Supplementary Material: HPLFlowNet: *H*ierarchical *P*ermutohedral *L*attice FlowNet for Scene Flow Estimation on Large-scale Point Clouds

In the supplementary material, we provide more implementation details in Sec. A. In Sec. B, we conduct a series of additional experiments, including more detailed analysis on single layer empirical efficiency, test on KITTI [7, 6] without depth thresholding to show if our model is translational invariant, test on all the points of KITTI without point subsampling to further demonstrate its capacity, *etc*. We also provide more qualitative results in Sec. C and conduct qualitative error analysis in Sec. D.

A. Implementation details

We apply the same data augmentation to both point clouds: random rotation within range $[-10°, 10°]$, random shift in range $[0, 1]$ and random scale in range $[0.95, 1.05]$ in each dimension; we then do an additional random shift in range $[0, 0.3]$ for PC_2 only.

We use Adam [4] for optimization with initial learning rate 0.0001, the default momentum setting, and batch size 1. We train our model with the initial learning rate for 85 epochs, and then multiply the learning rate by 0.7 every 35 epochs, for a total of 500 epochs.

Fig. 1 shows the more detailed HPLFlowNet architecture. Fig. 2 shows the detailed architecture of SPLATFlowNet and Ours-shallow we use in the main paper.

B. Additional experiments

Detailed single layer empirical efficiency. We measure runtime for each BCL variant in our architecture averaged on FlyingThings3D. We then replace them with original BCLs and do the same. Table 1 shows more detailed results. The fact that ours is more efficient is consistent across all layers. The upper layers with coarser lattice points have more advantage, since the numbers of non-empty lattice points decrease and our savings on splatting and slicing become more obvious (We not only reduce one step, but also do splatting and slicing on fewer points).

Table 1: Detailed single layer efficiency comparison with original BCLs on runtime (ms). The fact that ours is more efficient is consistent across all layers and the upper layers with coarser lattice points have more advantage.

Layer	Ours	Original BCL	Ratio
DownBCL1	2.65	2.91	91.1%
DownBCL2	1.81	2.26	80.1%
DownBCL3	1.36	2.05	66.3%
DownBCL4	1.05	2	52.5%
DownBCL5	0.96	1.99	48.2%
DownBCL6	0.9	2.01	44.8%
DownBCL7	0.87	2.04	42.6%
UpBCL1	18.11	21.09	85.9%
UpBCL2	5.93	7.34	80.8%
UpBCL3	2.55	4.4	58.0%
UpBCL4	1.02	3.3	30.9%
UpBCL5	0.64	2.57	24.9%
UpBCL6	0.65	2.61	24.9%
UpBCL7	0.67	2.45	27.3%
CorrBCL1	5.79	5.97	97.0%
CorrBCL2	3.43	4.33	79.2%
CorrBCL3	1.95	3.68	53.0%
CorrBCL4	1.49	3.6	41.4%
CorrBCL5	1.39	3.7	37.6%

Table 2: Evaluation results on KITTI without depth thresholding, $n = 8,192$ (n denotes the number of sampled points per frame), which shows that to a certain extent our model is translational invariant. Comparison with the No $Rel.\ Pos.$ variant shows that our translational invariance comes from taking relative positions as inputs.

Method	EPE3D	EPE2D
FlowNet3 [3]	1.4695	4.4613
ICP [2]	0.4615	22.3953
FlowNet3D [5]	0.2430	6.6149
No $Rel.\ Pos.$	1.5768	14.0311
Ours	**0.1365**	**4.3963**

Results on KITTI Scene Flow 2015 without depth thresholding, translational invariance test. Though trained with a depth threshold of 35 meters following [5], we conduct experiments on KITTI Scene Flow 2015 [7, 6] without this depth threshold and without finetuning, to see how methods generalize to input points with unseen coordinates, and to test whether they are translational invariant. We also evaluate on No $Rel.\ Pos.$ from the ablation studies, whose input signals are purely the global coordinates of the

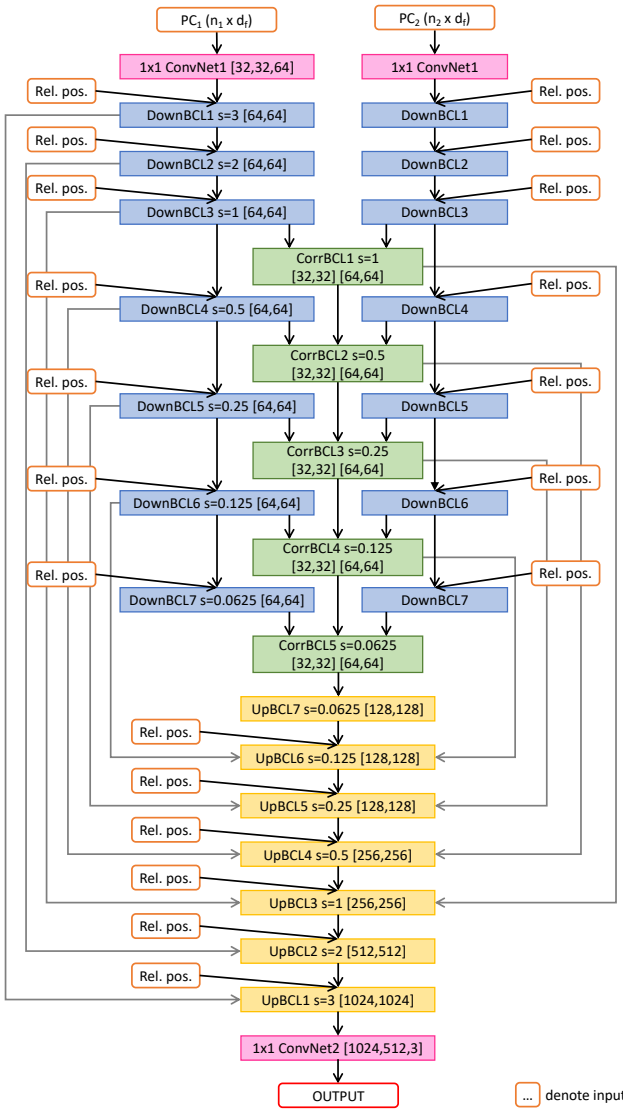


Figure 1: HPLFlowNet architecture with more details. The layers with the same names share weights. s denotes the scaling factor. $Rel.\ pos.$ stands for the relative positions (point's scaled lattice coordinates minus its nearest 0-remainder lattice point's coordinates. For explanation of "0-remainder", please refer to [1]). The numbers in the square brackets are the numbers of output channels for each convolution layer, with LeakyReLU in between; for CorrBCL, the first square bracket is for Patch correlation and the second is for Displacement filtering. For all brackets, the first convolution layer uses kernel size $(neighborhood\_size, 1)$ and the remaining convolution layers use kernel size $(1, 1)$, which is another place we adopt factorization for convolutional kernels.

input points, while our HPLFlowNet takes relative positions as input signals as well as the global coordinates.

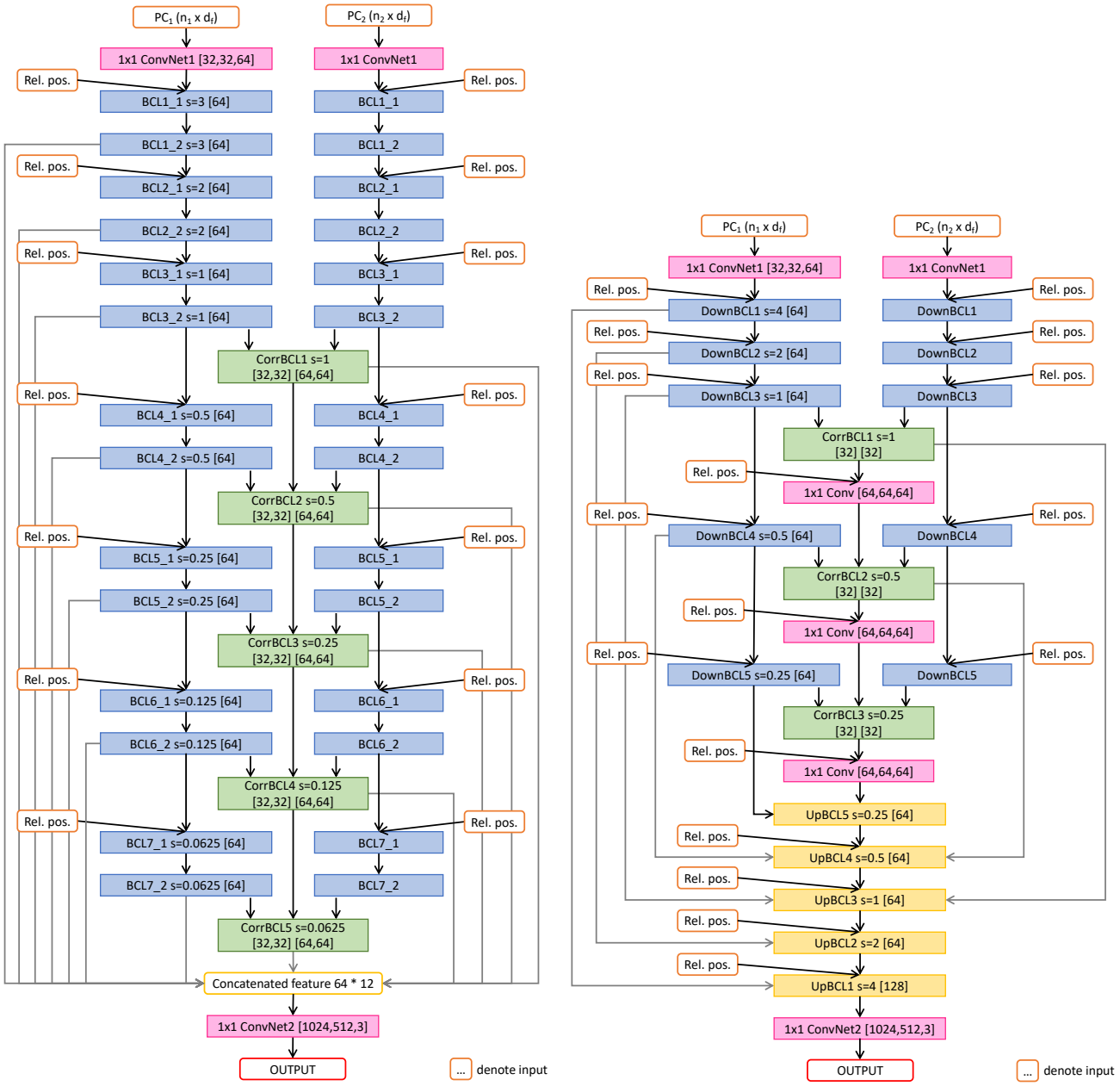
The results are shown in Table 2. Our architecture outperforms all the other methods on both metrics. The performance gap between Ours and No $Rel.\ Pos.$ under the 35m-depth-threshold setting is 0.0804 *vs.* 0.0989, while the gap increases significantly under the no-depth-threshold setting to 0.1365 *vs.* 1.5768. Recall that the only difference between Ours and No $Rel.\ Pos.$ is whether to concatenate input signals with relative positions at each DownBCL and UpBCL, so this results prove that by adopting such design, our model attains certain extent of translational invariance. Since FlowNet3D only uses relative positions as input signals, its performance does not decrease greatly. The performance of FlowNet3 on 3D metrics decreases. One reason is that under the 35m-depth-threshold setting, we remove the points with predicted depth $> 35m$ for FlowNet3 to avoid extreme errors, we cannot use this scheme under the no-depth-threshold setting (we only do the removing extreme operation for FlowNet3). Since more background points are included, and optical flows for faraway background pixels are usually small and easy to predict, it has better EPE2D.

Table 3: Evaluation results measured on all the points of KITTI without point subsampling and without depth thresholding. The robust performance of our methods under this setting further demonstrates its capacity on large-scale point clouds and generalization ability.

Method	EPE3D	EPE2D
FlowNet3 [3]	1.4787	4.4571
ICP [2]	0.4791	23.1040
Ours	**0.1209**	**4.1101**

Results on all the points of KITTI, without point subsampling and depth thresholding. By using our newly proposed DownBCLs, UpBCLs and CorrBCLs, our method is able to process a pair of complete KITTI frames in one pass without subsampling points. Since most baselines do not have this property, we only report the performance of ICP [2], FlowNet3 [3], and our method under this setting in Table 3. The good performance of our method further demonstrates its capacity on large-scale point clouds, which is due to our hierarchical architecture on permutohedral lattices of different scales. The good performance also shows our generalization ability and a certain degree of translational invariance again.

Results on KITTI 200. Following [5], we evaluated on all the 142 scenes in KITTI Scene Flow 2015 with publicly available raw 3D point clouds. The reason that some raw data are not included is "e.g., due to incomplete or noisy tracklet annotations" according to KITTI. Nevertheless, we present the evaluation results on all the 200 scenes in Table 4 for reference purposes. From the table, we see that all the methods have decreased performance, which shows that the scenes without publicly available raw data are more



(a) SPLATFlowNet (b) Ours-shallow

Figure 2: Detailed architectures of SPLATFlowNet we use for comparison and Ours-shallow, a shallower and faster version of ours we use to show the efficiency of our design. Though much shallower, Ours-shallow still outperforms SPLATFlowNet. Notations are the same as Fig 1.

noisy. Ours still outperforms all the other methods on all the metrics.

C. Additional qualitative results

We show additional qualitative results on the FlyingThings3D and KITTI datasets in Fig. 3 and Fig. 4 respectively. The results on FlyingThings3D demonstrate that our model is able to correctly predict scene flows for objects with complicated geometry, fine details and large 3D motions. The qualitative results on KITTI illustrate that the predicted flowed points preserve the patterns as the input objects (cars), and our model makes correct predictions for complicated objects like trees.

Table 4: Evaluation results on all the 200 KITTI scenes. Our method outperforms all the other methods on both metrics, though the data are more noisy.

Method	EPE3D	EPE2D
FlowNet3 [3]	0.6648	6.0822
ICP [2]	0.5930	32.5387
SPLATFlowNet	0.2386	9.4409
FlowNet3D [5]	0.2215	8.1638
Original BCL	0.2150	8.4043
Ours	**0.1506**	**5.6234**

D. Qualitative error analysis

Fig. 5 shows four typical errors in FlyingThings3D. In Fig. 5a, our model correctly predicts the translation but fails to predict the rotation. Predicting rotation is harder than translation, since points for the same object have different motions. In Fig. 5b, the model has worse performance when there is a messy cluster of objects without clear boundaries. Such clusters are common in FlyingThings3D since it is synthesized in a random way, but are less common in real-world scenarios. In Fig. 5c, the objects are actually two bicycles with different orientations. The seriously incomplete shapes make the model prone to mistakes. There is another common error in FlyingThings3D: when two or more objects are very close to each other in the first frame and have different motions. In Fig. 5d, the table (it is incomplete as well) at the upper left corner touches the shelf. The table flies to an upper position and the shelf flies to a lower position, which explains the prediction errors around the junction of the two objects: the model regards the two bordering objects as a single object and thus predicts incorrectly. Fig. 5e, 5f show the predicted flowed points and the ground truth flowed points from a different view point for better illustration.

Fig. 6 shows several common error patterns in KITTI. Since we remove the ground by height, remaining ground points have irregular patterns, and thus their motions are hard to predict, as shown in Fig. 6a. Fig. 6b shows the confusion errors for neighboring objects. Fig. 6c shows a unique error type for KITTI, our approach sometimes fails to make correct estimations on large-area messy bushes with large motions, since there are almost no such bushes in FlyingThings3D, the dataset the model is trained on.

E. Clarifications

Though the settings are different in different papers, we train and evaluate all the comparison methods on the same set of points and under the same setting, so the comparisons are fair, except for FlowNet3 [3], which use stereo inputs with RGB information and is not directly comparable to ours (reference purposes only).

Preprocessing. Current data filtering follows existing works and makes results more comparable. Besides, for FlyingThings3D, we evaluate on test set without hard sample removal or without removing occluded points (we do not remove occlusions for KITTI since it is the version that KITTI uses for ranking. However, the occlusion in the synthetic dataset is very different from real-world point clouds: all occluded points are there), the results are similar to current results.

Why scene flow estimation with stereo inputs does not perform well on 3D metrics? For scene flow estimation with stereo inputs, to reconstruct the 3D motions from the estimated optical flows and disparities, the transformation procedure is: 1) The depths of both frames are computed using the following formula: $depth = focal\_length * baseline/disparity$. When disparities are small, the multiplicative inversion magnifies small errors of the disparity estimation. 2) The depths of the second frame are warped to the reference frame (the first frame) by the estimated optical flows. If the optical flow is not correctly predicted, then the warped depth would be wrong. 3) The flowed pixel positions and warped depths are reconstructed into 3D coordinates with the camera matrix, and finally scene flow is estimated. At this step, the errors at different depths are changed differently. With errors accumulated through the three steps and without directly optimizing on 3D evaluation metrics, methods with stereo inputs do not perform as well on 3D metrics as they perform on optical flow and disparity metrics.

References

[1] A.B. Adams. *High-dimensional gaussian filtering for computational photography*. PhD thesis, Stanford University, 2011. 2

[2] P.J. Besl and N.D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, 1992. 1, 2, 4

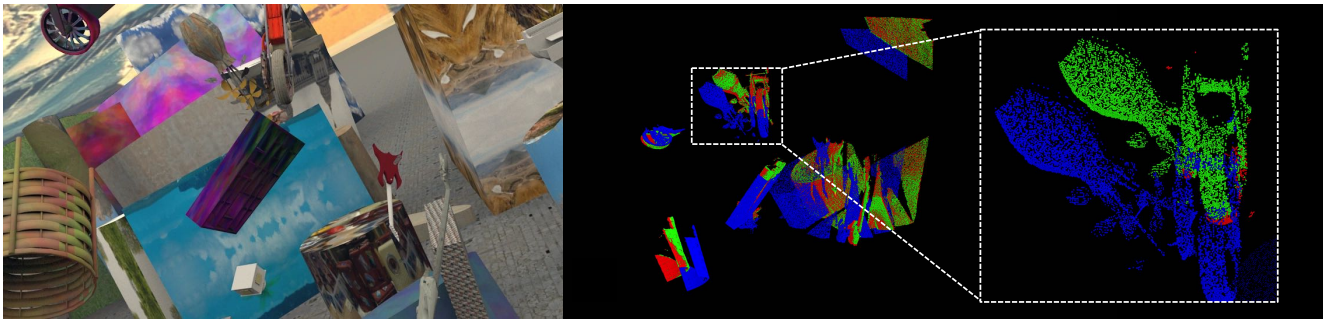
[3] E. Ilg, T. Saikia, M. Keuper, and T. Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*, 2018. 1, 2, 4

[4] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

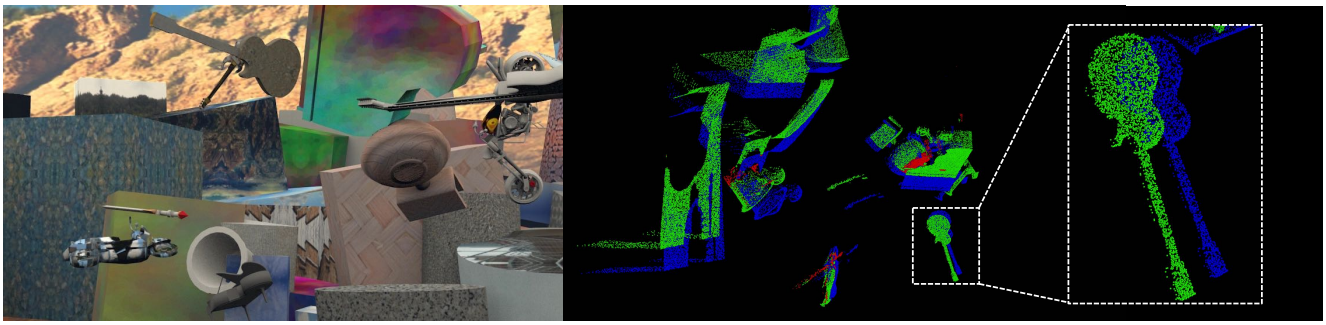
[5] X. Liu, C.R. Qi, and L.J. Guibas. Learning scene flow in 3d point clouds. *arXiv:1806.01411v1*, 2018. 1, 2, 4

[6] M. Menze, C. Heipke, and A. Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2015. 1

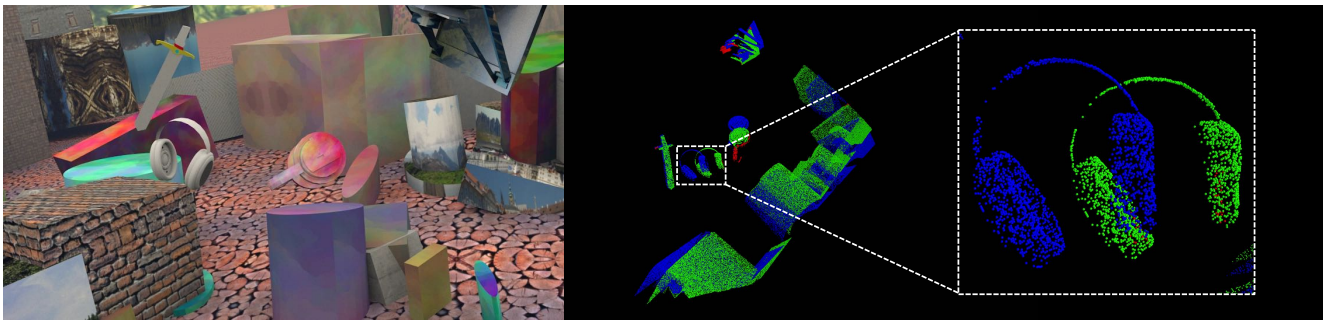
[7] M. Menze, C. Heipke, and A. Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018. 1



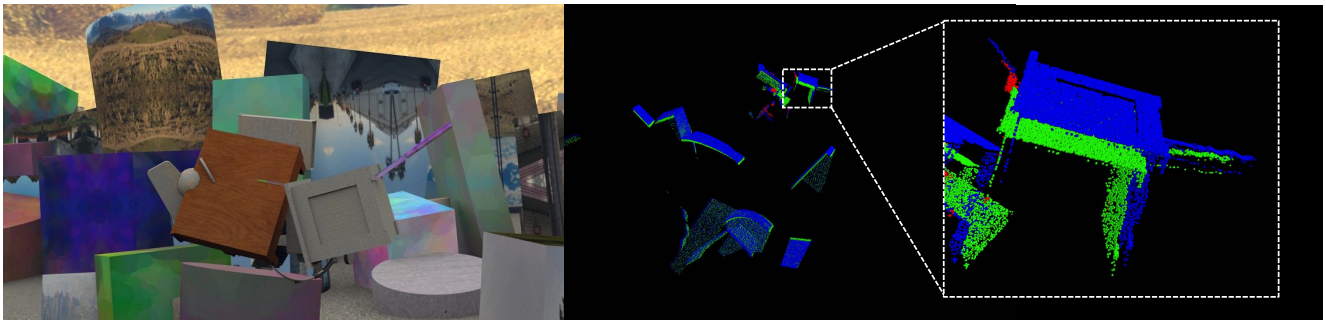
(a) FlyingThings3D 0001075, EPE3D=0.0875.



(b) FlyingThings3D 0001286, EPE3D=0.0514.

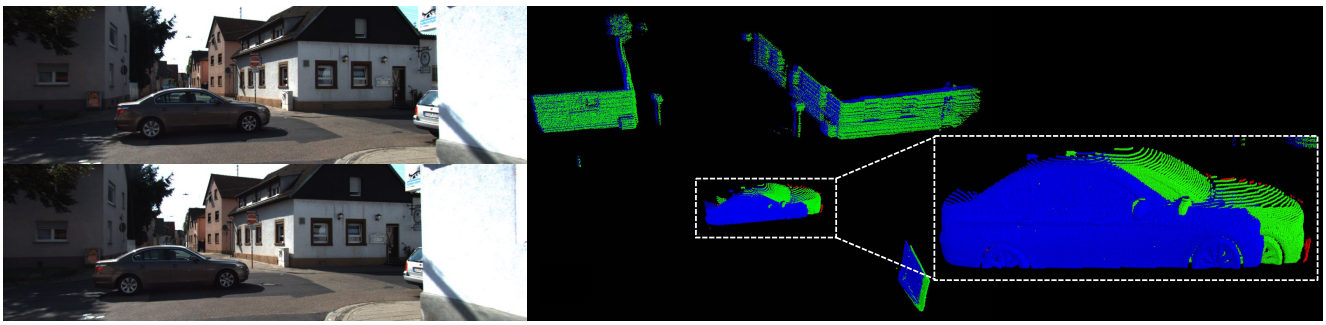


(c) FlyingThings3D 0001540, EPE3D=0.0366.

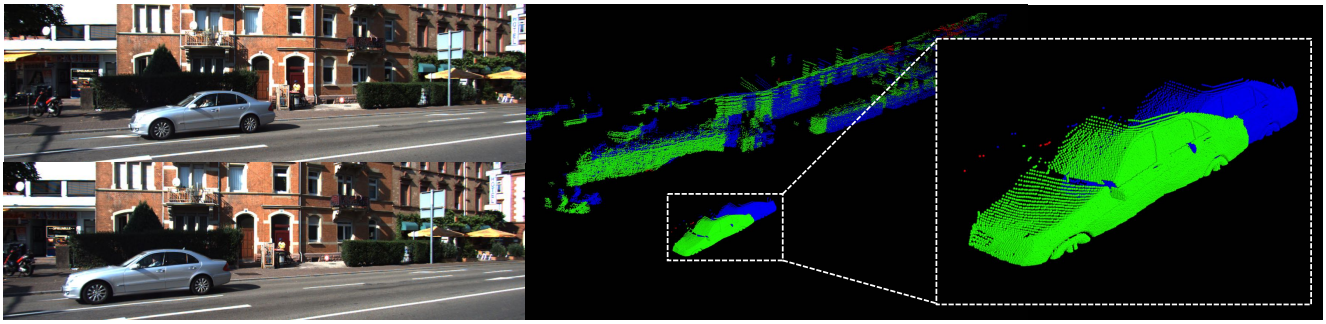


(d) FlyingThings3D 0003588, EPE3D=0.0394.

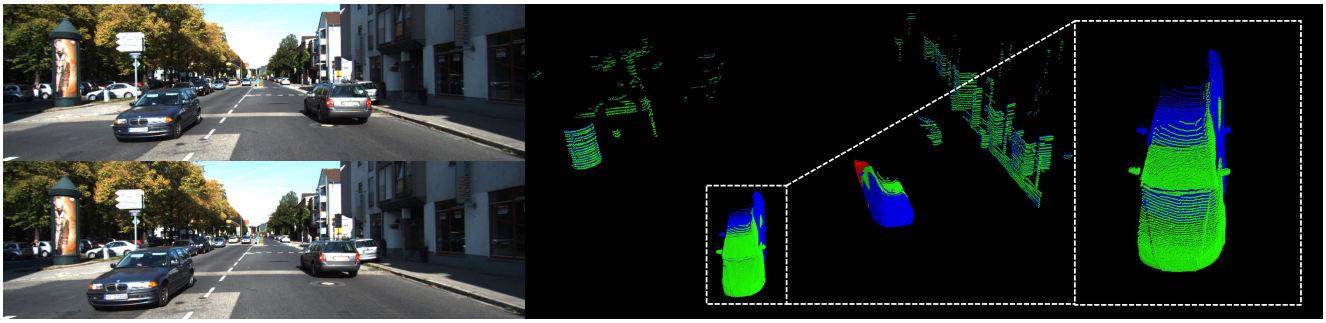
Figure 3: Visualization on FlyingThings3D. Left is the RGB image for the first frame, and right is the point cloud visualization: Blue points are PC_1, green points are *correctly predicted* (measured by Acc3D Relax) flowed points $PC_1 + \widehat{sf}$, and red points are ground-truth flowed points $PC_1 + sf$ which are *not correctly predicted*. We also report the EPE3D for each pair of frames.



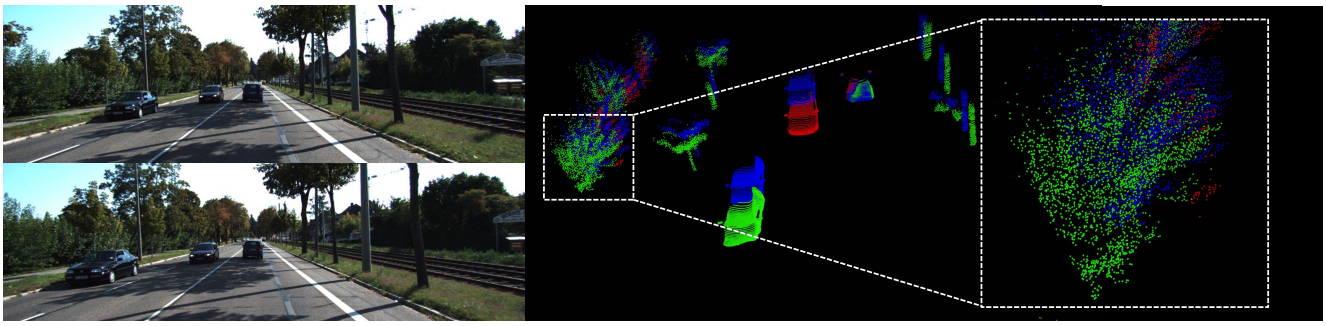
(a) KITTI 000086_10/000086_11, EPE3D=0.0449.



(b) KITTI 000133_10/000133_11, EPE3D=0.0641.

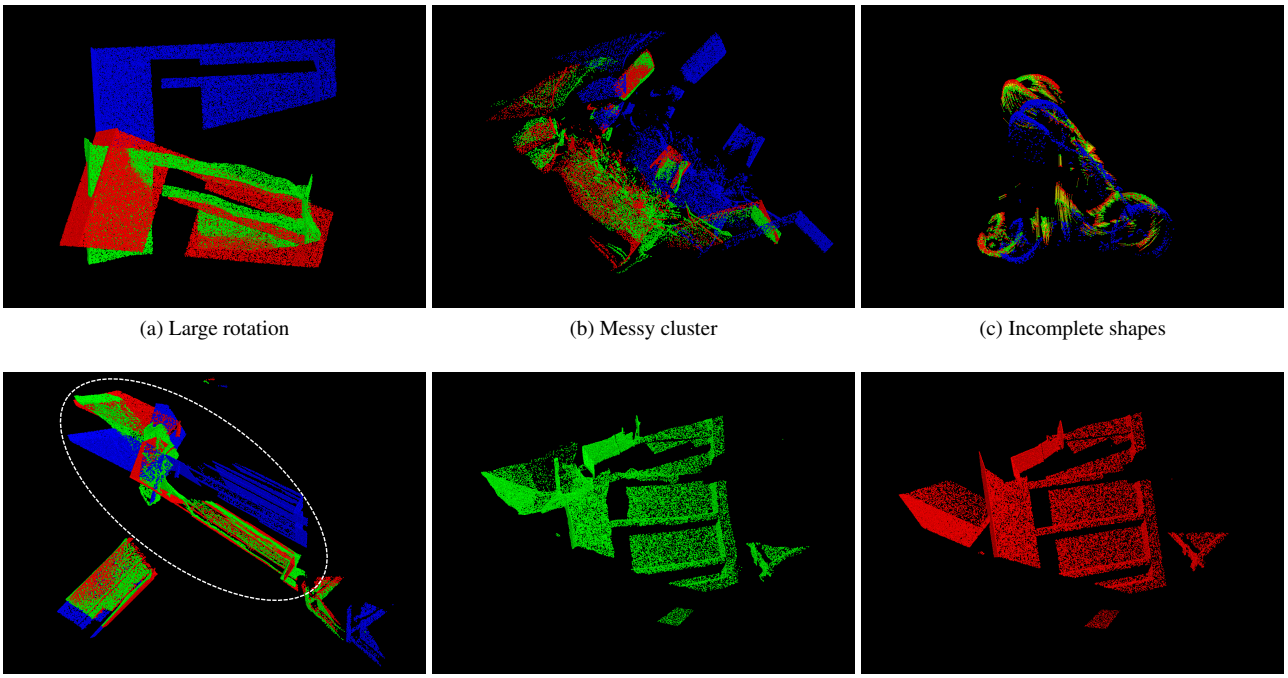


(c) KITTI 000137_10/000137_11, EPE3D=0.0364.



(d) KITTI 000106_10/000106_11, EPE3D=0.1389.

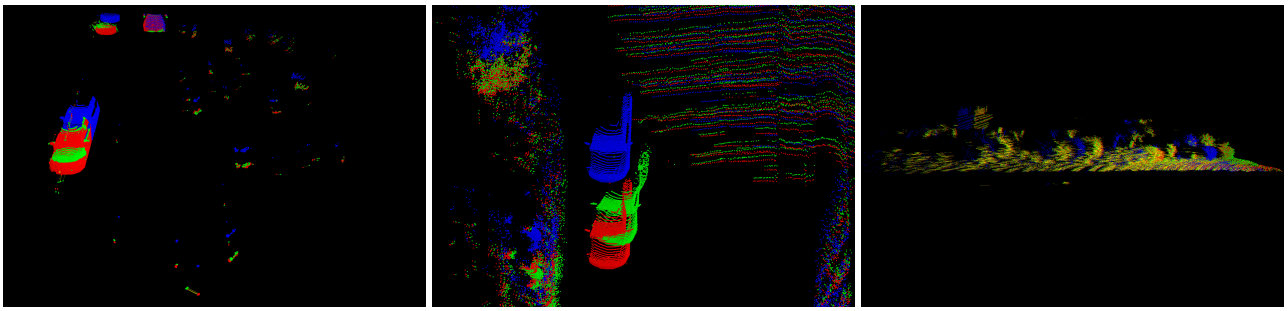
Figure 4: Visualization on KITTI. Left is the RGB pair for the two consecutive frames. The point cloud visualization scheme is the same as Fig. 3.



(a) Large rotation (b) Messy cluster (c) Incomplete shapes

(d) Confusion errors for neighboring objects (e) Predicted flowed points for Fig. 5d (f) Ground truth flowed points for Fig. 5d

Figure 5: Typical error types for FlyingThings3D. The visualization scheme is different from the previous ones: We plot PC_1 in blue, *ALL* the predicted flowed points $PC_1 + \widehat{sf}$ in green, and *ALL* the ground-truth flowed points $PC_1 + sf$ in red. For Fig. 5c, we additionally plot the error vectors in yellow for better illustration.



(a) Scattering ground points (b) Confusion errors for neighboring objects (c) Messy bushes

Figure 6: Typical error types for KITTI. We use the same visualization scheme as Fig. 5.