# BEV@DC: Bird's-Eye View Assisted Training for Depth Completion

Wending Zhou[1,2], Xu Yan[1,2], Yinghong Liao[1,2], Yuankai Lin[3], Jin Huang[4],
Gangming Zhao[5], Shuguang Cui[2,1], Zhen Li[2,1*]
[1]FNii, CUHK-Shenzhen, [2]SSE, CUHK-Shenzhen, [3]Huazhong University of Science and Technology,
[4]Cardiff University, [5]The University of Hong Kong
{wendingzhou@link.,lizhen@}cuhk.edu.cn

## Abstract

*Depth completion plays a crucial role in autonomous driving, in which cameras and LiDARs are two complementary sensors. Recent approaches attempt to exploit spatial geometric constraints hidden in LiDARs to enhance image-guided depth completion. However, only low efficiency and poor generalization can be achieved. In this paper, we propose **BEV@DC**, a more efficient and powerful multi-modal training scheme, to boost the performance of image-guided depth completion. In practice, the proposed BEV@DC model comprehensively takes advantage of LiDARs with rich geometric details in training, employing an enhanced depth completion manner in inference, which takes only images (RGB and depth) as input. Specifically, the geometric-aware LiDAR features are projected onto a unified BEV space, combining with RGB features to perform BEV completion. By equipping a newly proposed point-voxel spatial propagation network (PV-SPN), this auxiliary branch introduces strong guidance to the original image branches via 3D dense supervision and feature consistency. As a result, our baseline model demonstrates significant improvements with the sole image inputs. Concretely, it achieves state-of-the-art on several benchmarks, e.g., ranking **Top-1** on the challenging KITTI depth completion benchmark.*

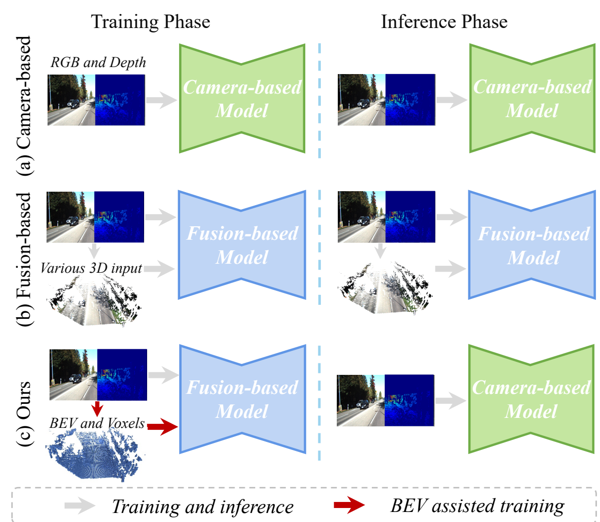


Figure 1. **BEV assisted training.** (a) Previous camera-based methods that take RGB and depth input. (b) Previous fusion-based methods introduce extra inputs and computation in both training and inference. (c) Our method takes additional LiDAR as input for assisted training. Only the 2D inputs are used during the inference, which reduces the computational burden.

## 1. Introduction

Dense depth estimation plays an essential role in various 3D vision tasks and self-driving applications, *e.g.*, 3D object detection and tracking, simultaneous localization and mapping (SLAM), and structure-from-motion (SFM) [14, 17, 19, 33, 37]. With the aid of outdoor LiDAR sensors or indoor RGBD cameras, 3D vision applications acquire depth maps for further industrial usage. However, the depth sensors cannot provide dense pixel-wise depth maps since their output is sparse and has numerous blank regions, especially in outdoor scenes. Therefore, it is necessary to fill the void areas of the depth maps in practice.

Recent depth completion methods [4, 12, 21, 47] leverage the RGB information as guidance since the RGB images contain scene structures, *e.g.*, textures, and monocular features, *e.g.*, vanishing points, to provide the cues for the missing pixels. However, the camera-based methods apply the 2D convolution on the irregularly distributed depth values, resulting in an implicit yet ineffective exploration of underlying 3D geometry, *i.e.*, over-smooth at the boundary of objects. Considering the deployment of cameras and LiDAR in commercial cars and the recent trend of cross-modal learning in the vision community, some methods [2, 3, 28, 41] introduce explicit 3D representations, *i.e.*, LiDAR point clouds generated by sparse depth,

[*]Corresponding author

to complement 2D appearance features with 3D structured priors. Despite the improvements, the fusion-based approaches still have the following issues: **1)** The 3D feature extraction and fusion are not efficacious, especially the critical spatial correlations between a depth point and its neighbors, which significantly affects the completion performance. **2)** Fusion-based methods are computation-intensive while processing sparse depths, RGB images, and additional 3D input such as LiDAR information, either occupying more memory storage or consuming more time in inference, which hinders real-time applications.

To address the above issues, we seek to boost image-guided depth completion performance by exploiting 3D representations via a more efficient and effective cross-representation training scheme. In training, we design an auxiliary LiDAR branch consisting of LiDAR encoder, cross-representation BEV decoder (CRBD) and point-voxel spatial propagation network (PV-SPN). Initially, we preprocess each LiDAR scan with the assigned voxel cells to alleviate the irregularity and sparseness and then extract its multi-scale features. After that, these features will be projected onto a unified BEV space. The following CRBD utilizes the above multi-scale BEV features and the ones from the camera branch to perform BEV fusion and completion. After that, the BEV completion is interpolated into the 3D space, and a point-voxel spatial propagation network is proposed to query the nearest neighbors for each coarse voxel and performs feature aggregation on all the adjacent points from LiDAR, refining the 3D geometric shapes. Moreover, to tackle the extra computational burden from the LiDAR branch, this plug-and-play component is only exploited in the training phase, enhancing the original camera branch through feature consistency and end-to-end backpropagation. Consequently, the trained model is independent of additional LiDAR inputs during the inference.

Compared with previous fusion-based methods, our proposed framework has the following advantages: **1) Generality**: Our plug-and-play solution can be incorporated into several camera-based depth completion models; **2) Flexibility**: The processing module for LiDAR representations only exists during training and is discarded in inference, as shown in Fig. 1(c), compared with previous camera-based models (a) and fusion-based models (b). There is no additional computational burden in the deployment. **3) Effectiveness**: It significantly boosts the performance upon the baseline approach, achieving state-of-the-art results on several benchmarks. To sum up, the main contributions are summarized as follows:

- **B**ird's-**E**ye **V**iew **A**ssisted **T**raining for **D**epth **C**ompletion (BEV@DC) is proposed, which assists camera-based depth completion with LiDAR representation during the training phase.
- Cross-representation BEV decoder (CRBD) and point-voxel spatial propagation network (PV-SPN) are proposed to gain fine-grained 3D geometric shapes and provide strong guidance to the RGB branch.
- Our solution achieves state-of-the-art on both outdoor KITTI depth completion benchmark and indoor NYU Depth v2 dataset.

## 2. Related Work

**RGB-Guided Depth Completion.** Compared with the unguided methods without the RGB inputs [8, 9, 15] , RGB-guided ones [4, 12, 21, 47] benefit from useful image features, *e.g.*, semantics, resulting in superior performances in the depth completion task. RGB-guided methods can be divided into two categories. One pattern is to utilize multiple branches to process the depth and RGB inputs, respectively, then fuse the processed information at different scales [28, 31, 38, 47, 50]. KBNet [38] presents a calibrated back-projection module to back-project spatial encodings of the depth map and RGB image onto 3D space. RigNet [47] introduces a repetitive design to RGB-guided networks to recover depth values. Another pattern [4, 5, 21, 25, 42] is that all the inputs are fed into a simple UNet [29] and then processed by the spatial propagation network (SPN) [23]. CSPN [5] is the first work that applies the SPN to depth completion, where the SPN learns spatial correlations between a depth point and its neighbors via propagation with the affinity matrix. Compared to the original SPN, CSPN uses a recursive convolution operation to increase efficiency. CSPN++ [4] further improves the CSPN by learning the adaptive convolutional kernel sizes and the number of iterations for SPN. Since CSPN and CSPN++ involve the unnecessary use of irrelevant local neighbors, non-local SPN [25] is proposed to handle relevant non-local neighbors during propagation. Recently, DySPN [21] presents a dynamic attention-based SPN that learns an adaptive affinity matrix by decoupling the neighborhood into parts in terms of the distances. Though these image-guided approaches are improved over time, they lack the ability to understand the 3D geometries and result in over-smooth boundaries.

**Fusion-Based Depth Completion.** Since 2D convolution fails to extract the 3D geometric information effectively, some depth completion methods [2, 3, 28, 41, 50] resort to explicit 3D representations. 2D-3D FuseNet [3] consists of two sub-networks that learn 2D and 3D representations via the multi-scale 2D convolutions and continuous convolutions, respectively and then fuse 2D and 3D representations into the 2D image space. PwP [41] predicts the surface normals, coarse depth, and confidence of LiDAR inputs simultaneously and feeds them to the diffusion refinement module to obtain the final results. DeepLIDAR [28] utilizes the surface normals as the intermediate representation and further effectively fuses the sparse depth and the dense color image via a modified encoder-decoder structure. ACM-

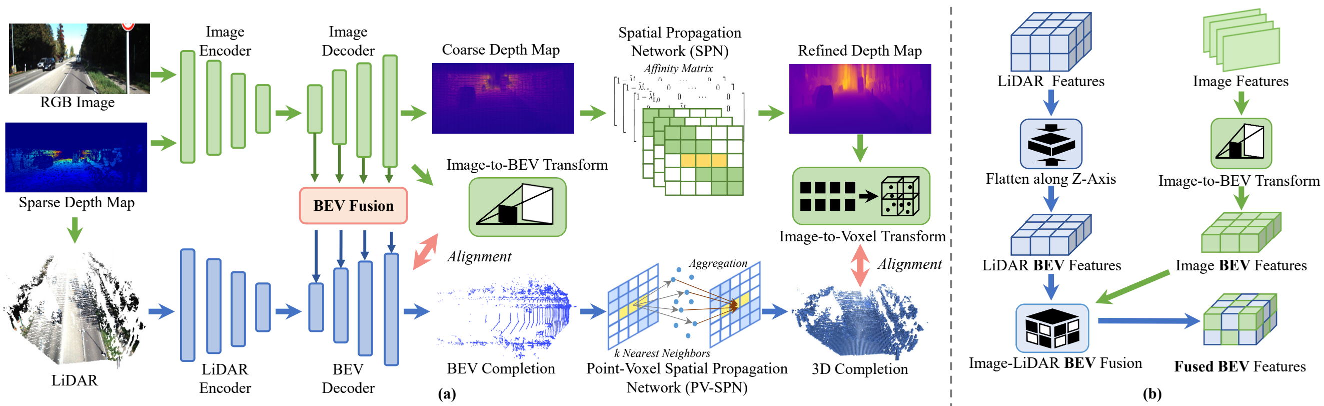


Figure 2. **Architecture of BEV@DC.** (a) There are two branches in the framework, namely camera and LiDAR branches. The former takes RGB image and sparse depth as input, which first produces a coarse depth completion through a 2D-UNet. Then a refined depth map is achieved by the assistance of auxiliary 3D completion together with a spatial propagation network (SPN). The LiDAR branch takes the point cloud as input, aggregates camera features in a BEV plane (refer to (b) for details), and conducts 3D completion through a point-voxel spatial propagation network (PV-SPN). (b) LiDAR and camera features are projected into a unified BEV space and fused.

Net [50] extracts the observed contextual information in a graph propagation manner. Recently, GAENet [2] learns the geometric-aware embedding from sparse LiDARs and further fuses the embedding with the 2D appearance features from RGB images to estimate dense depths. Although these fusion-based methods improve performance in a certain aspect, they introduce a huge computational burden at the same time, which inherently affects their real-world applications, such as autonomous driving.

**LiDAR Representation Learning.** The form of LiDAR data is represented as 3D point clouds, which are sparse and irregular scatter points in Euclidean space. To capture the geometric details in LiDAR point clouds, previous approaches learn the representation in the following manners. 1) Point-based methods: they directly learn 3D geometric details through point-wise MLPs [26, 45, 51], local aggregations [26, 27, 34], and non-local operators [46, 49]. 2) Voxel-based methods transform the point cloud into 3D voxel grids and apply 3D convolution. To accelerate the model speeds, the following approaches [11, 43, 44, 52] exploit sparse convolution that only calculate in the non-empty voxels. 3) Besides, there are projection-based methods, they project points onto 2D images by plane projection [1,18,32], spherical projection [39,40], or BEV projection [48], and thus the 2D-CNN can play a normal role. In this paper, we adopt voxel-based architecture in our LiDAR branch since it better balances effectiveness and efficiency.

# 3. Method

## 3.1. Overview

This paper focuses on boosting camera-based depth completion, which aims to generate dense depth maps with sparse depth maps and corresponding RGB images. To introduce the 3D geometric guidance to the network, we design an auxiliary LiDAR branch to boost the performance.

The architecture of our framework is illustrated in Fig. 2(a). There are two branches in our framework, namely the camera and LiDAR branches. The camera branch adopts traditional U-Net [29] architecture to perform coarse depth completion and refines the results with a spatial propagation network (SPN) [21]. The LiDAR point cloud, gained from the sparse depth map, is fed into the LiDAR encoder and the multi-scale BEV features are obtained. The cross-representation BEV decoder (CRBD) takes these features as the input and generates the BEV completion map in a cascaded manner by fusing the camera features. To perform a fine-grained 3D completion, we present a point-voxel spatial propagation network (PV-SPN). The outputs of BEV and 3D completion parts maintain feature consistency with that of the camera branch. The auxiliary branch is only applied in training and can be discarded in inference, which prevents the extra computational burden.

## 3.2. Multi-Scale BEV Generation

We obtain the 3D representations by transforming the sparse depth map to 3D coordinates, *i.e.*, LiDAR point clouds. To aggregate the camera and LiDAR features into a unified BEV space, we then transform them into the same BEV plane.

**LiDAR Transformation.** Given an input image with a size of $(H, W)$ and a sparse depth map $D$, we first generate the image coordinates $\mathbb{C}$ according to the depth map,

$$\mathbb{C} = \{(u, v, D_{uv}) |\, u \in [1, W], v \in [1, H]\}. \tag{1}$$

We then transform the image coordinates $\mathbb{C}$ into the 3D space, utilizing the camera intrinsic and extrinsic matrices

$K \in \mathbb{R}^{4\times4}$ and $T \in \mathbb{R}^{4\times4}$. Specifically, given the $i$-th image coordinate $\mathbb{C}_i = (u_i, v_i, d_i)$, its coordinate $(x_i, y_i, z_i)$ in the world system is calculated as

$$[x_i, y_i, z_i, 1]^T = T^{-1} \cdot K^{-1} \cdot [u_i \times d_i, v_i \times d_i, d_i, 1]^T. \quad (2)$$

After the transformation, we obtain a LiDAR point cloud $P = \{(x_i, y_i, z_i)\}_{i=1}^N$.

**BEV Features.** Unlike previous works [2,3] that utilize the point-based methods to process the LiDAR representations, we exploit more efficient sparse convolutions [11] to mine the LiDAR information. Foremost, we transform the original LiDAR coordinates to a sparse volumetric representation. Specifically, all the points are shifted to the local coordinate system with the geometric center as the origin. Then we normalize the points into a unit sphere by dividing all the coordinates by $max||P||_2$, and scaling the points to the range of $[0, 1]$. The normalized coordinates are denoted as $\hat{P}$. Subsequently, we transform the normalized point cloud to a voxel representation with the resolution $r$:

$$p_i^* = (x_i^*, y_i^*, z_i^*) = (\lfloor \hat{x}_i \times r \rfloor, \lfloor \hat{y}_i \times r \rfloor, \lfloor \hat{z}_i \times r \rfloor), \quad (3)$$

$$f_m^* = \frac{1}{N_m} \sum_{i=1}^{N} \mathbb{I}[x_i^* = \hat{x}_m, y_i^* = \hat{y}_m, z_i^* = \hat{z}_m] \cdot p_i, \quad (4)$$

where $\lfloor \cdot \rfloor$ is the floor function, and $\mathbb{I}(\cdot)$ is a binary indicator of whether $p_i^*$ belongs to the $m$-th voxel grid or not. $N_m$ is the number of points in the $m$-th voxel, and the original point coordinates are averaged as the features of each voxel. Via the operations in Eqn. (3) and (4), only the non-empty voxels are preserved ($N_m > 0$) in a hash table. The convolution operation only conducts on the non-empty voxels. In this way, the point cloud is in a larger volumetric resolution while maintaining the computational efficiency. The whole process of transforming depth images into the sparse voxels representation is referred to as Image-to-Voxel Transform in Fig. 2(a). The sparse voxels are input to a sparse convolution-based encoder that extracts the multi-scale features under different encoder scales. We then perform the average pooling operations in the height dimension to squeeze the feature maps, producing the multi-scale BEV features of LiDAR (*i.e.*, $\{F_l^L\}_{l=1}^L$).

**Camera Features.** We extract the multi-scale camera features from the U-Net decoder and transform them to the 3D space by Equation (2) and (3), which is shown as Image-to-BEV Transform in Fig. 2. Another set of the BEV features (*i.e.*, $\{F_l^C\}_{l=1}^L$) is obtained by pooling within the height dimension.

### 3.3. Cross-Representation BEV Decoder

The multi-scale camera and LiDAR BEV features are the inputs of our decoder architecture, performing completion in the BEV space. We adopt the U-Net decoder to upsample the features from the last encoder layer step-by-step. As shown in Fig. 2(b), the feature map $F_l^{Bev}$ of the $l$-th decoder layer is produced by

$$F_l^{Bev} = \mathcal{A}(\mathcal{U}(F_{l-1}^{Bev}); F_{L-l+1}^C; F_{L-l+1}^L) \quad (5)$$

where $\mathcal{A}(\cdot; ...; \cdot)$ and $\mathcal{U}(\cdot)$ are the fusion and upsampling operations, respectively. The feature map of the first decoder is skip-connected to the last encoder layer, $F_1^{Bev} = F_L^L$. The completion results in the BEV space $D^{Bev}$ is obtained by passing the feature map from the last decoder layer to a linear classifier.

### 3.4. Propagation for 3D Completion

To provide the camera branch with more fine-grained guidance, we propose a point-voxel spatial propagation network (PV-SPN) to project the BEV completion results to 3D voxels and refine the dense 3D completion with propagation.

**Revisit Spatial Propagation Network.** Spatial propagation network (SPN) [23] is widely used in the previous sparse-to-dense depth completion methods, which aims at refining the initial depth prediction in a recursive manner. With the initial depth completion input, the SPN refines the depth in several iteration steps, updating each pixel value via the aggregation of neighboring pixels and the inclusion of more detailed and accurate structure information. Specifically, the propagation process in the previous SPN is formulated as:

$$d_{i,j}^l = \mathcal{A}(D^{l-1} | A_{(i,j)}, \mathcal{N}(i,j)) \quad (6)$$

$$= a_{(i,j)\to(i,j)} d_{(i,j)}^{l-1} + \sum_{\mathcal{N}(i,j)} a_{(i,j)\to(p,q)} d_{(p,q)}^{l-1}, \quad (7)$$

where $d_{i,j}^l \in D^l$ denotes the depth value at pixel $(i, j)$ in the $l$-th iteration. $\mathcal{A}(\cdot)$ is a fusion function and $\mathcal{N}(i, j)$ represents the neighboring pixels of pixel $(i, j)$. The core component of SPN is the affinity matrix $A_{(i,j)}$, whose element $a_{(i,j)\to(p,q)} \in A_{(i,j)}$ contains a relational weight between pixels $(i, j)$ and $(p, q)$. These weights in an affinity matrix is calculated through the ad-hoc relationships [23] or in a learnable manner [21]. Moreover, the previous methods search the neighboring pixels $\mathcal{N}(i, j)$ by different coordinate shifts, which is further formulated as:

$$\mathcal{N}(i,j) = \{(i+m, j+n) | (m,n) \in \mathcal{S}(D|i,j)\}, \quad (8)$$

where $\mathcal{S}(\cdot)$ is a neighborhood searching function based on the depth map $D$ and pixel $(i, j)$. For instance, the original SPN [23] performs propagation in a fixed neighborhood coordinate set, *i.e.*, $\{-1, 0, 1\}$, and thus the $\mathcal{S}(\cdot)$ equals to searching all pixel in a $3 \times 3$ kernel. Further studies exploit different searching functions, such as searching in different kernel sizes in parallell [4] or nonlocal neighborhoods [25].

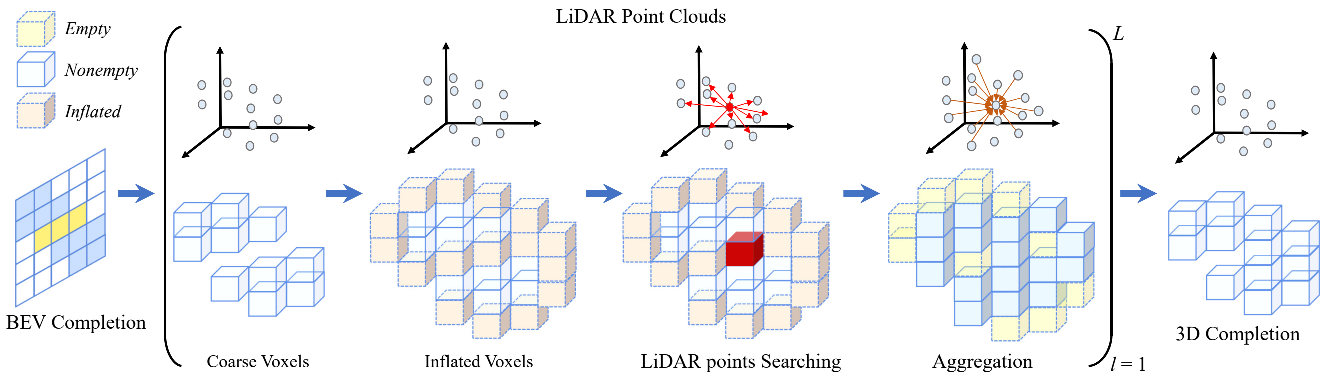


Figure 3. **Point-Voxel Spatial Propagation Network (PV-SPN).** It takes BEV completion as input and generates coarse voxels through height-dimensional ($Z$-Axis) MLPs. After that, it inflates the initial voxels and gains more nonempty grids. Then, it searches the K-nearest neighbors from the original LiDAR point cloud. At the end of each propagation, it conducts aggregation to predict the occupation of each grid. The PV-SPN will iteratively conduct L times.

**Point-Voxel Spatial Propagation Network.** PV-SPN takes the coarse voxel grids as input and refines the dense 3D completion results. As shown in Fig. 3, we first apply MLPs on the BEV completion $D^{Bev}$ along the height dimension where the original BEV is extended within the height dimension, and a coarse dense completion $V^{3D}$ is obtained. Subsequently, the goal of PV-SPN is to refine the dense 3D completion results via the spatial propagation. A naive implementation way is to extend SPN [23] (Eqn. (7) and (8)) to the 3D operation directly, *i.e.*, searching neighborhood through the 3D kernel and aggregating the neighboring voxels. However, such a manner is extremely time-consuming, since a large proportions of voxels are invalid in the 3D dense volume. Exploiting the sparse convolution might be more efficient, but it has poor generalization to generate invisible and new voxels.

To address this problem, we attempt to combine voxels and point cloud representations, conducting 3D completion via LiDAR guidance, as shown in Fig. 3. Initially, we conduct voxel inflation to obtain more nonzero voxels. We then transform the the coarse 3D completion $V^{3D}$ to the LiDAR coordinates, $V = \text{Voxel2LiDAR}(V^{3D})$. The operation Voxel2LiDAR$(\cdot)$ extracts the nonzero voxel coordinates and converts them to the LiDAR coordinates through an inversed process in Eqn. (3). The output $V$ is a point cloud containing the voxel centers. In each iteration, we search k-nearest neighbors (kNN) for each voxel center $v_i \in V$ towards the LiDAR point clouds $P$,

$$\mathcal{N}(v_i) = \{p_i = (x_i, y_i, z_i) \mid p_i \in \mathcal{S}^{3D}(v_i, P)\}, \quad (9)$$

where the searching function $\mathcal{S}^{3D}(\cdot)$ is defined as kNN. This way not only prevents the redundant computation in the invalid voxels but also utilizes the strong geometric guidance in the LiDAR point cloud. Furthermore, inspired by the graph convolution [36], a geometric-aware propagation in $l$-th iteration is applied between each voxel center and neighboring points as:

$$s_i^l = \mathcal{A}^{3D}(V^{l-1}|P, \mathcal{N}(v_i)) \quad (10)$$

$$= \sigma(\mathcal{T}_2\{\sum_{p_j \in \mathcal{N}(v_i)} a^{3D}_{v_i \to p_j} \mathcal{T}_1\{p_j\}\}), \quad (11)$$

where $s_i^l$ is a probability score of $i$-th voxel is nonempty. The operations $\mathcal{T}\{\cdot\}$ and $\sigma(\cdot)$ are MLP and sigmoid function, respectively. Notice that we design a learnable weight $a^{3D}_{v_i \to p_j}$ to aggregate the features adaptively:

$$a^{3D}_{v_i \to p_j} = \frac{\exp(\mathcal{T}((v_i - p_j) \mid\mid p_j))}{\sum_{p_k \in \mathcal{N}(v_i)} \exp(\mathcal{T}((v_i - p_k) \mid\mid p_k))}, \quad (12)$$

where $(\cdot \mid\mid \cdot)$ is the concatenation operation. The final output score $S = \{s_i^l\}$ generates voxels in the $l$-th iteration by the truncation on probability with a pre-defined threshold. Compared with the methods (Eqn. 7) that utilizes the relationships within adjacent pixels, our PV-SPN (Eqn. 11) fully explores the 3D geometric cues in the 3D LiDARs.

### 3.5. Training Objective and Inference

Following the previous camera-based approaches, we adopt the $L_1$ and $L_2$ losses for the camera branch. The ground truths of BEV and 3D completion are obtained by merging the consecutive frames of LiDAR sequences and the subsequent voxelization, which are aligned with the 2D ground truths. Notice that the voxels which cannot be observed in any LiDAR scan are labeled as 'ignored'. We then apply the Focal loss [20] in training and enable the network to focus on the nonempty grids. Moreover, the predictions of the camera and LiDAR branches are aligned to provide stronger guidance to the camera network, as shown in Fig. 2(a). In details, we extract the last feature from the

Table 1. Quantitative evaluation on KITTI DC benchmark. The upper part illustrates the results of camera-based methods and the middle part are those of fusion-based approaches. 'M', 'T' and 'C' denote 'modality', '3D representation' and 'camera', respectively. Only approaches published before 11/11/2022 are compared. The lower the metric values are, the better the estimation results are.

| Method | M-train | M-test | RMSE (mm) ↓ | MAE (mm) ↓ | iRMSE (1/km) ↓ | iMAE (1/km) ↓ | Reference (year) |
|---|---|---|---|---|---|---|---|
| CSPN [5] | C | C | 1019.64 | 279.46 | 2.93 | 1.15 | ECCV 2018 |
| FusionNet [10] | C | C | 772.87 | 215.02 | 2.19 | 0.93 | MVA 2019 |
| S2D [12] | C | C | 814.73 | 249.95 | 2.80 | 1.21 | TCI 2020 |
| DSPN [42] | C | C | 766.74 | 220.36 | 2.47 | 1.03 | ICIP 2020 |
| CSPN++ [4] | C | C | 743.69 | 209.28 | 2.07 | 0.90 | AAAI 2020 |
| NLSPN [25] | C | C | 741.68 | 199.59 | 1.99 | 0.84 | ECCV2020 |
| TWISE [16] | C | C | 840.20 | 195.58 | 2.08 | 0.82 | CVPR 2021 |
| GuideNet [31] | C | C | 736.24 | 218.83 | 2.25 | 0.99 | TIP 2021 |
| FCFRNet [22] | C | C | 735.81 | 217.15 | 2.20 | 0.98 | AAAI 2021 |
| PENet [13] | C | C | 730.08 | 210.55 | 2.17 | 0.94 | ICRA 2021 |
| RigNet [47] | C | C | 712.66 | 203.25 | 2.08 | 0.90 | ECCV 2022 |
| DySPN [21] | C | C | 709.12 | 192.71 | 1.88 | 0.82 | AAAI 2022 |
| DepthNormal [41] | T+C | T+C | 777.05 | 235.17 | 2.42 | 1.13 | ICCV 2019 |
| DeepLiDAR [28] | T+C | T+C | 758.38 | 226.50 | 2.56 | 1.15 | CVPR 2019 |
| FuseNet [3] | T+C | T+C | 752.88 | 221.19 | 2.34 | 1.14 | ICCV 2019 |
| ACMNet [50] | T+C | T+C | 744.91 | 206.09 | 2.08 | 0.90 | TIP 2021 |
| GraphCSPN [24] | T+C | T+C | 738.41 | 199.31 | 1.96 | 0.84 | ECCV 2022 |
| BEV@DC (ours) | T+C | C | **697.44** | **189.44** | **1.83** | **0.82** | CVPR2023 |

camera decoder, transform it to the BEV space, and align it with the last BEV feature. We also align the results of two SPNs by performing Image-to-Voxel Transform for the refined depth maps and guarantee their consistency with 3D completion results. Since the camera stream regresses the depth map while the LiDAR stream predicts the occupation states, it receives a direct alignment and only offers a hard constraint to each other. Therefore, we apply an additional classifier in the camera stream, which divides the depth value into several ranges, and then project the probabilities in each range into 3D voxels via Eqn. (3), which is aligned with the predictions of the LiDAR stream later. The $L_1$ loss is applied to constrain the consistency.

In training, since the camera features are fused into the LiDAR stream, which is supervised by BEV and 3D completion labels, the prior knowledge in 3D geometric shapes can inherently enhance the camera network through end-to-end backpropagation. Moreover, two feature constraints also boost knowledge transfer. After training, the enhanced camera stream can be independently deployed due to the unidirectional data flow. Our framework effectively improves performance while preventing the extra computational burden.

## 4. Experiments

This section describes the datasets, metrics, and implementation details in our experiments. We also demonstrate the effectiveness of our method by performing quantitative and qualitative analysis with the existing approaches. Moreover, the ablation studies show the effectiveness of the individual components of our method. Implementation details are provided in supplementary material.

### 4.1. Datasets and metrics

**Dataset.** KITTI Dataset is one of the largest real-world autonomous driving datasets [35], which contains over 90k RGB images with the corresponding LiDAR projected sparse depth measurements. It is split into 86k for training, 7k for validation, and 1k for testing by the official. The challenge of KITTI Depth Completion[1] lies in the sparsity of the input and ground truth depth, where only 5% pixels have valid depth values in the input and the 16% sparsity ground truths are annotated by accumulating 11 consecutive frames. The resolution of the image pairs is top cropped and center cropped to 1216×256 since there are nearly no LiDAR projections for the top 100 pixels.

NYUv2 Dataset [30] consists of paired color images and depth map captured from 464 indoor scenes by the Microsoft Kinect. We follow the previous works [5,21,25] that sample a subset of about 50k pairs from the official train split. The original images are downsampled to 320×240 and then center cropped to 304×228. We use the official test split, which contains 654 images for our evaluation.

**Metrics.** For KITTI depth completion dataset, we adopt the same evaluation metrics as the KITTI depth completion benchmark where root mean square error (RMSE), mean absolute error (MAE), inverse RMSE (iRMSE) and inverse MAE (iMAE) are utilized. While for NYUv2 dataset, RMSE, REL, and the percentage of pixels satisfying $\delta_\tau$ are

[1] KITTI Depth Completion Evaluation Benchmark

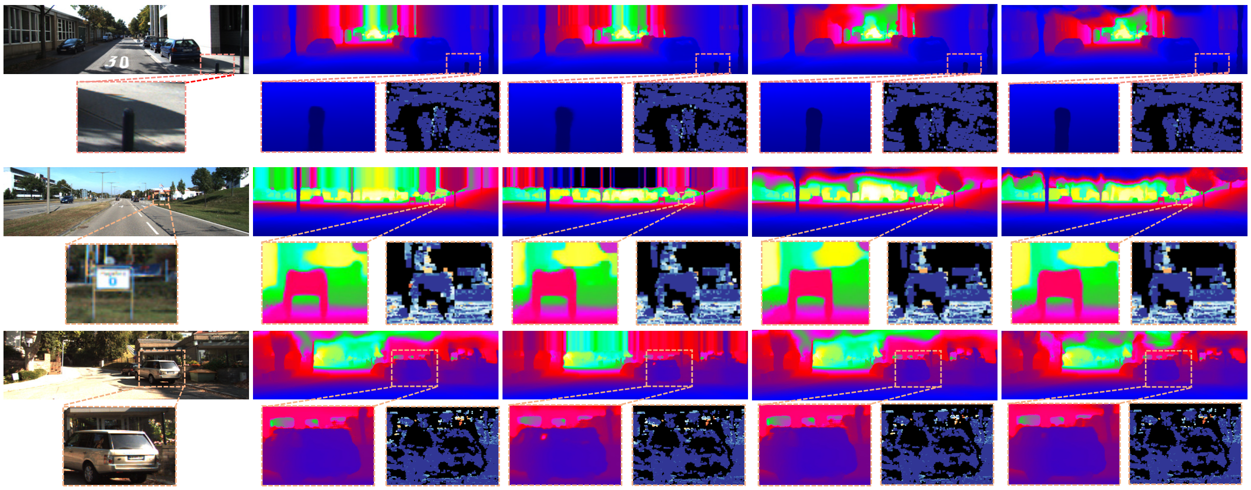


Figure 4. Qualitative results of BEV@DC on the KITTI DC benchmark. RigNet [47], ACMNet [50], and GraphCSPN [24] are selected for comparison. The zoom-in views' show closer details of the estimated depth maps and error maps (where darker is better). Our method is able to achieve clearer object boundaries and more fine-grained details with the help of introducing BEV level and 3D level alignment.

chosen for the evaluation metrics.

### 4.2. Comparison with State-of-the-arts

**KITTI Benchmark.** We evaluate our proposed BEV@DC on KITTI depth completion online benchmark where RMSE is adopted as the major evaluation metric to rank all the methods. The upper part of Tab. 1 illustrates the results of camera-based methods and the middle part are those of fusion-based approaches. Among all approaches, BEV@DC outperforms all the peer-reviewed methods in all evaluation metrics, including RMSE, MAE, iRMSE, and iMAE by the time of submission. In details, our proposed method obtains 697.44 mm in RMSE, which is significantly lower than that of the second-best method by 11.68 mm. Note that our model also outperforms other fusion-based models considerably [10, 28, 41]. Besides, Fusion-Net [10] employs an additional semantic segmentation network that is pre-trained on Cityscapes dataset [6], and DeepLiDAR [28] utilized the additional synthetic data produced by CARLA simulator [7]. In contrast, our method is solely trained on the KITTI dataset while achieving much better results, indicating that the proposed multi-modal training scheme can utilize the geometric-aware LiDAR features more effectively. We present the visualization results in Fig. 4, where RigNet [47], ACMNet [50], and GraphCSPN [24] are selected for comparison.

**NYUv2 Dataset.** Though our BEV@DC is proposed for outdoor scenarios, we also evaluate its generalization ability in indoor scenes. Tab. 2 displays the comparisons of the state-of-the-art on the NYUv2 dataset. The upper part of Tab. 2 illustrates the results of camera-based methods, and the middle part is those of fusion-based approaches. Our BEV@DC surpasses all the existing works, spanning camera and fusion-based approaches.

Table 2. Quantitative evaluation on NYUv2 dataset. The upper part illustrates the results of camera-based methods, and the middle part is those of fusion-based approaches.

| Method | RMSE (m) ↓ | REL (m) ↓ | $\delta_{1.25}$ ↑ | $\delta_{1.25^2}$ ↑ | $\delta_{1.25^3}$ ↑ |
|---|---|---|---|---|---|
| S2D [12] | 0.230 | 0.044 | 97.1 | 99.4 | 99.8 |
| CSPN [5] | 0.117 | 0.016 | 99.2 | 99.9 | 100.0 |
| CSPN++ [4] | 0.116 | - | - | - | - |
| FCFRNet [22] | 0.106 | 0.015 | 99.5 | 99.9 | 100.0 |
| GuideNet [31] | 0.101 | 0.015 | 99.5 | 99.9 | 100.0 |
| TWISE [16] | 0.097 | 0.013 | 99.6 | 99.9 | 100.0 |
| NLSPN [25] | 0.092 | 0.012 | 99.6 | 99.9 | 100.0 |
| RigNet [47] | 0.090 | 0.012 | 99.6 | 99.9 | 100.0 |
| DySPN [21] | 0.090 | 0.012 | 99.6 | 99.9 | 100.0 |
| DepthNormal [41] | 0.112 | 0.018 | 99.5 | 99.9 | 100.0 |
| DeepLiDAR [28] | 0.115 | 0.022 | 99.3 | 99.9 | 100.0 |
| ACMNet [50] | 0.105 | 0.015 | 99.4 | 99.9 | 100.0 |
| GraphCSPN [24] | 0.090 | 0.012 | 99.6 | 99.9 | 100.0 |
| BEV@DC (ours) | **0.089** | **0.012** | **99.6** | **99.9** | **100.0** |

### 4.3. Ablation Studies

**Design Analysis.** Tab. 3 presents the ablation study on the KITTI validation set. The table shows that our baseline only achieves a poor result of 762.21 RMSE. Simply using BEV completion without feature alignment (model A) cannot effectively improve the result, where the metric of RMSE is only decreased to 757.66. After exploiting feature alignment between the camera and LiDAR branches (model B), there is a significant improvement of RMSE to 736.57. This improvement mainly comes from the geometric prior provided by BEV fusion and completion. Fi-

Table 3. Ablation study on the KITTI DC validation set. The 'camera stream' denotes the architecture that only uses camera-based depth completion. The 'BEV Completion' means only conducting BEV completion without feature alignment. The lower the metric values are, the better the estimation results are.

| Method | Camera Stream | BEV Completion | Alignment | PV-SPN | RMSE ↓ | MAE ↓ | iRMSE ↓ | iMAE ↓ |
|---|---|---|---|---|---|---|---|---|
| baseline | ✓ | | | | 762.21 | 197.85 | 2.06 | 0.86 |
| model A | ✓ | ✓ | | | 757.66 | 195.26 | 2.06 | 0.85 |
| model B | ✓ | ✓ | ✓ | | 736.57 | 191.28 | 1.95 | 0.82 |
| full model | ✓ | ✓ | ✓ | ✓ | **719.62** | **187.14** | **1.88** | **0.80** |

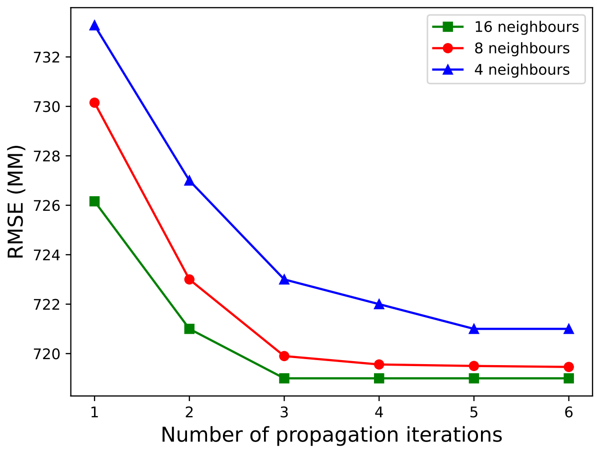


Figure 5. Impact of the number of propagation steps and neighbors on the prediction RMSE on KITTI validation set.

nally, PV-SPN greatly improves the performance to 719.62, which provides the fine-grained geometric details through 3D dense voxels, and inherently affects the results of SPN through consistency criterion. All of our proposed components manifest a positive effect to the camera-based model.

**Number of Neighborhoods and Iterations.** There are two important factors in the point-voxel spatial propagation network: the number of neighbors and iteration steps. To explore the impact of those factors on performance, we set the iteration steps from 1 to 6 and the number of neighbors to 4, 8, 16. The results are illustrated in Fig. 5. Exploiting limited iteration steps, *i.e.*, 1 or 2, disables the network to aggregate enough information, only achieving poor RMSE of 726.25 or 721.51, respectively. Also, we find out that the performance hits a bottleneck when the number of iterations is larger than 3. As for the neighborhood numbers, PV-SPN has a higher RMSE result when it equals to 4, and the lowest value while using 16 neighborhoods. To balance the efficiency and effectiveness, we finally set the number of neighbors and iterations as 16 and 3, respectively.

### 4.4. Training and Inference Speed

To demonstrate the superiority of our BEV-assisted training strategy, we show the comparison of training and inference speed with the previous fusion-based method [50]. As

Table 4. The cost of training and inference on KITTI validation set. Both methods are tested with the metric of 'sample/s'.

| Method | Training | Inference |
|---|---|---|
| ACMNet [50] | 2.72 FPS | 4.20 FPS |
| BEV@DC (ours) | 3.01 FPS | 7.87 FPS |

shown in Tab. 4, the proposed method not only achieves slightly faster speed than ACMNet in training. Moreover, it is much faster than ACMNet in inference, *i.e.*, 7.87 FPS v.s. 4.20 FPS. The reason is that our proposed components (*i.e.*, LiDAR stream) are fully discarded in inference and thus do not introduce any extra computational burden.

## 5. Conclusions

This work proposes the BEV-assisted training for depth completion (*i.e.*, BEV@DC), a general training scheme, to boost the performance of image-guided depth completion via a 3D prior-related training scheme. By leveraging an auxiliary BEV fusion and 3D dense completion with feature consistency, BEV@DC acquires structural information from the LiDAR, effectively enhancing the performance of a pure camera network. Eventually, it achieves state-of-the-art on two large-scale benchmarks (*i.e.*, KITTI DC benchmark and NYUv2 dataset). We believe that our work can be applied to a wider range of other scenarios in the future.

## Acknowledgment

This work was supported in part by Shenzhen General Program No. JCYJ20220530143600001, by the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen HK S&T Cooperation Zone, by Shenzhen-Hong Kong Joint Funding No. SGDX20211123112401002, by Shenzhen Outstanding Talents Training Fund, by Guangdong Research Project No. 2017ZT07X152 and No. 2019CX01X104, by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), by the Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen, by the NSFC 61931024&81922046, by zelixir biotechnology company Fund, by Tencent Open Fund.

## References


[1] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR*, 2:7, 2017. 3

[2] Hu Chen, Hongyu Yang, and Yi Zhang. Depth Completion using Geometry-Aware Embedding. In *International Conference on Robotics and Automation (ICRA)*, pages 8680–8686, 2022. 1, 2, 3, 4

[3] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning Joint 2D-3D Representations for Depth Completion. In *ICCV*, pages 10023–10032, 2019. 1, 2, 4, 6

[4] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. CSPN++: Learning Context and Resource Aware Convolutional Spatial Propagation Networks for Depth Completion. In *AAAI*, pages 10615–10622, 2020. 1, 2, 4, 6, 7

[5] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth Estimation via Affinity Learned with Convolutional Spatial Propagation Network. In *ECCV*, pages 108–125, 2018. 2, 6, 7

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, pages 3213–3223, 2016. 7

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning*, volume 78, pages 1–16, 2017. 7

[8] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-Aware CNNs for Depth Completion: Uncertainty from Beginning to End. In *CVPR*, pages 12014–12023, 2020. 2

[9] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Confidence Propagation through CNNs for Guided Sparse Depth Regression. *IEEE TPAMI*, 42(10):2423–2436, 2020. 2

[10] Wouter Gansbeke, Davy Neven, Bert Brabandere, and Luc Van Gool. Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty. In *International Conference on Machine Vision Applications*, pages 1–6, 2019. 6, 7

[11] Benjamin Graham and Laurens van der Maaten. Submanifold Sparse Convolutional Networks. *arXiv preprint arXiv:1706.01307*, 2017. 3, 4

[12] Praful Hambarde and Subrahmanyam Murala. S2DNet: Depth Estimation from Single Image and Sparse Samples. *IEEE Transactions on Computational Imaging*, 6:806–817, 2020. 1, 2, 6, 7

[13] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. PENet: Towards Precise and Efficient Image Guided Depth Completion. In *International Conference on Robotics and Automation (ICRA)*, pages 13656–13662, 2021. 6

[14] Kuan–Chih Huang, Tsung–Han Wu, Hung–Ting Su, and Winston H. Hsu. Monodtr: Monocular 3d object detection with depth-aware transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4002–4011, 2022. 1

[15] Zixuan Huang, Junming Fan, Shenggan Cheng, Shuai Yi, Xiaogang Wang, and Hongsheng Li. HMS-Net: Hierarchical Multi-Scale Sparsity-Invariant Network for Sparse Depth Completion. *IEEE TIP*, 29:3429–3441, 2020. 2

[16] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth Completion with Twin Surface Extrapolation at Occlusion Boundaries. In *CVPR*, pages 2583–2592, 2021. 6, 7

[17] Longlong Jing, Ruichi Yu, Henrik Kretzschmar, Kang Li, Charles R Qi, Hang Zhao, Alper Ayvaci, Xu Chen, Dillon Cower, Yingwei Li, et al. Depth estimation matters most: improving per-object depth estimation for monocular 3d detection and tracking. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 366–373. IEEE, 2022. 1

[18] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep Projective 3D Semantic Segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107, 2017. 3

[19] Yao-Chih Lee, Kuan-Wei Tseng, Yu-Ta Chen, Chien-Cheng Chen, Chu-Song Chen, and Yi-Ping Hung. 3d video stabilization with depth estimation by cnn-based optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10621–10630, June 2021. 1

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *ICCV*, pages 2980–2988, 2017. 5

[21] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic Spatial Propagation Network for Depth Completion. In *AAAI*, 2022. 1, 2, 3, 4, 6, 7

[22] Lina Liu, Xibin Song, Xiaoyang Lyu, Junwei Diao, Mengmeng Wang, Yong Liu, and Liangjun Zhang. FCFR-Net: Feature Fusion based Coarse-to-Fine Residual Learning for Monocular Depth Completion. In *AAAI*, 2021. 6, 7

[23] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning Affinity via Spatial Propagation Networks. In *NeurIPS*, volume 30, page 1519–1529, 2017. 2, 4, 5

[24] Xin Liu, Xiaofei Shao, Bo Wang, Yali Li, and Shengjin Wang. GraphCSPN: Geometry-Aware Depth Completion via Dynamic GCNs. In *ECCV*, pages 90–107, 2022. 6, 7

[25] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-kuei Liu, and In So Kweon. Non-Local Spatial Propagation Network for Depth Completion. In *ECCV*, pages 120–136, 2020. 2, 4, 6, 7

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, pages 652–660, 2017. 3

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, pages 5099–5108, 2017. 3

[28] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. DeepLiDAR: Deep Surface Normal Guided Depth Prediction for Outdoor Scene from Sparse LiDAR Data and Single Color Image. In *CVPR*, pages 3313–3322, 2019. 1, 2, 6, 7

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. 2, 3

[30] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012. 6

[31] Jie Tang, Fei-Peng Tian, Wei Feng, Li Jian, and Tan Ping. Learning Guided Convolutional Network for Depth Completion. *IEEE TIP*, 30:1116–1129, 2021. 2, 6, 7

[32] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent Convolutions for Dense Prediction in 3D. In *CVPR*, pages 3887–3896, 2018. 3

[33] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, 2017. 1

[34] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *ICCV*, 2019. 3

[35] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity Invariant CNNs. In *International Conference on 3D Vision (3DV)*, pages 11–20, 2017. 6

[36] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, pages 10296–10305, 2019. 5

[37] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[38] Alex Wong and Stefano Soatto. Unsupervised Depth Completion with Calibrated Backprojection Layers. In *ICCV*, pages 12747–12756, 2021. 2

[39] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *International Conference on Robotics and Automation (ICRA)*, pages 1887–1893, 2018. 3

[40] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019. 3

[41] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth Completion from Sparse LiDAR Data with Depth-Normal Constraints. In *ICCV*, pages 2811–2820, 2019. 1, 2, 6, 7

[42] Zheyuan Xu, Hongche Yin, and Jian Yao. Deformable Spatial Propagation Networks for Depth Completion. In *ICIP*, pages 913–917, 2020. 2, 6

[43] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3101–3109, 2021. 3

[44] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds. In *ECCV*, 2022. 3

[45] Xu Yan, Heshen Zhan, Chaoda Zheng, Jiantao Gao, Ruimao Zhang, Shuguang Cui, and Zhen Li. Let images give you more: Point cloud cross-modal training for shape analysis. In *NeurIPS*, 2022. 3

[46] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling. In *CVPR*, pages 5589–5598, 2020. 3

[47] Zhiqiang Yan, Kun Wang, Xiang Li, Zhenyu Zhang, Jun Li, and Jian Yang. RigNet: Repetitive Image Guided Network for Depth Completion. In *ECCV*, 2022. 1, 2, 6, 7

[48] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In *CVPR*, pages 9601–9610, 2020. 3

[49] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, pages 16259–16268, 2021. 3

[50] Shanshan Zhao, Graduate Student Member, Mingming Gong, and Huan Fu. Adaptive Context-Aware Multi-Modal Network for Depth Completion. *IEEE TIP*, 30:5264–5276, 2021. 2, 3, 6, 7, 8

[51] Chaoda Zheng, Xu Yan, Haiming Zhang, Baoyuan Wang, Shenghui Cheng, Shuguang Cui, and Zhen Li. Beyond 3D Siamese Tracking: A Motion-Centric Paradigm for 3D Single Object Tracking in Point Clouds. In *CVPR*, pages 8111–8120, 2022. 3

[52] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *CVPR*, pages 9939–9948, 2021. 3