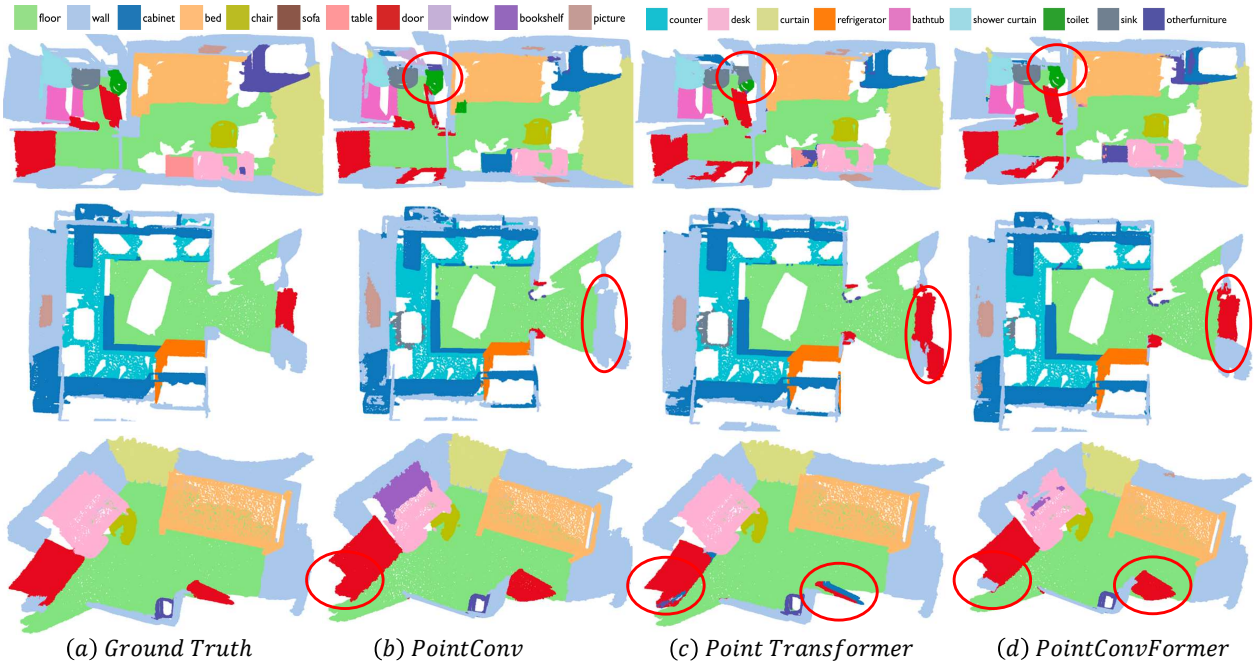

Figure 3. **ScanNet result visualization.** We visualize the ScanNet prediction results from our PointConvFormer, PointConv [9] and Point Transformer [11]. The **red** ellipses indicates the improvements of our PointConvFormer over other approaches. Points with ignore labels are filtered for a better visualization. (Best viewed in color)

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9621–9630, 2019. 3, 4

[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 *European Conference on Computer Vision*, pages 88–107. Springer, 2020. 1, 3

[11] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 2, 3, 4

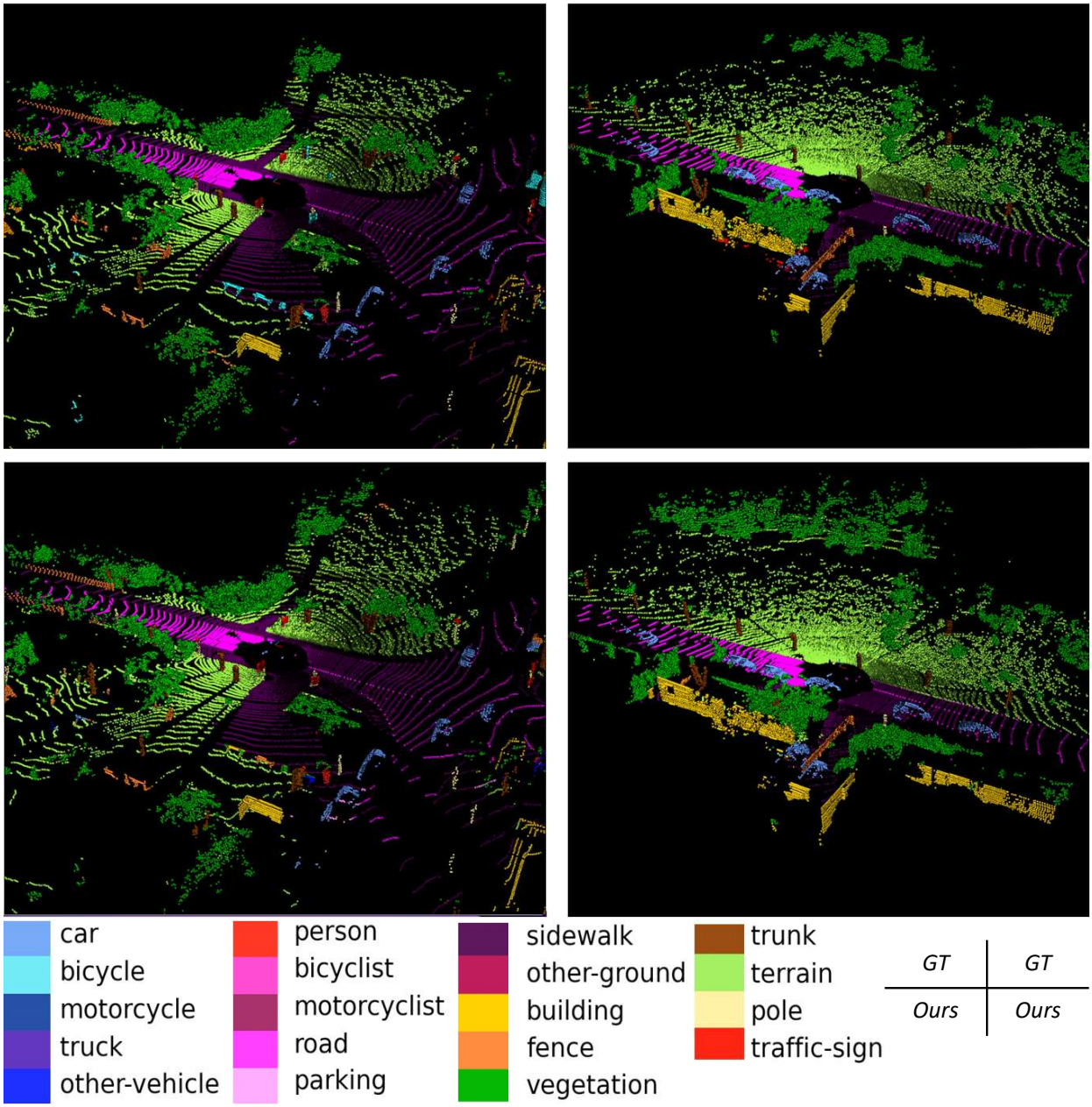

Figure 4. **SemanticKitti result visualization.** We visualize the SemanticKitti prediction results from our PointConvFormer. Each column is a scan from SemanticKitti validation set. The first row is the input, the second row is the ground truth, the third row is our prediction. (Best viewed in color.)

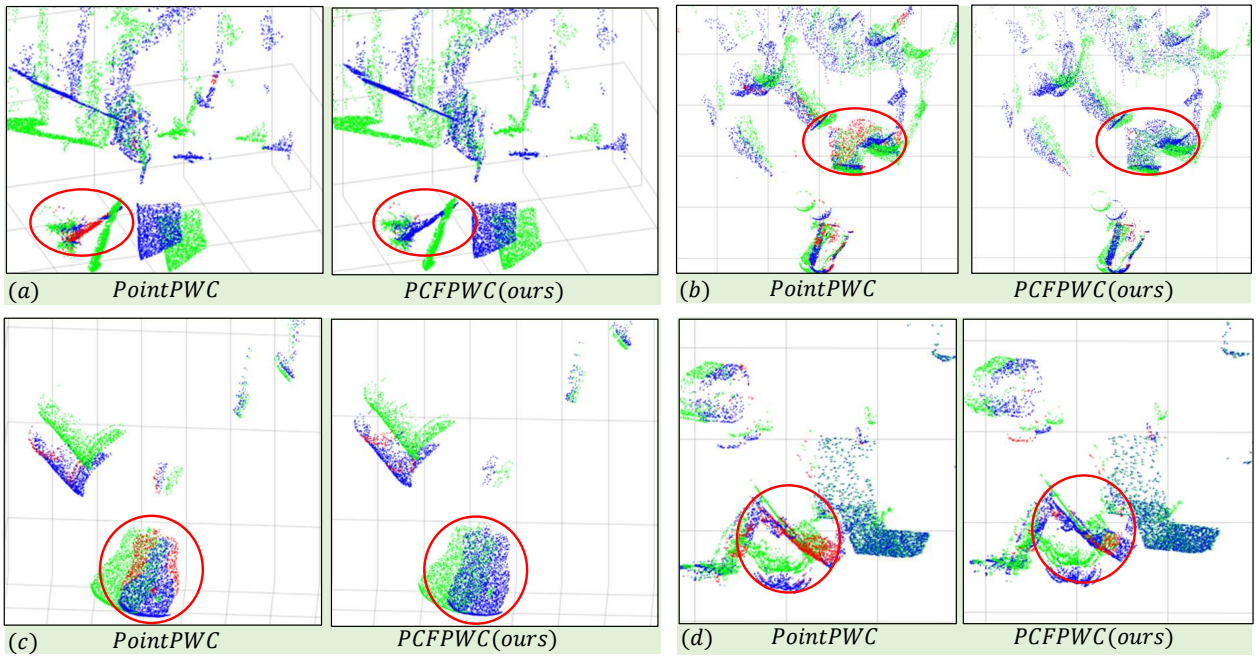

Figure 5. **Qualitative comparison between PointPWC-Net and PCFPWC-Net (FlyingThings3D [5]).** (a) is the visualization of the FlyingThings3D dataset. (b) is the visualization of the KITTI dataset. Green points are the source point cloud. Blue points are the points warped by the correctly predicted scene flow. The predicted scene flow belonging to Acc3DR is regarded as a correct prediction. For the points with incorrect predictions, we use the ground truth scene flow to warp them and the warped results are shown as red points. (Best viewed in color.)

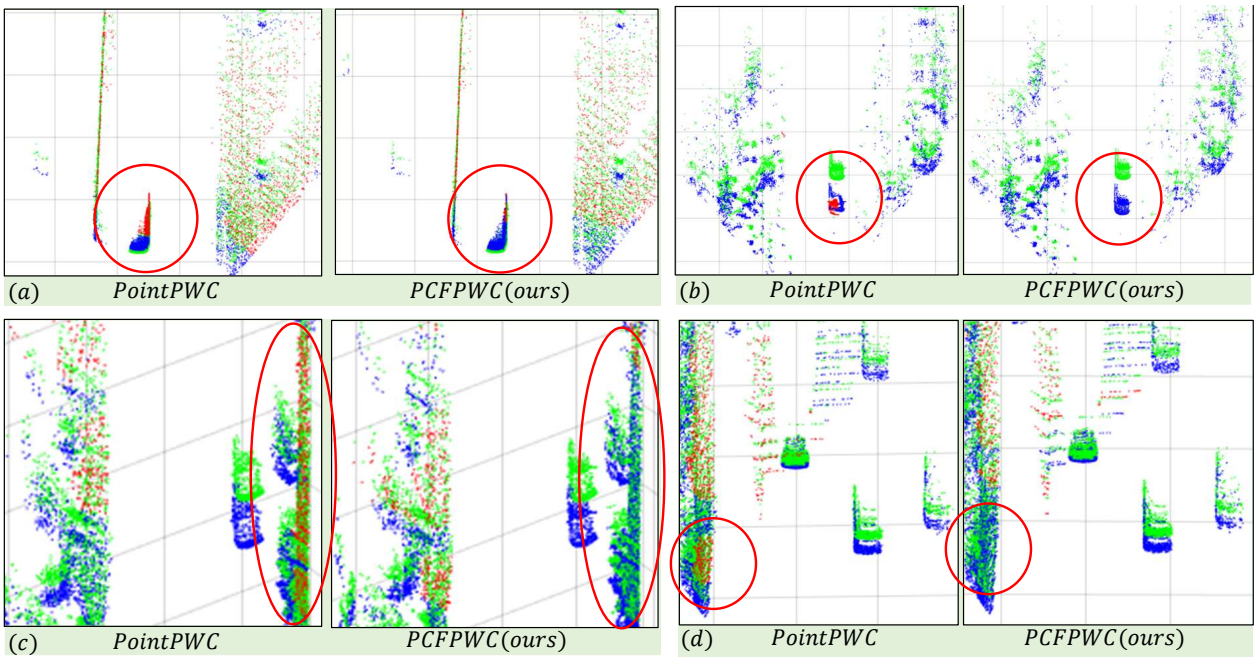

Figure 6. **Qualitative comparison between PointPWC-Net and PCFPWC-Net (KITTI [6]).** Green points are the source point cloud. Blue points are the points warped by the correctly predicted scene flow. The predicted scene flow belonging to Acc3DR is regarded as a correct prediction. For the points with incorrect predictions, we use the ground truth scene flow to warp them and the warped results are shown as red points. (d) is a failure case, where the points on the wall or ground/road are hard to find accurate correspondences for both PointPWC and PCFPWC. (Best viewed in color.)