Pseudo-LiDAR Point Cloud Interpolation Based on 3D Motion Representation and Spatial Supervision

Haojie Liu, Kang Liao, *Graduate Student Member, IEEE*, Chunyu Lin, *Member, IEEE*, Yao Zhao, *Senior Member, IEEE*, and Yulan Guo, *Senior Member, IEEE*

***Abstract*— Pseudo-LiDAR point cloud interpolation is a novel and challenging task in autonomous driving, which aims to address the frequency mismatching problem between a camera and a LiDAR. Previous works represent the 3D spatial motion relationship with a coarse 2D optical flow, and the quality of interpolated point clouds only depends on the supervision of depth maps. As a result, the generated point clouds suffer from inferior global distributions and local appearances. To solve the above problems, we propose a Pseudo-LiDAR point cloud interpolation network to generate temporally and spatially high-quality point cloud sequences. By exploiting the scene flow from point clouds, the proposed network is able to learn a more accurate representation of the 3D spatial motion relationship. For a more comprehensive perception of the distribution of a point cloud, we design a novel reconstruction loss function with the chamfer distance to supervise the generation of Pseudo-LiDAR point clouds in 3D space. In addition, we introduce a multi-modal deep aggregation module to facilitate the efficient fusion of texture and depth features. As the benefits of the improved motion representation, training loss function, and model structure, our approach gains significant improvements on the Pseudo-LiDAR point cloud interpolation task. The experimental results evaluated on KITTI dataset demonstrate the state-of-the-art quantitative and qualitative performance of the proposed network.**

***Index Terms*— Pseudo-LiDAR interpolation, 3D point cloud, depth completion, scene flow, video interpolation, convolutional neural networks.**

I. Introduction

RECENTLY, multi-sensor systems that sense both image and depth information have gained increasing attention, which is widely used in navigation applications such as autonomous driving and robotics. For these applications, the accurate and dense depth information is crucial for the obstacle avoidance [1], object detection [2], [3], and 3D scene reconstruction tasks [4]. On the perception platform of autonomous driving, the prerequisite of sensor fusion is synchronizing the system. However, there is an inherent limitation in LiDAR sensors, which provide reliable 3D spatial information at a low frequency (around 10Hz). To achieve time synchronization, the frequency of the camera (around 20Hz) has to be decreased, leading to an inefficient multi-sensor system. In addition, LiDAR sensors only get sparse depth measurements, e.g., 64 scan lines in the vertical direction. Such a low frequency and sparse depth sensing are insufficient for real applications. Therefore, for synchronous sensing of multi-sensor systems, it would be promising to increase the frequency of LiDAR data to match the high frequency of cameras. The high-frequency and dense point cloud sequences are of great significance in high-speed and complicated application scenarios.

Because of the enormous volume of point cloud captured by LiDARs, direct processing and learning in 3D space is time-consuming. PLIN [5] presents the first pipeline for the Pseudo-LiDAR point cloud interpolation task. Particularly, the Pseudo-LiDAR point cloud is obtained by the interpolated dense depth map and camera parameters. PLIN increases LiDAR sensors' frequency by interpolating adjacent point clouds to solve frequency mismatching between a LiDAR and a camera. Using a coarse-to-fine architecture, PLIN can progressively perceive multi-modal information and generate the intermediate Pseudo-LiDAR point clouds. However, PLIN has several limitations, as follows.

1) The spatial motion information is derived from the 2D optical flow between color images of adjacent consecutive frames. Nevertheless, the 2D optical flow only represents the planar pixels' movement deviation, and cannot represent the movement in 3D space. Thus, the motion relationship used in PLIN causes an inferior temporal interpolation of point cloud sequences.

2) PLIN only supervises the generation of intermediate depth maps during the training process. Consequently, the quality of generated point clouds only depends on the synthetic depth maps. Moreover, the network does not provide any spatial constraints on the point cloud generation and does not measure the quality of the point cloud.

3) The method for the fusion of multi-modal features is plain. PLIN roughly concatenates the texture and depth features and feeds these features into an interpolation neural

Manuscript received June 12, 2020; revised November 26, 2020 and January 19, 2021; accepted January 27, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61772066, Grant 61972028, and Grant 61972435, and in part by the Fundamental Research Funds for the Central Universities under Grant 2018JBZ001. The Associate Editor for this article was C. Wen. *(Haojie Liu and Kang Liao are co-first authors.) (Corresponding author: Chunyu Lin.)*

Haojie Liu, Kang Liao, Chunyu Lin, and Yao Zhao are with the Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China, and also with the Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China (e-mail: hj_liu@bjtu.edu.cn; kang_liao@bjtu.edu.cn; cylin@bjtu.edu.cn; yzhao@bjtu.edu.cn).

Yulan Guo is with the School of Electronics and Communication Engineering, Sun Yat-sen University, Guangzhou 510006, China, and also with the College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: yulan.guo@nudt.edu.cn).

Digital Object Identifier 10.1109/TITS.2021.3056048

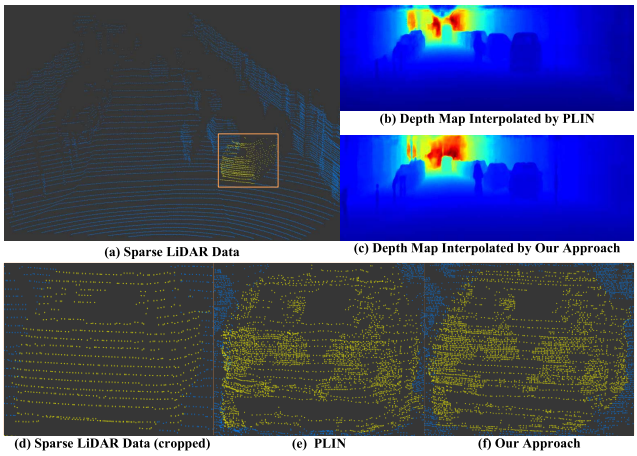


Fig. 1. A comparison of Pseudo-LiDAR point cloud interpolation methods. We visualize the interpolated depth maps and Pseudo-LiDAR point clouds obtained by PLIN [5] and our approach. The cropped region (d) indicates the ground truth point cloud from the sparse LiDAR data (a). Our result displays more accurate appearances than PLIN and is denser than the original point cloud.

network. However, this type of fusion cannot emphasize the effective complementary message passing between different features.

To address these limitations, we present a novel network to improve the motion representation and spatial supervision for Pseudo-LiDAR point cloud interpolation. In particular, since optical flow does not describe the motion information in 3D space, we use the scene flow to guide the generation of Pseudo-LiDAR point clouds. The scene flow represents a 3D motion field from two consecutive stereo pairs, and we design a spatial motion perception module to estimate it. In addition, we implement a point cloud reconstruction loss to constrain the interpolation of Pseudo-LiDAR point clouds, which enables us to generate more realistic results in terms of spatial distribution.

Further, we design a dual branch structure for texture completion and temporal interpolation. In the texture completion branch, the intermediate color image is used to provide rich textures for dense depth map generation. In the temporal interpolation branch, we exploit a warping layer with two adjacent point clouds and the estimated scene flow to synthesize the intermediate depth map. To facilitate efficient fusion of texture and depth features, we introduce a multi-modal deep aggregation module. As the benefits of the improved motion representation, loss function, and model structure, our approach gains significant improvements on the Pseudo-LiDAR point cloud interpolation task. As illustrated in Fig. 1, we compare the depth map and point cloud interpolated by PLIN and our approach. Our method produces more accurate appearances than PLIN and achieves denser distribution than the original point clouds.

The contributions of this work are summarized as follows.

- Considering the full representation of 3D motion information, we design a spatial motion perception module to guide the generation of Pseudo-LiDAR point clouds.
- We design a reconstruction loss to supervise and guide the generation of Pseudo-LiDAR point clouds in 3D space and further introduce a quality metric for point clouds.
- We propose a multi-modal deep aggregation module to effectively fuse the texture completion branch and temporal interpolation branch features.

The remainder of this paper is organized as follows. In Sec. II, we describe the related work of point cloud interpolation. In Sec. III, we introduce the overall model structure and describe each module in details. Finally, the experimental results are presented in Sec. IV.

II. Related Work

In this section, related works on the topic of depth estimation, depth completion, and video interpolation will be discussed.

A. Depth Estimation

Depth estimation focuses on obtaining the depth value of each pixel using a single color image. Earlier works used traditional methods [6], [7], which applied hand-crafted features to the depth of color images in probability map models. Recently, with the popularity of convolutional neural networks in image segmentation and detection, learning-based methods have been applied to the depth estimation task. For example, for supervised methods, Eigen *et al.* [8] adopted a multi-scale convolutional architecture to obtain depth information. Li *et al.* [9] used the conditional random field (CRF) post-processing refinement step to perform a regression on the features to obtain high-quality depth images. For unsupervised methods, the supervision is provided by perspective synthesis. Xie *et al.* [10] constructed a stereo pair to calculate and estimate an intermediate disparity image by generating corresponding perspectives. To further improve the performance, C. Godard *et al.* [11] used geometric constraints to constrain the differences between the left and right images. However, due to the inherent ambiguity and uncertainty of the depth information got from color images, the depth map obtained by these depth estimation methods are still inaccurate for navigation systems.

B. Depth Completion

Compared with depth estimation, the depth completion task aims to obtain an accurate dense depth map using a sparse depth map and possible image data. Depth completion covers a series of issues related to different input modalities. When the input modality is a relatively dense depth map that contains missing values, depth completion can be cast as a variety of techniques, such as the executive-based depth inpainting [12], [13], object-aware interpolation [14], Fourier-based depth filling [15], and depth enhancement [16].

LiDAR is an indispensable sensor in 3D sensing systems such as autonomous driving and robots. When the acquired LiDAR depth data is projected into the 2D image space, the available depth information only takes up about 4% of the image pixels [17]. To improve the application of such

sparse depth measurements, various methods try to use sparse depth values to estimate dense and accurate depth maps. For example, Uhrig *et al.* [17] proposed a simple and effective sparse convolutional layer to take data sparsity into account. Ma *et al.* [18] considered the depth completion task as a regression problem and fed the color image and sparse depth map into an encoder-decoder structure. A similar method in [19] used a self-supervised training mechanism to achieve the completion task. To extend the confidence of the convolution operation on the continuous network layer, A. Eldesokey *et al.* [20] proposed a constrained normalized convolution operation. Huang *et al.* [21] proposed boundary constraints to improve the structure and quality of the depth map. M. Jaritz *et al.* [22] jointly learned semantic segmentation and completion tasks to improve the performance. In [23], the surface normal estimation is used for the depth completion task. Chen *et al.* [24] designed an effective network fusion block, which can jointly learn 2D and 3D representations. Depth completion is based on interpolation in 2D space. In 3D space, a large number of point cloud inpainting (interpolation) methods [25], [26] [27], [28] are available in literature. Guo *et al.* [29] summarized the state-of-the-art methods for 3D point clouds processing. Compared with this spatial depth completion and inpainting methods for small object-oriented 3D point clouds, our method can generate temporally and spatially high-quality point cloud sequences. In the field of simultaneous localization and mapping (SLAM) [30], a laser frame can be projected and generated at any timestamp or any frequency based on the motion parameters. In the comparison experiment part, we compare the "projecting using motion parameters" + "depth completion" method with the proposed approach.

C. Video Interpolation

In the field of video processing, video interpolation is a popular research topic. Video frame interpolation aims to synthesize non-existent frames from the original adjacent frames. It makes sense to generate high-quality slow-motion videos from existing videos. Liu *et al.* [31] proposed a deep voxel flow network to synthesize video frames by flowing the pixel values of existing frames. To achieve real-time temporal interpolation, Peleg *et al.* [32] adopted an economic structured framework and considered the interpolation task as a classification problem rather than a regression problem. Jiang *et al.* [33] jointly modeled the motion interpretation and occlusion inference problems to achieve variable-length multi-frame video interpolation. Bao *et al.* [34] proposed a depth-aware stream projection layer to guide the detection of occlusion using depth information in the video frame interpolation task. Although there are many works in video frame interpolation, the point cloud interpolation task gains little attention due to the vast volume and complicated structure of point clouds.

PLIN [5] is the first work to interpolate an intermediate Pseudo-LiDAR point cloud from two consecutive stereo pairs. It utilized a coarse-to-fine network structure to facilitate the perception of multi-modal information such as optical flow and color images. Compared with PLIN, our approach uses the improved motion representation, training loss function, and model structure, achieving significant improvements on the Pseudo-LiDAR point cloud interpolation task.

III. Method

In this section, we introduce the overall model structure and describe each module in detail. Given two consecutive sparse depth maps (D_{t-1} and D_{t+1}) and RGB image (I_t), Pseudo-LiDAR point cloud interpolation aims to produce an intermediate dense depth map D_t, which is back-projected to obtain the intermediate Pseudo-LiDAR point cloud (PC_t) using known camera parameters. As illustrated in Fig. 2, the whole framework mainly comprises two branches: the texture completion branch and temporal interpolation branch. The texture completion branch takes the image, and consecutive sparse depth maps as inputs and produces feature maps encoded with rich textures. One channel of the feature maps is further combined with the sparse depth maps and the scene flow estimated by the spatial perception module, generating the feature maps guided by spatial motion information. The feature maps derived from the dual branch are then integrated by the fusion layer to produce the intermediate dense depth map. Finally, the intermediate Pseudo-LiDAR point cloud is generated by back-projecting the intermediate dense depth map.

A. Texture Completion Branch

The sparse depth map is difficult to represent the detailed relationship of context information due to its large number of missing pixel values. Therefore, the rich texture information of color images is conducive to the corresponding prediction depth, especially in boundary regions. Many works use color images to estimate depth information, which indicates that it can provide corresponding depth inference clues. The color image contains the low-level pixel information. To effectively guide the subsequent depth interpolation, the texture completion branch is designed to extract more rich features such as the high-level semantics feature from the color image. In particular, to extract the texture and semantic features, the adjacent sparse depth maps and color images are concatenated and fed into the texture completion branch. Moreover, the output of the texture completion branch can be used as a prior to guide the temporal interpolation branch.

We consider the interpolation of Pseudo-LiDAR point cloud as a regression problem. The texture completion network implements an encoder-decoder structure with skip connections. We concatenate the color image, and adjacent sparse depth maps into a tensor that is fed into the residual block of the encoder. In the encoder, the backbone network uses the residual network ResNet-34 [35]. In the decoder, the low-dimensional feature maps are up-sampled to the same size as the original feature map through five deconvolution operations. In addition, multiple skip connections are used to combine low-level features with high-level features. Except for the last layer of convolution, ReLU and Batch Normalization are performed after all convolutions. Finally, the last layer of the

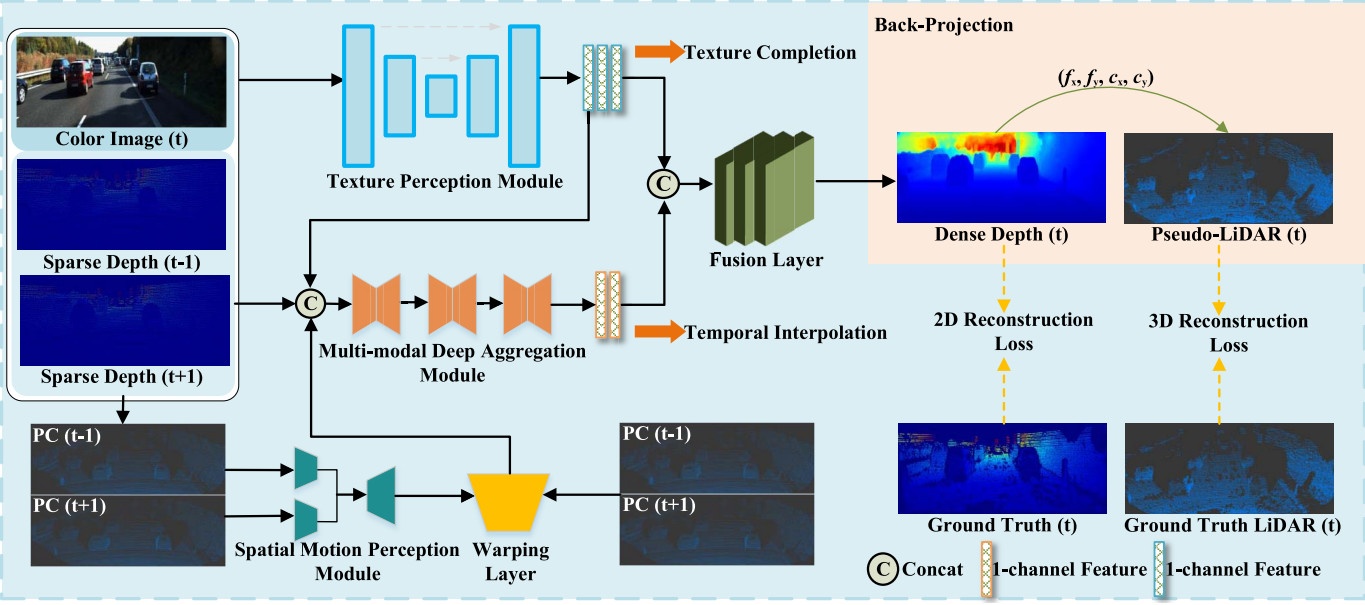


Fig. 2. The overview of the proposed Pseudo-LiDAR point cloud interpolation network. Taking an intermediate image at time t and two adjacent sparse depth maps at times $t-1$ and $t+1$ as its inputs, the texture completion branch produces rich texture feature maps.One feature map combines with the sparse depth maps and the scene flow estimated by the spatial perception module to produce the feature maps containing spatial motion information. Finally, a fusion layer fuses feature maps of the dual branch to generate an intermediate dense depth map, which is transformed by back-projecting to produce our Pseudo-LiDAR point cloud.

texture completion branch uses a 1×1 convolution kernel to reduce the multi-channel feature map into a 3-channel feature map. Note that these features only contain the texture and structure information, which cannot describe the accurate motion information yet. Thus, we introduce a spatial motion perception module to further improve the interpolation performance.

B. Spatial Motion Perception Module

In the video interpolation task, the optical flow is indispensable since it contains the motion relationship between adjacent frames. Optical flow represents the motion deviation $(\Delta x, \Delta y)$ in 2D image plane, while the scene flow is represented by a motion field $(\Delta x, \Delta y, \Delta z)$ in 3D space. Scene flow is the counterpart of optical flow in three-dimensional space, and it can represent the real spatial motion relationship of objects more explicitly. In PLIN, the optical flow between color images is used to represent the motion relationship of depth maps, but the optical flow only represents the movement of plane pixels and cannot fully describe the motion information in real 3D space. Therefore, our approach exploits the scene flow to generate a more realistic Pseudo-LiDAR point cloud.

The scene flow estimation method can be described as follows. The input is adjacent point clouds: PC_{t-1} at time $t-1$ and PC_{t+1} at time $t+1$. PC_{t-1} and PC_{t+1} are obtained by projecting the sparse depth map D_{t-1} and D_{t+1} using the camera's intrinsic parameters. For details, We will discuss the specific transformation formulas in Section III-D. Point cloud is a set of points $(x_i, y_i, z_i)_{i=1}^{n}$, where n is the number of points, and each point may also contain its own attribute features $(x_i, y_i, z_i, \ldots) \in \mathbb{R}^{d_f}$, where d_f refers to the dimension of the attribute feature, such as the reflection intensity, color, and normal. The output is the estimated scene flow $sf_i = (dx_i, dy_i, dz_i)$ for each point i in PC_{t-1}. FlowNet3D [36] explores the motion based on PointNet++ [37], it processes each point and aggregates information through the pooling layers. Our scene flow estimation network is based on FlowNet3D and the improved bilateral convolutional layer operations, which restore the spatial information from unstructured point clouds. The input of the network is 3D point clouds of consecutive frames, and the output is the corresponding deviation of each point. The scene flow calculations can be expressed as follows.

$$\begin{aligned} sf_{t-1\to t+1} &= H^{sf}(PC_{t-1}, PC_{t+1}), \\ sf_{t+1\to t-1} &= H^{sf}(PC_{t+1}, PC_{t-1}), \end{aligned} \tag{1}$$

where PC_{t-1} and PC_{t+1} denote the input point clouds of adjacent frames, H^{sf} is the scene flow estimation function, $sf_{t-1\to t+1}, sf_{t+1\to t-1}$ refer to the estimated scene flow.

Since the input of the scene flow estimation task is the adjacent point clouds, we first convert the adjacent depth maps into point clouds in terms of the prior camera parameters. The adjacent point clouds are then fed to our spatial motion perception module to estimate the scene flow. Fig. 3 shows the overall network structure of the spatial motion perception module. Our scene flow estimation network is similar to the encoder-decoder structure. In the downsampling stage, we adopt a dual-input structure, in which all layers share weights to extract the features of point clouds. By stacking improved bilateral convolutional layer (BCL) [38] to continuously reduce the scale. We also fuse features of different scales. In the upsampling phase, we gradually increase the scale by stacking the improved bilateral convolutional layers to improve the prediction performance. In each BCL, we consider the relative position of the input. Finally, our scene flow is obtained. The Warping Layer can be considered as an online

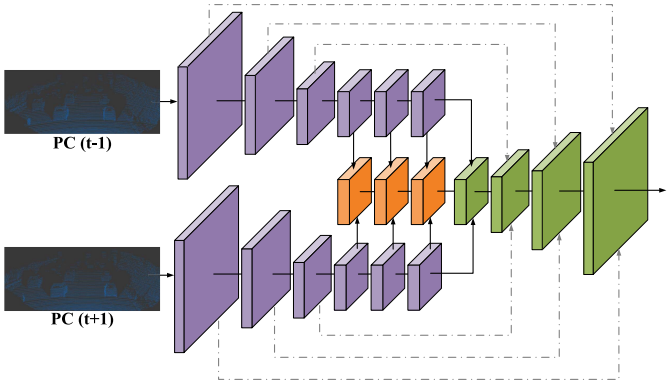


Fig. 3. The overall network structure of the spatial motion perception module.

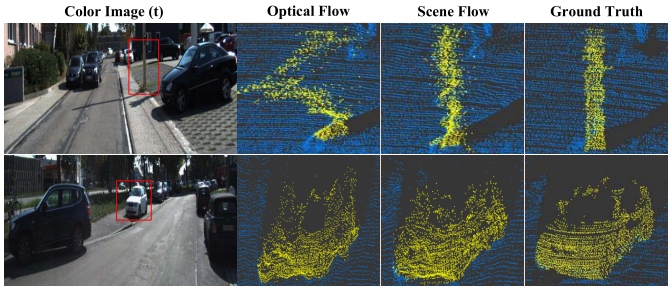


Fig. 4. Comparison of Pseudo-LiDAR point cloud guided by different motion information. The point cloud guided by scene flow is more similar to the ground truth, and the distribution is more reasonable.

operation. Its function consists of two parts. (i) First, it shifts PC_t-1 or PC_{t+1} to PC_t using scene flow, and (ii) it projects the point cloud to a sparse depth image. We use a warping operation on PC_{t-1} or PC_{t+1} to synthesize the point cloud PC_t at time t, which can be expressed as:

$$PC_t = PC_{t-1} + \frac{sf_{t-1\to t+1}}{2}, \tag{2}$$

or

$$PC_t = PC_{t+1} + \frac{sf_{t-1\to t+1}}{2}. \tag{3}$$

To boost the fast spatial information sensing, we project the obtained intermediate point cloud PC_t into the 2D image plane. In this part, we get the accurate but sparse intermediate depth map D_t. As shown in Fig. 4, we conducted a comparison experiment. With the same network structure, we use optical flow and scene flow to guide the interpolation of the Pseudo-LiDAR point cloud separately. The results show that the scene flow facilitates to generate a more realistic point cloud. Compared to optical flow, the point cloud guided by the scene flow has a more reasonable shape. We attribute this to better motion representation. To effectively integrate multi-modal features and generate an accurate and dense depth map, we introduce a multi-modal deep aggregation module to facilitate the efficient fusion of texture and depth features.

C. Multi-Modal Deep Aggregation Module

To generate an accurate and dense depth map, we design a multi-modal deep aggregation module to fuse the feature maps of the texture completion branch and the temporal interpolation branch. The texture feature can guide the network to pay more attention to the saliency objects, which contains a more clear structure and edge information. On the other hand, the depth feature can provide precise spatial information in terms of the estimated scene flow. Fig. 5 shows the overall network structure of the multi-modal deep aggregation module. The specific details are described as follows. In particular, we adopt a stacked aggregation unit architecture for the multi-modal deep aggregation module. The stacked aggregation unit consists of three aggregation units, each with a top-down and bottom-up process. Inspired by ResNet [35], we use a residual learning paradigm between aggregation units. In addition, the skip connection operations are applied to introduce the low-level feature into the high-level feature with the same dimension.

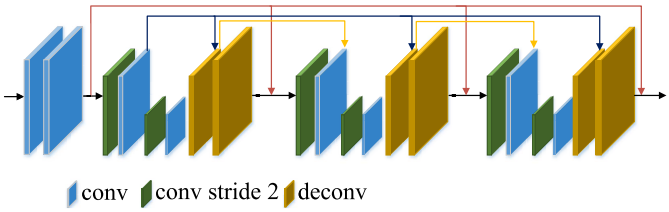


Fig. 5. The overall network structure of the multi-modal deep aggregation module.

In each aggregation unit, the encoder and decoder are composed of three layers of convolutions. The encoder uses two stride convolutions to downsample the feature resolution to the 1/4 original size; the decoder uses two deconvolution operations to upsample the features fused from the encoder and the previous network block. Considering the data's sparseness, the encoder in the first network block does not use the batch normalization operation after convolution. All convolutional layers use a 3×3 convolution kernel with a small receptive field. The output of the multi-modal deep aggregation module is a 2-channel feature map containing dense spatial distribution information. At the end of the dual branch architecture, we leverage a fusion layer to combine different feature maps further and obtain the final result. The fusion layer consists of three convolutional layers, and the number of filters per convolutional layer is 32, 32, and 1, respectively. Except for the last layer, the Batch Normalization layer with ReLU activation function is implemented after each convolutional layer.

There are two concat operation in our interpolation framework. First, considering the intermediate color image contains realistic structure and content information, we use 1-channel feature map derived from the texture completion branch to guide the multi-modal deep aggregation module, enabling a more accurate temporal interpolation. Subsequently, all feature maps of the texture completion branch are combined with the temporal interpolation's results to construct the final depth map, of which the rich texture and semantics information boosts a denser spatial interpolation.

D. Back-Projection

In this part, we obtain the 3D point cloud by back-projecting the generated intermediate depth map to 3D space. According

to the pinhole camera imaging principle, if the depth value $Z_t(u, v)$ of each pixel coordinate (u, v) exists in the image, we can derive the corresponding 3D position (x, y, z). That is:

$$x = \frac{(u - c_u) \times z}{f_u}, \tag{4}$$

$$y = \frac{(v - c_v) \times z}{f_v}, \tag{5}$$

$$z = Z_t(u, v), \tag{6}$$

where f_v and f_u are the vertical and horizontal focal lengths, respectively. (c_u, c_v) is the center of the camera aperture. Based on the prior camera parameters, the generated depth map is back-projected into a 3D point cloud. Since this point cloud is obtained by transforming the depth map, we refer to the point cloud as a Pseudo-LiDAR point cloud [39].

E. Loss Function

Previous work only supervises the generated dense depth maps, which does not constrain the 3D structure of the target point cloud. To this end, we design a point cloud reconstruction loss to supervise the generation of Pseudo-LiDAR point clouds. Constructing the distance function between the predicted point cloud and the ground truth point cloud is an important step. A suitable distance function should meet at least two conditions: 1) the calculation is differentiable; 2) since data needs to be forwarded and back-propagated for many times, effective calculations are required [40]. The goal of our efforts can be expressed as:

$$L\left(\left\{PC_i^{pred}\right\}, \left\{PC_i^{gt}\right\}\right) = \sum d\left(PC_i^{pred}, PC_i^{gt}\right), \tag{7}$$

where PC_i^{pred} and PC_i^{gt} indicate the prediction and ground truth of each sample, respectively.

We need to find a distance metric $d \subseteq \mathbb{R}^3$ to minimize the difference between the generated point cloud and the ground truth point cloud. There are two candidates for the measurement: the Earth Mover's distance (EMD) and the Chamfer distance (CD):

Earth Mover's distance: if two point sets $PC_1, PC_2 \subseteq \mathbb{R}^3$ have the same size, EMD can be defined as:

$$d_{EMD}(PC_1, PC_2) = \min_{\phi: PC_1 \to PC_2} \sum_{x \in PC_1} \|x - \phi(x)\|_2, \tag{8}$$

where $\phi : PC_1 \to PC_2$ is a bijection. EMD is almost differentiable everywhere, but its accurate calculation is expensive for learning models.

Chamfer distance: we can define it between $PC_1, PC_2 \subseteq \mathbb{R}^3$ as:

$$d_{CD}(PC_1, PC_2) = \sum_{x \in PC_1} \min_{y \in PC_2} \|x - y\|_2^2 + \sum_{y \in PC_2} \min_{x \in PC_1} \|x - y\|_2^2. \tag{9}$$

This algorithm finds the nearest point of each point PC_1 in another point set PC_2 and adds up the squared distances. For each point, searching for the nearest point is independent and easily parallelized. To speed up the nearest point search, a similar KD-tree data structure can be applied. Since EDM has a limitation on the number of input points, we use the simple and effective CD distance as our reconstruction loss to evaluate the similarity between generated point cloud and ground truth point cloud. Our reconstruction loss is formed as follows.

$$\mathcal{L}_{rec}(PC_{pred}, PC_{gt}) = d_{CD}\left(PC_{pred}, PC_{gt}\right) + d_{CD}\left(PC_{gt}, PC_{pred}\right), \tag{10}$$

where d_{CD} denotes the chamfer distance metric. PC_{pred} and PC_{gt} are the predicted and ground truth point clouds, respectively.

In addition to the point cloud supervision, we also perform 2D supervision on dense depth maps. We use $\mathcal{L}_2$ loss for the generated depth map D_{pred} and ground truth D_{gt}:

$$\mathcal{L}_D(D_{pred}, D_{gt}) = \|1_{\{D_{gt} > 0\}} \cdot (D_{pred} - D_{gt})\|_2^2. \tag{11}$$

Our entire loss function linearly combines the point cloud reconstruction loss and the depth map reconstruction loss, which can be expressed as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_D(D_{pred}, D_{gt}) + \lambda_2 \mathcal{L}_{rec}(PC_{pred}, PC_{gt}), \tag{12}$$

where λ_1 and λ_2 are the balance weights. The weights have been set empirically as $\lambda_1 = 1$ and $\lambda_2 = 1$.

IV. Experiments

In this section, we conduct extensive experiments to verify the effectiveness of our proposed approach. We compare with previous works and perform a series of ablation studies to show the effectiveness of each module. Since our model's main application is on-board LiDARs in a multi-sensor system, our experiments are based on the KITTI dataset [41]. As illustrated in Fig. 6, the depth maps obtained by our approach show clear boundaries in visual effects and are denser than the ground truth.

A. Experimental Setting

1) Dataset: Our experiments are performed on the KITTI depth completion dataset and the raw dataset. The KITTI dataset contains 86,898 frames of training data, 6,852 frames of evaluation data, and 1,000 frames of test data. This dataset provides sparse depth maps and color images. Each frame contains LiDAR scan data and RGB color images, in which the sparse depth map corresponds to the projection of the 3D LiDAR scan point cloud. The ground truth corresponding to each sparse depth map is a relatively dense depth map. Our application scenario is based on the outdoor on-board LiDAR, which is generally a scene of relative motion. Since there are scenes where the frames are still in the training dataset, we choose 48,000 frames with apparent motions.

2) Evaluation Metrics: Although our task is not depth completion, we can use the evaluation metrics of depth completion to evaluate the quality of the generated dense depth map. There are four evaluation metrics in the depth completion task: the root mean square error (RMSE), mean absolute error (MAE),

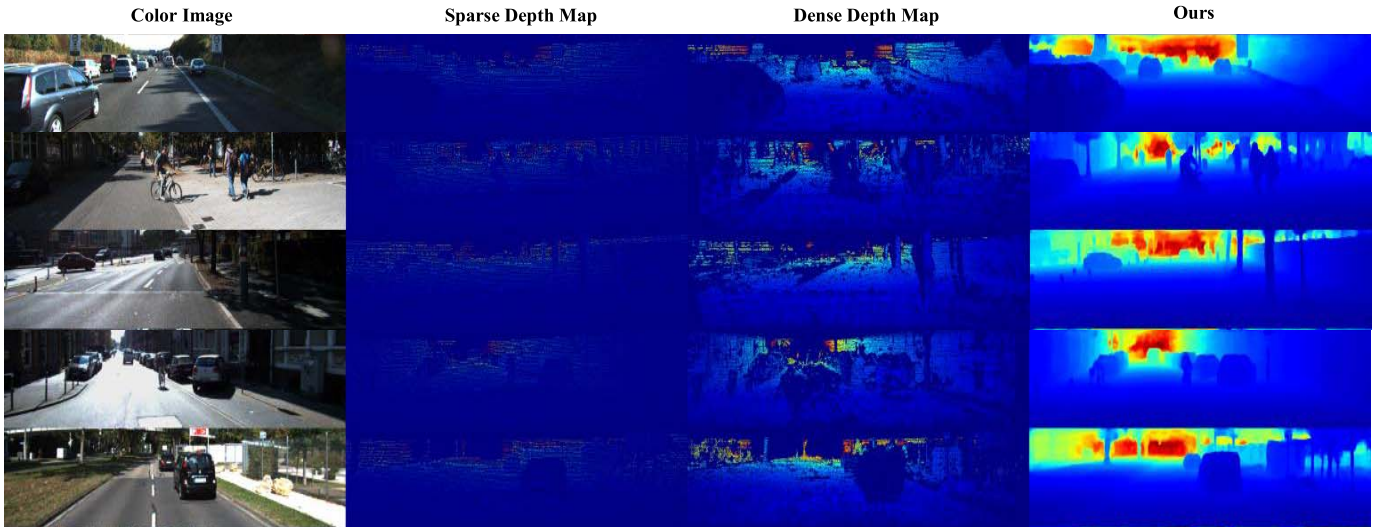


Fig. 6. Results of the interpolated depth map obtained by our approach. For each example, we show the color image, sparse depth map, dense depth map, and our interpolated result. The dense depth map represents the ground truth of training. Our depth map results have denser distributions and clear boundaries of objects.

root mean square error inverse depth (iRMSE), and mean absolute error inverse depth (iMAE). We mainly focus on the RMSE when comparing methods because RMSE directly measures the error in depth, penalizes larger errors, and is the leading metric for depth completion. These four evaluation indicators are defined by the following formulas:

- Root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum \left(D_{pred} - D_{gt}\right)^2} \tag{13}$$

- Mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum \left|D_{pred} - D_{gt}\right| \tag{14}$$

- Root mean squared error of the inverse depth [1/km](iRMSE):

$$iRMSE = \sqrt{\frac{1}{n}\sum \left(\frac{1}{D_{pred}} - \frac{1}{D_{gt}}\right)^2} \tag{15}$$

- Mean absolute error of the inverse depth [1/km](iMAE):

$$iMAE = \frac{1}{n}\sum \left|\frac{1}{D_{pred}} - \frac{1}{D_{gt}}\right| \tag{16}$$

Since the point clouds are discrete samples of surfaces of objects or scenes, a metric such as the point-to-plane metric may better reflect surface reconstruction quality. To this end, we use the point-to-plane ICP algorithm to measure the quality of surface reconstruction. The Inlier-RMSE is used to measure the RMSE of all inline correspondences. Note that a lower Inlier-RMSE value represents better reconstruction quality. In order to evaluate the similarity of the overall structure of the generated point cloud, we introduce new evaluation metrics, i.e., CD as follows:

$$CD = d_{CD}\left(PC_1, PC_2\right) + d_{CD}\left(PC_2, PC_1\right) \tag{17}$$

TABLE I

ABLATION STUDY: PERFORMANCE ACHIEVED BY OUR NETWORK WITH AND WITHOUT EACH MODULE

Configuration	RMSE	MAE	iRMSE	iMAE	CD	POINT-to-PLANE (Inlier-Rmse)
Baseline	1408.80	513.06	7.63	3.01	0.21	0.1603
+Aggregation module	1224.91	409.69	4.69	1.95	0.16	0.1645
+Scene flow	1124.76	382.15	4.39	1.89	0.14	0.1558
+Reconstruction loss	**1091.99**	**371.56**	**4.21**	**1.83**	**0.12**	**0.1488**

3) Implementation Details: The depth values at the upper end of the depth map are all zero, and this section does not provide any depth information. Therefore, all our data (RGB, sparse depth, and dense depth map) are cropped from the bottom to a uniform 1216×256 size. Data enhancement operations are also applied to the training data, such as random flips and color adjustments. In the calculation of scene flow, we randomly sample 17,500 points in the point cloud of each frame as the input of the scene flow network, which is designed based on the HPLFlowNet [42]. Adam optimizer is applied during our training phase with 10^{-4} initial learning rate, which is decayed by 0.1 every 4 epochs. We train our network on a 1080Ti GPU with a batch size of 2 for about 60 hours, which is completed by PyTorch [43].

B. Ablation Study

We perform an extensive ablation study to verify the effectiveness of each module. The performance comparison of the proposed approach is shown in Table I. Specifically, we perform four ablation experiments, each of which is based on the addition of a new network element or module to the previous network configuration.

As listed in Table I, the result shows that the complete network achieves the best interpolation performance. For the baseline network, we take two consecutive sparse depth maps and the intermediate color image as the texture completion

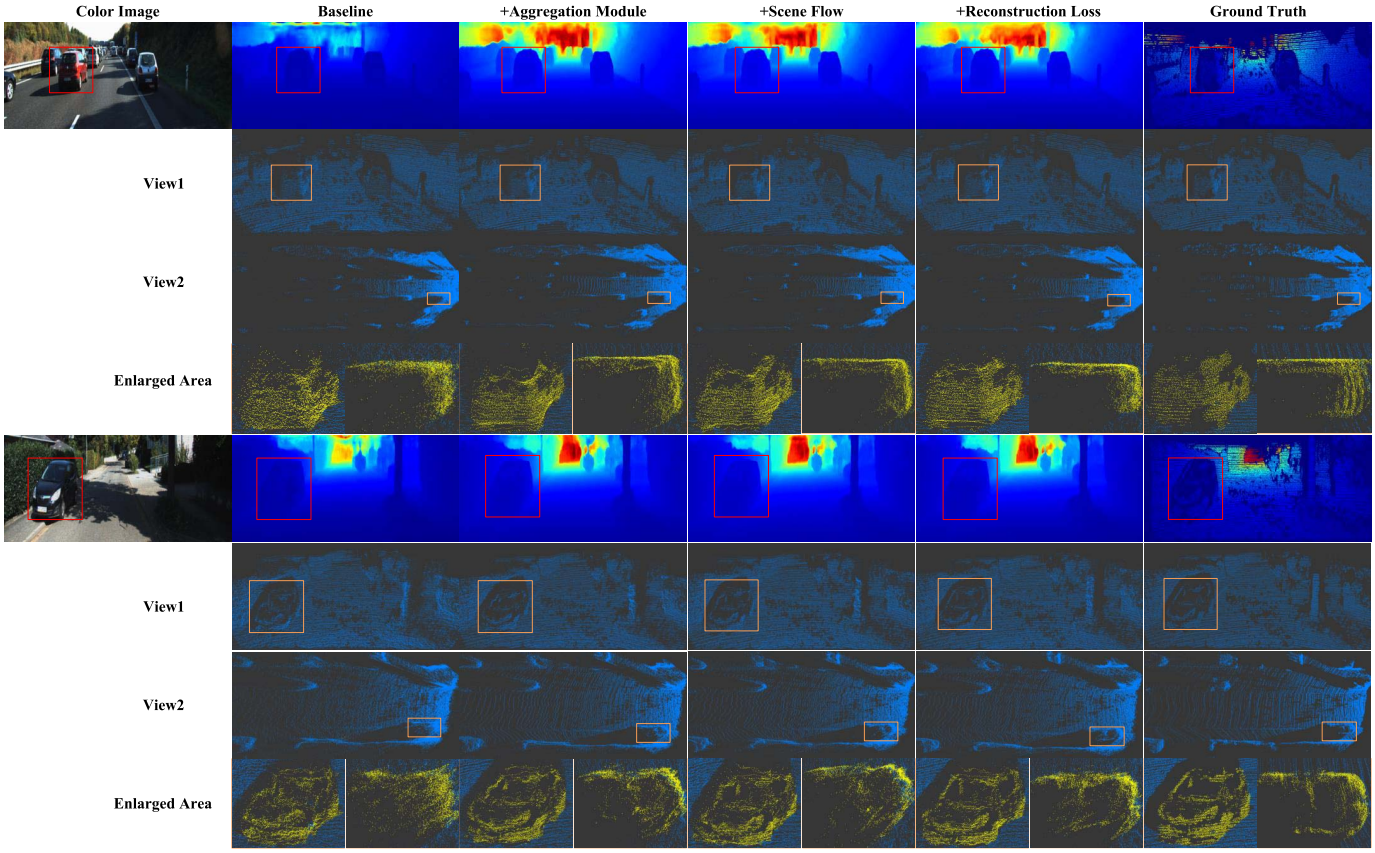


Fig. 7. Visual results of the ablation study. For each configuration, from top to bottom are depth maps corresponding to color images, point cloud view 1, point cloud view 2, and partially zoomed regions. Our method produces a reasonable distribution and shape of Pseudo-LiDAR point clouds.

branch's input and obtain the intermediate dense depth map. By comparing the experimental results, we have the following observations: 1) Without the spatial motion guidance, our multi-modal deep aggregation module can also produce good interpolation results, as it combines the features of the dual branch and is more conducive to the fusion of features. 2) Under the guidance of the scene flow containing motion information, we have greatly improved the performance of interpolation. This benefits from a better representation of spatial motion information. 3) Point cloud reconstruction constraints further improve interpolation performance. It can be observed that the value of our evaluation metrics decreases as the number of modules increases, which also proves the effectiveness of each of our network modules. To intuitively compare these different performances, we visualize the interpolated results of two scenes obtained by the above methods in Fig. 7. The complete network generates the most realistic details and distributions of the intermediate point cloud. Note that, in the enlarged area, the car's shape distribution obtained by the complete network is the most similar to the ground truth.

C. Comparison With State-of-the-Art

We evaluate our model on the KITTI depth completion dataset. We show the comparison results with other state-of-the-art point cloud interpolation methods in Table II. Since

TABLE II

QUANTITATIVE EVALUATION RESULTS OF THE TRADITIONAL INTERPOLATION METHOD, SUPER SLOMO [33], MOTION + COMPLETION [19], PLIN [5], AND OUR METHOD

Method	RMSE	MAE	iRMSE	iMAE	CD	POINT-to-PLANE (Inlier-Rmse)
Traditional Interpolation	12552.46	3868.80	-	-	-	-
Super Slomo [33]	16055.19	11692.72	-	-	27.38	-
Motion+Completion [19]	1636.10.19	422.00	**4.03**	**1.63**	0.36	0.1584
PLIN [5]	1168.27	546.37	6.84	3.68	0.21	0.1890
Ours	**1091.99**	**371.56**	4.21	1.83	**0.12**	**0.1488**

PLIN is a pioneer work in this field, it is used for comparison in this section. Inspired by the idea of "motion compensation" in LOAM [30], we compared the method of "projecting using motion parameters" + "sparse LiDAR completion" with our approach. Using the GPS/IMU data provided by the KITTI dataset, we got frame-to-frame motion parameters $(R|T)$. With the camera's intrinsic and extrinsic parameters, we projected the sparse depth map to the LiDAR coordinate system consistent with the motion parameters, and then used the motion parameters to transform and project the data from $t-1$ to t. Furthermore, we completed the projected data at time t using the state-of-the-art pre-trained depth completion network [19]. The comparison results are shown in Table II. In addition, we also compare the traditional average depth interpolation method and video interpolation method. For the compared traditional interpolation method, vehicles' the motion can be considered a pure translation with uniform speed, which usually happens in the self-driving scenes.

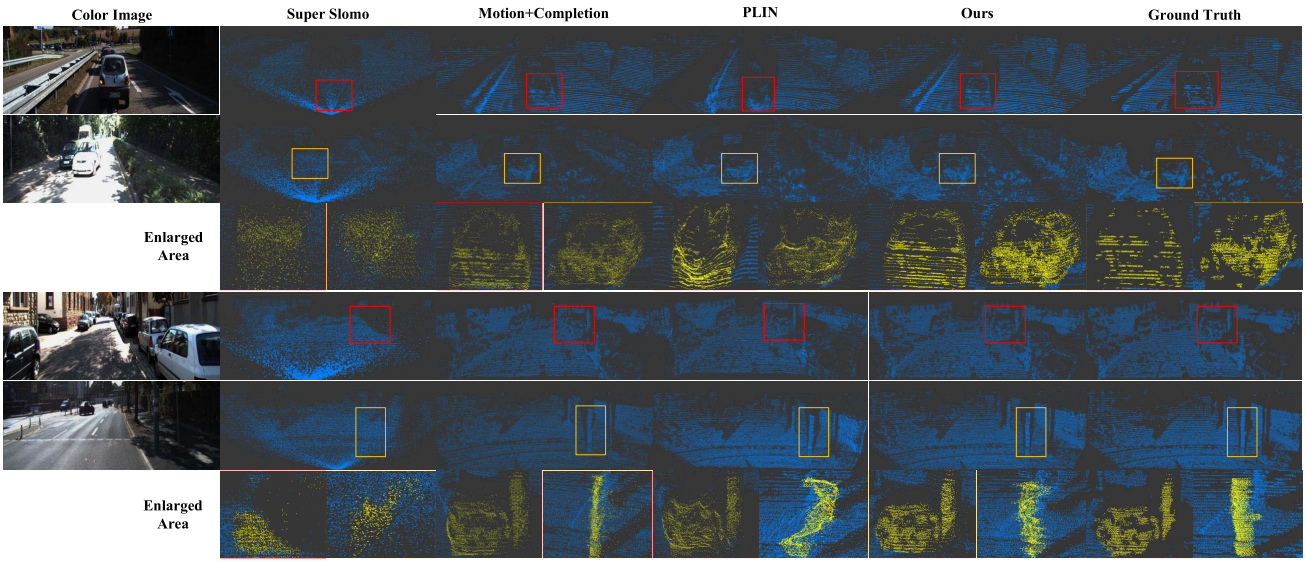


Fig. 8. Visual comparison with state-of-the-art methods (better viewed in color). For each scene, we show the color image and the Pseudo-LiDAR point cloud obtained by different methods. In the third row, we enlarged the local regions for better observation. Our method produces a more reasonable distribution and shape. In the zoomed regions, our method recovers better 3D details.

For the video frame interpolation method, the Super Slomo [33] network is retrained on the depth completion dataset.

1) Quantitative Comparison: We show some quantitative results to compare our approach with existing methods in Table II. Experimental results show that our approach is superior to other methods in learning the interpolation of the point cloud from RGBD data. In particular, we achieve state-of-the-art results in mainly concerned metrics, i.e., RMSE; the metrics to point cloud quality, i.e., CD and Inlier-RMSE. For the traditional method, the intermediate depth map is obtained by averaging consecutive depth maps. Its poor performance is understandable because the pixel values of continuous depth maps do not have a corresponding relationship. For the video frame interpolation method, since the motion relationship between depth maps cannot be obtained, it is difficult to generate satisfactory results. In the "motion" + "completion" method, we use the real motion parameters and the pre-trained depth completion network [19]. Even with this configuration, our approach outperforms the projection-based method on quantitative evaluation, which demonstrates that the effectiveness of the proposed interpolation module and learning architecture. We can conclude that our method can adaptively perceive the dynamic environment of the self-driving scene, but the projection-based method only provides an overall offset of the current environment. Guided by color images and bidirectional optical flows, PLIN is designed for the task of point cloud interpolation and achieves good performance, but it lacks the point cloud supervision and spatial motion representation. Compared with these methods, our approach improves the Pseudo-LiDAR point cloud interpolation task by adopting the scene flow, 3D space supervision mechanism, and multi-modal deep aggregation module. As a result, our approach outperforms these classical methods.

2) Visual Comparison: For visual comparison, we compare different interpolation results in Fig. 8. In PLIN, it is shown that the traditional interpolation method cannot handle the point cloud interpolation problem well, and the visual performance is poor. Therefore, we only show the comparison between Super Slomo [33], motion + completion, PLIN, our approach, and ground truth. As illustrated in Fig. 8, our approach produces a more reasonable distribution and shape than PLIN and motion + completion. The whole distribution of Pseudo-LiDAR point clouds is more similar to that of the ground truth point clouds. In the zoomed regions, our method recovers better 3D details for car, road, and tree. This is benefited from optimized motion representation, 3D space supervision mechanism, and model structure.

V. Conclusion

In this paper, we propose a novel Pseudo-LiDAR point cloud interpolation network with better interpolation performance than previous works. To represent the spatial motion information more accurately, we use point cloud scene flow to guide the point cloud interpolation task. We design a multi-modal deep aggregation module to facilitate the efficient fusion of features of the dual branch. In addition, we adopt a supervision mechanism in 3D space to supervise the generation of Pseudo-LiDAR point clouds. As the benefits of the optimized motion representation, training loss function, and model structure, the proposed pipeline significantly improves interpolation performance. We have shown the effectiveness of our approach on the KITTI dataset, outperforming the state-of-the-art point cloud interpolation techniques with a large margin.

References

[1] X. Yang, H. Luo, Y. Wu, Y. Gao, C. Liao, and K.-T. Cheng, "Reactive obstacle avoidance of monocular quadrotors with online adapted depth prediction network," *Neurocomputing*, vol. 325, pp. 142–158, Jan. 2019.

[2] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[3] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[4] D. Shin, Z. Ren, E. Sudderth, and C. Fowlkes, "3D scene reconstruction with multi-layer depth and epipolar transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2172–2182.

[5] H. Liu, K. Liao, C. Lin, Y. Zhao, and M. Liu, "PLIN: A network for pseudo-LiDAR point cloud interpolation," *Sensors*, vol. 20, no. 6, p. 1573, Mar. 2020.

[6] M. Liu, M. Salzmann, and X. He, "Discrete-continuous depth estimation from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 716–723.

[7] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2144–2158, Nov. 2014.

[8] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2650–2658.

[9] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1119–1127.

[10] J. Xie, R. Girshick, and A. Farhadi, "Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 842–857.

[11] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 270–279.

[12] A. Atapour-Abarghouei and T. P. Breckon, "Extended patch prioritization for depth filling within constrained exemplar-based RGB-D image completion," in *Proc. Int. Conf. Image Anal. Recognit.* Cham, Switzerland: Springer, 2018, pp. 306–314.

[13] M. Kulkarni and A. N. Rajagopalan, "Depth inpainting by tensor voting," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 30, no. 6, pp. 1155–1165, Jun. 2013.

[14] A. Atapour-Abarghouei and T. P. Breckon, "Depthcomp: Real-time depth image completion based on prior semantic scene segmentation," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2017.

[15] A. Atapour-Abarghouei, G. P. de La Garanderie, and T. P. Breckon, "Back to butterworth—A Fourier basis for 3D surface relief hole filling within RGB-D imagery," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2813–2818.

[16] M. Camplani and L. Salgado, "Efficient spatio-temporal hole filling strategy for kinect depth maps," *Proc. SPIE*, vol. 8290, Feb. 2012, Art. no. 82900E.

[17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant CNNs," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 11–20.

[18] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 1–8.

[19] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 3288–3295.

[20] A. Eldesokey, M. Felsberg, and F. S. Khan, "Propagating confidences through CNNs for sparse data regression," 2018, *arXiv:1805.11913*. [Online]. Available: http://arxiv.org/abs/1805.11913

[21] Y.-K. Huang, T.-H. Wu, Y.-C. Liu, and W. H. Hsu, "Indoor depth completion with boundary consistency and self-attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–9.

[22] M. Jaritz, R. D. Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with CNNs: Depth completion and semantic segmentation," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 52–60.

[23] J. Qiu *et al.*, "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3313–3322.

[24] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2D-3D representations for depth completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 10023–10032.

[25] C. Dinesh, I. V. Bajic, and G. Cheung, "Adaptive nonrigid inpainting of three-dimensional point cloud geometry," *IEEE Signal Process. Lett.*, vol. 25, no. 6, pp. 878–882, Jun. 2018.

[26] Z. Fu, W. Hu, and Z. Guo, "3D dynamic point cloud inpainting via temporal consistency on graphs," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2020, pp. 1–6.

[27] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "GRNet: Gridding residual network for dense point cloud completion," 2020, *arXiv:2006.03761*. [Online]. Available: http://arxiv.org/abs/2006.03761

[28] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4087–4100, Aug. 2019.

[29] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jun. 29, 2020, doi: 10.1109/TPAMI.2020.3005434.

[30] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," *Robot., Sci. Syst.*, vol. 2, no. 9, pp. 1–9, 2014.

[31] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4463–4471.

[32] T. Peleg, P. Szekely, D. Sabo, and O. Sendik, "IM-Net for high resolution video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2398–2407.

[33] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super SloMo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9000–9008.

[34] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3703–3712.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[36] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 529–537.

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*. [Online]. Available: https://arxiv.org/abs/1706.02413

[38] M. Kiefel, V. Jampani, and P. V. Gehler, "Permutohedral lattice CNNs," 2014, *arXiv:1412.6618*. [Online]. Available: https://arxiv.org/abs/1412.6618

[39] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8445–8453.

[40] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 605–613.

[41] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[42] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "HPLFlowNet: Hierarchical permutohedral lattice FlowNet for scene flow estimation on large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3254–3263.

[43] A. Paszke *et al.*, "Automatic differentiation in Pytorch," in *Proc. Neural Inf. Process. Syst. (NeurIPS) Workshop*, 2017.

Haojie Liu was born in Zhoukou, China. He is currently pursuing the master's degree in signal and information processing with the Institute of Information Science, Beijing Jiaotong University, Beijing, China.

His current research interests include image processing, depth completion, and point cloud processing.

Kang Liao (Graduate Student Member, IEEE) received the B.S. degree in software engineering from Shaanxi Normal University, Xi'an, China, in 2017. He is currently pursuing the Ph.D. degree in signal and information processing with the Institute of Information Science, Beijing Jiaotong University, Beijing, China.

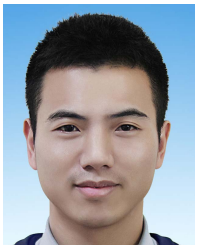
His current research interests include image and video processing, 3-D scene understanding, and adversarial learning.

Chunyu Lin (Member, IEEE) received the Ph.D. degree from Beijing Jiaotong University (BJTU), Beijing, China, in 2011.

From 2009 to 2010, he was a Visiting Researcher with the ICT Group, Delft University of Technology, Netherlands. From 2011 to 2012, he was a Post-Doctoral Researcher with the Multimedia Laboratory, Gent University, Belgium. He is currently a Professor with BJTU. His research interests include image/video compression and robust transmission, 3-D video coding, virtual reality video processing, and ADAS.

Yao Zhao (Senior Member, IEEE) received the B.S. degree from the Radio Engineering Department, Fuzhou University, Fuzhou, China, in 1989, the M.E. degree from the Department, Southeast University, Nanjing, China, in 1992, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University (BJTU), Beijing, China, in 1996.

He became an Associate Professor with BJTU in 1998 and became a Professor in 2001. From 2001 to 2002, he was a Senior Research Fellow with the Information and Communication Theory Group, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands. He is currently the Director of the Institute of Information Science, BJTU. His current research interests include image/video coding, digital watermarking and forensics, and video analysis and understanding. He is a fellow of the IET. He was named as a Distinguished Young Scholar by the National Science Foundation of China in 2010. He was elected as a Chang Jiang Scholar of Ministry of Education of China in 2013. He serves on the Editorial Board of several international journals, including as an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS, a Senior Associate Editor of the IEEE SIGNAL PROCESSING LETTERS, and an Area Editor of *Signal Processing: Image Communication*.



Yulan Guo (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees from the National University of Defense Technology (NUDT), in 2008 and 2015, respectively.

He is currently an Associate Professor. He has authored over 100 articles in journals and conferences, such as the IEEE TPAMI and IJCV. His current research interests include 3D vision, particularly on 3D feature learning, 3D modeling, 3D object recognition, and scene understanding. He received the ACM China SIGAI Rising Star Award in 2019 and the Wu-Wenjun Outstanding AI Youth Award in 2019. He served as an Associate Editor for *IET Computer Vision* and *IET Image Processing*, a Guest Editor for IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), and an Area Chair for several conferences, such as CVPR'21 and ICCV'21.