3-D Scene Flow Estimation on Pseudo-LiDAR: Bridging the Gap on Estimating Point Motion

Chaokang Jiang ©, Guangming Wang ©, Graduate Student Member, IEEE, Yanzi Miao ©, Member, IEEE, and Hesheng Wang ©, Senior Member, IEEE

Abstract—3-D scene flow characterizes how the points at the current time flow to the next time in the 3-D Euclidean space, which possesses the capacity to infer autonomously the nonrigid motion of all objects in the scene. The previous methods for estimating scene flow from images have limitations, which split the holistic nature of 3-D scene flow by estimating optical flow and disparity separately. Learning 3-D scene flow from point clouds also faces the difficulties of the gap between synthesized and real data and the sparsity of LiDAR point clouds. In this article, the generated dense depth map is utilized to obtain explicit 3-D coordinates, which achieves direct learning of 3-D scene flow from 2-D images. The stability of the predicted scene flow is improved by introducing the dense nature of 2-D pixels into the 3-D space. Outliers in the generated 3-D point cloud are removed by statistical methods to weaken the impact of noisy points on the 3-D scene flow estimation task. Disparity consistency loss is proposed to achieve more effective unsupervised learning of 3-D scene flow. The proposed method of self-supervised learning of 3-D scene flow on real-world images is compared with a variety of methods for learning on the synthesized dataset and learning on LiDAR point clouds. The comparisons of multiple scene flow metrics are shown to demonstrate the effectiveness and superiority of introducing pseudo-LiDAR point cloud to scene flow estimation.

Index Terms—Deep learning, depth estimation, pseudo-LiDAR, scene flow, self-supervised learning.

I. INTRODUCTION

The scene flow [1] estimates the motion of a 3-D point in the scene, which is different from LiDAR odometry [2] that estimates the consistent pose transformation of the entire scene. 3-D scene flow is more flexible. The flexibility of scene flow makes it capable of assisting in many different tasks, such as object tracking [3] and LiDAR odometry [2]. Generally, depth and optical flow together represent the scene flow in the scene flow estimation method based on a 2-D image [4], [5] or RGBD image [6]. Mono-SF [4] infers the scene flow of visible points using constraints on motion invariance of multiview geometric and depth distribution of a single view. To obtain more reliable scene flow, Hur et al. [5] propagated temporal constraints to continuous multiframe images. RAFT-3D [6] estimates the scene flow by soft grouping pixels into rigid objects. The nature of the scene flow being the 3-D motion vectors is split by the methods for estimating scene flow by optical flow and pixel depth in image space. Because of the lack of explicit 3-D geometry information in the image space, these methods often cause large pixel matching errors during scene flow inference. For example, two unrelated points that are far apart in 3-D space may be very close to each other in the image plane. Dewan et al. [7] predicted 3-D scene flow from adjacent frames of LiDAR data in a without-learning method. This method requires various strong assumptions, such as that the local structure will not be deformed by the motion in the 3-D scene. Some recent works [8], [9], [10], [11] learn 3-D scene flow from point cloud pairs based on deep neural network. However, these methods of 3-D scene flow estimation are invariably self-supervised/supervised on the synthesized dataset FlyingThings3D [12] and evaluated the generalization of the model on the real-world dataset KITTI Scene Flow [13]. The models trained on the synthesized dataset will cause accuracy degradation in the real world [9].

3-D scene flow annotations are very scarce in real-world datasets. Some works [9], [11] propose some excellent self-supervised losses, but these are difficult to achieve success on LiDAR signals. LiDAR signals are recognized to have two weaknesses, sparsity and point cloud distortion. First, the existing self-supervised losses [9], [11] imply a strong assumption of point-by-point correspondence. But point clouds from different moments are inherently discrete. The point-by-point correspondence and the discrete nature of the point cloud are contradictory. The sparsity of the LiDAR signal further exacerbates this fact. Second, the data collection process for mechanical LiDAR is...
flow estimation. A novel disparity consistency loss is proposed by exploiting the coupling relationship between 3-D scene flow and stereo depth estimation.

II. RELATED WORK

A. Pseudo-LiDAR

In recent years, many works [15], [16], [17] built a pseudo-LiDAR-based 3-D object detection pipeline. Pseudo-LiDAR has shown significant advantages in the field of object detection. Wang et al. [15] introduced pseudo-LiDAR into an existing LiDAR object detection model and demonstrate that the main reason for the performance gap between stereo and LiDAR is the representation of the data rather than the quality of the depth estimate. Pseudo-LiDAR++ [16] optimizes the structure and loss function of the depth estimation network based on Wang et al. [15] to enable the pseudo-LiDAR framework to accurately estimate distant objects. Qian et al. [17] built Pseudo-LiDAR++ [16] to address the problem that the depth estimation network and the object detection network must be trained separately. The previous pseudo-LiDAR framework focuses more on scene perception with the single frame, while this article focuses on the motion relationship between two frames.

B. Scene Flow Estimation

Some works study the estimation of dense scene flow from images of consecutive frames. Mono-SF [4] proposes Prob-DepthNet to estimate the pixel depth distribution of the single image. Geometric information from multiple views and depth distribution information from a single view are used to jointly estimate the scene flow. Hur et al. [5] introduced a multi-frame temporal constraint to the scene flow estimation network. Chen et al. [1] developed a coarse-grained software framework for scene-flow methods and realized real-time cross-platform embedded scene flow algorithms. In addition, Rishav et al. [18] fused LiDAR and images to estimate dense scene flow, but they still perform feature fusion in image space. These methods rely on 2-D representations and cannot learn geometric motion from explicit 3-D coordinates. The pseudo-LiDAR is the bridge between the 2-D signal and 3-D signal, which provides the basis for directly learning the 3-D scene flow from the 2-D data.

The original geometric information is preserved in the 3-D point cloud, which is the preferred representation for many scene understanding applications in self-driving and robotics. Some researchers [7] estimate 3-D scene flow from LiDAR point clouds by using the classical method. Dewan et al. [7] introduced local geometric constancy during the motion and introduced a triangular grid to determine the relationship of the points. Benefiting from the point cloud deep learning, some recent works [8], [9], [10], [11] proposed to learn 3-D scene flow from raw point clouds. FlowNet3D [8] first proposes the flow-embedding layer, which finds point correspondence and implicitly represents the 3-D scene flow. FLOT [10] studies lightweight structures for optimizing scene flow estimation.
using optimal transport modules. PointPWC-Net [9] proposes novel learnable cost volume layers to learn 3-D scene flow in a coarse-to-fine approach, and introduces three self-supervised losses to learn the 3-D scene flow without accessing the ground truth. Mittal et al. [11] proposed two new self-supervised losses.

III. SELF-SUPERVISED LEARNING OF THE 3-D SCENE FLOW FROM PSEUDO-LIDAR

A. Overview

The main purpose of this article is to recover 3-D scene flow from stereo images. The stereo images are represented by $I_l$ and $I_r$, respectively. As shown in Fig. 2, given a pair of stereo images, which contains reference frames $\{I_l^t, I_r^t\}$ and target frames $\{I_l^{t+1}, I_r^{t+1}\}$. Each image is represented by a matrix of dimension $H \times W \times 3$. Depth map $D_t$ at time $t$ is predicted by feeding the stereo image $\{I_l^t, I_r^t\}$ into the depth estimation network $D_{net}$. Each pixel value of $D$ represents the distance $d$ between a certain point in the scene and the left camera. Pseudo-LiDAR point cloud comes from backprojecting the generated depth map to a 3-D point cloud, as follows:

$$
\begin{align*}
  x_w &= d \times (u - c_x) \times \frac{1}{f_x} \\
  y_w &= d \times (v - c_y) \times \frac{1}{f_y} \\
  z_w &= d,
\end{align*}
$$

where $f_x$ and $f_y$ are the horizontal and vertical focal lengths of the camera, and $c_x$ and $c_y$ are the coordinate centers of the image, respectively. The 3-D point coordinate $(x_w, y_w, z_w)$ in the pseudo-LiDAR point cloud $PL$ is calculated by pixel coordinates $(u, v), d$, and camera intrinsics.

$$
PL_t = \{c_{1,i} \in \mathbb{R}^3 \}_{i=1}^{N_1}
$$

with $N_1$ points and

$$
PL_{t+1} = \{c_{2,j} \in \mathbb{R}^3 \}_{j=1}^{N_2}
$$

with $N_2$ points are generated from the depth maps $D_t$ and $D_{t+1}$, where $c_{1,i}$ and $c_{2,j}$ are the 3-D coordinates of the points. $PL_t$ and $PL_{t+1}$ are randomly sampled to $N$ points, respectively. The sampled pseudo-LiDAR point clouds are passed into the scene flow estimator $F_{sf}$ to extract the scene flow vector $SF_t = \{s_f[i] | i = 1, 2, \ldots, N\}$ for each 3-D point in frame $t$.

$$
SF_t = F_{sf}(\{D_{net}(I_l^t; \theta_1), D_{net}(I_l^{t+1}; \theta_1); \theta_2\}
$$

where $\theta_1$ and $\theta_2$ are the parameters of the network. $P$ represents the backprojection by (1).

![Diagram of the framework for estimating 3-D scene flow from stereo images. Depth estimator workflow is described in Section III-B. Pseudo-LiDAR pipeline is described in detail in Section III-A. Refinement of Pseudo-LiDAR Point Clouds is described in Section III-C, where the red and blue points represent the point clouds at frame $t$ and frame $t+1$, respectively. “w/ edge constraints” denote the operation of filtering edge error points of the pseudo-LiDAR point cloud, which are explained specifically in Section III-C2. “w/o outliers” denote the operation of filtering outliers, which are explained specifically in Section III-C3. Scene flow estimator is described in Section III-D. Pseudocode for the overall framework is also provided in this article in Algorithm 1.](image-url)
It is difficult to obtain the ground truth scene flow of pseudo-LiDAR point clouds. Mining a priori knowledge from the scene itself to self-supervised learning of 3-D scene flow is essential. Ideally, \( \mathcal{PL}_{t+1} \) and estimated point cloud \( \mathcal{PL}_{t+1}^* \) have the same structure. With this priori knowledge, point cloud \( \mathcal{PL}_t \) is warped to point cloud \( \mathcal{PL}_{t+1}^* \) through the predicted scene flow \( \mathcal{SF}_t \) as follows:

\[
\mathcal{PL}_{t+1}^* = \mathcal{PL}_t + \mathcal{SF}_t.
\]

Based on the consistency of \( \mathcal{PL}_{t+1} \) and \( \mathcal{PL}_{t+1}^* \), the loss functions (7) and (10) are utilized to implement self-supervised learning. We provide the pseudocode in Algorithm 1 for our method, where \( L, CV_{pc}, \) and \( \text{PredSF} \) are described in detail in Section III-D.

### B. Depth Estimation

The disparity \( z \) is the horizontal offset of the corresponding pixel in the stereo image, which represents the difference caused by viewing the same object from a stereo camera. \( I_r(u, v) \) and \( I_c(u, v + z) \) represent the observation of the same 3-D point in space. Two cameras are connected by a line called the baseline. The distance \( d \) between the object and the observation point can be calculated by knowing the disparity \( z \), the baseline length \( b \), and the horizontal focal length \( f \)

\[
d = \frac{b \times f}{z}.
\]

Disparity estimation networks, such as PSMNet [19], extract deep feature maps \( \{F_r^t, F_c^t\} \) and \( \{F_{r+1}^t, F_{c+1}^t\} \) from \( \{I_r^t, I_c^t\} \) and \( \{I_{r+1}^t, I_{c+1}^t\} \), respectively. As shown in Fig. 2, the features \( F_r^t (u, v) \) and \( F_c^t (u, v + z) \) are concatenated to construct 4-D tensor \( C_{\text{disp}}(u, v, z,:) \), namely the disparity cost volume. Then, the disparity cost volume \( H_{\text{disp}}(u, v, z) \) is calculated by feeding \( C_{\text{disp}}(u, v, z,:) \) into the 3-D convolutional neural network (CNN). The predicted pixel disparity \( D^*(u,v) \) is calculated by softmax weighting \( \sum_z \text{softmax}(H_{\text{disp}}(u, v, z)) \times z \) [19]. Based on the fact that disparity and depth are inversely proportional to each other, the convolution operation in the disparity cost volume has disadvantages. The same convolution kernel is applied to a few pixels with small disparity (i.e., large depth) resulting in an easy skipping and ignoring many 3-D points. It is more reasonable to run the convolution kernel on the depth grid that produces the same effect on neighbor depths, rather than overemphasizing objects with large disparity (i.e., small depth) on the disparity cost volume. Based on this insight, the disparity cost volume \( C_{\text{disp}}(u, v, z,:) \) is reconstructed as depth cost volume \( C_{\text{dep}}(u, v, d,:) \) [16]. Finally, the depth of the pixel is calculated through a similar weighting operation as abovementioned.

The sparse LiDAR points are projected onto the 2-D image as the ground truth depth map \( D_{gt} \). The depth loss is constructed by minimizing the depth error

\[
\mathcal{L}_1 = \sum_{(u,v) \in D_{gt}} \tau(D_{gt}(u,v) - D^*(u,v))
\]

where \( D^* \) represents the predicted depth map, and \( \tau \) represents smooth L1 loss.
2) Constraints on Pseudo-LiDAR Point Cloud Edges: A large number of incorrectly estimated points are distributed at the scene boundaries. For example, the generated pseudo-LiDAR point cloud has long tails on the far left and far right. Weakly textured areas, such as white sky, also result in a lot of depth estimation errors. Appropriate boundaries are specified for pseudo-LiDAR point clouds to remove as many edge error points as possible and not to lose important structural information.

3) Remove Outliers From the Pseudo-LiDAR Point Cloud: It is also very important to remove the noise points inside the pseudo-LiDAR point cloud. As shown in Fig. 2, a long tail is formed on the edge of the car, and these estimated points have deviated from the car itself. Statistical analysis is performed on the neighborhood of each point. The average distance \( \bar{v}_m \) from the \( m \) neighboring points is calculated. The obtained result is assumed to be a Gaussian distribution, whose mean is determined by the mean and standard deviation. This statistical method is useful to find discordant points in the whole point cloud. A point in the point cloud is considered as an outlier when its distance from its nearest point is larger than a distance threshold \( d_{\text{max}} \)

\[
d_{\text{max}} = \bar{v}_m + \alpha \times \sqrt{\frac{1}{m-1} \sum_{k=1}^{m} (v_k - \bar{v}_m)^2}
\]

where \( d_{\text{max}} \) is determined by the scaling factor \( \alpha \) and the number \( m \) of nearest neighbors.

D. 3-D Scene Flow Estimator

The image convolution process is the continuous multiplication and summation of the convolution kernel in the image space. This operation is flawed for matching points in a real-world space. Points that are far apart in 3-D space may be close together on the image, which leads to incorrect feature representation or feature matching. Convolution on 3-D point clouds better avoids that flaw.

The generated pseudo-LiDAR point clouds \( PL_t \) and \( PL_{t+1} \) are encoded and downsampled by PointConv [9] to obtain the point cloud features \( F_{PL_t} = \{ F_{t}^{i} | i = 1, 2, \ldots, n_1 \} \) and \( F_{PL_{t+1}} = \{ F_{t+1}^{i} | i = 1, 2, \ldots, n_2 \} \). Then, the matching cost between point \( pl_{t,i} \) and point \( pl_{t+1,j} \) is calculated by concatenating the features \( F_t^{i} \) of \( pl_{t,i} \), the features \( F_{t+1}^{i} \) of \( pl_{t+1,j} \), and the direction vector \( pl_{t,i} - pl_{t+1,j} \) [9], where \( pl_{t,j} \in PL_t \) and \( pl_{t+1,j} \in PL_{t+1} \). The nonlinear relationship between \( pl_{t,i} \) and \( pl_{t+1,j} \) is learned by using multilayer perceptron. According to the obtained matching costs, the point cloud cost volume \( CV_{pc} \) used to estimate the movement between points is aggregated. The scene flow estimator constructs a coarse-to-fine network for scene flow estimation. The coarse scene flow is estimated by feeding point cloud features and \( CV_{pc} \) into the scene flow predictor PredSF. The input of PredSF is the point cloud features of the first frame of the current level, \( CV_{pc} \), scene flow \( SF_{t+1}^{ep} \) from the last level of upsampling, and point cloud features from the last level of upsampling. The output is the scene flow and point cloud features of the current level. The local features of the four variables of PredSF input are merged by using the PointConv [9] layer, and the new \( N_l \times C \)-dimensional features are output. The new \( N_l \times C \)-dimensional features are used as input to the multilayer perceptron to predict the current level of scene flow. The final output is the 3-D scene flow \( SF_{t} = \{ s_{f_j} | i = 1, 2, \ldots, n_1 \} \) of each point in \( PL_t \).

The self-supervised loss is used at each level to optimize the prediction of the scene flow. The proposed network has four unsupervised losses Chamfer loss \( \mathcal{L}_{C} \), smoothness constraint \( \mathcal{L}_{SC} \), Laplacian regularization \( \mathcal{L}_{LR} \), and disparity consistency \( \mathcal{L}_{DC} \).\( \mathcal{P}L_t \) is warped using the predicted scene flow \( SF_{t} \) to obtain the estimated point cloud \( \mathcal{P}L_{t+1}^{\omega} \) at time \( t + 1 \). The \( \mathcal{L}_{C} \) loss function is designed to calculate the chamfer distance between \( \mathcal{P}L_{t+1} \) and \( \mathcal{P}L_{t+1}^{\omega} \). The formula is described as

\[
\mathcal{L}_{C}(\mathcal{P}L_{t+1}^{\omega}, \mathcal{P}L_{t+1}) = \sum_{pl_{t+1}^{\omega} \in \mathcal{P}L_{t+1}^{\omega}} \min_{pl_{t+1} \in \mathcal{P}L_{t+1}} \| pl_{t+1}^{\omega} - pl_{t+1} \|^2 + \sum_{pl_{t+1} \in \mathcal{P}L_{t+1}} \min_{pl_{t+1}^{\omega} \in \mathcal{P}L_{t+1}^{\omega}} \| pl_{t+1}^{\omega} - pl_{t+1} \|^2
\]

where \( \| \cdot \|_2 \) represents the operation of Euclidean distance. The design of smoothness constraint \( \mathcal{L}_{SC} \) is inspired by the prior knowledge of smooth scene flow in real-world local space as follows:

\[
\mathcal{L}_{SC} = \sum_{s_{f_j} \in SF} \frac{1}{|R(s_{f_j})|} \sum_{s_{f_i} \in R(s_{f_j})} \| s_{f_i} - s_{f_j} \|_2^2
\]

where \( R(s_{f_j}) \) means the set of all scene flow in the local space around \( s_{f_i} \), and \( |R(s_{f_i})| \) represents the number of points in \( R(s_{f_i}) \). Similar to \( \mathcal{L}_{C} \), the goal of Laplacian regularization \( \mathcal{L}_{LR} \) is to make the Laplace coordinate vectors of the same position in \( \mathcal{P}L_{t+1}^{\omega} \) and \( \mathcal{P}L_{t+1} \) consistent. The Laplace coordinate vector \( \omega(s_{f_j}) \) of the point in \( \mathcal{P}L_{t+1}^{\omega} \) is calculated as follows:

\[
\omega(s_{f_j}) = \frac{1}{|R(s_{f_j})|} \sum_{pl_{t+1}^{\omega} \in R(s_{f_j})} \| pl_{t+1}^{\omega} - pl_{t+1} \|_2^2
\]

where \( |R(s_{f_j})| \) is the number of points in the local space around \( s_{f_j} \) and \( |R(s_{f_j})| \) is the number of points in \( R(s_{f_j}) \). \( \omega \) is the interpolated Laplace coordinate vector from \( \mathcal{P}L_{t+1} \) at the same position as \( pl_{t+1}^{\omega} \) by using the inverse distance weight. \( \mathcal{L}_{LR} \) is described as

\[
\mathcal{L}_{LR} = \sum_{pl_{t+1}^{\omega} \in \mathcal{P}L_{t+1}^{\omega}} \| \omega(s_{f_j}) - \omega(pl_{t+1}) \|_2^2
\]

Inspired by the coupling relationship between depth and pose in unsupervised depth pose estimation tasks [21], we propose a disparity consistency loss \( \mathcal{L}_{DC} \). Specifically, each point on the first frame image is warped into the second frame by an estimated 3-D scene flow, and the disparity or depth values from the warped points and the points in the real second frame should be the same. The disparity consistency loss is specifically described as

\[
\mathcal{L}_{DC} = \mu(|B| \cdot [P_{f}(\mathcal{P}L_{t+1}^{\omega}), D_{t+1}] - P_{f}(\mathcal{P}L_{t}))
\]
TABLE I
ALL METHODS ARE EVALUATED ON SFKITTI [13]

| Methods       | Training Set | Sup | Input | EPE3D(m) | Acc3D† | Acc3D† | Outliers3D | EPE3D(px) | Acc2D† |
|---------------|--------------|-----|-------|----------|--------|--------|------------|----------|--------|
| FlowNet3 [22] | FlyingC, FT3D| Full| Stereo| 0.9111   | 0.2039 | 0.3587 | 0.7463     | 5.1023   | 0.7803 |
| FlowNet3D [8] | FT3D         | Full| Points| 0.1767   | 0.3738 | 0.6677 | 0.5271     | 7.2141   | 0.5093 |
| Pontes et al. [23] | FT3D | Self| Points| 0.1690   | 0.2171 | 0.4775 | —          | —        | —      |
| PointPW-CNet [9] | FT3D | Self| Points| 0.2549   | 0.2379 | 0.4957 | 0.6863     | 8.9439   | 0.3299 |
| PointPW-CNet [9] | FT3D, odKITTI| Self| Points| 0.1699   | 0.2593 | 0.5718 | 0.5584     | 7.2800   | 0.3971 |
| Self-Point-Flow [24] | FT3D | Self| Points| 0.1120   | 0.5276 | 0.7936 | 0.4086     | —        | —      |
| Matal et al. (8) [11] | FT3D, odKITTI| Self| Points| 0.1260   | 0.3200 | 0.7364 | —          | —        | —      |
| Ours PL (with pretrain) | FT3D, odKITTI| Self| Stereo| 0.1103   | 0.4568 | 0.7412 | 0.4211     | 4.9141   | 0.5532 |
| Ours PL (with pretrain) | FT3D, odKITTI| Self| Mono   | 0.0955   | 0.5118 | 0.7790 | 0.3812     | 4.2671   | 0.6046 |

*“Full” and “self” represent the supervised training and the self-supervised training, respectively. “Stereo,” “mono,” and “points” represent stereo images, monocular images, and point clouds, respectively. “L” and “†” are used to help the reader understand whether a larger or smaller value of the metric is better. The network is trained on the LiDAR point cloud pseudo-LiDAR point cloud denoted as “L” and “PL,” respectively. “H” represents fine-tuning the model on sFKITTI, where the model is trained on 100 frames in sFKITTI and is evaluated on the remaining 42 frames.

The bold values in tables mean the best evaluation results of the compared methods.

where $D_{t+1}$ represents the depth map at frame $t+1$. $P_j(\cdot)$ means the projection of the point cloud onto the image using the camera internal parameters. $B_l(\cdot)$ means the index of bilinear interpolation. $\mu(\cdot)$ means averaging over the tensor.

The overall loss of the scene flow estimator is as follows:

$$
L = \sum_{l=0}^{L} \Lambda_l (\lambda_1 L_C^L + \lambda_2 L_{SC}^L + \lambda_3 L_{LR}^L + \lambda_4 L_{DC}^L).
$$

(12)

The loss of the $l$th level is a weighted sum of four losses. The total loss $L$ is a weighted sum of the losses at each level. $\Lambda_l$ represents the weight of the loss in the $l$th level.

IV. EXPERIMENTS

A. Experimental Details

1) Training Settings: The proposed algorithm is written in Python and PyTorch and runs on Linux. On a single NVIDIA TITAN RTX GPU, we train for 40 epochs. The initial learning rate is set to 0.0001, and the learning rate decreases by 50% every five training epochs. The batch size is set to 4. The generated pseudo-LiDAR is randomly sampled to 4096 points as input of the scene flow estimator. With the same parameter settings as PointPWC-Net, there are four levels of the feature pyramid in the scene flow estimator in this article. In (12), the first-level weight $\alpha_1$ to the fourth-level weight $\alpha_4$ are 0.02, 0.04, 0.08, and 0.16. The self-supervised loss weights are $\lambda_1 = 1.0$, $\lambda_2 = 0.2$, $\lambda_3 = 0.2$, and $\lambda_4 = 1.0$, respectively.

Depth annotations in the synthesized dataset [12] are used to supervise the depth estimator, similar to Pseudo-LiDAR++ [16]. The pretrained depth estimator is fine-tuned utilizing LiDAR points from the KITTI [20] as sparse ground truth, as indicated in (5). During the scene flow estimator training stage, the depth estimator weights will be fixed. The scene flow estimator is first pretrained on FT3D [12] with the self-supervision method. Stereo images from the 00-09 sequence of the KITTI odometry dataset (odKITTI) [20] are selected to train our scene flow estimation model. To further improve the applicability of the method, we also explored a framework for monocular vision estimation of 3-D scene flow, where the depth estimator uses the advanced monocular depth model AdaBins [29].

To further demonstrate the denseness advantage of the pseudo-LiDAR point cloud proposed in Section III-C1, the scene flow estimator is trained on a denser LiDAR point clouds from the high-fidelity 128-Channel LiDAR Dataset (DurLAR) [14]. To be fair, we perform the same processing as PointPWC-Net [9] for the LiDAR point cloud in DurLAR. The results are presented at the bottom of Table I.

2) Evaluation Settings: Following PointPWC-Net [9], we evaluate the model performance on the KITTI Scene Flow dataset (sFKITTI) [13], where sFKITTI is obtained through 142 pairs annotations of disparity maps and optical flow. The lidarKITTI [13], with the same 142 pairs as sFKITTI, is generated by projecting the LiDAR point clouds of 64-beam onto the images. The 142 frame scenes are all used as test samples. Following Pontes et al. [23], we also evaluate the generalizability of the proposed method on two real-world datasets, Argoverse [26] (containing 212 test samples) and nuScenes [27] (containing 310 test samples). Different from Pontes et al. [23], our methods have not accessed any data from Argoverse [26] and nuScenes [27] in the training process. All methods in the table evaluate the performance of the scene flow directly on Argoverse and nuScenes. To be fair, we use the same evaluation metrics as PointPWC-Net [9].

B. Results

Tables I and II give the quantitative results of our method evaluated at sFKITTI [13]. The accuracy of our method is substantially ahead of supervised learning methods FlowNet3 [22], FlowNet3D [8], and FLOT [10]. Compared with the self-supervised learning method [9], [11], [23], [24], [25], [28] based on point clouds, learning 3-D scene flow on pseudo-LiDAR from...
### TABLE II

| Dataset | lidarKITTI [13] | Argoverse [26] | nuScenes [27] |
|---------|----------------|----------------|----------------|
| **Metrics** | **EPE3D↓** | **Acc3DS↑** | **Acc3DR↑** | **Outliers3D↓** | **EPE3D↓** | **Acc3DS↑** | **Acc3DR↑** | **Outliers3D↓** |
| PointPWC-Net [9] | 1.1944 | 0.0384 | 0.1410 | 0.9336 | 0.4288 | 0.0462 | 0.2164 | 0.9199 | 0.7883 | 0.0287 | 0.1333 | 0.9410 |
| Mittal et al. (ft) [11] | 0.9773 | 0.0096 | 0.0524 | 0.9936 | 0.6520 | 0.0319 | 0.1159 | 0.9621 | 0.8422 | 0.0289 | 0.1041 | 0.9615 |
| FLOT (Full) [10] | 0.6532 | 0.1554 | 0.3130 | 0.8371 | **0.2491** | 0.0946 | 0.3126 | 0.8657 | 0.4885 | 0.0821 | 0.2669 | 0.8547 |
| DCA-SRSPFE [28] | 0.5900 | 0.1505 | 0.3331 | 0.8485 | 0.7957 | 0.0712 | 0.1468 | 0.9799 | 0.7042 | 0.0538 | 0.1183 | 0.9766 |
| Ours (Stereo) | 0.5265 | 0.1752 | 0.3858 | 0.7638 | 0.2690 | 0.0768 | 0.2760 | 0.8440 | 0.4893 | 0.0354 | 0.2171 | 0.8649 |
| Ours (Mono) | **0.4908** | **0.2052** | **0.4238** | **0.7286** | 0.2517 | **0.1236** | **0.3666** | **0.8114** | **0.4709** | **0.1034** | **0.3175** | **0.8191** |

All networks learn the scene flow in a self-supervised approach and are not fine-tuned on lidarKITTI, argoverse, or nuScenes. The bold values in tables mean the best evaluation results of the compared methods.

---

The accuracy of learning 3-D scene flow on the 128-beam LiDAR signals [14] is improved compared with the 64-beam LiDAR signals. According to the quantitative results of Tables I and II, the proposed framework for learning 3-D scene flow from pseudo-LiDAR signals still presents greater advantages. To qualitatively demonstrate the effectiveness of our method, some visualizations are shown in Fig. 4. Compared with PointPWC-Net [9], the estimated points by our method are mostly correct points on the Acc3DR metric. From the details in Fig. 4, the point clouds estimated from our method overlap well with the ground truth point clouds, confirming the reliability of our method. Finally, the methods in this article also show excellent perceptual performance in the real world, as shown in Fig. 5.
TABLE III

| Edges | Outliers | $L_{DC}$ | Acc3DS | EPE2D(px) |
|-------|----------|---------|-------|----------|
| ×     | ×        | 0.2655  | 0.3319 | 11.4530  |
| ✓     | ×        | 0.1191  | 0.7181 | 5.3741   |
| ✓     | ✓        | 0.1156  | 0.7298 | 5.0802   |
| ✓     | ✓        | 0.1103  | 0.7412 | 4.9141   |

“Outliers” represents the elimination of outlier points within pseudo-LiDAR point clouds.
The bold values in tables mean the best evaluation results of the compared methods.

V. CONCLUSION

The method in this article achieves accurate perception of 3-D dynamic scenes on 2-D images. The pseudo-LiDAR point cloud is used as a bridge to compensate for the disadvantages of estimating 3-D scene flow from LiDAR point clouds. The points in the pseudo-LiDAR point cloud that affect the scene flow estimation are filtered out. In addition, disparity consistency loss was proposed and achieved better self-supervised learning results. The evaluation results demonstrate the advanced performance of our method in the real-world datasets.
[23] J. K. Pontes, J. Hays, and S. Lucey, “Scene flow from point clouds with or without learning,” in Proc. IEEE Int. Conf. 3D Vis., 2020, pp. 261–270.

[24] R. Li, G. Lin, and L. Xie, “Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 15577–15586.

[25] G. Wang, C. Jiang, Z. Shen, Y. Miao, and H. Wang, “SFGAN: Unsupervised generative adversarial learning of 3D scene flow from the 3D scene self,” Adv. Intell. Syst., vol. 4, no. 4, 2022, Art. no. 2100197.

[26] M.-F. Chang et al., “Argoverse: 3D tracking and forecasting with rich maps,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 8748–8757.

[27] H. Caesar et al., “muScenes: A multimodal dataset for autonomous driving,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11621–11631.

[28] Z. Jin, Y. Lei, N. Akhtar, H. Li, and M. Hayat, “Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 7233–7243.

[29] S. F. Bhat, I. Alhashim, and P. Wonka, “AdaBins: Depth estimation using adaptive bins,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 4009–4018.

Chaokang Jiang received the B.S. degree in automation from the Department of Engineering, Jingdezhen Ceramic Institute College of Technology and Art, Jingdezhen, China, in 2020. He is currently working toward the M.E. degree in control science and engineering with the China University of Mining and Technology, Beijing, China. His current research interests include SLAM and computer vision.

Guangming Wang (Graduate Student Member, IEEE) received the B.S. degree from the Department of Automation, Central South University, Changsha, China, in 2018. He is currently working toward the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. His current research interests include SLAM and computer vision, in particular, 2-D optical flow estimation, and 3-D scene flow estimation.

Yanzi Miao (Member, IEEE) received the Ph.D. degree in control science and engineering from the China University of Mining and Technology, Xuzhou, China, in 2019. She is currently a Professor with the School of Information and Control Engineering, China University of Mining and Technology, Beijing, China. Her current research interests include intelligent perception and fusion, machine vision, and active olfaction. As a joint-Ph.D. Candidate and a Visiting Scholar, she was with the Department of Informatics, University of Hamburg, Hamburg, Germany, in 2007 and 2017, respectively. Dr. Miao was the Technical Co-Chair of IEEE RCAR 2019.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively. He is currently a Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, service robot, computer vision, and autonomous driving.

Dr. Wang is an Associate Editor for IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, IEEE ROBOTICS AND AUTOMATION LETTERS, ASSEMBLY AUTOMATION, and INTERNATIONAL JOURNAL OF HUMANOID ROBOTICS, a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS, and the Editor of the Conference Editorial Board (CEB) of IEEE Robotics and Automation Society. He was an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS from 2015 to 2019. He was the General Chair of IEEE ROBIO 2022 and IEEE RCAR 2016, and the Program Chair of the IEEE ROBIO 2014 and IEEE/ASME AIM 2019. He will be the General Chair of IEEE/RSJ IROS 2025.