DifFlow3D: Toward Robust Uncertainty-Aware Scene Flow Estimation with Iterative Diffusion-Based Refinement (Supplementary Materials)

Jiuming Liu[1], Guangming Wang[2], Weicai Ye[3], Chaokang Jiang[4], Jinru Han[1],
Zhe Liu[5], Guofeng Zhang[3], Dalong Du[4], Hesheng Wang[1*]
[1]Department of Automation, Shanghai Jiao Tong University [2]University of Cambridge
[3]State Key Lab of CAD & CG, Zhejiang University [4]PhiGent Robotics
[5] MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{liujiuming,wangguangming,liuzhesjtu,wanghesheng}@sjtu.edu.cn
ts20060079a31@cumt.edu.cn maikeyeweicai@gmail.com

1. Overview

The supplementary materials are briefly distributed as follows:

- we first illustrate the motivation for why we leverage the diffusion model into the scene flow estimation task in Section 2.
- we show more model details including the network architecture designs in Section 3, datatsets and evaluation metrics in Section 4.
- Additional experimental results are also presented in Section 5 to demonstrate the superiority and universality of our method.
- Furthermore, quantities of visualization figures are shown in Section 6.

2. Motivation

Quite a few previous scene flow networks utilize the coarse-to-fine structure [4, 8, 14, 18] or recurrent iterations [3, 7, 17], which fit in well with the principles of diffusion models intrinsically since diffusion models progressively diffuse and generate samples step-by-step through a series of Markov chain. Furthermore, the iterative denoising steps of the diffusion model can enhance the model's robustness to noisy points, as demonstrated in our main manuscript.

Basically, the diffusion model is attributed to the field of generation models. Recently, an increasing number of researchers have tried to adapt it to deterministic tasks, such as object detection [1, 5], semantic segmentation [13, 19], and pose estimation [6, 16, 20]. The powerful ability and denoising intrinsic of diffusion models boost the advancements in these domains. Inspired by their successes, we also resort to diffusion models for providing a robust and universal refinement module to further improve the scene flow estimation accuracy.

*Corresponding Authors.

Algorithm 1 Training Process

Input: timestamp t, ground truth flow residual s_0, condition information C.

1: **repeat**
2: $s_0 \sim q(s_0)$
3: $t \sim$ Uniform({1,...,T})
4: Sample the intermediate flow residual s_t at the timestamp t through sampling noise $\epsilon \sim \mathcal{N}(0, I)$ in the forward process.
5: Take gradient descent step on $\bigtriangledown_\theta \parallel s_0 - M_\theta(s_t, C, t) \parallel$
6: **until** converged

Algorithm 2 Sampling Process

Input: total timestamps T, Gaussian noise s_T, condition information C.
Output: refined scene flow residual s_0.

1: $s_T \sim \mathcal{N}(0, I)$
2: **for** $t = T, ..., 1$ **do**
3: $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
4: $s_{t-1} = \frac{1}{\sqrt{\alpha_t}}(s_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} M_\theta(s_t, C, t)) + \sigma_t z$
5: **end for**
6: **return** s_0

Module	Layer	Point number	M_θ	Iteration	MLP width	C_{in}	C_{out}
Scene Flow Initialization	BGRU	256	-	4	[256,256]	320+320+3	256
	FE	256	-	-	[128,64],[3],[1]	256+256+3	3,1
Iterative Diffusion-based Scene Flow Refinement	$l=1$	512	GRU	4	[128,128],[3],[1]	128+128+3	64,3,1
	$l=2$	2048	GRU	4	[64,64],[3],[1]	64+64+3	64,3,1
	$l=3$	8192	GRU	4	[32,32],[3],[1]	32+32+3	64,3,1

Table 1. **Detailed network parameters with MSBRN [3] as our baseline.** MLP width indicates the number of output channels for each layer of MLP. C_{in} and C_{out} respectively denote the number of input and output channels.

Module		Layer type	K	Sample rate	MLP width
Scene Flow Initialization		All-to-all point gathering	4,256	1	[256,128,128], [256,128]
		Shared MLP for FE	—	1	[128,128,256] [256,256,512]
		FC layer for flow and uncertainty	—	1	[3] [1]
Iterative Diffusion-based Scene Flow Refinement	$l=1$	Attentive cost volume	4, 6	1	[512,256,256], [512,256]
		M_θ	—	1	[512,256,256]
		FC layer for flow and uncertainty	—	1	[3], [1]
	$l=2$	Set upconv	8	4	[256,128,128], [128]
		Attentive cost volume	4, 6	1	[256,128,128], [256,128]
		M_θ	—	1	[256,128,128]
		FC layer for flow and uncertainty	—	1	[3], [1]
	$l=3$	Set upconv	8	4	[256,128,128], [128]
		Attentive cost volume	4, 6	1	[128,64,64], [128,64]
		M_θ	—	1	[256,128,128]
		FC layer for flow and uncertainty	—	1	[3], [1]
	$l=4$	Set upconv	8	2	[128,64,64], [64]
		Attentive cost volume	4, 6	1	[64,32,32], [64,32]
		M_θ	—	1	[128,64,64]
		FC layer for flow and uncertainty	—	1	[3], [1]

Table 2. **Detailed network parameters with 3DFlow [14] as our baseline.** K points are selected in the K Nearest Neighbors (KNN) of the all-to-all point gathering layer, set upconv layer, and attentive cost volume layer. MLP width means the number of output channels for each layer of MLP. FC means fully connected layer.

3. Implementation Details

3.1. Training and Inference Process

Training. Our diffusion model follows the same sampling and training strategy as in DDIM [12]. Specifically, our training process can be represented by Algorithm 1. The intermediate flow residual s_t is first sampled from the forward process by adding a pre-defined noise schedule onto the flow residual ground truth s_0. Then, condition information C, intermediate flow residual s_t, and time embedding from t will all be treated as inputs of the denoising network M_θ. The optimization objective is designed by narrowing the distribution between the ground truth flow residual s_0 and predicted one $M_\theta(s_t, C, t)$. Furthermore, we propose a per-flow corresponding uncertainty estimation within the same diffusion model. Its training process remains the same as the scene flow. For better clarity, we omit its description in the Algorithm 1. Finally, our network employs multi-scale supervision combining scene flow, scene flow residual, and uncertainty residual. Their weights λ_{sf}, λ_{res}, and λ_{un} are set as 1, 1, 1 respectively. For different refinement layers, our designed loss weights are $0.02, 0.04, 0.08, 0.16$. All the variables are defined as described in the main manuscript.

Sampling. We iteratively generate the flow residuals from a total Gaussian noise s_T as in Algorithm 2. Our proposed diffusion-based refinement module recovers the denser flow residuals from coarse to fine. Each layer has the same sampling process in Algorithm 2.

3.2. Network Architecture Details

Basically, our whole network consists of three components: feature extraction, scene flow initialization, and iterative diffusion-based flow refinement. Among these, feature extraction and flow initialization are not our main concerns in this paper. Instead, we try to investigate whether there exists a relatively universal method that can refine coarse estimated scene flow for more accurate and robust denser flow estimation. With the diffusion model, our proposed refinement layer can be adapted to a series of recent SOTA methods. Here, we show network settings based on two representative methods: coarse-to-fine structure based—3DFlow [14], and the recurrent network based—MSBRN [3]. We combine the initialization manners in their networks with our iterative diffusion-based refinement module as follows.

Method	FT3D$_s$				KITTI$_s$				Runtime(ms)	Memory (G)
	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓		
GMSF [21] (NeurIPS2023)	**0.0104**	**0.9891**	**0.9963**	**0.0320**	0.0272	0.9330	0.9822	0.1316	2305.7	162.3
Ours	0.0114	0.9836	0.9949	0.0350	**0.0078**	**0.9817**	**0.9924**	**0.0795**	**228.3**	**10.8**

Table 3. **Comparison with GMSF [21] on the FT3D$_s$ and KITTI$_s$ datasets.** We compare the performance of our model with the recent SOTA method–GMSF on both two datasets. GMSF is based on the global matching with transformer architecture. Although their model has slightly better accuracy when evaluated on the Flyingthings3D dataset, it suffers from dramatically decreasing accuracy when generalized on the KITTI dataset, and also much lower efficiency. Runtime is evaluated on the same RTX 3090 GPU. Memory indicates the memory requirement for training. The best results are in bold.

Method	Sup.	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓
SFGAN [15]	Self	0.075	0.4980	0.8117	0.4530
Ours	Full	**0.008**	**0.9817**	**0.9924**	**0.0795**

Table 4. **Comparison with SFGAN on the KITTI$_s$ dataset.** SFGAN is a kind of self-supervised GAN-based scene flow estimation method, which is also trained on the FT3D$_s$ dataset.

With the initialization method in MSBRN. MSBRN [3] proposes a multi-scale bi-directional recurrent network by combining the GRU with hybrid correlation. We adopt its flow initialization manner as in Table 1. The flow initialization mainly consists of two parts: BGRU and Flow embedding (FE) layers. BGRU is proposed to augment point features with bi-directional frame association and iteratively optimize the state correlations. Then, augmented point correlation features from BGRU will be the inputs of the FE layer together with down-sampled point features from PC_1. In Table 1, we list the main network details of its initialization. After the initialization, our proposed iterative diffusion-based refinement is leveraged to recover the denser flow residuals. The denoising network is designed as GRU and iterations of GRU are all set as 3. In each refinement layer, we input correlation features, up-sampled coarse flow features, and up-sampled point features into the module. After concatenating the time embedding and latent variables, these features are all processed by the denoising network. Finally, refined point features, refined scene flow residual, and uncertainty residual are generated from layer l and then serve as the inputs of layer $l+1$.

With the initialization method in 3DFlow. 3DFlow [14] designs an all-to-all initial correlation layer with the backward validation, where its refinement is based on five components: up-sampled dense flow embedding, flow re-embedding, point feature from PC_1, dense flow, and dense flow feature. Here, we apply the same flow initialization method using the all-to-all point gathering module. Then, all five flow-related features are utilized as condition information within our diffusion model for guiding the flow residual generation. Similarly, we will concatenate the condition information, time embedding, and latent variables together to go through the denoising network as in Table 2. Finally, a fully connected layer is leveraged to regress the flow and uncertainty residuals.

4. Datasets and Evaluation Metrics

4.1. Datasets and Data Pre-processing

In this section, we supplement more details about the datasets and data pre-processing methods used in the main manuscript.

Datatsets. Following previous works [2, 4, 8, 11, 14, 17], we train our DifFlow3D on synthetic FlyingThings3D dataset [9]. To demonstrate the generalization ability of our model, we conduct the evaluation experiments not only on FlyingThings3D but also on real-world LiDAR scans from the KITTI [10] dataset. For a fair comparison, we follow previous works [2, 4, 8, 11, 14, 17] to feed only XYZ coordinates into our network as inputs, and randomly sample 8192 points for both two point cloud frames.

Data pre-processing. Common data processing of the above two datasets in the scene flow task has two versions: 1) Data without occlusion. This is proposed by HPLFlowNet [4], where each point in PC_1 has its corresponding point in PC_2. 2) Data with occlusion, which is proposed by FlowNet3D [8]. This version has occluded points and masks as inputs. Masks indicate invalid points that have no correspondence in the other frame, which are also used to compute training loss and evaluation metrics. For a comprehensive and fair comparison with previous works, we follow both two data processing versions for training and testing our model.

4.2. Evaluation Metrics

For a fair comparison with previous scene flow methods, we evaluate our DifFlow3D on the same evaluation metrics as [2, 4, 8, 11, 14, 17], including both 3D and 2D metrics.

EPE3D (m): $\| sf^l - sf^{GT} \|_2$ indicates the average end-point-error between the estimated scene flow positions and ground truth ones. It is commonly used as the most representative metric in the scene flow estimation task.

Acc3DS: the percentage of points which satisfy EPE3D $< 0.05m$ or relative error $< 5\%$.

Acc3DR: the percentage of points which satisfy EPE3D $< 0.1m$ or relative error $< 10\%$.

Method	Training	FT3D$_s$ EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE2D↓	Acc2D↑	KITTI$_s$ EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE2D↓	Acc2D↑
PointPWC [18]	Complete	0.0588	0.7379	0.9276	0.3424	3.2390	0.7994	0.0694	0.7281	0.8884	0.2648	3.0062	0.7673
PointPWC+DSFR	Complete	**0.0476** (↓ 19.0%)	**0.8357**	**0.9557**	**0.2515**	**2.5840**	**0.8607**	**0.0518** (↓ 25.4%)	**0.8314**	**0.9468**	**0.2009**	2.1435	**0.8465**

Table 5. **The plug-and-play results with the initialization of PointPWC [18] on the FT3D$_s$ and KITTI$_s$ dataset.** Our Diffusion-based Scene Flow Refinement (DSFR) can effectively improve the accuracy introduced into PointPWC. The best results are in bold.

Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE2D↓	Acc2D↑
Only initialized flow	0.1559	0.0777	0.4462	0.8637	6.2390	0.3759
Predicted flow with one refinement layer	0.0883	0.4071	0.8197	0.5817	4.7682	0.5759
Predicted flow with two refinement layers	0.0274	0.9284	0.9799	0.1356	1.5036	0.9228
Predicted flow with three refinement layers	**0.0114**	**0.9836**	**0.9949**	**0.0350**	**0.6220**	**0.9824**

Table 6. **Ablation studies about the layer number of diffusion-based refinement on the FT3D$_s$ dataset.**

Iteration	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE2D↓	Acc2D↑	Runtime(ms)
2	0.0339	0.8909	0.9822	0.2476	1.7726	0.9211	**152.3**
3	0.0178	0.9680	0.9910	0.0767	0.9679	0.9667	212.9
4	0.0114	**0.9836**	0.9949	0.0350	0.6220	**0.9824**	228.3
5	**0.0113**	0.9774	**0.9950**	**0.0345**	**0.6163**	0.9758	335.3
6	0.0157	0.9761	0.9933	0.0693	0.8371	0.9752	395.4

Table 7. **Ablation studies about the iteration times in denoising GRU on the FT3D$_s$ dataset.**

Outliers: the percentage of points which satisfy EPE3D $> 0.3m$ or relative error $> 10\%$.

EPE2D (px): the average 2D end-point-error measured by projecting back onto the image plane.

Acc2D: the percentage of points which satisfy EPE2D $< 3px$ or relative error $< 5\%$.

5. Additional Experiments

5.1. More Comparison Results

In this section, we present more comparison results with recent methods as follows.

Comparison results with GMSF. GMSF [21] proposes a global matching approach by leveraging the transformer architecture into scene flow estimation. As in Table 3, although the performance on the synthetic dataset Flyingthings3D has already saturated because of their per-point dense matching between two frames with cross attention, the drawbacks are also obvious. On the one hand, their running time is extremely large due to the quadratic complexity of transformer to the point number, particularly for all dense points. Typically, their model is trained on four NVIDIA A40 GPUs, but our DifFlow3D can be only trained on a single RTX 3090 GPU with rather competitive accuracy. On the other hand, their generalization capability is much worse than ours on the KITTI dataset, partly because dense global matching can learn less about the universal and underlying rules across different data domains.

Comparison results with SFGAN. Similar to this work, SFGAN [15] also proposes a scene flow network with another generative model—GAN. They synthesize indistinguishable fake point clouds through the adversarial training of the generator, and discriminate the real points and the synthesized fake points by the discriminator. Here, we also compare our diffusion-based method with their GAN-based one in Table 4. Obviously, our DifFlow3D can outperform SFGAN by a large margin in terms of EPE3D, Acc3DR, Acc3DR, and Outliers metrics. Both two networks are trained on the Flyingthings3D dataset and evaluated on the KITTI dataset. The difference is that SFGAN is self-supervised trained on FT3D$_s$ and then finetuned on KITTI$_s$, but our method is fully-supervised trained on FT3D$_s$ and evaluated on KITTI$_s$ without any fine-tuning. We leave the self-supervised learning of the diffusion-based refinement module as our future work.

5.2. Plug-and-play Results on PointPWC

As shown in the main manuscript, our Diffusion-based Scene Flow Refinement (DSFR) module can serve as a plug-and-play module into a series of recent methods, significantly improving their estimation accuracy. Here, we also show the results by applying our DSFR module to one classic learning-based method–PointPWC [18] in Table 5. Our DSFR can improve the estimation accuracy by 7.3% on the synthetic Flyingthings3D dataset.

5.3. Ablation studies

In this section, we further conduct ablation studies about the number of refinement layers, the iteration times in GRU, and uncertainty threshold settings.

Refinement layer number. We first conduct experiments about different layer numbers of the diffusion-based refinement in our model. As illustrated in Table 6, with increasing coarse-to-fine refinement layers, our predicted scene flow has more EPE3D reduction progressively. However, more additional refinement layers can not further improve the estimation accuracy but have much larger computational burdens.

Iteration times of denoising network. We leverage GRU as our denoising network. Here, we also show how

Method	Training	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers↓	EPE2D↓	Acc2D↑
Ours w/o uncertainty	Quarter	0.0318	0.9136	0.9769	0.1640	1.7374	0.9158
Ours with uncertainty	Quarter	**0.0297**	**0.9207**	**0.9785**	**0.1548**	**1.6344**	**0.9188**
Ours w/o uncertainty	Complete	0.0290	0.9296	0.9826	0.1403	1.5458	0.9321
Ours with uncertainty	Complete	**0.0242**	**0.9494**	**0.9860**	**0.1166**	**1.3201**	**0.9459**

Table 8. **Ablation studies about uncertainty (with the flow initialization based on 3DFlow) on the FT3D$_s$ dataset.**

EPE3D (m)		E_{end}			
		0.05	0.03	0.02	0.01
E_{start}	0.5	0.024	0.021	0.011	0.027
	0.3	0.020	0.024	0.020	0.021
	0.1	0.018	0.020	0.023	0.019

Table 9. **Comparison with different uncertainty threshold configurations on the FT3D$_s$ dataset.**

the number of iterations in GRU influences our model performance in Table 7. Specifically, our model is trained with four iterations, but evaluated with different iteration settings. The increasing iteration times of GRU lead to improved estimation accuracy but at the cost of decreased efficiency. Once the iteration number exceeds 4, the accuracy improvement becomes less significant while the runtime dramatically increases. Thus, our final iteration choice is set as 4, which is the trade-off between accuracy and efficiency.

Significance of the uncertainty (with the initialization in 3DFlow). We propose an uncertainty-aware scene flow estimation framework, where uncertainty can evaluate and reflect the reliability of our estimated scene flow. In the main manuscript, we have conducted ablation studies about with or without uncertainty based on the initialization in MSBRN [3]. However, we notice that the performance of MSBRN has relatively saturated. Therefore, we also show the comparison results about how our proposed uncertainty influences the estimation accuracy with the initialization in 3DFlow [14] in Table 8. Without designed uncertainty estimation, our network has 7.1% and 19.8% increasing estimation errors on quarter and complete datasets in terms of the EPE3D metric.

Uncertainty thresholds. As in the main manuscript, the ground truth uncertainty is constrained by absolute and relative estimation error thresholds which dynamically decay during the training. This leads the scene flow estimation to progressively approach the ground truth flow, and makes the network aware of which flow matching is unreliable. Here, we conduct experiments by comparing different threshold settings. As in Table 9, we can witness the best result when the threshold starts at 0.5 and ends at 0.02.

6. Visualization

In this section, we show more qualitative results, including the distribution of our proposed uncertainty, plug-and-play

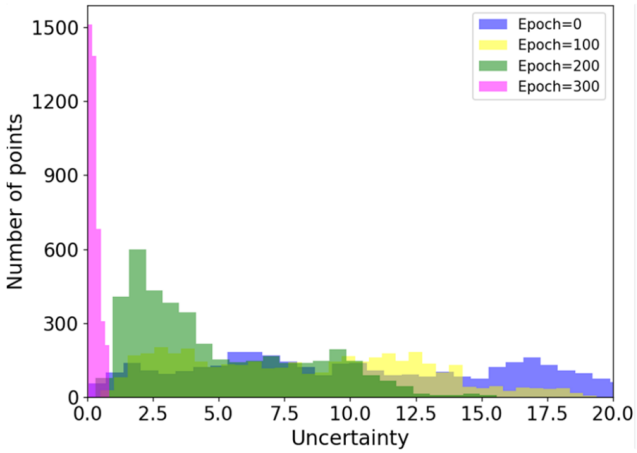

Figure 1. **Visualization of our predicted uncertainty during the training process.** We calculate the distribution histogram to investigate how the uncertainty varies during the training process.

results, and scene flow estimation results on both Flyingthings3D and KITTI datasets.

Uncertainty distribution. We visualize the uncertainty distribution at different training stages in Fig.1. At the initial training stages, the distribution variance of our uncertainty is rather large because the network has almost no capability to predict accurate scene flow. With the training process, our estimated flow tends to approach the ground truth flow. In this case, our estimated uncertainty will decline toward 0, since the uncertainty ground truth is determined by the estimation errors. When estimation errors are less than certain thresholds, the uncertainty ground truth will be set as 0. In this case, uncertainty will get smaller with the training process, which progressively leads the scene flow estimation toward a more accurate value.

Plug-and-play visualization of our proposed DSFR. Our proposed Diffusion-based Scene Flow Refinement (DSFR) module proves to be effective on a series of recent methods as in Table 3 of the main manuscript and Table 5 of the supplementary materials. To more intuitively demonstrate the effectiveness and superiority of our method, we also display the visualization results of the plug-and-play application on 3DFlow [14] and MSBRN [3] in Fig. 2. From the figure, it is obvious that the estimation accuracy of their predicted scene flow is improved dramatically when combined with our proposed DSFR module. Notably, for dynamic cars or brushwood with complex and repetitive

patterns, our DSFR can alleviate the mismatching and further enhance the estimation reliability as in Fig. 2.

Comparison with previous scene flow networks on challenging cases. As stated in the main manuscript, our DifFlow3D has better robustness and accuracy on challenging cases due to the denoising intrinsic of diffusion and uncertainty estimation. Here, we present more samples in Fig. 3. Previous scene flow networks commonly tend to have flow mismatching on dynamics, such as the moving car in the sample (a). Also, they have poor capability to estimate accurate scene flow in regions with repetitive patterns, such as the brushwood in samples (b) and (c). However, we notice that our DifFlow3D has wonderful performance on these challenging cases, which can be partly attributed to the robustness of our finding correct point correspondences.

Qualitative scene flow estimation results. In Fig. 4 and Fig. 5, we respectively list five samples to illustrate our predicted scene flow results on Flyingthings3D and KITTI datasets. The scene flow estimation task aims to predict per-point flow vectors. Inputs are point cloud frames captured from two consecutive time steps as in the first column. In the second column, we add the predicted scene flow onto the first frame PC_1 to form predicted PC_2. In the last column, we put predicted PC_2 and original PC_2 together. These two point clouds are almost overlapped, which indicates that our predicted flow is extremely accurate.

References

[1] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19830–19843, 2023. 1

[2] Wencan Cheng and Jong Hwan Ko. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *European Conference on Computer Vision*, pages 108–124. Springer, 2022. 3

[3] Wencan Cheng and Jong Hwan Ko. Multi-scale bidirectional recurrent network with hybrid correlation for point cloud based scene flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10041–10050, 2023. 1, 2, 3, 5, 8

[4] 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. 1, 3

[5] Cheng-Ju Ho, Chen-Hsuan Tai, Yen-Yu Lin, Ming-Hsuan Yang, and Yi-Hsuan Tsai. Diffusion-ss3d: Diffusion model for semi-supervised 3d object detection. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[6] Karl Holmquist and Bastian Wandt. Diffpose: Multi-hypothesis human pose estimation using diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15977–15987, 2023. 1

[7] 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. 1

[8] 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, 3

[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. 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. 3

[11] 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

[12] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[13] Haoru Tan, Sitong Wu, and Jimin Pi. Semantic diffusion network for semantic segmentation. *Advances in Neural Information Processing Systems*, 35:8702–8716, 2022. 1

[14] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What matters for 3d scene flow network. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 38–55. Springer, 2022. 1, 2, 3, 5, 8

[15] Guangming Wang, Chaokang Jiang, Zehang Shen, Yanzi Miao, and Hesheng Wang. Sfgan: Unsupervised generative adversarial learning of 3d scene flow from the 3d scene self. *Advanced Intelligent Systems*, 4(4):2100197, 2022. 3, 4

[16] Jianyuan Wang, Christian Rupprecht, and David Novotny. Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9773–9783, 2023. 1

[17] 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. 1, 3

[18] 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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 88–107. Springer, 2020. 1, 4

[19] Weijia Wu, Yuzhong Zhao, Mike Zheng Shou, Hong Zhou, and Chunhua Shen. Diffumask: Synthesizing images with

pixel-level annotations for semantic segmentation using diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1206–1217, 2023. 1

[20] Jiyao Zhang, Mingdong Wu, and Hao Dong. Generative category-level object pose estimation via diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[21] Yushan Zhang, Johan Edstedt, Bastian Wandt, Per-Erik Forssén, Maria Magnusson, and Michael Felsberg. Gmsf: Global matching scene flow. *Advances in Neural Information Processing Systems*, 36, 2024. 3, 4

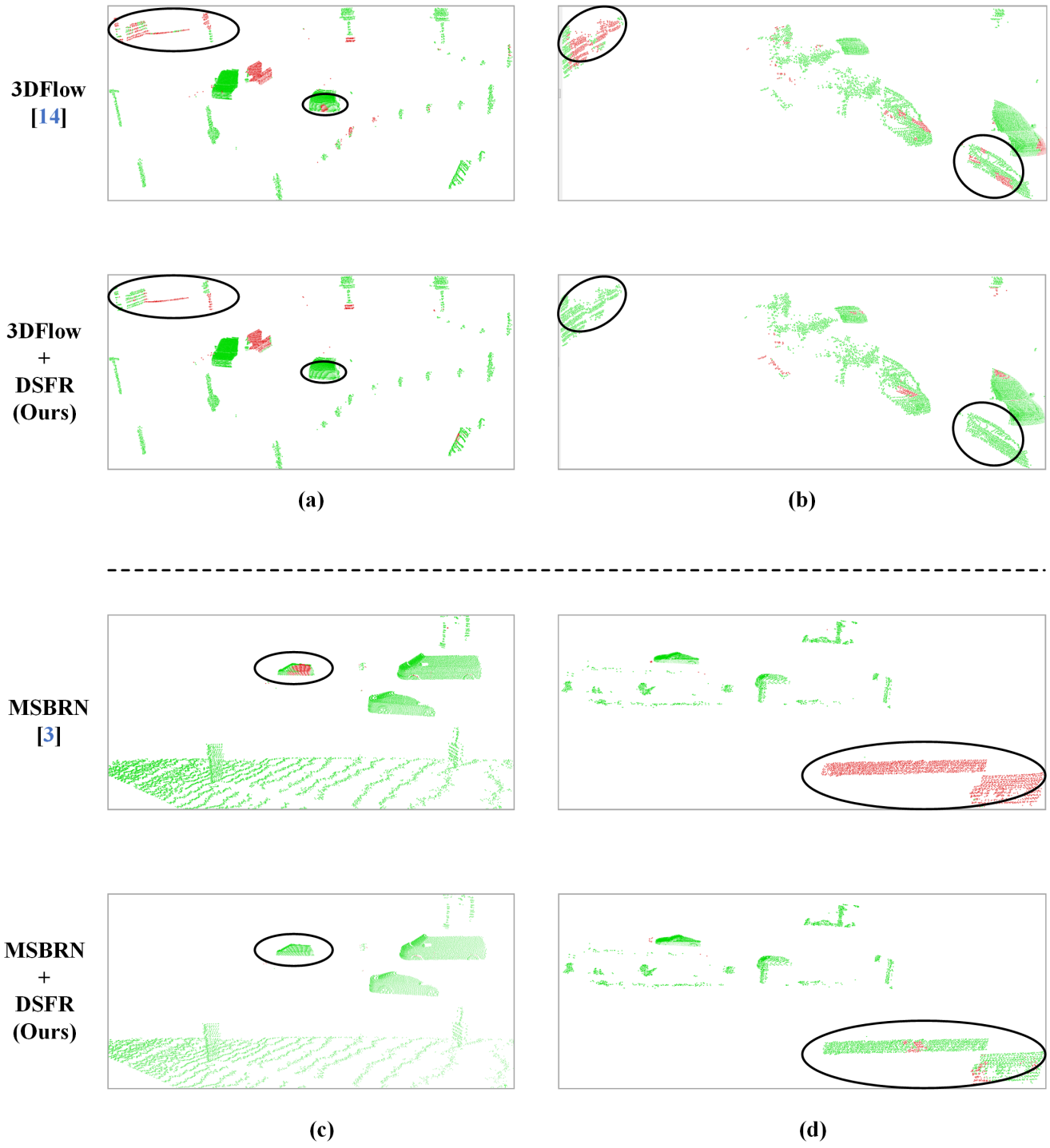

Figure 2. **Visualization of plug-and-play results on 3DFlow [14] and MSBRN [3].** green, red points respectively indicate accurately estimated PC_2 (PC_1 warped by estimated flow), and inaccurately estimated PC_2. The accuracy here is measured by Acc3DR, which means that if the EPE3D or relative error is lower than certain thresholds, the estimation is defined as an accurate one. Our proposed DSFR can significantly improve the estimation accuracy of previous networks to the dynamic cars as in samples (a) and (c) or the brushwood with repetitive patterns as in samples (b) and (d).

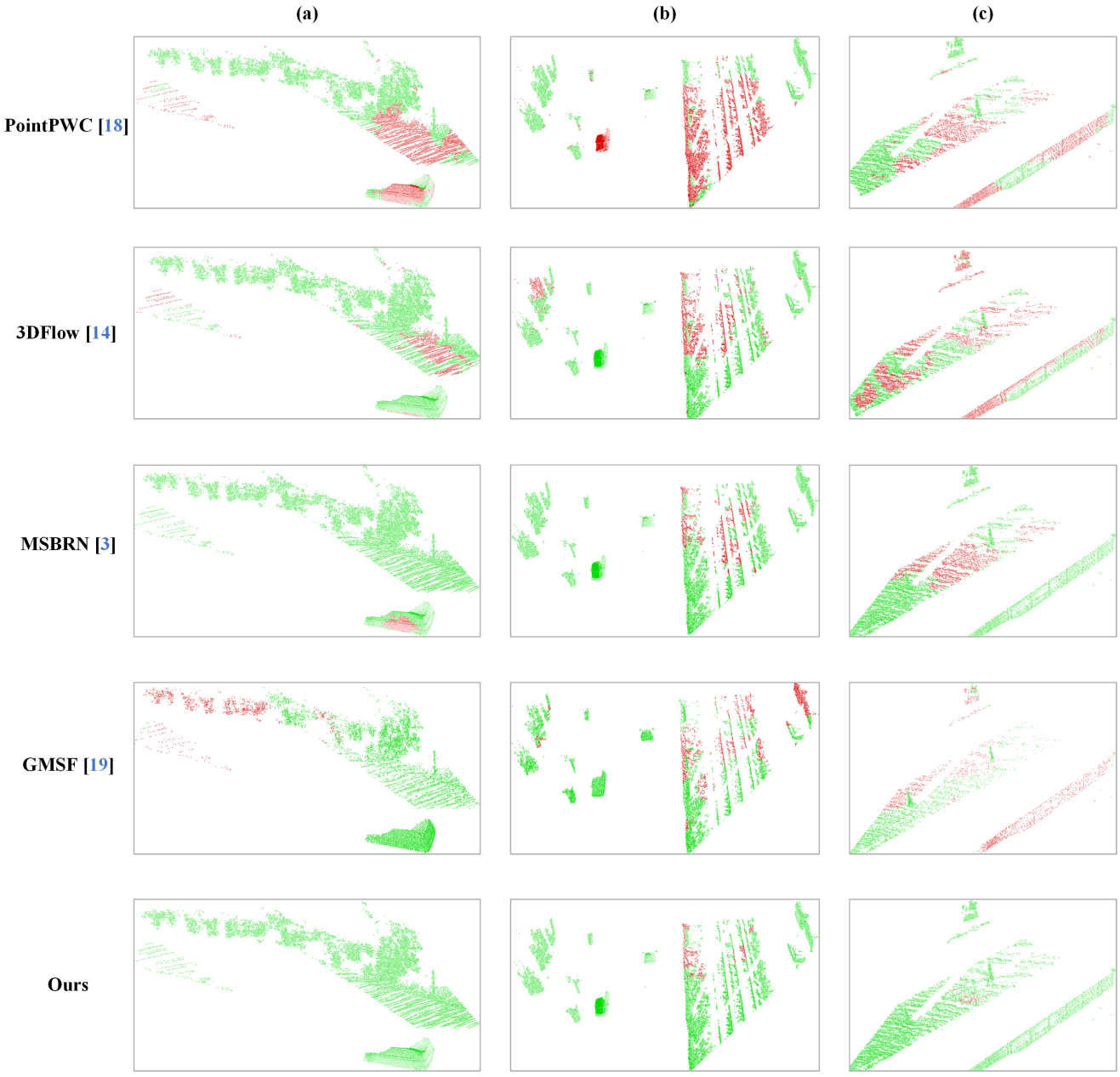

Figure 3. **Comparison results with previous works on challenging cases of the KITTI dataset.** green, red points respectively indicate accurately estimated PC_2 (PC_1 warped by estimated flow), and inaccurately estimated PC_2. Previous scene flow networks commonly tend to have flow mismatching on dynamics, such as the moving car in sample (a). Also, they have poor flow estimation capability in regions with repetitive patterns, such as the brushwood in samples (b) and (c). However, we notice that our DifFlow3D has wonderful performance on these challenging cases, which can be partly attributed to the robustness of our finding correct point correspondences.

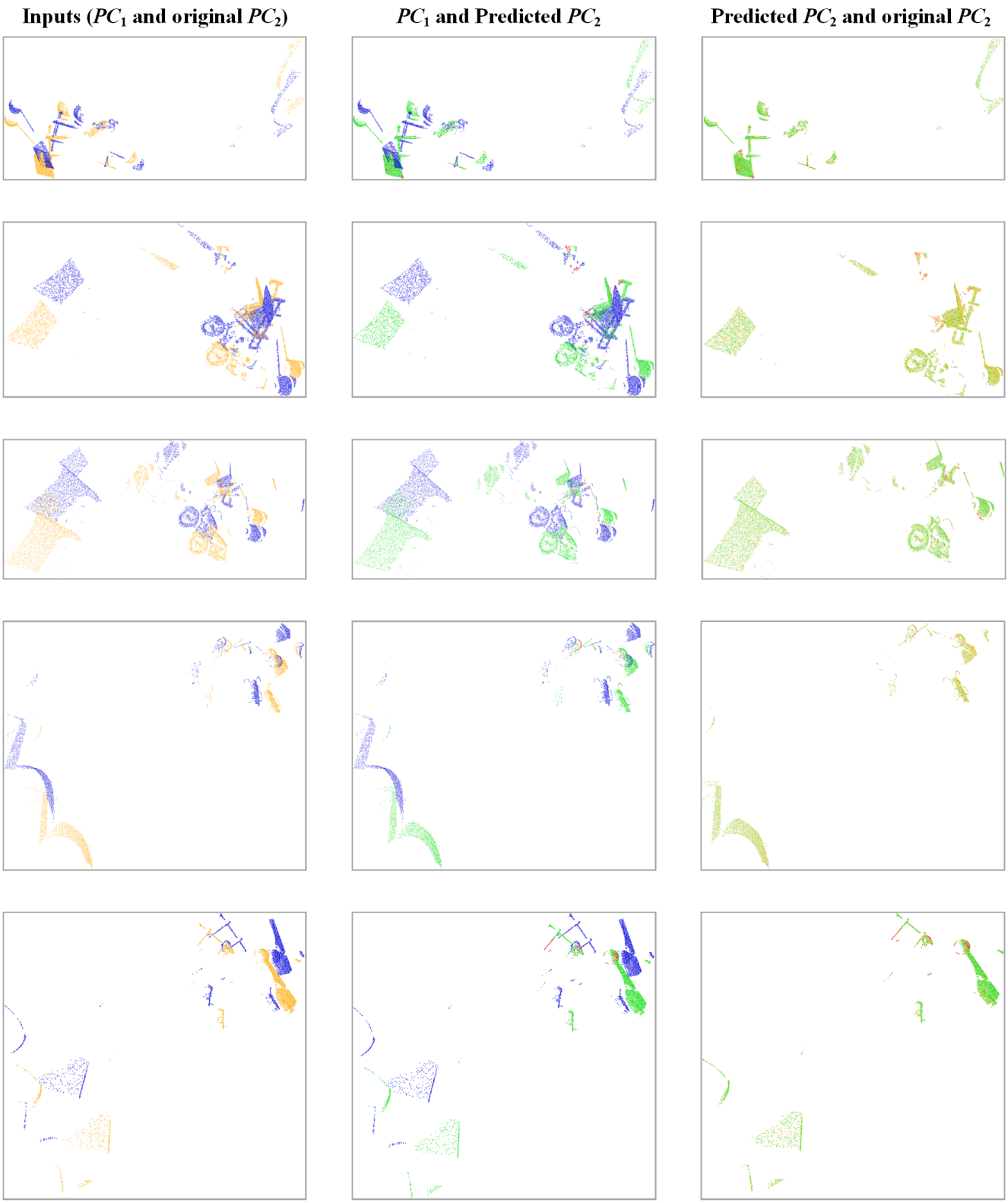

Figure 4. **Visualization of the predicted scene flow on the FlyingThings3D dataset.** Blue, Orange, green, red points respectively indicate the first frame PC_1, the second frame PC_2, accurately estimated PC_2 (PC_1 warped by estimated flow), and inaccurately estimated PC_2.

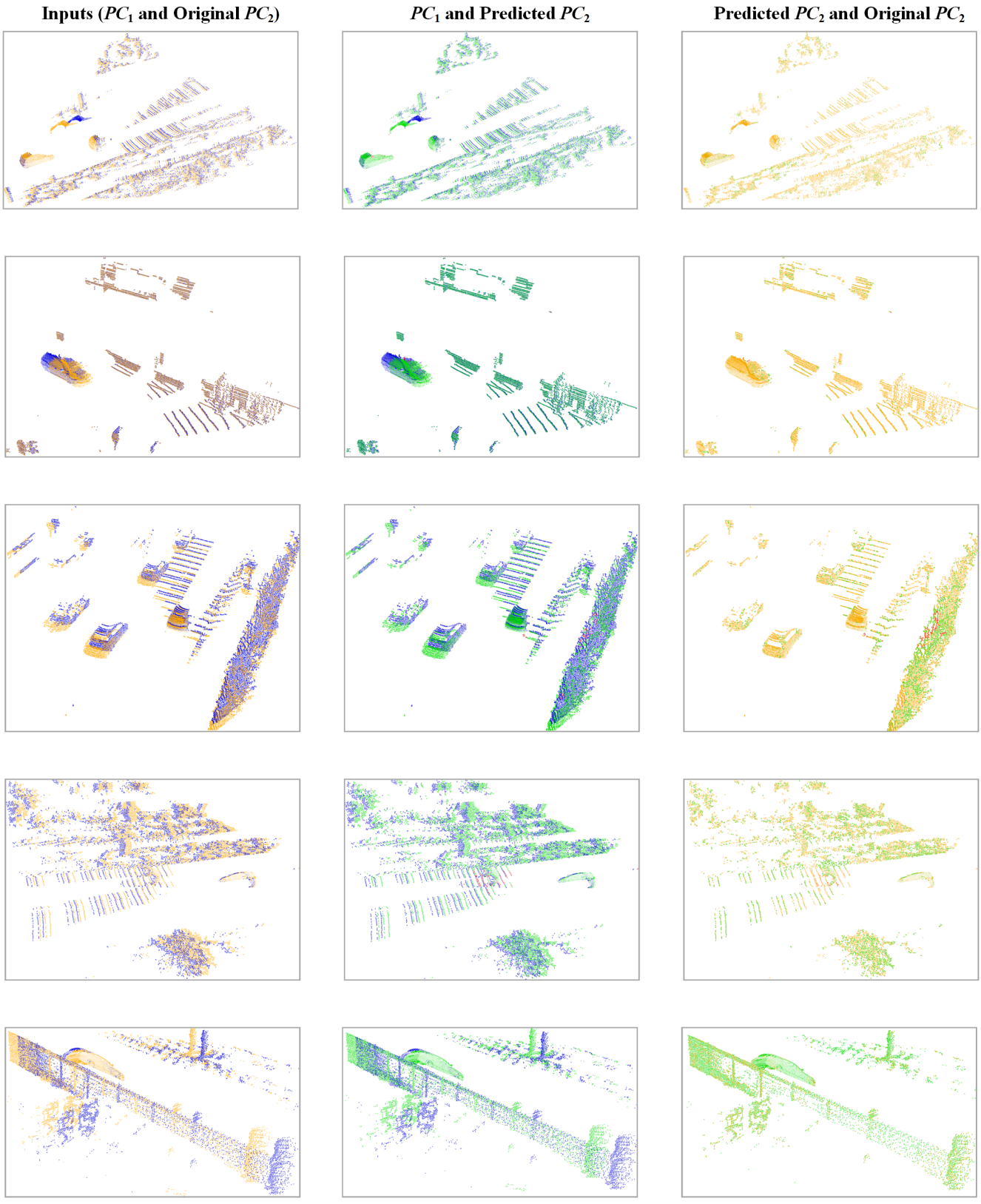

Figure 5. **Visualization of the predicted scene flow on the KITTI dataset.** Blue, Orange, green, red points respectively indicate the first frame PC_1, the second frame PC_2, accurately estimated PC_2 (PC_1 warped by estimated flow), and inaccurately estimated PC_2.