FH-Net: A Fast Hierarchical Network for Scene Flow Estimation on Real-world Point Clouds

Lihe Ding*, Shaocong Dong*, Tingfa Xu†, Xinli Xu, Jie Wang, and Jianan Li†

Beijing Institute of Technology
{dean.dinglihe,dscdyc1010295799,xxlbigbrother,
jwang123bit}@gmail.com, {lijianan,ciom_xtf1}@bit.edu.cn

Abstract Estimating scene flow from real-world point clouds is a fundamental task for practical 3D vision. Previous methods often rely on deep models to first extract expensive per-point features at full resolution, and then get the flow either from complex matching mechanism or feature decoding, suffering high computational cost and latency. In this work, we propose a fast hierarchical network, FH-Net, which directly gets the key points flow through a lightweight Trans-flow layer utilizing the reliable local geometry prior, and optionally back-propagates the computed sparse flows through an inverse Trans-up layer to obtain hierarchical flows at different resolutions. To focus more on challenging dynamic objects, we also provide a new copy-and-paste data augmentation technique based on dynamic object pairs generation. Moreover, to alleviate the chronic shortage of real-world training data, we establish two new large-scale datasets to this field by collecting lidar-scanned point clouds from public autonomous driving datasets and annotating the collected data through novel pseudo-labeling. Extensive experiments on both public and proposed datasets show that our method outperforms prior state-of-the-arts while running at least $\mathbf{7}\times$ faster at $\mathbf{113}$ FPS. Code and data are released at https://github.com/pigtigger/FH-Net.

Keywords: Scene flow, Real-world point cloud, Transformer, Copy-and-paste

1 Introduction

Scene flow estimation from point clouds, which accurately measures point movement between consecutive frames, serves as an fundamental step for downstream tasks like 3D motion segmentation [32] and tracking [45], and thus plays an increasingly important role in robotics [3] and autonomous driving [46]. Previous works [18,10] for scene flow estimation, however, mainly focus on synthetic datasets [20] with CAD models and pseudo lidar points [21] obtained from disparity images where point clouds are dense and under direct correspondence, totally different from the lidar-scanned and leading to catastrophic performance when move to real world.

Conventional scene flow estimation paradigm Fig. 1a [18,10] uses a U-Net [31] style deep model to obtain per-point fused feature from two consecutive frames at full resolution, and then decode flow from the fused feature, one other line of works Fig. 1b [27] try to extract point feature without temporal fusion and apply complex feature

* Equal contribution. †Correspondence to: Jianan Li and Tingfa Xu.

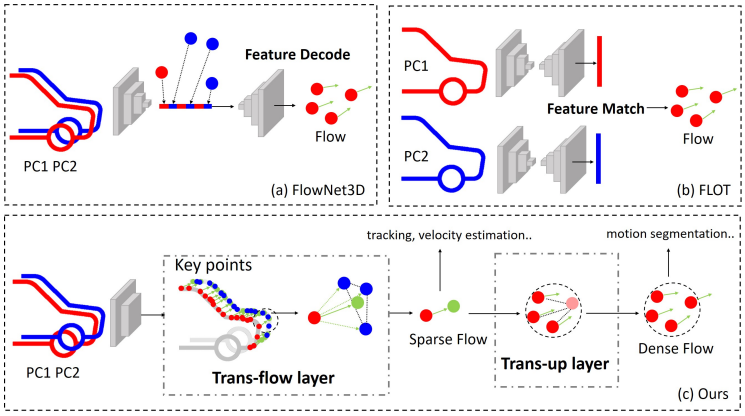

Figure 1. Most previous works (a) fuse two consecutive point clouds in deep feature space from which to decode the flow, or (b) match the features extracted from two point cloud separately by a deep network to predict the flow. Our approach (c) directly computes key points flow (green) by aggregating the spatial offsets from the source point (red) to its neighboring target points (blue) via the well-designed local relationship modeling then use trans-up layer to get the hierarchical flow.

matching algorithm to compute the flow. However, both of these methods need to get the full resolution deep point feature at first then predict the flow, which not only can not well meet the various demand of downstream tasks (i.e., velocity estimation needs real-time instance-level flow while motion segmentation needs higher resolution flow) but also brings high computational cost and latency thus hard to be deployed on real-world autonomous vehicle and robots. To deal with these problems, we propose a fast hierarchical network Fig. 1c, FH-Net, which can output different resolution flows from different layers while running much faster than real-time.

Specifically, we first extract the key points' geometric features for each frame through shallow hierarchical set abstraction layers [28], then utilize a novel transformer-based Trans-flow layer to directly compute key points' flow by applying point-wise attention in the local region between two consecutive frames. We take the offset from source point to every target point as the so-called value while get query and key from the feature of source and target point respectively. Followed by a well-designed subtraction relation mechanism, we can obtain the source key point flow directly by leveraging the strong guide from local relative position instead of decoding from deep feature or applying expensive feature matching mechanism after fully up-sampling like previous methods as shown in Fig. 1.

Through Trans-flow layer, we can get the sparse keypoint flows which well meet the need of object level flow in scenarios like 3D tracking. However, dense prediction tasks like segmentation, requires more fine-grained flows at higher resolution. So optionally, we need to up-sample the predicted flow from Trans-flow layer. A trial solution is interpolation up-sampling, which however only weights flow by distance and suffers large propagation error when close points have different flow. To address this problem, we propose Trans-up layer sharing the same mechanism as Trans-flow layer to up sample

the flow. For arbitrary high resolution point, by attending each of its neighbor points in low resolution point set both in spatial and feature space, we can dynamically weight the sparse flows and aggregate them to get the dense flow. Benefit from this design, we can get up-sampled flow at each resolution through hierarchical Trans-up layers and fed it into downstream tasks with higher flexibility and efficiency. Benefit from the hierarchical flow outputs, we further present a Hierarchical loss which calculates the EPE (every point error) of intermediate flows in each up-sampling layer to better constrain the flow accuracy and speed up the convergence.

Based on the proposed fundamental Trans-flow and Trans-up layers, we build our FH-Net with series version, FH-L, FH-R and FH-S with flexible combinations of the above modules to trade off between complexity and efficiency, which can adapt to various tasks. Additionally, dynamic objects is commonly of small number in a single frame, which poses challenges to learn the prediction of more important foreground flow. To tackle this problem, inspired by [44,8], we propose a new data augmentation strategy which during training randomly select objects from pre-built dynamic objects database and augment them with different strategy, i.e., dropout, down-sampling, to generate object pairs then paste them in two frames respectively.

Furthermore, to mitigate the problem of lacking real-world training data for scene flow estimation, we build two new point cloud scene flow datasets named SF-KITTI and SF-Waymo by annotating general autonomous driving dataset i.e. KITTI [7], Waymo [34] with pseudo sceneflow labels. Specifically, we compute every foreground point's label from its gt box's pose transform between two frames and annotate background point via ego-motion obtained from IMU. We note that the recent released dataset [15] shares similar ideas. The key difference are it only focuses on foreground point and different processing strategies for ground point.

Extensive experiments show our FH-Net achieves state-of-the-art lidar-scanned point cloud scene flow estimation results both on public Lidar-KITTI [9,7] and our more challenging large scale dataset SF-Waymo with faster inference speed and significant reduced parameters compared with previous methods [9,43,18]. Furthermore, we also achieve competitive results on no-lidar-scanned dataset FlyingThings3D [20] and Stereo-KITTI [21], which shows the great generality of our method.

The key contributions of this work are as follows:

- We propose a fast hierarchical architecture, which outputs accurate hierarchical scene flow on real-world point clouds with high efficiency.
- We introduce a new data augmentation strategy which improves predicted flow's accuracy especially on more challenging dynamic objects.
- We construct two lidar-scanned point cloud scene flow datasets to the community, which can facilitate the research on real-world scene flow estimation.
- We establish new state-of-the-art results on Lidar-KITTI and SF-Waymo.

2 Related Works

Scene flow on Point Clouds. While there is extensive literature on traditional 3D scene flow [38,13,42,14,36,37,25], we pay more attention to recent learning-based methods[20,40,6,26,30,24] which have shown advantages in many aspects. Pioneer

work FlowNet3D[18] directly consumed point cloud and estimated scene flow through an encoder-decoder architecture. FlowNet++[41] further used geometric constraints to get more accurate flow. HPLFlowNet[10] projected point clouds into permutohedral lattices then got flow from Bilateral Convolutional Layers. Inspired by PWC-Net[33], PointPWCNet[43] estimated scene flow in a coarse-to-fine fashion. FLOT[27] estimated scene flow through optimal transport and Meteor-Net[19] improved the accuracy of the inferred flow utilizing multiple temporal information. Recently, Gojcic et al.[9] estimated foreground flow and background flow respectively with weakly supervision. Other works[23,16] also explored the unsupervised/self-supervised sceneflow estimation because of the absence of annotated real-world scene flow dataset.

Transformer and attention. Transformer architectures with attention mechanisms have recently revolutionized computer vision research. For 2D image, Dosovitskiy et al. [5] treat images as sequences of patches. Hu et al. [12] and Ramachandran et al.[29] applied scalar dot-product self-attention within local image patches and Zhao et al. [47] developed a family of vector self-attention operators. Recently, Zhao et al. [48] and Guo et al.[11] introduced vector and dot-product transformer into 3D point cloud to better extract geometric features in a single frame. While in this work we use attention mechanism to better model the local relationship across consecutive frames.

3 Methodology

Given two consecutive point clouds, $\boldsymbol{P}_0$ at frame t and $\boldsymbol{Q}_0$ at frame $t+1$, the task of scene flow estimation aims to predict the translation of every point in $\boldsymbol{P}_0$ from frame t to frame $t+1$.

3.1 Network Architectures

Fig. 2 presents the overall workflow of our FH-Net, which comprises four key steps: *i) Feature embedding:* get key-points with shallow features for each frame; *ii) Trans-flow layer:* compute every key point's flow by directly aggregating the spatial vectors between source point and target point set; *iii) Trans-up layer:* upsample previous flow based on the similarity between sparse and dense points both in spatial and feature space; *iv) Flow propagation:* propagate the flow to raw resolution.

Feature embedding. This step aims to sample a subset of keypoints from the input point cloud, and enrich each of the keypoints with local geometric features. We apply two sequential set abstractions [28] to $\boldsymbol{P}_0$ and $\boldsymbol{Q}_0$, outputting two keypoint sets, $\boldsymbol{P}_2 = \{(\boldsymbol{x}_i, \boldsymbol{u}_i)\}_{i=1}^{N_2}$ and $\boldsymbol{Q}_2 = \{(\boldsymbol{y}_i, \boldsymbol{v}_i)\}_{i=1}^{N_2}$, respectively, where $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{N}^3$ denote *xyz* coordinates and $\boldsymbol{u}_i, \boldsymbol{v}_i \in \mathbb{R}^C$ denote embedded local features.

Trans-flow layer. The trans-flow layer outputs the scene flow $\boldsymbol{F}_2 = \left\{\boldsymbol{f}_i^2 \in \mathbb{R}^3\right\}_{i=1}^{N_2}$ for $\boldsymbol{P}_2$. It computes each flow element as a weighted sum of the spatial offsets from a source keypoint in $\boldsymbol{P}_2$ to its neighboring target keypoints in $\boldsymbol{Q}_2$ using transformer attention mechanism, through the following two steps:

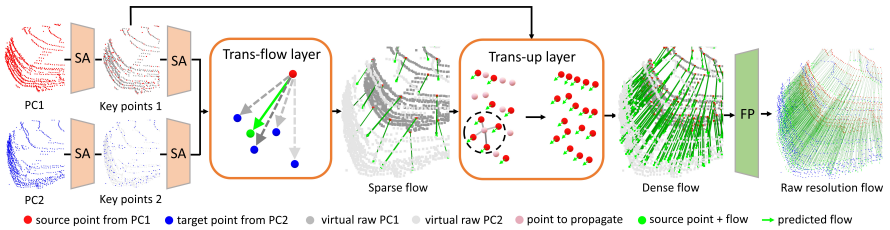

Figure 2. Overall architecture. FH-Net first obtains keypoints from set abstraction layers (SA), then uses Trans-flow layer to compute each key point's flow by directly aggregating the spatial vectors between source point and target neighbor point set through local region relationship modeling. The following trans-up layer is adopted to dynamically upsample the key points' flow which are then propagated to the raw resolution by Flow propagation module (FP).

Neighborhood grouping: For source keypoint $\boldsymbol{p}_i = (\boldsymbol{x}_i, \boldsymbol{u}_i)$ in $\boldsymbol{P}_2$, we first use ball query to group its spatially neighboring points in $\boldsymbol{Q}_2$, forming a target candidate set:

$$\boldsymbol{Q}_2^i = \{\boldsymbol{q}_j = (\boldsymbol{y}_j, \boldsymbol{v}_j) \mid \| \boldsymbol{y_j} - \boldsymbol{x_i} \|_2 < R\}_{j=1}^{M}, \tag{1}$$

where R is a preset radius and M is the number of neighboring points.

Cross-attention: We take $\boldsymbol{p}_i$ as *query* point and $\{\boldsymbol{q}_j\}_{j=1}^{M}$ as *key* points, and compute pair-wise attention weight as:

$$\alpha_{i,j} = \gamma\left(\varphi(\boldsymbol{u}_i) - \psi(\boldsymbol{v}_j) + \sigma(\boldsymbol{y}_j - \boldsymbol{x}_i)\right), \tag{2}$$

where φ and ψ are learnable point-wise linear transformations, σ is a relative position encoding. γ is a mapping function, learned by a MLP, that produces an attention weight by considering both feature and spatial relations.

Attentional aggregation: Given the attention weight, we directly take the positional offset between $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ as *value*, and compute the translation $\boldsymbol{f}_i^2$ for $\boldsymbol{p}_i$ as a weighted sum of the positional offsets with all the *key* points:

$$\boldsymbol{f}_i^2 = \sum_{j=1}^{M} \rho(\alpha_{i,j})(\boldsymbol{y}_j - \boldsymbol{x}_i), \tag{3}$$

where ρ is a normalization function, *softmax* hereby.

Cross-frame feature enhancement: After obtaining the low-resolution flow, it will inevitably introduce noise if we upsample the flow rely on structure feature from single frame only. Therefore, in addition to the aggregation of the flow, the Trans-flow layer also fuse the cross-frame features for further usage. Specifically, for source keypoint $\boldsymbol{p}_i$, we integrate its corresponding local context from $\boldsymbol{Q}_2$:

$$\boldsymbol{u}_i' = \underset{j=1,\ldots,M}{\mathrm{MAX}} \left\{h\left(\mathcal{C}\left(\boldsymbol{u}_i, \boldsymbol{v}_j, \sigma\left(\boldsymbol{y}_j - \boldsymbol{x}_i\right)\right)\right)\right\}, \tag{4}$$

where $\boldsymbol{u}_i'$ denotes enhanced features, $\mathcal{C}$ denotes concatenation in channel dimension, h denotes learnable linear transformation, MAX is element-wise max pooling.

Trans-up layer. Let $\boldsymbol{P}_1 = \left\{(\boldsymbol{x}_i^1, \boldsymbol{u}_i^1)\right\}_{i=1}^{N_1}$ be the keypoint set ($N_1 > N_2$), output by the first set abstraction layer on $\boldsymbol{P}_0$. The trans-up layer aims to get the scene flow $\boldsymbol{F}_1 = \left\{\boldsymbol{f}_i^1 \in \mathbb{R}^3\right\}_{i=1}^{N_1}$ for $\boldsymbol{P}_1$, given the predicted flow $\boldsymbol{F}_2$ for $\boldsymbol{P}_2$.

Generally, we follow the similar transformer attention mechanism as in trans-flow layer. For a target point $\boldsymbol{x}_i^1$ in $\boldsymbol{P}_1$, we first construct its neighboring point set in $\boldsymbol{P}_2$ through *neighborhood grouping*. Then we take the target point $\boldsymbol{x}_i^1$ as *query*, and the source points in the neighboring point set as *keys*, and compute pair-wise attention weights through *cross attention*. Finally, we take the given flows of these source points as *values*, and compute the flow $\boldsymbol{f}_i^1$ for $\boldsymbol{x}_i^1$ through *attentional aggregation*. Optionally, we update the feature $\boldsymbol{u_i^1}$ by applying Eq. (4) to $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ for further flow refinement.

Fig. 4 shows that benefited from Trans-up layer, the intermediate flow not only becomes denser but also gets refined by considering the integrity and correlation of local regions which can conquer the influence of outliers.

Flow propagation layer. We simply use 3D interpolations to upsample scene flow $\boldsymbol{F_1}$ to the final output $\boldsymbol{F_0}$ instead of Trans-up layer since there are no extracted features corresponded to the raw resolution points. Optionally, to mitigate the propagation error, a flow refinement residual learned from the last layers' feature can be added to the output flow. Specifically, we upsample the last layers' feature to the raw resolution by 3D interpolations and feed it into a MLP layer then added to the interpolation result:

$$\boldsymbol{f_j^0} = \sum_{\boldsymbol{x_i^1} \in \chi} \frac{1}{\| \boldsymbol{x_i^1} - \boldsymbol{x_j^0} \|} \boldsymbol{f_i^1} + MLP\Big(\sum_{\boldsymbol{x_i^1} \in \chi} \frac{1}{\| \boldsymbol{x_i^1} - \boldsymbol{x_j^0} \|} \boldsymbol{u_i^1}\Big), \tag{5}$$

where χ is the local region satisfies $\| \boldsymbol{x_i^1} - \boldsymbol{x_j^0} \|_2 < R$.

3.2 Hierarchical Loss

Our FH-Net can predict hierarchical scene flows $\left\{\left\{\boldsymbol{f}_i^k \mid_{i=1,\ldots,N_k}\right\}\right\}_{k=0}^{2}$, which enables us to add supervisions on each flow hierarchy during optimization, to improve predictive accuracy and speed up convergence:

$$\mathcal{L}_h = \sum_{k=0}^{2} \lambda_k \frac{1}{N_k} \sum_{i=1}^{N_k} \beta_i \left\| \boldsymbol{f}_i^k - \hat{\boldsymbol{f}}_i^k \right\|_2^2, \tag{6}$$

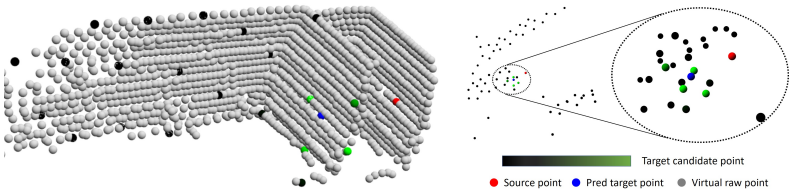

Figure 3. Attention visualization of Trans-flow layer. Brighter green points have higher attention weights and prediction is obtained by aggregating the offset from source point to these four activated point (we draw raw points in gray here for auxiliary understanding).

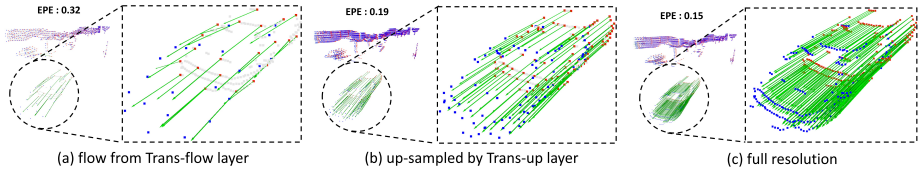

Figure 4. Hirachical flow visualization in Lidar-KITTI. red denotes source key point in a certain resolution, blue denotes the ground truth target points and green arrows denotes the flow. **gray** points are raw $\boldsymbol{P}_0$ points for reader to better understanding local structure.

where $\hat{\boldsymbol{f}}_i^k$ denotes the ground-truth flow corresponding to $\boldsymbol{f}_i^k$, λ_k is the balance weight for hierarchy k, and β_i is set to 5 and 0.5 for foreground and background points, respectively.

3.3 FH-Net Families

Based on the above fundamental layers, we provide three versions of FH-Net: **FH-R** is a standard version whose architecture specifications are listed in Tab. 1 **FH-L** is a lite version where residual refinement process is excluded during flow propagation, hence with less parameters and higher speed; **FH-S** is an advanced venison equipped with an extra segmentation module [4,35] to divide the point cloud into foreground and background points [9,1], whose flows are predicted by our model and by using the ego-motion from IMU sensors, respectively.

Table 1. Architecture specs of FH-R. SA: set abstraction.

layer type	R (m)	sampling rate	group num	MLP width
SA layer	0.5	0.5×	16	[32,32,64]
SA layer	1	0.25×	16	[64,64,128]
Trans-flow	10	1×	64	[128,256], [3,64,128], [128,512,1]
Trans-up	0.6	2×	64	[64,64], [256,128,64], [64,256,1], [3,64,64]
Flow propagation	0.5	4×	3	[256,256,128,3]

3.4 Data Augmentation

Our proposed data augmentation method first generates dynamic object pairs from a established object database then pastes them to the well designed safe area.

Dynamic object pairs generation. We first establish a dynamic object database by cropping a large number of independent dynamic object point clouds from all training data. During training, we take out the point cloud of one or more dynamic objects, then for each object we apply random downsampling and part dropout strategy to generate a pair of objects which is closer to the lidar scanned pattern as shown in Fig. 5 (a). Then we paste one of the object pairs to the current frame with random pose $\boldsymbol{T}_1$, and paste the

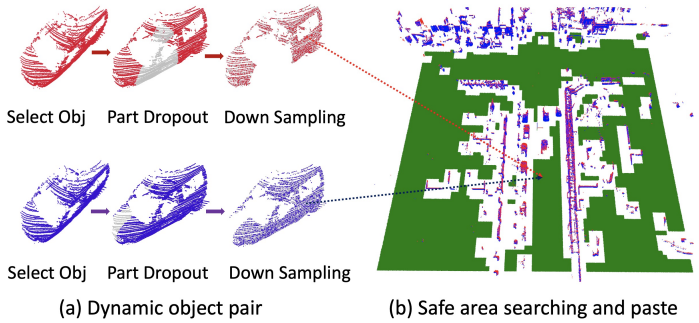

Figure 5. Data augmentation pipeline. green denotes safe area, red and blue denote points from two frames respectively.

other one with random pose $\boldsymbol{T}_2$ in the next frame. At the same time, we must ensure that the transformation from $\boldsymbol{T}_1$ to $\boldsymbol{T}_2$ conforms to the motion law of real object. In this way, a dynamic object pair is established, and the scene flow can be calculated as follows:

$$\boldsymbol{F}_{1\to 2} = \boldsymbol{T}_2\boldsymbol{P}_1 - \boldsymbol{T}_1\boldsymbol{P}_1, \tag{7}$$

where $\boldsymbol{P}_1 \in \mathbb{R}^4$ denotes the homogeneous points of first object in box coordinate and $\boldsymbol{F}_{1\to 2} \in \mathbb{R}^4$ is the homogeneous flow.

Safe area searching for object paste. Importantly, we need to prevent the pasted objects from overlapping with existing scenes in the raw frame, which does not happen in the real world. Inspired by [17], we first pillarize the points of current frame, and divide them into $h * w$ pillars in BEV (bird eye view). The pillars with points less than threshold is noted as *P-empty*, the safe area where object can be pasted. Both box_1 and box_2 should be located in the safe area. We further carry out morphological corrosion on the 2D *P-empty* image from BEV perspective to get stricter safe area of current frame as shown in Fig. 5 (b). As we first paste box_1 then paste box_2 according to the motion law, which is more convenient to implement, it can be ensured to great extent that box_2 is also located in safe area as long as the constraints of motion law are met.

4 Real-world Dataset Creation

We build a large-scale 360° point cloud scene flow dataset named SF-Waymo as well as a smaller one SF-KITTI, which are more in line with the real scene based on the general automatic driving dataset, Waymo and KITTI.

Data collection. We collect real-world point cloud pairs from Waymo and KITTI as well as 3D gt boxes and ego-motion annotations, then process them by removing the ground points with preset height threshold as ground points are useless for understanding scene motion. For KITTI, we also crop the point cloud within the view of front camera where annotations like 3D boxes are available. In general, our SF-Waymo contains 100 scenes, total 20k pairs and SF-KITTI has 20 scenes, total 7k pairs.

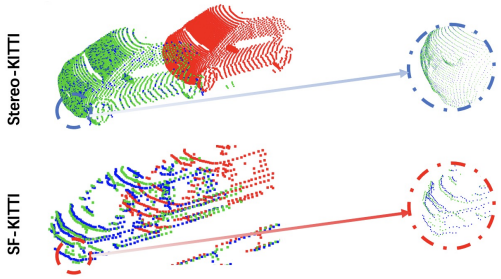

Figure 6. Comparisons of point density and correspondence between our SF-KITTI and previous Stereo-KITTI dataset. Red denotes source point, green denotes gt flow plus source point and blue denotes target point.

Tab. 2 provides more details and specific comparisons between our new proposed datasets with existing ones. Our datasets are closer to real scene with obvious advantages in scale, temporal frames, providing bidirectional flow and points semantic label.

Pseudo label. Observing that the scene flow of background is mainly generated by ego motion, while foreground flow is the combination of ego motion and dynamic object's movement, we divide points into background and foreground using gt boxes and compute label respectively. In addition, we also compute bidirectional flow between two frames. For background points $\boldsymbol{P}_1^b$ at frame t, the forward flow from time t to time $t+1$ is

$$\boldsymbol{F}_{for}^b = \boldsymbol{T}_{21}\boldsymbol{P}_1^b - \boldsymbol{P}_1^b, \tag{8}$$

where $\boldsymbol{T}_{21}$ denotes the pose transform from PC_1 to PC_2. For foreground dynamic object points $\boldsymbol{P}_1^f$ cropped by gt box, the forward flow can be written as:

$$\boldsymbol{F}_{for}^f = \boldsymbol{T}_{box,2}\boldsymbol{T}_{box,1}^{-1}\boldsymbol{P}_1^f - \boldsymbol{P}_1^f, \tag{9}$$

where $\boldsymbol{T}_{box,2}$ and $\boldsymbol{T}_{box,1}$ respectively denote the bbox's posture in PC_1 and PC_2 with vehicle coordinate, box_1 and box_2 can be matched by object ID. The backword flow can be calculated symmetrically:

$$\boldsymbol{F}_{back}^b = \boldsymbol{T}_{12}\boldsymbol{P}_2^b - \boldsymbol{P}_2^b, \tag{10}$$

$$\boldsymbol{F}_{back}^f = \boldsymbol{T}_{box,1}\boldsymbol{T}_{box,2}^{-1}\boldsymbol{P}_2^f - \boldsymbol{P}_2^f. \tag{11}$$

Table 2. Statistics of Different Datasets. DC: Direct point-to-point correspondence.

Dataset	Total	Train	Test	Pattern	Temporal	DC	FOV
FlyingThings3D	23k	19k	3824	synthetic	✗	✓	360°
Stereo KITTI	150	✗	150	disparity projection	✗	✓	90°
Lidar-KITTI	142	✗	142	lidar-scanned	✗	✗	90°
SF-KITTI	7k	6400	600	lidar-scanned	✓	✗	90°
SF-Waymo	20k	17k	3k	lidar-scanned	✓	✗	360°

Table 3. Datasets set-loss comparisons. Foreground set-loss is only computed by foreground points from two frames.

Dataset	FlyingThings3D	Stereo KITTI	Lidar-KITTI	SF-KITTI	SF-Waymo
Set-loss↓ $[m]$	0	0	0.205	0.149	**0.117**
Foreground set-loss↓ $[m]$	0	0	-	0.137	**0.077**

Visualization. Fig. 6 provide the visualization of some objects in Stereo-KITTI and our SF-KITTI. It can be seen that the points in Stereo-KITTI are in direct correspondence: for every source point in the first frame, we can find its matching point in the second frame, thus the green points and blue points almost coincide with each other in the first row of the figure. In addition, Stereo-KITTI is obtained from disparity images, where the points are dense and regular, it also has a large gap compared with the real-world. While in our SF-KITTI, since points are scanned by Lidar, they are sparse and not in direct correspondence, which is more inline with the real-world.

Label accuracy. We further verify that our pseudo labels are also accurate compared with Lidar-KITTI (labels obtained from carefully annotated KITTI-Sceneflow dataset). As GT flow is unavailable in real-world, we introduce **Set-loss** as a metric to evaluate the accuracy of flow annotations. Set-loss measures the degree of overlap between two point clouds pc_1, pc_2 (with n_1 and n_2 points, respectively) by averaging the point-to-set distance for every point in pc_1 and pc_2:

$$\frac{1}{n_1+n_2}\Big(\sum_{\boldsymbol{x_i}\in pc_1}\min_{\boldsymbol{x_j}\in pc_2}(\parallel \boldsymbol{x_i}-\boldsymbol{x_j} \parallel_2)+\sum_{\boldsymbol{x_j}\in pc_2}\min_{\boldsymbol{x_i}\in pc_1}(\parallel \boldsymbol{x_j}-\boldsymbol{x_i} \parallel_2)\Big). \tag{12}$$

Given two consecutive point cloud frames $\boldsymbol{P}$ and $\boldsymbol{Q}$ as well as the flow labels $\boldsymbol{f}$, we then compute the set loss between $\boldsymbol{P}+\boldsymbol{f}$ and $\boldsymbol{Q}$ to evaluate the accuracy of flow annotations. Ideally, for synthetic datasets FlyingThings3D and Stereo-KITTI (KITTI-Sceneflow), where points are in one to one correspondence, the set loss is **zero**. For real-world datasets like Lidar-KITTI, SF-KITTI and SF-Waymo, more accurate flow labels should also have smaller set loss. Significantly, the labels of Lidar-KITTI are computed by projecting points to image plane then assigning them the carefully annotated labels from KITTI-Sceneflow, which have been used widely. As shown in Tab. 3, the set-loss of SF-Waymo and SF-KITTI is lower than Lidar-KITTI and becomes smaller on foreground objects, which proves that the annotations of SF-Waymo and SF-KITTI, though generated from bounding boxes, are satisfactorily accurate and even better than those of existing real-world dataset.

5 Experiments

To evaluate the behavior of our FH-Net, we conduct comprehensive experiments on Lidar-KITTI [9,7], Stereo-KITTI and FlyingThings3D and the proposed SF-Waymo datasets. First, we describe settings of experiments, and then extend into results of our method. Further, we conduct extensive ablation studies to verify the effectiveness of FH-Net in detail.

Table 4. Evaluation results on Lidar-KITTI.

Method	Training set	EPE3D↓ $[m]$	Acc3DS↑	Acc3DR↑	Outliers↓	Speed (Hz)	Parameters
FlowNet3D[18]	FT3D	0.722	0.03	0.122	0.965	15.6	4,800k
PointPWC-Net[43]	FT3D	**0.39**	**0.387**	**0.55**	**0.653**	13.2	30,000k
FLOT[27]	FT3D	0.653	0.155	0.313	0.837	0.9	**440k**
Rigid3DSceneFlow[9]	FT3D	0.535	0.262	0.437	0.742	7.2	7,700k
FH-L	FT3D	0.531	0.20	0.397	0.825	**112.6**	680k
FlowNet3D[18]	SF-KITTI	0.289	0.107	0.334	0.749	15.6	4,800k
PointPWC-Net[43]	SF-KITTI	0.275	0.151	0.405	0.737	13.2	30,000k
FLOT[27]	SF-KITTI	0.271	0.133	0.424	0.725	0.9	**440k**
FH-L	SF-KITTI	0.255	0.241	0.538	0.683	**112.6**	680k
FH-R	SF-KITTI	**0.156**	**0.341**	**0.636**	**0.612**	57.3	1,500k
Rigid3DSceneFlow[9]	FT3D + SemanticKITTI	0.15	0.521	0.744	0.45	7.2	7,700k
FH-S	SF-KITTI + SemantcKITTI	**0.096**	**0.698**	**0.812**	**0.367**	**33.8**	**5,300k**

Table 5. Evaluation results on SF-Waymo.

Method	Training set	EPE3D↓ $[m]$	Acc3DS↑	Acc3DR↑	Outliers↓	Speed (Hz)	Parameters
FlowNet3D[18]	SF-Waymo	0.225	0.230	0.486	0.779	9.3	4,800k
PointPWC-Net[43]	SF-Waymo	0.307	0.103	0.231	0.786	1.0	30,000k
FESTA[39]	SF-Waymo	0.223	0.245	0.272	0.765	6.2	5,400k
FH-L	SF-Waymo	0.243	0.212	0.508	0.652	**66.7**	**680k**
FH-R	SF-Waymo	0.211	0.209	0.522	0.621	33.6	1,500k
FH-S	SF-Waymo	**0.175**	**0.358**	**0.674**	**0.603**	24.4	5,300k

5.1 Experimental Settings

Implementation Details. In our experiments, we set epochs to 150, batch size to 8, 4 for experiments on SF-KITTI and SF-Waymo, respectively. We train FH-Net by Adam optimizer with learning rate of 0.001. For hierarchical loss in Eq. (6), we set $\alpha_0 = 0.2$, $\alpha_1 = 0.2$ and $\alpha_2 = 0.6$, especially. In terms of data augmentation strategy, we paste 3 cars, 1-5 pedestrian and 1-4 cyclists with 0.4 probability of random down-sampling, as well as dropout. Notably, all models are trained end-to-end from scratch.
Evaluation Metrics. We take 3D end-point-error (EPE3D) as the main evaluation metric, defined as the mean L_2 distance between the ground truth and predicted scene flow. In addition, strict accuracy (Acc3DS), relaxed accuracy (Acc3DR) and Outliers are used to describe the performance of models.

5.2 Results on Lidar-KITTI

Setup. Lidar-KITTI [9,7] is a lidar-scanned point cloud scene flow dataset with ground truth from back projecting 3D points to Stereo-KITTI [22,21]. We train FH-Net on SF-KITTI with data-augmentation strategy referred in Sec. 3.4 and compare it with current SoTA methods [18,43,27,9], on Lidar-KITTI dataset. The results of previous methods trained on FlyingThings3D are from [9] and we also re-train them on SF-KITTI with the same settings as FH-Net for fair comparision. We train the segmentation branch of FH-S with semantic-KITTI[2] dataset. In addition, following [18,10,9], we remove the ground points by a preset height threshold, cause they are meaningless to scene motion understanding. All methods are measured on a single RTX 3090 GPU.

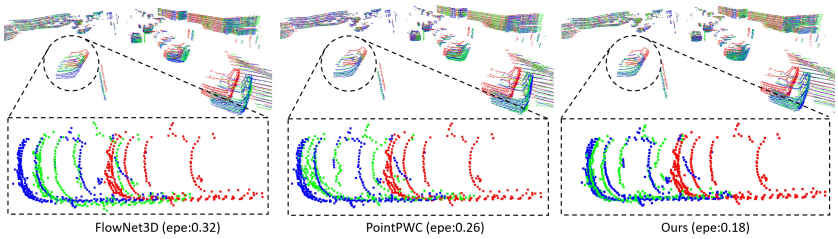

Figure 7. Qualitative results on Lidar-KITTI. red denotes PC1 points, green denotes predicted points and blue points are the ground truth.

Results. Tab. 4 shows our FH-Net achieves an EPE3D accuracy of 0.096, superior to SoTA method Rigid3DSceneFlow [9], as well as obvious advantages in terms of speed (about $5\times$). In addition, the elite version FH-L achieves competitive performance with fewer parameters ($0.68M$) and higher inference speed (112.6 fps). Compared with [9], the *params* of FH-L are reduced by 91.2% and the *speedup* is measured $15\times$.
Speed. Compared with current methods, FH-Net shows tremendous advantages in terms of speed. All three versions of FH-Net (FH-L, FH-R, FH-S) outperform all other methods in inference speed, achieving an extremely outstanding performance. With elite version (FH-L), we further speed up the inference to an amazing level of 112.6 fps at a sacrifice of little accuracy drop, while maintaining high-performance 0.255 EPE3D.
Visualization. We select some representative scenarios to visualize our performance and give some analysis. As shown in Fig. 7, previous works fail to fit the wide range object movement for lacking local position information, while ours predicts behave well with the help of strong local relative position.

5.3 Results on SF-Waymo

Setup. We train all methods on SF-waymo training set and evaluate them on SF-Waymo validation set. Other settings are the same as Sec. 5.1.
Results. Empirically, our method surpasses all other methods and achieves the state-of-the-art EPE3D accuracy of **0.175** with higher inference speed and fewer parameters. For example, we outperform FESTA [39] by 0.048 EPE3ED and $4\times$ speedup in inference

Table 6. Evaluation results on no-lidar-scanned dataset

Method	Training set	EPE3D
FlowNet3D [18]	FT3D	0.177
HPLFlowNet [10]	FT3D	0.117
Rigid3DSceneflow [9]	FT3D	**0.042**
FH-R	FT3D	0.098
FH-S	SF-KITTI + FT3D	0.072

(a) **Results on Stereo-KITTI**

Method	EPE3D
FlowNet3D [18]	0.114
PointPWC-Net [43]	0.059
FLOT [27]	**0.052**
Rigid3DSceneflow [9]	**0.052**
FH-R	0.054

(b) **Results on FlyingThings3D**

efficiency. Notably, the lite version FH-L achieves amazing speed (**66.7Hz**) with slight drop in accuracy, Tab. 5 illustrates all results on SF-Waymo.
Speed. Due to the lightweight design of architecture, our method outperforms all other methods in inference speed. Compared with FESTA ($6.2fps$), our FH-Net still performs much better ($24.4fps, 33.6fps$ and $66.7fps$). Moreover, we notice even in the large-scale SF-Waymo dataset, our FH-Net still exceeds the requirement of real-time scene flow prediction, showing great possibilities for autonomous driving.

5.4 Results on No-lidar-scanned Dataset

We also report the results on no-lidar scanned datasets such as Stereo-KITTI and FlyingThings3D. Results in Tab. 6 indicate that our well-designed architecture for real-world scene flow estimation can well generalize to human-made data. Notably, on such datasets our method does not perform as well as on real-world datasets like Lidar-KITTI and SF-Waymo. Cause points in Stereo-KITTI and FlyingThings3D datasets always have direct correspondence, our method that estimates target points has poorer performance than Rigid3DSceneflow[9] that adopts matching strategy.

5.5 Ablation Studies

Tab. 7 shows the ablation results based on FH-L. Models are trained on SF-KITTI and evaluated on Lidar-KITTI, unless otherwise specified, analyzed as follows:
Effect of Trans-flow layer. We replace Trans-flow layer with other operators such as taking the average of candidate flows or directly using the nearest point's offset as flow. Tab. 7d shows that Trans-flow layer has obvious superiority.
Effect of Trans-up layer. Tab. 7b demonstrates that upsampling the sparse flow through Trans-up layer can better modeling the local region's relationship and get more accurate flow than simply linear interpolation. We also observe that the interacted features is beneficial in the Trans-up layer.
Effect of position encoding. Tab. 7h shows that considering the semantic feature similarity as well as the spatial similarity by adding positional encoding in trans-flow layer leads to apparent improvement, which indicates that the local geometry distribution plays an important role when predict the flow.
Refinement in flow propagation Tab. 7f illustrates that adding residual refinement in flow propagation module can help mitigate some propagation errors as we expected. We think that the refinement module plays an important role in case of occlusion.
Effect of hierarchical loss. Tab. 7c shows that intermediate flow supervision are beneficial, which is consistent with our assumptions that the more accurate intermediate flows also lead to better performance of the final prediction. We also find that expanding foreground weight by five times significantly boost the performance.
Effect of data augmentation. Tab. 7a shows the effectiveness of our new proposed data augmentation method with different number of pasted dynamic objects. Furthermore, we also evaluate the foreground EPE3D to investigate the performance gain from foreground dynamic object pair paste strategy. The result on Tab. 7a is consistent with our assumptions that increasing the number and diversity of dynamic objects can significantly boost

Table 7. Ablations studies

Car	Ped	Cyc	EPE3D*	EPE3D
✗	✗	✗	0.278	0.308
0	1-5	1-4	0.276	0.301
3	1-5	1-4	0.127	**0.255**
6	1-5	1-4	**0.106**	0.259

(a) **Data augmentation**: * denotes the error of foreground.

Linear	Trans	F.I	EPE3D
✓	✗	✗	0.459
✗	✓	✗	0.314
✗	✓	✓	**0.255**

(b) **Trans-up layer**:F.I denotes feature interaction.

$[\lambda_0, \lambda_1, \lambda_2]$	$[\beta_0, \beta_1]$	EPE3D
✗	✗	0.306
0.5, 0.5, 0	✗	0.298
0.6, 0.2, 0.2	1, 1	0.265
0.6, 0.2, 0.2	5, 0.5	**0.255**

(c) **Hierarchical loss**: λ and β are hyperparameters of loss.

TF	Avg	N	EPE3D
✗	✓	✗	1.384
✗	✗	✓	1.025
✓	✗	✗	**0.255**

(d) **Trans-flow layer**: TF,Avg,N denote trans-flow, average, nearest.

Method	Train set	Validation set	EPE3D
FlowNet3D	FT3D	SF-Waymo/eval	0.834
FlowNet3D	SF-Waymo/train	SF-Waymo/eval	**0.225**
FH-L	FT3D	SF-Waymo/eval	0.673
FH-L	SF-Waymo/train	SF-Waymo/eval	**0.243**

(e) **Training data**: We train baseline model and FH-L on commonly used training set FT3D and SF-Waymo.

FR	EPE3D
✗	0.255
✓	**0.223**

(f) **FR**: residual flow refinement.

Method	Pre-train set	Fine-tune set	Validation set	EPE3D
FlowNet3D	FT3D	Stereo-KITTI	Stereo-KITTI	0.144
FH-L	FT3D	Stereo-KITTI	Stereo-KITTI	0.110
FlowNet3D	SF-KITTI	Stereo-KITTI	Stereo-KITTI	**0.114**
FH-L	SF-KITTI	Stereo-KITTI	Stereo-KITTI	**0.079**

(g) **Pre-train and fine-tune**: We pre-train the baseline and our FH-L on our dataset then fine-tune on others.

Pos.enc	EPE3D
✗	0.278
✓	**0.255**

(h) **Position encoding**

the performance especially on more challenging foreground objects.

Effect of training data. In Tab. 7e and Tab. 4, we analyze the effect of our new proposed SF-KITTI and SF-Waymo by re-training models on them and compare the results with those trained on FT3D. It can be seen that the EPE3D is obviously reduced both on Lidar-KITTI and SF-Waymo, which proves that our new dataset is more friendly to the real-world sceneflow esitamtion. Furthermore, we evaluate the effectiveness of our datasets by using it to pretrain the model then fine-tune on other public datasets, Tab. 7g shows that the network can achieve better performance by learning more challenging real-world scenes in our new proposed dataset.

6 Conclusion

In this paper, we present a fast hierarchical framework named FH-Net which can output hierarchical flows and maintains high inference speed. We also propose a new copy-and-paste data augmentation method to focus more on challenging dynamic objects and establish two real-world sceneflow dataset for real-world scene flow estimation.

Acknowledgements This work was financially supported by the National Natural Science Foundation of China (No. 62101032), the Postdoctoral Science Foundation of China (No. 2021M690015), and Beijing Institute of Technology Research Fund Program for Young Scholars (No. 3040011182111).

References

1. Behl, A., Paschalidou, D., Donné, S., Geiger, A.: Pointflownet: Learning representations for rigid motion estimation from point clouds. In: CVPR (2019)
2. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: ICCV (2019)
3. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: A survey of recent developments. IJRR (2011)
4. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR (2019)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Fan, H., Yang, Y.: Pointrnn: Point recurrent neural network for moving point cloud processing. arXiv preprint arXiv:1910.08287 (2019)
7. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
8. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.Y., Cubuk, E.D., Le, Q.V., Zoph, B.: Simple copy-paste is a strong data augmentation method for instance segmentation. In: CVPR (2021)
9. Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T.: Weakly supervised learning of rigid 3d scene flow. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5692–5703 (2021)
10. Gu, X., Wang, Y., Wu, C., Lee, Y.J., Wang, P.: Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: CVPR (2019)
11. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (2021)
12. Hu, H., Zhang, Z., Xie, Z., Lin, S.: Local relation networks for image recognition. In: ICCV (2019)
13. Huguet, F., Devernay, F.: A variational method for scene flow estimation from stereo sequences. In: ICCV (2007)
14. Jaimez, M., Souiai, M., Gonzalez-Jimenez, J., Cremers, D.: A primal-dual framework for real-time dense rgb-d scene flow. In: ICRA (2015)
15. Jund, P., Sweeney, C., Abdo, N., Chen, Z., Shlens, J.: Scalable scene flow from point clouds in the real world. IEEE Robotics and Automation Letters (2021)
16. Kittenplon, Y., Eldar, Y.C., Raviv, D.: Flowstep3d: Model unrolling for self-supervised scene flow estimation. In: CVPR (2021)
17. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019)
18. Liu, X., Qi, C.R., Guibas, L.J.: Flownet3d: Learning scene flow in 3d point clouds. In: CVPR (2019)
19. Liu, X., Yan, M., Bohg, J.: Meteornet: Deep learning on dynamic 3d point cloud sequences. In: ICCV (2019)
20. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR (2016)
21. Menze, M., Heipke, C., Geiger, A.: Joint 3d estimation of vehicles and scene flow. ISPRS (2015)
22. Menze, M., Heipke, C., Geiger, A.: Object scene flow. ISPRS (2018)
23. Mittal, H., Okorn, B., Held, D.: Just go with the flow: Self-supervised scene flow estimation. In: CVPR (2020)

24. Mustafa, A., Hilton, A.: Semantically coherent 4d scene flow of dynamic scenes. IJCV (2020)
25. Newcombe, R.A., Fox, D., Seitz, S.M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: CVPR (2015)
26. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4d reconstruction by learning particle dynamics. In: ICCV (2019)
27. Puy, G., Boulch, A., Marlet, R.: Flot: Scene flow on point clouds guided by optimal transport. In: ECCV (2020)
28. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
29. Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., Shlens, J.: Stand-alone self-attention in vision models. arXiv preprint arXiv:1906.05909 (2019)
30. Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., Guibas, L.J.: Caspr: Learning canonical spatiotemporal point cloud representations. arXiv preprint arXiv:2008.02792 (2020)
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
32. Shao, L., Shah, P., Dwaracherla, V., Bohg, J.: Motion-based object segmentation based on dense rgb-d scene flow. IEEE Robotics and Automation Letters **3**(4), 3797–3804 (2018)
33. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: CVPR (2018)
34. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020)
35. Sun, P., Wang, W., Chai, Y., Elsayed, G., Bewley, A., Zhang, X., Sminchisescu, C., Anguelov, D.: Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In: CVPR (2021)
36. Tanzmeister, G., Thomas, J., Wollherr, D., Buss, M.: Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation. In: ICRA (2014)
37. Ushani, A.K., Wolcott, R.W., Walls, J.M., Eustice, R.M.: A learning approach for real-time temporal scene flow estimation from lidar data. In: ICRA (2017)
38. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. In: ICCV (1999)
39. Wang, H., Pang, J., Lodhi, M.A., Tian, Y., Tian, D.: Festa: Flow estimation via spatial-temporal attention for scene point clouds. In: CVPR (2021)
40. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: CVPR (2018)
41. Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., Chen, M.: Flownet3d++: Geometric losses for deep scene flow estimation. In: WACV (2020)
42. Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D.: Efficient dense scene flow from sparse or dense stereo data. In: ECCV (2008)
43. Wu, W., Wang, Z., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. arXiv preprint arXiv:1911.12408 (2019)
44. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
45. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: CVPR (2021)
46. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: Common practices and emerging technologies. IEEE access **8**, 58443–58469 (2020)
47. Zhao, H., Jia, J., Koltun, V.: Exploring self-attention for image recognition. In: CVPR (2020)
48. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: ICCV (2021)