DELFlow: Dense Efficient Learning of Scene Flow for Large-Scale Point Clouds

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

Abstract

Point clouds are naturally sparse, while image pixels are dense. The inconsistency limits feature fusion from both modalities for point-wise scene flow estimation. Previous methods rarely predict scene flow from the entire point clouds of the scene with one-time inference due to the memory inefficiency and heavy overhead from distance calculation and sorting involved in commonly used farthest point sampling, KNN, and ball query algorithms for local feature aggregation. To mitigate these issues in scene flow learning, we regularize raw points to a dense format by storing 3D coordinates in 2D grids. Unlike the sampling operation commonly used in existing works, the dense 2D representation 1) preserves most points in the given scene, 2) brings in a significant boost of efficiency, and 3) eliminates the density gap between points and pixels, allowing us to perform effective feature fusion. We also present a novel warping projection technique to alleviate the information loss problem resulting from the fact that multiple points could be mapped into one grid during projection when computing cost volume. Sufficient experiments demonstrate the efficiency and effectiveness of our method, outperforming the prior-arts on the FlyingThings3D and KITTI dataset. Our source codes will be released on `https://github.com/IRMVLab/DELFlow`.

1. Introduction

Scene flow represents the point-wise motion between a pair of frames, specifically the magnitude and direction of the 3D motion. As a low-level task, 3D scene flow estimation is beneficial to numerous high-level scene understanding tasks in autonomous driving, such as LiDAR odometry [46], object tracking [47], and semantic segmentation [18].

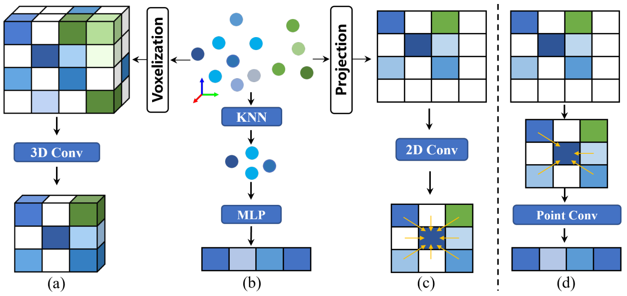

Figure 1. Comparison of current point cloud processing frameworks. (a) 3D grid-based methods. (b) 3D point-based methods. (c) 2D perspective grid-based methods. (d) Our outlier-aware point-based methods, where outliers are filtered out.

Early works [7, 31] convert point cloud into 3D voxel grids and process them with 3D Convolutional Neural Networks (CNNs). However, the computational costs of such voxel-based methods grow drastically with the resolution. Besides voxelization (Fig. 1a), recent studies [9, 17, 51, 43] resort to learning scene flow directly from the raw coordinates of point clouds in 3D space. These point-based methods [17, 51, 45, 40] (Fig. 1b) relies on feature aggregation from neighborhood points with a PointNet [28, 29] structure. Nevertheless, the most commonly used K Nearest Neighbors (KNN) and ball query algorithms in solutions [9, 17, 51, 41] require frequent distance calculation and sorting between point clouds, which are memory-inefficient and time-consuming. Therefore, only a limited number of points are taken as inputs because large-scale input point clouds indicate more GPU memory consumption. The inefficiency led to the proposal of projection-based methods [4, 48, 49], where 2D convolutions are applied to the projected point cloud on image plane. As shown in Fig. 1c, a convolution kernel is used to perform feature extraction among nine neighbors. Such 2D perspective grid-based methods suffer from the neighboring outliers and are not flexible since effective 3D methods cannot be applied.

[*]Corresponding Authors. The first two authors contributed equally.

To deal with the heavy computational cost of 3D methods and the limited accuracy of 2D methods, we propose a projection-based framework for scene flow learning from dense point clouds. We store raw 3D point clouds ($n \times 3$) in the corresponding 2D pixels by projecting them onto image plane ($H \times W \times 3$). Such representation allows us to take the complete point clouds from input scene. For the second stage, we propose *Kernel Based Grouping* to capture 3D geometric information on 2D grid. The operations are conducted on 2D grids, but the output is in the 3D space. Different from 3D point-based methods, we are able to process point clouds with less memory consumption as more prior spatial information from the 2D format reduces the local query complexity. By reducing the grouping range in search of neighboring points, our proposed network achieves single-frame feature extraction and inter-frame correlation for tens of thousands of LiDAR point clouds. In addition, the far-away points can be removed using our outlier-aware method (Fig. 1d), solving the problem of 2D methods caused by outliers.

The second challenge pertains to the feature fusion between point clouds and images, as raw point clouds in 3D space are naturally sparse, unordered and unstructured, while pixels of images are dense and adjacent to each other. CamLiFlow [16] projects a small number of sparse points onto image plane to fuse features from these two modalities. Instead, we take all points as input using the dense format of projected point clouds, which eliminates the density gap between sparse points and dense pixels, enabling effective feature fusion. As a result, we explore the potential of attentive feature fusion between dense points and pixels for more accurate scene flow prediction.

In addition, we adopt a warping operation in cost volume to refine the predicted flow. During the refinement process, multiple warped points can be projected into the same grid on 2D plane. Previous methods merge extra points, but cause information loss which affects the final accuracy. On the contrary, our proposed cost volume module equipped with a novel warping projection technique enables us to avoid such information loss by using the warped coordinates as an intermediate indexing variable.

Overall, our contributions are as follows:

- We propose an efficient scene flow learning framework operated on projected point clouds to process points in 3D space, which takes the entire point clouds as input and predicts flow with one-time inference.
- We present a novel cost volume module with a warping projection technique, allowing us to compute the cost volume without information loss caused by merging points during the refinement process.
- We design a pixel-point feature fusion module, which encodes color information from images to guide the decoding of point-wise motion in point clouds, improving the accuracy of scene flow estimation.

2. Related work

Scene Flow Learning from 3D Data. With the development of deep learning for raw 3D point cloud processing [28, 29], FlowNet3D [17] pioneers in directly processing point clouds and predicts 3D scene flow in an end-to-end fashion. A flow embedding layer is proposed to compute the correlation between a pair of point clouds. PointPWC-Net [51] proposes a patch-to-patch method by considering more than just one point of the first frame during the correlation process, and extends the coarse-to-fine structure from optical flow [33] to scene flow estimation. Inspired by permutohedral lattice [1] and Bilateral Convolutional Layers (BCL) [11], HPLFlowNet [9] proposes DownBCL, UpBCL, and CorrBCL to restore rich information from point clouds. However, the interpolation from points to permutohedral lattice leads to information loss. To assign different weights to the correlated points in a patch, HALFLOW [45] proposes a hierarchical neural network with a double attentive embedding layer. Using optimal transport [25, 26, 37] tools, FLOT [27] achieves competitive performance with much less parameters. The rigidity assumption is widely used in other works [38, 39, 23, 20]. HCRF-Flow [15] formulates the rigidity constraint as a high order term. Built on the architecture of RAFT [35], RAFT-3D [36] utilizes rigid-motion embeddings to group neighbors into rigid objects and refines the 2D flow field with a recurrent structure.

Efficient Learning of Point Clouds. Previous works [7, 31] convert point clouds into voxel grids and process them with 3D CNNs. The information loss happens during voxelization because multiple points can be mapped into one grid, and only one point is kept [34]. Such voxel-based methods are not memory-efficient due to high computational requirements with increasing voxel resolution [19]. Therefore, projection-based methods [4, 48, 49, 50, 52] are proposed to tackle this problem. The 3D point clouds are first projected onto a 2D plane, and then 2D CNNs are applied. DarkNet53 [3] applies a fully convolutional neural network on the projected points after spherical projection. However, projection-based methods usually come with loss of geometry information [34]. Recent point-based methods operate on raw point clouds without a voxelization process. For example, PointNet [28] uses Multi-Layer Perception (MLP) to extract global features directly from raw point clouds. PointNet++ [29] explored the ability to learn local features from neighborhood points. Methods [17, 45, 51] based on PointNet++ are efficient and effective for small-scale point clouds, but the computational cost increases greatly when it comes to large-scale points. YOGO [53] accelerates KNN and ball query algorithms by dividing points into several sub-regions and operating on the tokens extracted from sub-regions. Wang *et al.* [44] proposes a cylindrical projection method in LiDAR odometry.

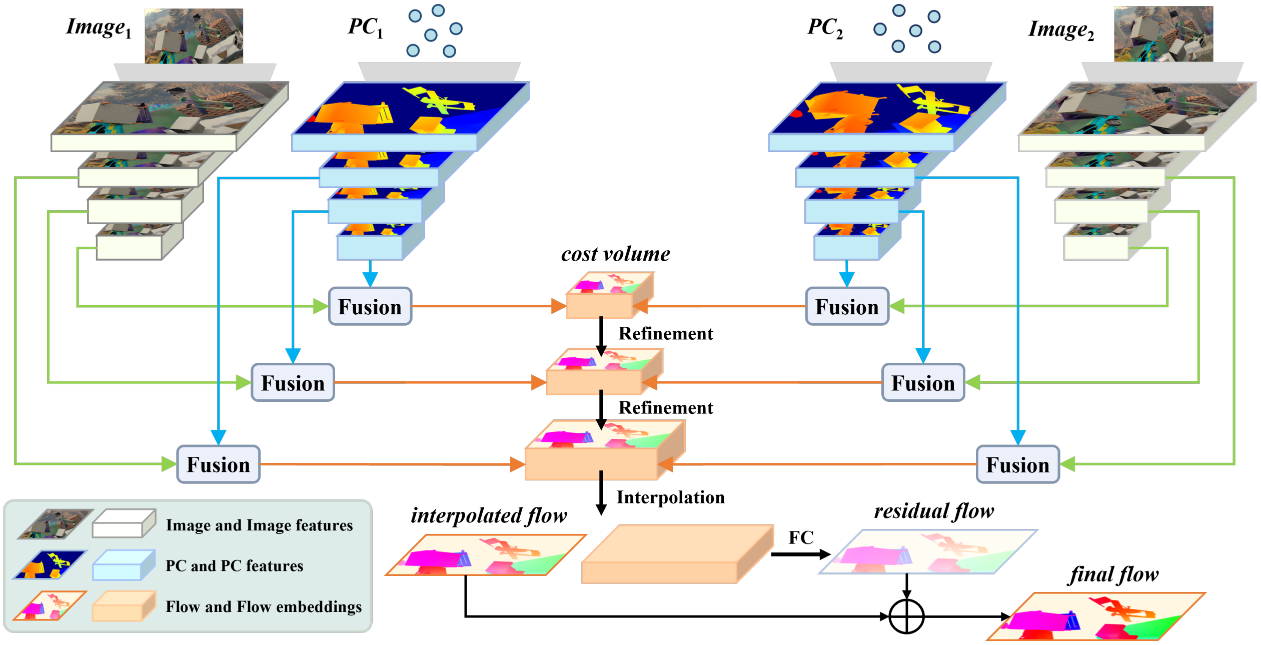

Figure 2. Structure overview of our proposed network. Given a pair of consecutive point clouds and images, down-sampling, correlation, and up-sampling is performed in order. The network predicts flow in a coarse-to-fine manner with the refinement module. Feature fusion between image and point cloud is performed before the refinement process. We adopt the residual scene flow prediction manner, by predicting a residual flow from the flow embeddings, and then adding the residual flow to the coarse flow, generating a refined flow.

3. Efficient Learning of Dense Point Clouds

Let $PC_1 = \{x_i | x_i \in \mathbb{R}^3\}_{i=1}^{N_1}$ and $PC_2 = \{y_j | y_j \in \mathbb{R}^3\}_{j=1}^{N_2}$ represent the point clouds from a pair of consecutive frames, the goal of scene flow estimation is to predict the motion $F = \{\Delta x_i | \Delta x_i \in \mathbb{R}^3\}_{i=1}^{N_1}$ of every point in PC_1. Local feature aggregation from neighboring points is widely used to extract high-level point features. To achieve efficient learning of dense point clouds, we introduce a dense format of point clouds, which brings in more prior spatial information and reduces computational complexity. Moreover, an efficient network with a new cost volume module is constructed to further improve the performance.

3.1. 2D Representation of Dense Point Clouds

The original 3D point clouds ($n \times 3$) are unordered, where n is the number of points. Such representation provides no prior knowledge and requires much effort to structure the irregular data [34]. To avoid unnecessary calculations, a representation of point clouds with more prior spatial information is essential for efficient processing of large-scale point clouds. Inspired by a cylindrical projection technique from [44], we introduce a dense representation of 3D point clouds to scene flow prediction. We conduct pixelization by projecting the raw point clouds (x, y, z) onto the 2D plane (u, v) with a calibration matrix.

Different from previous projection-based methods which contain RGB and depth information in the pixel grids, we fill (u, v) with the 3D coordinates (x, y, z) of each point, generating a $H \times W \times 3$ matrix, where H and W are height and width of the projected image plane. Not every pixel has its corresponding point in 3D space, therefore, we fill empty pixels with $(0, 0, 0)$. Our pixelization technique enables us to preserve complete information from raw point clouds compared to the interpolation in [9] and sampling operation in [17, 51, 45]. It allows us to process the entire LiDAR scans of the scene at once. Our scene flow prediction network takes the entire point clouds as input like an image, but with 3D coordinates instead of RGB value in the corresponding pixels.

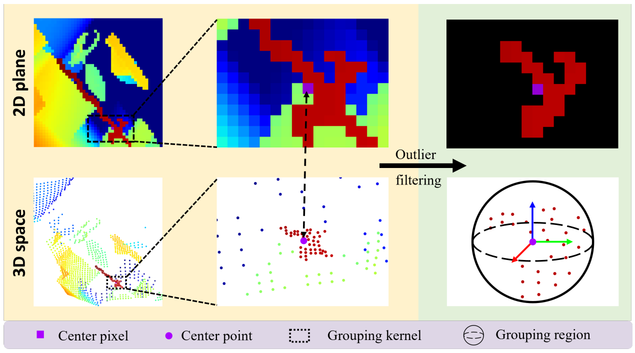

Figure 3. Kernel Based Grouping. Purple pixel denotes the central point $x_i^{'}$, and the black rectangle is the given kernel. Red pixels in the third column are the selected neighbors and far-away outliers are removed during the filtering process.

Instead of applying 2D solutions on the projected point clouds as other projection-based methods [5, 8], our approach addresses the scene flow prediction problem in a 3D manner, yet with operations on the 2D plane, ensuring the efficiency of 2D methods and effectiveness of 3D methods.

3.2. Scene Flow Prediction Network

Following the projection technique in Section 3.1, each point's coordinate is stored in the corresponding projected pixel position, and all points of the scene are taken as input. Since the resolution of point cloud is currently much higher than that of randomly sampling a small number of points, an efficient algorithm to reduce the computational cost is in need. Therefore, we construct an efficient 3D scene flow estimation network based on the 2D representation form as shown in Fig. 2. We adopt a hierarchical structure to predict the scene flow in a coarse-to-fine manner.

Kernel Based Neighbors Grouping: Upon the 2D map of 3D point clouds, a kernel based grouping technique is adopted from [44] to select neighboring points. In most cases, two nearby points in 3D space are also close on 2D plane after projection, which helps filter out many inappropriate neighborhood points without extra calculations.

We implement a CUDA operator to search for neighboring points in 3D space around the center on 2D plane. As shown in Fig. 3, given the central point p_c with corresponding 2D coordinate (u_c, v_c), a 2D kernel is placed around the center. We can easily locate its neighboring points within the kernel on 2D plane. The kernel size for grouping is denoted as $k_s = [k_h \times k_w]$, so we only need to search the points within range: $[u_c - k_w/2 : u_c + k_w/2, v - k_h/2 : v + k_h/2]$ instead of the whole point clouds. This technique enables us to reduce the searching complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \cdot k_s)$. Unlike traditional 2D kernels that operate on projected points directly, our approach allows us to remove far-away outliers that could be harmful to feature extraction. Then, the features of valid neighboring pixels are aggregated to the center. The grouping and filtering operations are conducted on 2D plane, but the feature extraction process happens in 3D space. As a result, our proposed method shares the efficiency of 2D methods and accuracy of 3D methods. It is noteworthy that larger kernel indicates more searching time and reducing the kernel size results in a lower latency. However, if the kernel size is too small, the algorithm cannot find enough appropriate neighboring points. Through experiments, we find that an appropriate kernel size is usually between $3K \sim 4K$ where K is the number of neighbors to select.

Point Cloud Encoder: The input point clouds are stored in a form of image, with the size of $h \times w \times 3$. For high-level feature learning, we attempt to aggregate local features from neighboring points to the central point using MLP.

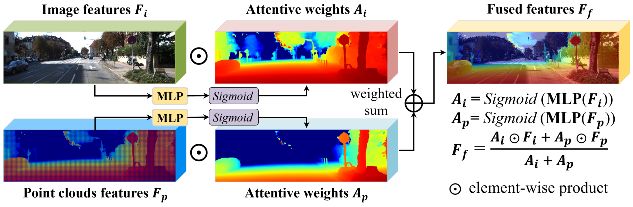
Image features F_i
Attentive weights A_i
Fused features F_f
MLP
Sigmoid
MLP
Sigmoid
weighted sum
Point clouds features F_p
Attentive weights A_p
$A_i = Sigmoid\,(\mathbf{MLP}(F_i))$
$A_p = Sigmoid\,(\mathbf{MLP}(F_p))$
$F_f = \frac{A_i \odot F_i + A_p \odot F_p}{A_i + A_p}$
⊙ element-wise product

Figure 4. Feature fusion between image and point cloud features. We utilize a self-attention mechanism to align 2D and 3D features.

Given the raw point sets $P = \{x_i | x_i \in \mathbb{R}^3\}_{i=1}^{h \times w}$, we first select a number of central points $P_c = \{x_i^{'} | x_i^{'} \in \mathbb{R}^3\}_{i=1}^{h^{'} \times w^{'}}$ by setting a fixed stride. The corresponding features of P and P_c are denoted as $\{f_i | f_i \in \mathbb{R}^C\}_{i=1}^{h \times w}$ and $\{f_i^{'} | f_i^{'} \in \mathbb{R}^C\}_{i=1}^{h^{'} \times w^{'}}$. We use kernel based grouping method to select K neighbors for each point $x_i^{'}$. Then, MLP is used to extract the neighboring points' features $\{(x_{ik}^{'}, f_{ik}^{'}) | x_{ik}^{'} \in \mathbb{R}^3, f_{ik}^{'} \in \mathbb{R}^C\}_{k=1}^{K}$, generating high-level features:

$$h_i^{'} = \underset{k}{MaxPool}\,\{\mathbf{MLP}((x_{ik}^{'} - x_i^{'}) \oplus f_{ik}^{'})\}. \tag{1}$$

Multi-modal Feature Fusion: Given the image features $F_i \in \mathbb{R}^{H \times W \times C_1}$ and point cloud features $F_p \in \mathbb{R}^{H \times W \times C_2}$, we utilize a self-attention mechanism to perform feature fusion between two modalities to improve the accuracy of scene flow prediction as shown in Fig. 4. Since the point cloud and image share the same size of $H \times W$, each image pixel corresponds to a point and vice versa. Therefore, it is straightforward to integrate color information from 2D images to guide the prediction of 3D point-wise motion with the feature fusion module.

Attentive Feature Correlation of Point Clouds: After obtaining the high-level features of PC_1 and PC_2, we need to compute their motion correlation. We use the attentive cost volume in [45] to learn the hidden motion pattern between two consecutive frames of point clouds.

Given point clouds $PC_1 = \{(x_i, f_i) | x_i \in \mathbb{R}^3, f_i \in \mathbb{R}^C\}_{i=1}^{N_1}$ and $PC_2 = \{(y_j, g_j) | y_j \in \mathbb{R}^3, g_j \in \mathbb{R}^C\}_{j=1}^{N_2}$, the features of neighboring PC_2 are first aggregated to the centering PC_1 via attention mechanism and each neighbor is weighed differently. In the second stage, the features of aggregated PC_1 will be updated again by aggregating the neighboring PC_1, generating the attentive flow embedding features $E = \{(x_i, e_i) | x_i \in \mathbb{R}^3, e_i \in \mathbb{R}^{C^{'}}\}_{i=1}^{N_1}$. We reimplement the double attentive flow embedding layers under 2D representation to accelerate the correlation between two frames of projected point clouds.

Scene Flow Predictor: The scene flow predictor takes three inputs: warped PC_1 features, attentive flow embedding features and the upsampled coarse dense flow embedding features. Following [51, 45, 9], these three inputs are concatenated together and then processed by a shared MLP, generating the refined flow embedding features.

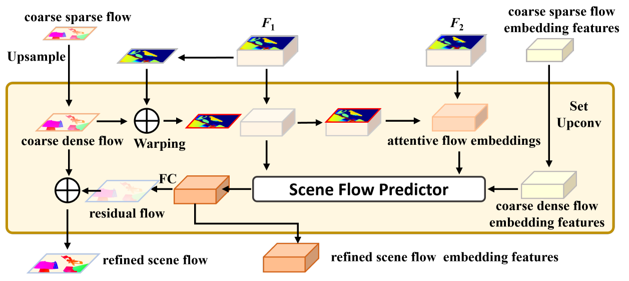
coarse sparse flow
F_1
F_2
coarse sparse flow embedding features
Upsample
Set Upconv
coarse dense flow
Warping
attentive flow embeddings
FC
Scene Flow Predictor
residual flow
coarse dense flow embedding features
refined scene flow
refined scene flow embedding features

Figure 5. Residual refinement module. Set Upconv denotes up-sampling layers with learnable parameters.

Hierarchical Flow Refinement: We adopt the pyramid structure for supervised learning of scene flow estimation. Both the low-level and high-level scene flow is used to compute the training loss. We first predict a flow with the least number of points from the highest-level features. Then level by level, scene flow with higher resolution is predicted through up-sampling and refinement modules. For up-sampling, similar operations in equation (1) are performed among low-resolution points for high-resolution points to obtain neighboring features $b_i^{'}$, which are then concatenated with the fused features $s_i^{'}$ of central points. A shared MLP is used to generate up-sampled features.

After up-sampling operation, we obtain a coarse flow embeddings of higher-resolution points. Next, a warping operation in Section 3.3 is adopted to generate the flow re-embedding features. As shown in Fig. 5, residual scene flow can be predicted by the scene flow predictor. Finally, we obtain a refined flow of higher accuracy by adding the estimated residual flow to the coarse flow.

3.3. A New Cost Volume with Warping Projection

Assuming that $p \in \mathbb{R}^3$ of PC_1 flows to $q \in \mathbb{R}^3$ of PC_2, their corresponding 2D coordinates are (u_1, v_1) and (u_2, v_2). When there is a large motion of p across frames, (u_1, v_1) is far away from (u_2, v_2). This requires a large searching kernel because we aim to correctly locate the corresponding (u_2, v_2) in the bounding kernel around (u_1, v_1). However, increasing kernel size leads to longer searching time and heavier computational cost.

To address the issue, we incorporate the warping idea by adding the predicted coarse flow F_c to PC_1, which generates $PC_{1w} = \{\hat{p} | \hat{p} \in \mathbb{R}^3\}_{i=1}^N$. The warped points PC_{1w} are closer to PC_2 than the original PC_1. Next, the cost volume is computed between PC_{1w} and PC_2 to predict a residual flow F_Δ. By iteratively refining the predicted flow, the warped points become progressively closer to PC_2. We can obtain a more accurate flow F_a by adding the residual flow to the coarse flow level by level.

$$F_a = F_c + \sum_{i=1}^{l} F_{\Delta i} \tag{2}$$

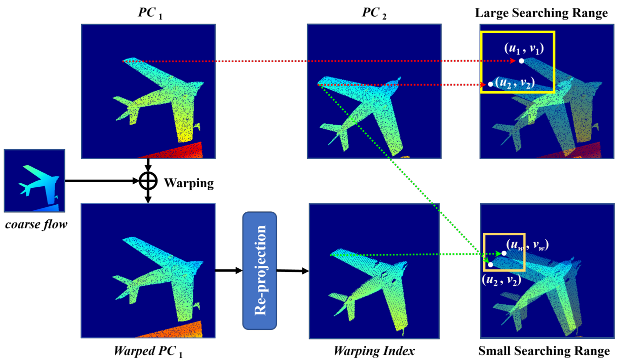

Figure 6. Cost volume computation with warping projection technique. The searching range can be reduced significantly.

In our setting, p and q are stored in a $h \times w \times 3$ matrix, located at their corresponding 2D grid (u_1, v_1) and (u_2, v_2) respectively. The warped points are obtained by adding the coarse flow F_c $(h \times w \times 3)$ to PC_1. In this way, $\hat{p} = p + f_c$ is stored in (u_1, v_1) rather than their corresponding 2D grid (u_w, v_w). We present a new cost volume module equipped with a warping projection technique to deal with this problem. As illustrated in Fig. 6, we create a $h \times w \times 2$ matrix to store the warping index (u_w, v_w) of every warped point in PC_{1w}. (u_w, v_w) acts as an indexing variable between (u_1, v_1) and (u_2, v_2) during the calculation of cost volume. For each warped point in PC_{1w} located at (u_1, v_1), a kernel is placed around the corresponding indexing position (u_w, v_w) on PC_2. Within the kernel, the distances between PC_{1w} and PC_2 are calculated, and distant PC_2 points are filtered out while K neighbors are retained.

During the computation of cost volume, each warped point has a corresponding 2D position due to our warping projection method. If we directly re-project PC_{1w} to get (u_w, v_w) and store them in the corresponding pixel, multiple points in PC_{1w} could be mapped into the same grid and extra points would be merged. Our technique effectively avoids the loss of points resulted from merging operations. As a result, the complete point could be preserved. Our designed cost volume module also eliminates the need for a large kernel, thereby ensuring the processing efficiency.

4. Experiments

4.1. FlyingThings3D

Data Preprocessing: FlyingThings3D dataset is a synthetic dataset, consisting of 19640 training samples and 3824 testing samples. Following the preprocessing methods described in [51, 9, 45], the point clouds and ground truth scene flow are constructed from the depth map and optical flow. We remove points with depth exceeding $35m$ as [51, 9, 45]. We first train the model on a quarter of FlyingThings3D dataset to reduce training time, and then finetune the model on the complete dataset.

Method	Input	2D Metrics		3D Metrics		
		EPE_{2D}	ACC_{1px}	EPE_{3D}	$\text{ACC}_{0.05}$	$\text{ACC}_{0.10}$
FlowNet2.0 [10]	Image	5.05	72.8 %	-	-	-
PWC-Net [33]	Image	6.55	64.3 %	-	-	-
RAFT [35]	Image	3.12	81.1 %	-	-	-
FlowNet3D [17]	LiDAR	-	-	0.169	25.4 %	57.9 %
PointPWC-Net [51]	LiDAR	-	-	0.132	44.3 %	67.4 %
FLOT [27]	LiDAR	-	-	0.156	34.3 %	64.3 %
RAFT-3D [36]	Image+Depth	2.37	87.1 %	0.094	80.6 %	-
CamLiFlow [16]	Image+LiDAR	2.20	87.3 %	0.061	85.6 %	91.9 %
Ours	Image+LiDAR	**2.02**	85.9 %	**0.058**	**86.7 %**	**93.2 %**

Table 1. Quantitative results compared with recent methods on the FlyingThings3D dataset [21]. The performances are evaluated on all points (including occluded points and non-occluded points).

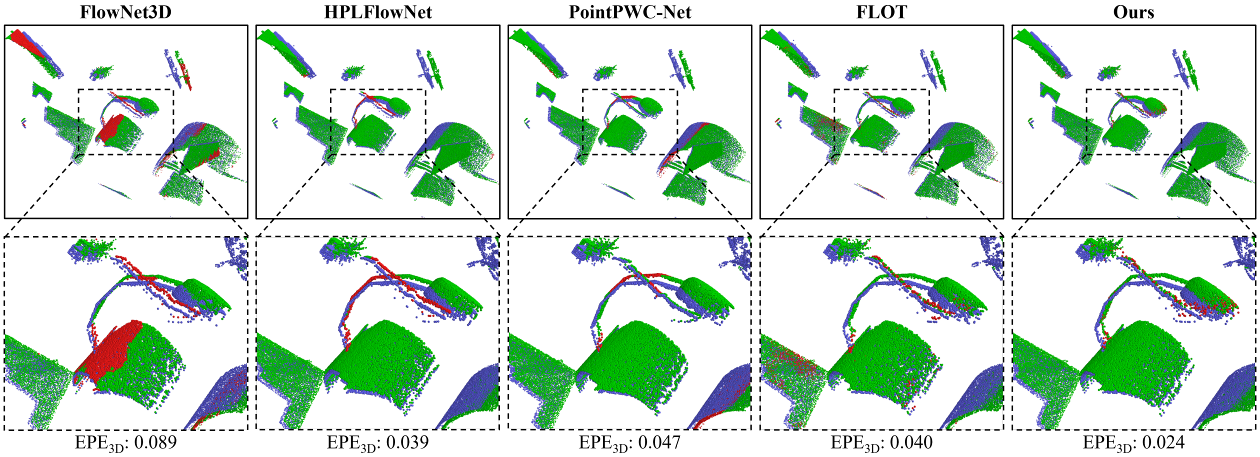
FlowNet3D
HPLFlowNet
PointPWC-Net
FLOT
Ours
EPE_{3D}: 0.089
EPE_{3D}: 0.039
EPE_{3D}: 0.047
EPE_{3D}: 0.040
EPE_{3D}: 0.024

Figure 7. Qualitative visualization results of scene flow estimation. We compare our methods with FlowNet3D [17], HPLFlowNet [9] and PointPWC-Net [51] and FLOT [27]. Blue points represent PC_1. Green points represent accurate predictions and red points represent inaccurate predictions. (Accuracy is measured using $\text{ACC}_{0.10}$.)

Evaluation Metrics: For a fair comparison, we adopt the following evaluation metrics as [17, 51, 9, 42]: EPE_{3D}, $\text{ACC}_{.05}$, $\text{ACC}_{0.10}$ for 3D scene flow evaluation and EPE_{2D}, ACC_{1px} for 2D optical flow evaluation.

Quantitative results: The quantitative results on FlyingThings3D dataset are listed in Tab. 1. We compare our method with other baselines [10, 33, 35, 17, 51, 27, 32, 36, 16] in terms of the evaluation metrics described above. Results show that our method demonstrates comparable performance both in 2D metrics and 3D metrics.

Qualitative results: To demonstrate the effectiveness of our method for 3D scene flow prediction, we compare the performance of our model with other point-based methods [17, 9, 51, 27] on FlyingThings3D dataset. The qualitative results shown in Fig. 7 show that our method achieves scene flow prediction with least errors.

4.2. KITTI Scene Flow 2015

Training: KITTI Scene Flow 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.

Evaluation: The dense depth maps are obtained from LEAStereo [6] for testing. We reconstruct point clouds from the estimated depth with the calibration parameters. We submit our results to the official KITTI website, which adopts the percentage of outliers as evaluation metrics.

Comparison with State-of-the-Arts: The quantitative results on KITTI Scene Flow dataset are listed in Tab. 2, which show that our method outperforms prior-arts. The visualization results are illustrated in Fig. 8.

4.3. Implementation Details

All experiments are conducted on NVIDIA Quadro RTX 8000 GPU with PyTorch 1.10.0. We use Adam [13] as our optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. The initial learning rate of 0.001 decays exponentially with a decaying rate $\gamma = 0.8$, and the decaying step is 80. Our designed network adopts the hierarchical structure, where both low-level and high-level flow are predicted. The optical flow and scene

Method	Input	Disparity 1			Disparity 2			Optical Flow			Scene Flow		
		bg	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all
SSF [30]	Stereo	3.55	8.75	4.42	4.94	17.48	7.02	5.63	14.71	7.14	7.18	24.58	10.07
Sense [12]	Stereo	2.07	3.01	2.22	4.90	10.83	5.89	7.30	9.33	7.64	8.36	15.49	9.55
PRSM [39]	Stereo	3.02	10.52	4.27	5.13	15.11	6.79	5.33	13.40	6.68	6.61	20.79	8.97
ACOSF [14]	Stereo	2.79	7.56	3.58	3.82	12.74	5.31	4.56	12.00	5.79	5.61	19.38	7.90
DRISF [20]	Stereo	2.16	4.49	2.55	2.90	9.73	4.04	3.59	10.40	4.73	4.39	15.94	6.31
M-FUSE [22]	Stereo	**1.40**	**2.91**	**1.65**	2.14	8.10	3.13	2.66	7.47	3.46	3.43	11.84	4.83
OpticalExp [54]	Mono + LiDAR	1.48	3.46	1.81	3.39	8.54	4.25	5.83	8.66	6.30	7.06	13.44	8.12
ISF [2]	Stereo + LiDAR	4.12	6.17	4.46	4.88	11.34	5.95	5.40	10.29	6.22	6.58	15.63	8.08
RigidMask+ISF [55]	Stereo + LiDAR	1.53	3.65	1.89	2.09	8.92	3.23	2.63	7.85	3.50	3.25	13.08	4.89
RAFT-3D [36]	Mono + Depth	1.48	3.46	1.81	2.51	9.46	3.67	3.39	8.79	4.29	4.27	13.27	5.77
CamLiFlow [16]	Mono + LiDAR	1.48	3.46	1.81	1.92	8.14	2.95	2.31	**7.04**	3.10	2.87	12.23	4.43
Ours	Mono + LiDAR	**1.40**	**2.91**	**1.65**	**1.90**	**7.50**	**2.84**	**2.27**	7.10	**3.07**	**2.87**	**11.69**	**4.34**

Table 2. Quantitative results compared with recent methods on KITTI Scene Flow dataset [24] evaluated on all pixels (including occluded pixels). In the table, bg and fg denote the background and foreground, respectively. A lower value indicates better performance.

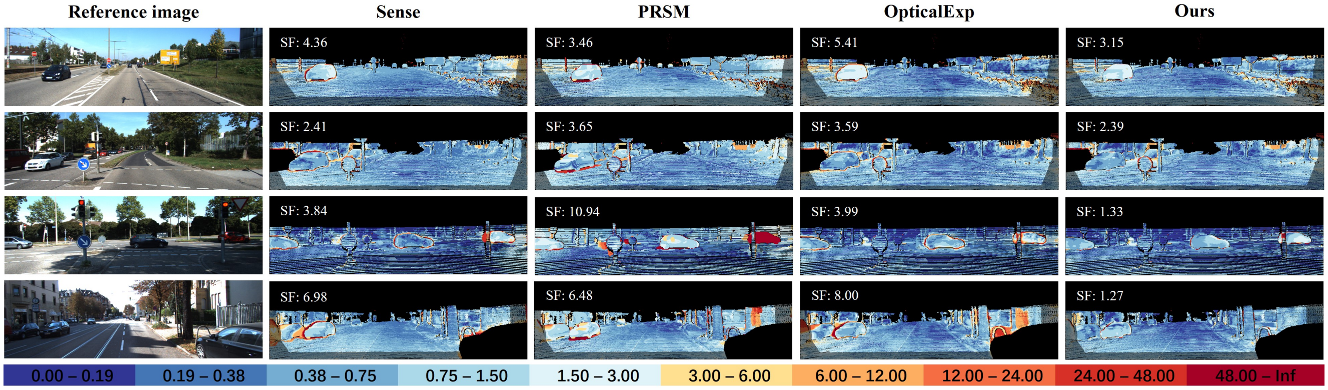

Figure 8. Qualitative visualization results of scene flow estimation on testing KITTI dataset.

flow are predicted similarly, both in a pyramid, warping, and cost volume (PWC) structure as Fig. 2. Therefore, we adopt a joint multi-scale supervised manner to train the model by adding optical flow loss and scene flow loss. Let $F^l = \{f_i^l | f_i^l \in \mathbb{R}^3\}_{i=1}^{N_l}$ be the predicted flow at level l, and $\hat{F}^l = \{\hat{f}_i^l | \hat{f}_i^l \in \mathbb{R}^3\}_{i=1}^{N_l}$ be the corresponding ground truth flow. The training loss of optical flow and scene flow is calculated in the same way as follows:

$$loss = \sum_{l=1}^{4} w_l \frac{1}{N_l} \sum_{i=1}^{N_l} \left\| \hat{f}_i^l - f_i^l \right\|_2 , \tag{3}$$

where N_l denotes the number of valid points or pixels at level l. w_l denotes the weighting factor of losses at different levels, and $w_1 = 0.1, w_2 = 0.2, w_3 = 0.3, w_4 = 0.8$.

4.4. Ablation Study

We conduct a series of ablation studies to verify the effectiveness of each component of our proposed network on the validation set of FlyingThings3D dataset.

Dense or Sparse Representation: We compare the performances of our method with or without 2D dense representation of point clouds. For experiments without the 2D dense representation, following [51, 45], we randomly select 8192 points and take them as input for the neural network. As 2D representation allows us to input more information, the model using dense representation outperforms the sparse 3D representation as shown in Tab. 3 (a).

Warping Projection in Cost Volume: For experiments without our proposed warping projection technique, we project directly the PC_{1w} onto 2D plane and merge multiple projected points in the same grid. The flow embedding features for merged points are obtained from interpolation. Results in Tab. 3 (b) demonstrate that the warping projection technique is useful in avoiding information loss and thus improve the model's performance.

Attentive Feature Fusion: For experiments without feature fusion, the network takes only point clouds as input. Non-attentive feature fusion simply concatenates the image features and point cloud features. Our feature fusion approach utilizes an attention mechanism to aggregate features in a pixel-point manner. We find that feature fusion improves prediction accuracy, and the attention-based fusion reports the best performance, as shown in Tab. 3 (c).

Kernel Based Grouping: For experiments without kernel based grouping, the network use KNN algorithm to group

	Method	2D Metrics		3D Metrics			
		EPE_{2D}	ACC_{1px}	EPE_{3D}	$\text{ACC}_{0.05}$	$\text{ACC}_{0.10}$	Outliers
(a)	Ours (w/o projection, only 8192 input points)	2.23	82.2 %	0.061	84.3 %	92.7 %	25.0 %
	Ours (full, with projection, all points)	**2.02**	**85.9 %**	**0.058**	**86.7 %**	**93.2 %**	**20.8 %**
(b)	Ours (w/o warping operation)	12.36	32.4 %	0.433	5.8 %	20.0 %	97.6 %
	Ours (w/o warping projection in cost volume)	4.53	74.9 %	0.105	49.9 %	80.9 %	57.5 %
	Ours (full, with warping projection)	**2.02**	**85.9 %**	**0.058**	**86.7 %**	**93.2 %**	**20.8 %**
(c)	Ours (w/o feature fusion)	4.32	76.8 %	0.081	69.5 %	87.6 %	40.1 %
	Ours (with concatenation feature fusion)	3.26	81.9 %	0.064	85.4 %	92.6 %	21.5 %
	Ours (full, with attentive feature fusion)	**2.02**	**85.9 %**	**0.058**	**86.7 %**	**93.2 %**	**20.8 %**
(d)	Ours (w/o kernel-based grouping)	2.09	84.2 %	**0.056**	85.9 %	**93.5 %**	**19.9 %**
	Ours (full, with kernel-based grouping)	**2.02**	**85.9 %**	0.058	**86.7 %**	93.2 %	20.8 %

Table 3. The ablation study results of 3D scene flow learning on FlyingThings3D dataset [21].

neighbors among all points. As shown in Tab. 3 (d), similar or even better performance is obtained because the searching range includes all points. However, the searching process becomes computationally expensive. Therefore, we compare the latency of our model with and without the kernel based grouping technique. The results in Tab. 4 show that the efficiency drops by a factor of 20.

4.5. Analysis

The experimental results demonstrate that our proposed method achieves comparable performance on both FlyingThings3D and KITTI dataset. We attribute this improvement to three key components of our approach: First, our approach benefits from the dense representation. Previous methods often take a sparse set of points as input, typically 8192 points, which is much less than the total points in a scene. This downsampling operation leads to a loss of details and may result in an incomplete representation of the scene. In contrast, we store unordered 3D points in regular 2D grids. The dense representation provides a more precise structure of the scene, allowing us to preserve complete information without losing points. Second, the proposed cost volume with the warping projection technique significantly reduces the size of the search kernel, thus improving the computational efficiency. Unlike previous warping methods that directly project the warped points onto 2D plane and merge extra points projected into the same grid, our new warping projection technique enables us to avoid such information loss by storing the corresponding warped 2D coordinates as intermediate indexes. Third, the multi-modal feature fusion further improves the motion prediction. The integration of image and point cloud is advantageous as they are complementary to each other. Images contain rich color information, while point clouds provide abundant 3D geometrical information. Our attentive feature fusion between these modalities contributes to a more precise and distinctive representation of points. Overall, our approach leverages more input data and reduces information loss to improve the prediction accuracy of point-wise motion.

Methods	Input point number	Size (MB)	Time (ms)
HPLFlowNet [9]	8192	231.8	93.67
HPLFlowNet [9]	49152	231.8	132.58
PointPWC-Net [51]	8192	30.1	100.05
PointPWC-Net [51]	49152	30.1	2643.61
Ours (w/o KBG)	56269 (480 × 270)	**20.1**	1324.29
Ours	56269 (480 × 270)	**20.1**	**52.91**

Table 4. Comparison of model size and running time. KBG denotes the Kernel Based Grouping technique.

Latency: We further evaluate the performance of our model by comparing the running time and size with other methods. As shown in Tab. 4, our framework outperforms others in terms of efficiency. Since the number of valid points may vary across frames, we calculate the average number of valid points in the validation split of the dataset.

Limitations: The projection-based framework requires the input point cloud to be 2.5D format, otherwise multiple points could be mapped into the same grid. In the future, we plan to introduce hash map to store multiple points to address this issue. Currently, our method is applicable to FlyingThings3D and KITTI dataset since the point clouds are converted from depth map. Considering the surface-scanning characteristic of LiDAR sensor, our method is still practical in autonomous driving.

5. Conclusion

In this paper, we propose DELFlow, allowing us to take the entire point clouds of a scene as input and efficiently estimate scene flow at once. To preserve the complete information of raw point clouds without losing details, we introduce a cost volume module based on warping projection. Besides, the attentive feature fusion from images and point clouds further improves the accuracy. By leveraging more data input and reducing information loss, our approach outperforms prior-arts in efficiency and effectiveness.

Acknowledgement. This work was supported in part by the Natural Science Foundation of China under Grant 62225309, 62073222, U21A20480 and U1913204.

References

[1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer graphics forum*, volume 29, pages 753–762. Wiley Online Library, 2010. 2

[2] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *ICCV*, pages 2574–2583, 2017. 7

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, pages 9297–9307, 2019. 2

[4] A. Boulch, B. Le Saux, and N. Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. In *Proceedings of the Workshop on 3D Object Retrieval*, page 17–24. Eurographics Association, 2017. 1, 2

[5] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the point: Efficient 3d object detection in the range image with graph convolution kernels. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 16000–16009, 2021. 4

[6] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *Proc. Adv. Neural Inform. Process. Syst.*, pages 22158–22169, 2020. 6

[7] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 1, 2

[8] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proc. Conf. Comput. Vis.*, pages 2918–2927, 2021. 4

[9] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 3254–3263, 2019. 1, 2, 3, 4, 5, 6, 8

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, July 2017. 6

[11] Varun Jampani, Martin Kiefel, and Peter V Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 4452–4461, 2016. 2

[12] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik Learned-Miller, and Jan Kautz. Sense: A shared encoder network for scene-flow estimation. In *ICCV*, pages 3195–3204, 2019. 7

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015. 6

[14] Congcong Li, Haoyu Ma, and Qingmin Liao. Two-stage adaptive object scene flow using hybrid cnn-crf model. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3876–3883. IEEE, 2021. 7

[15] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 364–373, 2021. 2

[16] 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, 6, 7

[17] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 529–537, 2019. 1, 2, 3, 6

[18] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, pages 9246–9255, 2019. 1

[19] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[20] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 3614–3622, 2019. 2, 7

[21] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 4040–4048, 2016. 6, 8

[22] Lukas Mehl, Azin Jahedi, Jenny Schmalfuss, and Andrés Bruhn. M-fuse: Multi-frame fusion for scene flow estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2020–2029, 2023. 7

[23] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 3061–3070, 2015. 2

[24] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 7

[25] Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. Got: an optimal transport framework for graph comparison. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[26] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR, 2016. 2

[27] 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. 2, 6

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 652–660, 2017. 1, 2

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 5099–5108, 2017. 1, 2

[30] Zhile Ren, Deqing Sun, Jan Kautz, and Erik Sudderth. Cascaded scene flow prediction using semantic segmentation. In *2017 International Conference on 3D Vision (3DV)*, pages 225–233. IEEE, 2017. 7

[31] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 3577–3586, 2017. 1, 2

[32] Rishav Rishav, Ramy Battrawy, René Schuster, Oliver Wasenmüller, and Didier Stricker. Deeplidarflow: A deep learning architecture for scene flow estimation using monocular camera and sparse lidar. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10460–10467. IEEE, 2020. 6

[33] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 8934–8943, 2018. 2, 6

[34] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. 2, 3

[35] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 2, 6

[36] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 8375–8384, 2021. 2, 6, 7

[37] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019. 2

[38] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *ICCV*, pages 1377–1384, 2013. 2

[39] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *IJCV*, 115(1):1–28, 2015. 2, 7

[40] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What matters for 3d scene flow network. In *European Conference on Computer Vision*, pages 38–55. Springer, 2022. 1

[41] Guangming Wang, Yunzhe Hu, Xinrui Wu, and Hesheng Wang. Residual 3-d scene flow learning with context-aware feature extraction. *IEEE Transactions on Instrumentation and Measurement*, 71:1–9, 2022. 1

[42] 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. 6

[43] Guangming Wang, Xiaoyu Tian, Ruiqi Ding, and Hesheng Wang. Unsupervised learning of 3d scene flow from monocular camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4325–4331. IEEE, 2021. 1

[44] Guangming Wang, Xinrui Wu, Shuyang Jiang, Zhe Liu, and Hesheng Wang. Efficient 3d deep lidar odometry. *PAMI*, 2021. 2, 3, 4

[45] 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. 1, 2, 3, 4, 5, 7

[46] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Pwclo-net: Deep lidar odometry in 3d point clouds using hierarchical embedding mask optimization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 15910–15919, 2021. 1

[47] Sukai Wang, Yuxiang Sun, Chengju Liu, and Ming Liu. Pointtracknet: An end-to-end network for 3-d object detection and tracking from point clouds. *IEEE Robotics and Automation Letters*, 5(2):3206–3212, 2020. 1

[48] Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1–6. IEEE, 2018. 1, 2

[49] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 1, 2

[50] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019. 2

[51] 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. In *ECCV*, 2019. 1, 2, 3, 4, 5, 6, 7, 8

[52] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020. 2

[53] Chenfeng Xu, Bohan Zhai, Bichen Wu, Tian Li, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. You only group once: Efficient point-cloud processing with token representation and relation inference module. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4589–4596. IEEE, 2021. 2

[54] Gengshan Yang and Deva Ramanan. Upgrading optical flow to 3d scene flow through optical expansion. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 1334–1343, 2020. 7

[55] Gengshan Yang and Deva Ramanan. Learning to segment rigid motions from two frames. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 1266–1275, 2021. 7