# OGNI-DC: Robust Depth Completion with Optimization-Guided Neural Iterations

Yiming Zuo and Jia Deng

Department of Computer Science, Princeton University
{zuoym,jiadeng}@princeton.edu

**Abstract.** Depth completion is the task of generating a dense depth map given an image and a sparse depth map as inputs. In this paper, we present OGNI-DC, a novel framework for depth completion. The key to our method is "**O**ptimization-**G**uided **N**eural **I**terations" (OGNI). It consists of a recurrent unit that refines a depth gradient field and a differentiable depth integrator that integrates the depth gradients into a depth map. OGNI-DC exhibits strong generalization, outperforming baselines by a large margin on unseen datasets and across various sparsity levels. Moreover, OGNI-DC has high accuracy, achieving state-of-the-art performance on the NYUv2 and the KITTI benchmarks. Code is available at `https://github.com/princeton-vl/OGNI-DC`.

**Keywords:** Depth Completion · Optimization-inspired Design

## 1 Introduction

Depth completion is the task of predicting a pixel-wise depth map from a single RGB image and known sparse depth. The sparse depth can come from depth sensors such as Lidar [8, 45] and structured light [32], or multiview systems such as SLAM or Structure-from-Motion [47]. Depth completion has important applications such as autonomous driving [3, 9], robotics [19], and augmented reality [12].

For a depth completion system to be maximally useful, it needs to be not only accurate but also robust, meaning that it continues to perform well even with large distribution shifts in input, such as the type of scenes and the sparsity patterns of known depth. Such robustness is desirable because it allows a single system to perform well across a wide range of conditions.

However, achieving both accuracy and robustness at the same time remains challenging. Early works [10, 23, 52] on depth completion formulate it as an optimization problem with hand-crafted energy terms. Specifically, Zhang *et al*. [52] propose to solve a global optimization problem, where the depth needs to agree with the sparse observations at valid locations while being constrained by surface normals and local smoothness. While optimization-based approaches exhibit strong cross-dataset generalization, they are not accurate enough.

More recent methods are based on deep learning [4, 20, 33, 48, 53, 56] and usually directly regress depth values with deep neural networks. Such methods achieve impressive accuracy on benchmarks such as NYUv2 [32] and KITTI [45]

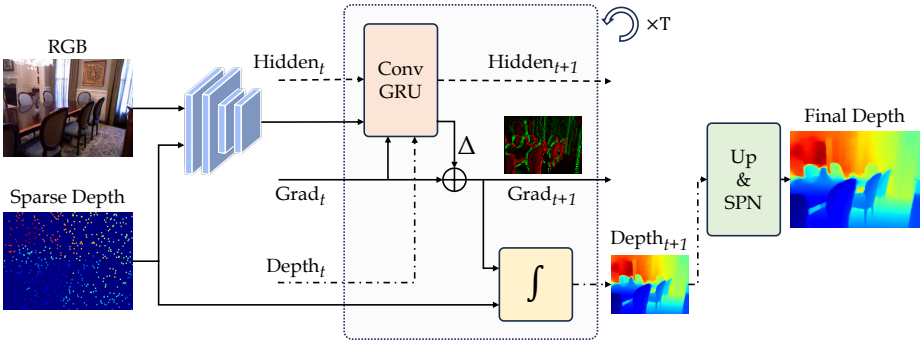


**Fig. 1:** The overall pipeline of OGNI-DC. We first extract features at 1/4 resolution from the concatenation of the image and the sparse depth map. After that, a ConvGRU iteratively refines a depth gradient field based on current predictions, and the DDI (Sec. 4) integrates the depth gradient field into an intermediate depth map. Finally, we up-sample the intermediate depth map and enhance it with a Spatial Propagation Network (SPN) [20] to get the full-resolution depth map.

when trained with data of the same domain. However, the models are not robust to shifts in scene distributions or sparsity patterns of known depth, and often fail catastrophically when tested on new datasets, especially those with different depth ranges or sparsity levels [2, 7].

In this paper, we propose a novel depth completion method, OGNI-DC, which achieves both the superior accuracy of deep neural networks and the robustness of optimization-based approaches. The core of our method is "**O**ptimization-**G**uided **N**eural **I**terations" (OGNI); it consists of a recurrent network that iteratively refines a field of *depth gradients* (*i.e.*, depth differences between neighboring pixels) and a novel *Differentiable Depth Integrator* (DDI) that integrates the depth gradients into a dense depth map. In OGNI-DC, refinement and integration are tightly coupled, meaning that the neural refinement of the depth gradients depends on the current depth integration result, and depth integration is guided by the neural refinement.

DDI is the key to our design. DDI integrates the depth gradients into a depth map while satisfying the boundary conditions set by the sparse depth observations. DDI is *differentiable*, meaning that the errors on the predicted depth map can be back-propagated to the depth gradients, making the full pipeline end-to-end trainable.

DDI improves robustness by exploiting the fact that depth can be recovered from depth gradients and known sparse depth through optimization. Such optimization explicitly constrains the depth to be consistent with the sparse depth, and thus enables the model to easily adapt to varying patterns of sparse depth observations without retraining. In addition, unlike metric depth prediction which usually requires reasoning on the entire image, depth gradients can often be inferred from a local window and are thus easier to predict. An easier learning task leads to better generalization given the same amount of training data.

Another important design is recurrent refinement through a convolutional gated recurrent unit (ConvGRU). We feed the integration result from the previous step back to the ConvGRU, which then refines the depth gradients. The recurrent refinement makes the network aware of the consequences of its depth gradients outputs and thus provides stronger guidance and regularization.

The design of OGNI-DC is substantially different from the previous deep-learning-based depth completion methods. First, instead of directly predicting depth, OGNI-DC predicts depth gradients, which is equally expressive but easier to learn. Second, the constraints on depth from the sparse depth observations are explicitly enforced, rather than from a brittle, learned identity mapping [33, 53].

The high-level idea of OGNI-DC draws inspiration from previous works that use coupled optimization and iterative refinement, such as DROID-SLAM [43] and DPVO [44]. However, prior works were limited to multiview tasks, and we are the first to apply it to a single-view task. Therefore, our designs of the optimization layer are completely different: our optimization enforces constraints on depth and depth gradients from a single image, whereas prior works enforce constraints on depth and pixel correspondences across multiple views.

To prove the effectiveness of our system, we conduct extensive experiments on common benchmarks. We evaluate the model's accuracy under both the zero-shot generalization setting on the DADD [8] and the VOID [47] datasets, as well as the in-domain setting on the NYUv2 [32] and the KITTI [45] datasets.

OGNI-DC exhibits strong generalization. When trained on NYUv2 [32] and tested on VOID [47], our model reduces MAE by 35.5% compared to prior works. Similarly, when trained on KITTI [45] alone, our model reduces RMSE by 25.0% on DDAD [8] from the best previous model trained on the same data. OGNI-DC is also robust to the density changes of the sparse depth observations. On NYUv2 [32], our one single model works the best across a wide density range of $50 \sim 20\,000$ points. On KITTI [45], our model trained with 64 Lidar scanning lines outperforms previous methods by a large margin when tested with $8 \sim 32$ lines, reducing MAE by 25.5% on the 16-Lines input.

Finally, OGNI-DC achieves state-of-the-art in-domain accuracy. It achieves the best performance on the NYUv2 [32] benchmark, improving RMSE from 0.089m to 0.087m. On the KITTI [45] online benchmark, OGNI-DC outperforms all previous methods on 3 out of the 4 metrics when trained with an $L_1$ loss.

## 2 Related Works

### 2.1 Depth Completion

Deep learning-based methods have achieved impressive accuracy for depth completion. Early works directly predict depth values from image and sparse depth inputs [24, 29, 41, 45]. These methods are feed-forward and have difficulties in reasoning long-range depth dependencies because of the limited receptive field.

To mitigate this issue, more recent works [4–6,13,20,25,33,48] propose a series of Spatial Propagation Networks (SPNs). SPNs iteratively update the regressed

depth map based on the predictions of each pixel and its neighbors. Specifically, NLSPN [33] predicts a non-local neighborhood by utilizing deformable convolutions. DySPN [20] predicts different propagation weights for each iteration and achieves better performance with fewer iterations and neighbors. BEV@DC [56] uses 3D SPN on the unprotected Lidar point cloud at training time to regularize the solution space. Although SPN-based method and OGNI-DC both involve iterative updates, they are significantly different: the SPN-based method predicts depth values and propagation guidance only once, and uses SPN to propagate depth predictions without explicit constraints around observed locations. In contrast, our method predicts depth gradients iteratively based on current depth, and explicitly enforces the observation constraints through optimization.

A recent work LRRU [46] generates an initial dense depth map with handcrafted heuristics instead of neural networks, and propagates the depth values iteratively through spatially-variant kernels. While both LRRU and our method avoid direct depth regression, we are different in: 1) LRRU is an SPN variant that directly propagates depth values, while our recurrent unit performs updates on the depth gradients and is coupled with DDI to produce depth outputs. 2) The updates in LRRU are coarse-to-fine instead of recurrent, only being applied once at each resolution, whereas our recurrent update can be unrolled arbitrary times at the same resolution.

Several works focus on generalization [2,7,26]. SpAgNet [7] merges the sparse observations into the multi-resolution depth maps predicted by a network and can deal with extremely sparse inputs. VPP4DC [2] repurposes a stereo matching network for depth completion by projecting random patterns onto a virtual neighboring view, and achieves decent zero-shot performance on the DDAD [8] and the VOID [47] datasets. The generalization of all these methods comes with a cost of accuracy, as none of these works achieves comparable performance on NYUv2 [32] or KITTI [45] as ours. Furthermore, some methods focus on cross-dataset generalization, while others focus on cross-density generalization, but none of these methods achieve satisfactory results in both scenarios.

### 2.2 Geometry Reconstruction from Local Properties

Some prior works propose to solve 3D vision tasks with constraints from local properties, such as surface normals [14, 27, 35, 37, 51, 55], occlusion boundaries [16, 38, 52], and principle directions [15]. Such local properties are easier to learn and generalize better to unseen domains, compared to global properties such as depth [52]. Here we mainly introduce prior works on monocular depth estimation, as it is the most closely related task. Long *et al*. [27] employs the depth-normal constraints by sampling reliable planar regions in the pixel space. DeepLiDAR [37] predicts surface normals as an intermediate representation and converts it to depth with a neural network. Compared to them, OGNI-DC predicts depth gradients instead of normals, as depth gradients are defined everywhere and are capable of reconstructing scenes with many thin structures such as trees and fences, whereas surface normals are not defined on occlusion

boundaries. While depth gradients have been used in previous works as a loss term [18,39], we are the first to directly predict depth gradients from the network.

Some previous methods solve geometry reconstruction problems with iterative updates [1,21,36,40,42,43]. Among them, GeoNet++ [36] iteratively refines normal-from-depth and depth-from-normal. IronDepth [1] proposes depth propagation candidates based on local planes defined by the normals. NDDepth [40] performs contrastive iterative updates on the two depth maps directly predicted and computed from normals. Compared to ours, none of the iterative units of these methods involve an optimization-based layer, and cannot be easily extended to the depth completion task.

Several works utilize optimization-inspired designs to solve various computer vision tasks [14,22,50]. Specifically, VA-DepthNet [22] solves an integration problem in the feature space at 1/16 resolution. Compared to them, our DDI layer solves an optimization problem at a larger scale directly in the depth space.

Some other methods employ coupled iterative refinement and differentiable optimization [43, 44]. DROID-SLAM [43] iteratively updates the optical flow, and uses a Dense Bundle Adjustment (DBA) layer to optimize camera poses and depths. Compared to DROID-SLAM, the nature of the tasks and the designs of our differentiable optimization layers are different. The DBA layer solves a non-linear optimization problem and performs only two Gauss-Newton steps at each iteration. Our DDI layer solves a linear least-squares problem and finds the global minimum through the conjugate gradient method [11]. This difference is non-trivial and poses additional challenges in the backward pass.

## 3 Approach Overview

In this section, we describe our depth completion pipeline. Our model takes an RGB image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$, a sparse depth observation map $\mathbf{O} \in \mathbb{R}_{+}^{H\times W}$, and a mask $\mathbf{M} \in \{0,1\}^{H\times W}$ as input, where $\mathbf{M}_{i,j} = 1$ if and only if there is a valid observation at location $(i, j)$. Our model outputs a dense depth map $\hat{\mathbf{D}} \in \mathbb{R}_{+}^{H\times W}$.

Our pipeline is illustrated in Fig. 1. There are three main components, *i.e.*, a backbone for feature extraction, a coupled ConvGRU and DDI for intermediate depth maps prediction, and an up-sample and enhancement layer that produces the final depth map. We describe each component below in this section, and leave the details of DDI to Sec. 4.

### 3.1 Feature Extraction

We use a deep neural network as the backbone to extract features. The input to the backbone is the *concatenation* of the RGB image $\mathbf{I}$ and the sparse observations $\mathbf{O}$. The backbone outputs features at two resolutions: 1/4 resolution for intermediate depth predictions and full resolution for final depth enhancement:

$$\hat{\mathbf{F}}^{\text{full}}, \hat{\mathbf{F}}^{\frac{1}{4}} = \text{Backbone}(\mathbf{I}, \mathbf{O}). \tag{1}$$

We use CompletionFormer [53] as our backbone for its state-of-the-art performance. We take the decoder features at the full and 1/4 resolution as outputs.

### 3.2 Intermediate Depth Prediction

We predict $T$ intermediate depth maps in an iterative manner, where $T$ is the total number of steps. In the $t$-th step, we first predict the depth gradients $\hat{\mathbf{G}}_t \in \mathbb{R}^{2\times\frac{H}{4}\times\frac{W}{4}}$, and then convert $\hat{\mathbf{G}}_t$ into an intermediate depth $\hat{\mathbf{D}}_t^{\frac{1}{4}} \in \mathbb{R}_+^{\frac{H}{4}\times\frac{W}{4}}$ with DDI (see Sec. 4). We describe the details of the two modules below.

**Depth Gradients Prediction.** This module performs iterative updates on the depth gradients map through a convolutional gated recurrent unit (ConvGRU) [42]. For each step, it predicts a refinement $\Delta\hat{\mathbf{G}}$ which is applied to the current depth gradients: $\hat{\mathbf{G}}_t = \Delta\hat{\mathbf{G}} + \hat{\mathbf{G}}_{t-1}$.

This module takes as input the backbone features, the hidden state, the predicted depth gradients, and the predicted intermediate depth map from the previous step. It outputs a depth gradients refinement map and an updated hidden state:

$$\Delta\hat{\mathbf{G}}, \mathbf{h}_t = \mathrm{ConvGRU}(\hat{\mathbf{F}}^{\frac{1}{4}}, \mathbf{h}_{t-1}, \hat{\mathbf{D}}_{t-1}^{\frac{1}{4}}, \hat{\mathbf{G}}_{t-1}). \tag{2}$$

We initialize $\hat{\mathbf{G}}_0$ as an all $\mathbf{0}$ tensor, and $\mathbf{h}_0$ from a 2-layers CNN. $\hat{\mathbf{D}}_{t-1}^{\frac{1}{4}}$ and $\hat{\mathbf{G}}_{t-1}$ are encoded by a 2-layers CNN before feeding into the GRU. The network architecture of ConvGRU is adopted from RAFT [42].

**Gradients Integration.** We use DDI (see Sec. 4) to integrate the depth gradients and the down-sampled observations into an intermediate depth map:

$$\hat{\mathbf{D}}_t^{\frac{1}{4}} = \mathrm{DDI}(\hat{\mathbf{G}}_t, \mathbf{O}^{\frac{1}{4}}, \mathbf{M}^{\frac{1}{4}}), \tag{3}$$

where $\mathbf{O}^{\frac{1}{4}}$ and $\mathbf{M}^{\frac{1}{4}}$ are obtained by performing an average-pooling on valid pixels from the observation $\mathbf{O}$ and the mask $\mathbf{M}$ respectively.

All computations are done at 1/4 resolution. The intermediate depth prediction involves iterative refinements and differentiable integrations, which put a heavy burden on computation and memory when running at full resolution. Therefore, 1/4 resolution is a good balance between performance and computational cost. Fig. 2 illustrates the iterative refinement process.

### 3.3 Depth Up-sample and Enhancement

We up-sample the intermediate depth map $\hat{\mathbf{D}}_t^{\frac{1}{4}}$ into a full resolution depth map $\hat{\mathbf{D}}_t^{\mathrm{up}}$. We use the convex up-sampling [42], where we predict a mask of shape $(H/4) \times (W/4) \times (4 \times 4 \times 9)$. Each value in $\hat{\mathbf{D}}_t^{\mathrm{up}}$ is calculated as the convex combination of the values of its $3 \times 3$ neighbors in $\hat{\mathbf{D}}_t^{\frac{1}{4}}$.

Finally, we use a spatial propagation network (SPN) to enhance the up-sampled depth map and get the final prediction $\hat{\mathbf{D}}_t$: $\hat{\mathbf{D}}_t = \mathrm{SPN}(\hat{\mathbf{D}}_t^{\mathrm{up}}, \hat{\mathbf{F}}^{\mathrm{full}})$. SPN helps our model retain high-resolution details, especially at object boundaries. We use DySPN [20] in our implementation, but our model can also work well with other SPNs or even without an SPN, because DDI plays the role of propagating the sparse depth information globally.

At training time, we up-sample and enhance all intermediate depth maps for supervision. At test time, we only up-sample and enhance the last prediction.

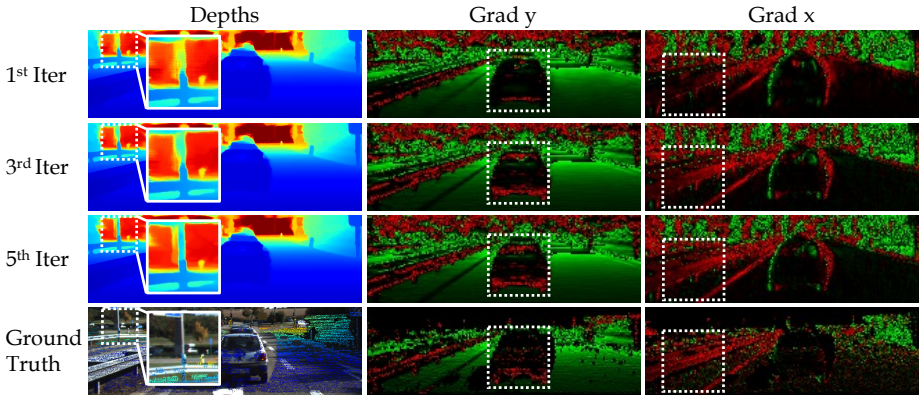


**Fig. 2:** We demonstrate the effectiveness of our iterative refinement. The depth and the gradients predictions in the highlighted areas gradually improved. Red means negative gradients and green means positive gradients. Brighter colors mean larger absolute values. Some ground truths are missing in the gradients map due to incomplete depth.

### 3.4 Loss Functions

We supervise the final outputs $\{\hat{\mathbf{D}}_1, \cdots, \hat{\mathbf{D}}_T\}$ at all iterations. We also supervise the up-sampled depth maps $\{\hat{\mathbf{D}}_1^{\text{up}}, \cdots, \hat{\mathbf{D}}_T^{\text{up}}\}$ and the depth gradients $\{\hat{\mathbf{G}}_1, \cdots, \hat{\mathbf{G}}_T\}$. Given a ground-truth depth map $\mathbf{D}$, we compute the ground-truth depth gradients $\mathbf{G}$ by down-sampling $\mathbf{D}$ and taking its finite difference.

We supervise $\hat{\mathbf{D}}$ and $\hat{\mathbf{D}}^{\text{up}}$ with a combination of an $L_1$ and an $L_2$ loss, and $\hat{\mathbf{G}}$ with an $L_1$ loss. We use a loss decay $\gamma = 0.9$ and $\lambda = 1.0$ in all experiments:

$$\mathcal{L}_{\mathbf{D}} = \sum_{t=1}^{T} \gamma^{T-t} \left( \left\| \hat{\mathbf{D}}_t - \mathbf{D} \right\|_2^2 + \left\| \hat{\mathbf{D}}_t - \mathbf{D} \right\|_1 + \left\| \hat{\mathbf{D}}_t^{\text{up}} - \mathbf{D} \right\|_2^2 + \left\| \hat{\mathbf{D}}_t^{\text{up}} - \mathbf{D} \right\|_1 \right), \tag{4}$$

$$\mathcal{L}_{\mathbf{G}} = \sum_{t=1}^{T} \gamma^{T-t} \left\| \hat{\mathbf{G}}_t - \mathbf{G} \right\|_1, \ \mathcal{L} = \mathcal{L}_{\mathbf{D}} + \lambda \cdot \mathcal{L}_{\mathbf{G}}. \tag{5}$$

## 4 Differentiable Depth Integrator (DDI)

In this section, we introduce our differentiable depth integrator (DDI), which converts the predicted depth gradients and the given sparse observations into a depth map. We define DDI as a function:

$$\hat{\mathbf{D}} = \text{DDI}(\hat{\mathbf{G}}, \mathbf{O}, \mathbf{M}). \tag{6}$$

Let $H \times W$ be the depth map resolution. $\hat{\mathbf{G}} = \{\hat{\mathbf{G}}^{\mathbf{x}}, \hat{\mathbf{G}}^{\mathbf{y}}\} \in \mathbb{R}^{2 \times H \times W}$ is the predicted depth gradients along the $x$ and $y$ directions. $\mathbf{O} \in \mathbb{R}_+^{H \times W}$ is the sparse

depth observations and $\mathbf{M} \in \{0,1\}^{H \times W}$ is its corresponding mask. $\hat{\mathbf{D}} \in \mathbb{R}_{+}^{H \times W}$ is the predicted depth map.

DDI is the key to achieving strong generalization and high accuracy, as it explicitly enforces the predicted depth map to be consistent with the sparse observations, introducing a strong inductive bias to the network. Moreover, DDI is differentiable, allowing us to train the network end-to-end.

**Optimization Problem Definition.** DDI solves a 2D numerical integration problem, integrating depth gradients into depth with boundary conditions set by the sparse observations. We formulate depth integration as a linear least squares problem with two energy terms:

$$\hat{\mathbf{D}} = \underset{\mathbf{D}}{\arg\min} \left( E_G(\mathbf{D}, \hat{\mathbf{G}}) + \alpha \cdot E_O(\mathbf{D}, \mathbf{O}, \mathbf{M}) \right), \tag{7}$$

where $E_G(\cdot)$ corresponds to the gradient conditions, $E_O(\cdot)$ corresponds to the observation conditions, and $\alpha$ is a hyperparameter balancing the relative impact of the two terms. The performance of OGNI-DC is not sensitive to $\alpha$ and we use $\alpha = 5.0$ for all experiments.

$E_G(\cdot)$ encourages the finite differences between neighboring pixels in $\mathbf{D}$ to be close to the predicted depth gradients $\hat{\mathbf{G}}$:

$$E_G(\mathbf{D}, \hat{\mathbf{G}}) = \sum_{i=2}^{W} \sum_{j=1}^{H} \left( \mathbf{D}_{i,j} - \mathbf{D}_{i-1,j} - \hat{\mathbf{G}}_{i,j}^{\mathbf{x}} \right)^2 + \sum_{i=1}^{W} \sum_{j=2}^{H} \left( \mathbf{D}_{i,j} - \mathbf{D}_{i,j-1} - \hat{\mathbf{G}}_{i,j}^{\mathbf{y}} \right)^2. \tag{8}$$

$E_O(\cdot)$ encourages the predicted depth values to be consistent with the sparse observations at valid locations:

$$E_O(\mathbf{D}, \mathbf{O}, \mathbf{M}) = \sum_{i=1}^{W} \sum_{j=1}^{H} \mathbf{M}_{i,j} \cdot (\mathbf{D}_{i,j} - \mathbf{O}_{i,j})^2. \tag{9}$$

We model the observation conditions as energy terms rather than hard constraints for two reasons. 1) The optimization problem with soft constraints can be solved more efficiently. 2) The sparse observations can contain noise, such as the bleeding artifacts caused by blended foreground and background depth in KITTI [45]. We solve this problem by predicting a confidence map for the sparse depth input, which allows DDI to ignore noisy sparse depth values. We omit it here for brevity. Please refer to the Appendix for details.

**Conjugate Gradient Solver.** We rewrite Eq. (7) in the matrix form of linear least squares:

$$\overline{\hat{\mathbf{D}}} = \underset{\overline{\mathbf{D}}}{\arg\min} \left\| \underbrace{\begin{pmatrix} \overline{\boldsymbol{\Delta}_{\mathbf{x}}} \\ \overline{\boldsymbol{\Delta}_{\mathbf{y}}} \\ \operatorname{diag}\left(\sqrt{\alpha} \cdot \overline{\mathbf{M}}\right) \end{pmatrix}}_{\mathbf{A}} \overline{\mathbf{D}} - \underbrace{\begin{pmatrix} \overline{\hat{\mathbf{G}}^{\mathbf{x}}} \\ \overline{\hat{\mathbf{G}}^{\mathbf{y}}} \\ \sqrt{\alpha} \cdot \overline{\mathbf{M}} \cdot \overline{\mathbf{O}} \end{pmatrix}}_{\mathbf{b}} \right\|_2^2, \tag{10}$$

where $\mathbf{\Delta_x}$ is the finite difference operator in the $x$ direction, *i.e.*, $(\mathbf{\Delta_x} \circ \mathbf{D})_{i,j} = \mathbf{D}_{i,j} - \mathbf{D}_{i-1,j}$, and $\mathbf{\Delta_y}$ is defined accordingly. The bar $(\overline{\cdot})$ over a matrix or an operator means the flattened version of it.

Eq. (10) has the closed-form solution of $\overline{\hat{\mathbf{D}}} = (\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{A}^\intercal\mathbf{b}$. However, solving $\overline{\hat{\mathbf{D}}}$ by directly computing $(\mathbf{A}^\intercal\mathbf{A})^{-1}$ is intractable for high-resolution images, as $\mathbf{A}^\intercal\mathbf{A}$ is a giant matrix with shape $HW \times HW$. Therefore, we use the conjugate gradient method [11] to solve it efficiently. Note that neither $\mathbf{A}$ nor $\mathbf{A}^\intercal\mathbf{A}$ needs to be explicitly stored, as we only need to implement its matrix multiplication with a vector. All operations are implemented in PyTorch [34] and naturally support GPU acceleration.

**Backward Pass.** To make DDI differentiable, we have to compute the Jacobian matrix $\partial\hat{\mathbf{D}}/\partial\hat{\mathbf{G}}$. One option is to trace through the whole conjugate gradient process in the forward pass. However, the memory cost is intractable as the number of conjugate gradient steps is large. Instead, we compute $\partial\hat{\mathbf{D}}/\partial\hat{\mathbf{G}}$ by the chain rule:

$$\frac{\partial\overline{\hat{\mathbf{D}}}}{\partial\overline{\hat{\mathbf{G}}}} = \frac{\partial\overline{\hat{\mathbf{D}}}}{\partial\mathbf{b}} \cdot \frac{\partial\mathbf{b}}{\partial\overline{\hat{\mathbf{G}}}}, \tag{11}$$

$$\frac{\partial\overline{\hat{\mathbf{D}}}}{\partial\mathbf{b}} = (\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{A}^\intercal, \ \frac{\partial\mathbf{b}}{\partial\overline{\hat{\mathbf{G}}}} = \begin{pmatrix} \mathbf{I}_{H(W-1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(H-1)W} & \mathbf{0} \end{pmatrix}^\intercal, \tag{12}$$

where $\mathbf{I}$ is the identity matrix. Note again we do not have to explicitly compute or store $\partial\overline{\hat{\mathbf{D}}}/\partial\mathbf{b}$, as we only need its matrix multiplication with a vector. This can be done effectively using the same conjugate gradient solver as in the forward pass. Details can be found in the Appendix.

**Initialization from Previous Solution.** Since we solve the optimization problem multiple times in each forward and backward pass, each time with slightly refined depth gradients, the solution from the previous round can be used as an initial guess to accelerate convergence. This reduces the overall DDI latency by up to 62.1%. Please see Tab. 5 for detailed analysis of speed.

## 5 Experiments

### 5.1 Implementation Details

We implement OGNI-DC in PyTorch [34]. We use the AdamW [28] optimizer with an initial learning rate of 0.001. We use 5 GRU iterations for all experiments. On NYUv2 [32], we train the model on a single RTX 3090 GPU for 36 epochs with a batch size of 12, which takes about 3 days. On KITTI [45], we train the model on 8×L40 GPUs for 100 epochs with a batch size of 40, which takes about 1 week. Following previous works [41, 46, 54], we average the predictions for the original and the horizontal-flipped inputs only for the KITTI online submissions.

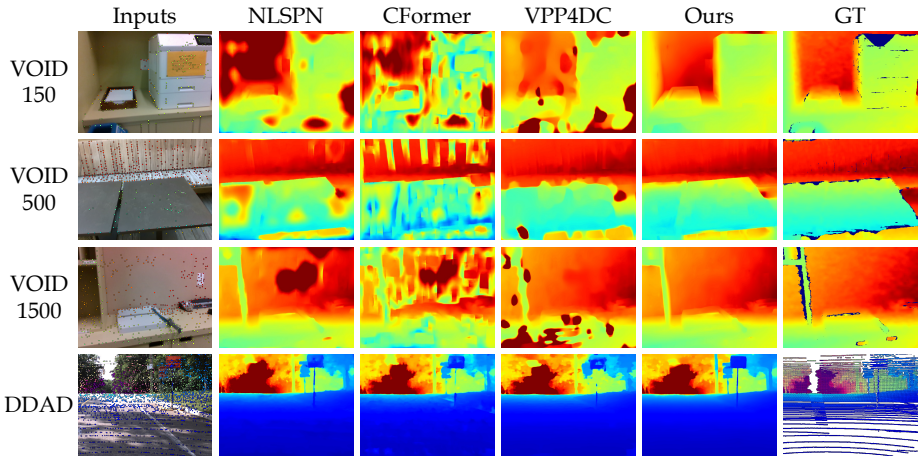


**Fig. 3:** Qualitative comparisons of zero-shot generalization on the VOID [47] and the DDAD [8] datasets. The sparse depth observations are superimposed on the RGB images. Compared to baselines, our results are less noisy and sharper at boundaries.

Inspired by previous works [7,26], to make the backbone more robust to depth density changes, we use a simple random masking technique on the sparse depth input: at training time, we randomly drop $0 \sim 100\%$ observed depth values for 50% training samples, and keep the other 50% untouched.

### 5.2 Datasets and Evaluation Metrics

We evaluate our method on 4 commonly used datasets. To provide good coverage of both indoor and outdoor scenes, we use NYUv2 [32] and VOID [47] for room environments, and KITTI [45] and DDAD [8] for autonomous driving environments. Detailed descriptions of the datasets' statistics, split, and pre-processing can be found in the Appendix.

**Evaluation Metrics.** We evaluate under the standard metrics: *Root Mean Squared Error (RMSE)*, *Mean Absolute Error (MAE)*, *Root Mean Squared Error of Inverse Depth (iRMSE)*, *Mean Absolute Error of Inverse Depth (iMAE)*, and *Mean Absolute Relative Error (REL)*. Definitions can be found in [17].

### 5.3 Zero-shot Generalization to VOID and DDAD

The generalization to unseen environments is critical to the users of the depth completion systems. Following VPP4DC [2], we test the *zero-shot* generalization of our model in both indoor and outdoor environments. We train models on NYUv2 [32] and KITTI [45] and test them on VOID [47] and DDAD [8], respectively. We compare against state-of-the-art baselines, including 3 models with the best in-domain performance [33, 46, 53], and VPP4DC [2] specially designed for cross-dataset generalization.

**Table 1:** Zero-shot generalization to VOID [47] and DDAD [8]. All metrics in [m].

| Train Datasets | NYU | | | | | | KITTI | |
|---|---|---|---|---|---|---|---|---|
| Test Datasets | VOID1500 [47] | | VOID500 [47] | | VOID150 [47] | | DDAD [8] | |
| Points/Density | 1 500/0.5% | | 500/0.15% | | 150/0.05% | | 5 000/0.21% | |
| Methods | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE |
| NLSPN [33] | 0.737 | 0.298 | 0.802 | 0.381 | 0.963 | 0.492 | 11.646 | 4.621 |
| SpAgNet [7] | 0.706 | 0.244 | 0.752 | 0.326 | 0.866 | 0.408 | 18.247 | 9.130 |
| CFormer [53] | 0.726 | 0.261 | 0.821 | 0.385 | 0.956 | 0.487 | 9.606 | 3.328 |
| LRRU [46] | - | - | - | - | - | - | 9.164 | 2.738 |
| VPP4DC [2] | 0.800 | 0.253 | 0.840 | 0.307 | 0.960 | 0.397 | 10.247 | 2.290 |
| Ours | **0.593** | **0.175** | **0.589** | **0.198** | **0.693** | **0.261** | **6.876** | **1.867** |

**Table 2:** Robustness to the number of Lidar lines on the KITTI [45] validation set. All methods have a single model trained with 64 lines. All metrics are in [mm].

| Lidar Scans | 64-Lines | | 32-Lines | | 16-Lines | | 8-Lines | |
|---|---|---|---|---|---|---|---|---|
| Methods | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE |
| NLSPN [33] | 778.0 | 199.5 | 1217.2 | 367.5 | 1988.5 | 693.1 | 3234.9 | 1491.3 |
| SpAgNet [7] | 844.8 | 218.4 | 1164.2 | 339.2 | 1863.3 | 606.9 | 2691.3 | 1087.2 |
| LRRU [46] | **729.5** | **188.8** | 1082.9 | 315.6 | 1978.5 | 700.5 | 3512.8 | 1624.0 |
| CFormer [53] | 741.4 | 195.0 | 1241.8 | 384.9 | 2236.0 | 880.4 | 3638.8 | 1698.8 |
| Ours | 749.8 | 192.9 | **1017.2** | **267.4** | **1661.8** | **451.9** | **2389.8** | **784.9** |

Results are shown in Tab. 1. OGNI-DC outperforms baselines by a large margin on all metrics. On the most challenging VOID150 [47] benchmark (about 0.05% depth coverage), OGNI-DC improves MAE by 34.2% (from 0.397 to 0.261) compared to the best-performing prior work VPP4DC [2]. On DDAD [8], OGNI-DC improves RMSE by 25.0% (from 9.164 to 6.876) compared to LRRU [46]. We visualize the depth predictions of different methods in Fig. 3. Compared to baselines, our method generates sharp and accurate results on DDAD [8] and at all sparsity levels on VOID [47].

### 5.4 Robustness to Different Sparsity Levels

Robustness to different sparsity levels is also important for downstream applications. Previous works [16, 53] *retrain* different models at different sparsity levels. We find this setting impractical, as the depth density cannot be known *a priori* in real testing environments. For example, when the sparse observations come from a SLAM system [31], the sparsity level can vary drastically from frame to frame as various numbers of key points are being tracked. Therefore, we want *one model* to perform well across all sparsity levels.

We adopt the experiment setup from SpAgNet [7], where all models are trained under the standard setting (500 points on NYUv2 [32], 64-Lines Lidar on KITTI [45]), and tested with different sparsity levels. We compare against

**Table 3:** Generalization to different sparsity levels on the NYUv2 [32] test set. All methods have a single model trained with 500 points. The RMSE metric is in [m]. "X+Mask" are the baselines retrained by randomly masking out $0 \sim 100\%$ of the sparse depth observations during training.

| Samples | 20000 | | 5000 | | 1000 | | 500 | |
|---|---|---|---|---|---|---|---|---|
| Methods | RMSE | REL | RMSE | REL | RMSE | REL | RMSE | REL |
| NLSPN [33] | 0.034 | 0.004 | **0.042** | **0.005** | 0.071 | **0.009** | 0.092 | **0.012** |
| SpAgNet [7] | - | - | - | - | - | - | 0.114 | 0.015 |
| CFormer [53] | 0.203 | 0.046 | 0.045 | **0.005** | **0.070** | **0.009** | 0.090 | **0.012** |
| NLSPN+Mask | 0.796 | 0.133 | 0.086 | 0.013 | 0.082 | 0.011 | 0.103 | 0.014 |
| CFormer+Mask | 0.161 | 0.022 | 0.050 | 0.007 | 0.075 | 0.010 | 0.096 | 0.013 |
| Ours | **0.028** | **0.003** | **0.042** | **0.005** | **0.070** | **0.009** | **0.089** | **0.012** |

| Samples | 200 | | 100 | | 50 | | 5 | |
|---|---|---|---|---|---|---|---|---|
| Methods | RMSE | REL | RMSE | REL | RMSE | REL | RMSE | REL |
| NLSPN [33] | 0.136 | 0.019 | 0.245 | 0.037 | 0.431 | 0.081 | 1.043 | 0.262 |
| SpAgNet [7] | 0.155 | 0.024 | 0.209 | 0.038 | 0.272 | 0.058 | **0.467** | 0.131 |
| CFormer [53] | 0.141 | 0.021 | 0.429 | 0.092 | 0.707 | 0.181 | 1.141 | 0.307 |
| NLSPN+Mask | 0.142 | 0.022 | 0.183 | 0.031 | 0.241 | 0.046 | 0.728 | 0.175 |
| CFormer+Mask | 0.135 | 0.021 | 0.174 | 0.030 | 0.227 | 0.044 | 0.487 | **0.129** |
| Ours | **0.124** | **0.018** | **0.157** | **0.025** | **0.207** | **0.038** | 0.633 | 0.171 |

several baselines, including SpAgNet [7] which is specialized for sparsity generalization. We build two stronger baselines, where we retrain the NLSPN [33] and CFormer [53] with the same random masking we use, which we denote as "X+Mask".

On KITTI [45], we sub-sample the Lidar points following [16] to 32, 16, and 8 lines. On NYUv2 [32], we test across a wide range of $5 \sim 20\,000$ randomly sampled sparse points. Results are shown in Tab. 2 and Tab. 3.

On KITTI [45], OGNI-DC achives comparable performance as baselines on 64-Lines input and outperforms baselines by a large margin on sparser inputs, reducing MAE by 25.5% compared to SpAgNet [7] (451.9 versus 606.9) on 16-Lines input.

On NYUv2 [32], our method works better than baselines under almost all sparsity levels. Interestingly, our method can also work well with more than 500 points, although it has never seen these cases during training. The generalization of CFormer [53] and NLSPN [33] improves with random masking, but is still worse than ours. Under the extremely sparse case (5 points), our method still works better than NLSPN [33] and CFormer [53], but slightly worse than SpAgNet [7]. That's probably because the estimation errors on depth gradients can accumulate in the integration process in large regions without any depth observations. However, we don't find this to be a significant limitation of our system, since these cases are very unlikely in real-world scenarios (*e.g.*, ORB-SLAM [30] cannot add new key-frames when it tracks fewer than 50 key-points).

**Table 4:** Comparison with prior works on the NYUv2 [32] and KITTI [45] datasets. We mark the best method in **bold** and the second-best method with an underline.

| Datasets | NYUv2 [32] | | KITTI [45] | | | |
|---|---|---|---|---|---|---|
| Methods | RMSE [m] | REL | iRMSE [1/km] | iMAE [1/km] | RMSE [mm] | MAE [mm] |
| CSPN [5] | 0.117 | 0.016 | 2.93 | 1.15 | 1019.64 | 279.46 |
| DeepLiDAR [37] | 0.115 | 0.022 | 2.56 | 1.15 | 758.38 | 226.50 |
| GuideNet [41] | 0.101 | 0.015 | 2.25 | 0.99 | 736.24 | 218.83 |
| NLSPN [33] | 0.092 | 0.012 | 1.99 | 0.84 | 741.68 | 199.59 |
| ACMNet [54] | 0.105 | 0.015 | 2.08 | 0.90 | 744.91 | 206.09 |
| RigNet [49] | 0.090 | 0.012 | 2.08 | 0.90 | 712.66 | 203.25 |
| DySPN [20] | 0.090 | 0.012 | 1.88 | 0.82 | 709.12 | 192.71 |
| LRRU [46] | 0.091 | **0.011** | 1.87 | 0.81 | **696.51** | 189.96 |
| BEV@DC [56] | 0.089 | 0.012 | 1.83 | 0.82 | 697.44 | 189.44 |
| CFormer [53] ($L_1$) | - | - | 1.89 | 0.80 | 764.87 | 183.88 |
| CFormer [53] ($L_1 + L_2$) | 0.090 | 0.012 | 2.01 | 0.88 | 708.87 | 203.45 |
| Ours ($L_1$) | - | - | **1.81** | **0.79** | 747.64 | **182.29** |
| Ours ($L_1 + L_2$) | **0.087** | **0.011** | 1.86 | 0.83 | 708.38 | 193.20 |

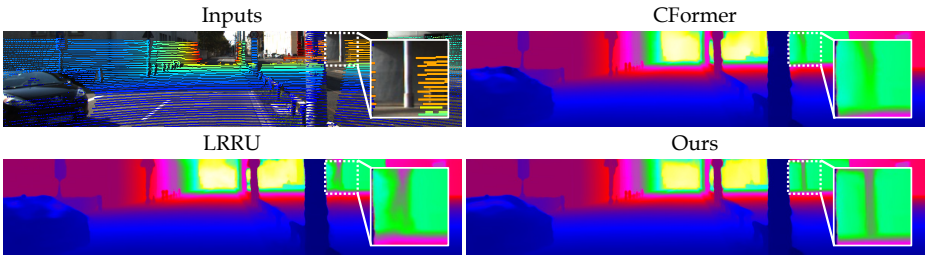


**Fig. 4:** Qualitative comparison with LRRU [46] and CFormer [53] on the KITTI [45] test split. Our method reconstructs the telephone pole better than the baselines.

### 5.5 In-domain Performance on NYUv2 and KITTI

We compare the in-domain performance of OGNI-DC with other state-of-the-art methods on the standard NYUv2 [32] and KITTI [45] benchmarks.

Quantitative results are shown in Tab. 4. On the NYUv2 [32] dataset, our model achieves RMSE of 0.087 and REL of 0.011, the best among all previous methods. On the KITTI [45] dataset, following CFormer [53], we train two models, one with only $L_1$ loss and the other with combined $L_1 + L_2$ loss. Our model trained with $L_1$ loss archives the state-of-the-art MAE of 182.29, iRMSE of 1.81, and iMAE of 0.79. Our model trained with combined $L_1 + L_2$ loss achieves a better RMSE of 708.38, which is close to the prior best-performing model. Note that in Tab. 4 we disable masking to achieve the best in-domain performance and for a fair comparison with previous methods. The difference caused by masking is marginal: our model with masking still archives the state-of-the-art performance on the NYUv2 [32] with RMSE = 0.089 (see Tab. 3, 500 samples).

**Table 5:** We collect statistics with PyTorch Profiler on a L40 GPU under the KITTI resolution ($240 \times 1216$). All metrics are in mm.

| Metrics | NYUv2 [32] | | KITTI [45] | | Inference Time (ms) | | | | Mem (GB) |
|---|---|---|---|---|---|---|---|---|---|
| Methods | RMSE | MAE | RMSE | MAE | **Total** | Backbone | GRU | DDI | |
| CFormer+DySPN | 123.6 | 43.2 | 825.1 | 208.8 | 268.0 | 268.0 | 0.0 | 0.0 | 5.4 |
| No DDI | 128.6 | 44.9 | 824.0 | 207.5 | 298.7 | 256.5 | 42.2 | 0.0 | 5.9 |
| DDI zeros init | **112.2** | 38.0 | 813.7 | **205.4** | 646.5 | 256.5 | 50.5 | 339.5 | 6.0 |
| DDI pre-filled init | **112.2** | 38.0 | 813.7 | **205.4** | 601.9 | 256.5 | 50.5 | 294.9 | 6.0 |
| 1 GRU iteration | 114.0 | 39.9 | 820.1 | 208.8 | 317.6 | 256.5 | 9.8 | 51.3 | 5.2 |
| 3 GRU iterations | 112.4 | 38.2 | 818.6 | 207.8 | 379.7 | 256.5 | 28.8 | 94.4 | 5.6 |
| 7 GRU iterations | 112.3 | 38.0 | **811.3** | 205.5 | 484.2 | 256.5 | 68.0 | 159.7 | 6.3 |
| ConvRNN | 112.7 | 38.1 | 817.7 | 205.8 | 408.7 | 256.5 | 34.0 | 118.2 | 5.6 |
| Cascaded | 112.5 | **37.9** | 814.4 | 206.0 | 444.7 | 256.5 | 50.5 | 137.7 | 5.9 |
| Ours | **112.2** | 38.0 | 813.7 | **205.4** | 435.8 | 256.5 | 50.5 | 128.8 | 5.9 |

Qualitative comparisons on the KITTI [45] test split are shown in Fig. 4. Our method can reconstruct the thin telephone pole, on which both baselines fail.

### 5.6 Ablation Studies

To study the effects of the key designs in OGNI-DC, we conduct ablation studies on the NYUv2 [32] and the KITTI [45] validation set. On KITTI, following CFormer [53], we train models on 10,000 images. Results are shown in Tab. 5.

**CFormer+DySPN.** Our model works significantly better than this baseline, improving the RMSE from 123.6mm to 112.2mm on NYU. Our model requires slightly more computation than the baseline: the FPS drops by 38% and the memory usage increases by 10%. We believe our model's state-of-the-art accuracy and generalization is worth this trade-off in computation.

**DDI.** We train a model without DDI, where the ConvGRU directly generates updates on the depth map. Comparing (No DDI) to ours, DDI significantly improves MAE from 44.9mm to 38.0mm on NYU. We also examine the convergence speed of DDI using different initialization strategies. Our init. from current solution (Sec. 4) effectively reduce the DDI latency by 62.1% and 56.3% compared to init. from zeros and using the heuristic-based pre-filled depth as in LRRU [46].

**Iterative Refinement.** To prove the effectiveness of iterative refinement, we unroll the ConvGRU for $1 \sim 7$ times. On NYU, 5 iterations improve MAE from 39.9mm to 38.0mm compared to 1 iteration. Unrolling 7 iterations provides no further improvements. Therefore, we use 5 iterations as a balance between performance and speed. See Fig. 2 for qualitative effect of the iterative refinement.

**ConvGRU** To show the advantage of ConvGRU over other iterative designs, we compare against ConvRNN and a ConvGRU without weighting tying among iterations (Cascaded). ConvGRU has higher capacity with the gated mechanism and therefore achieves better performance than ConvRNN. Ours achieves similar performance with fewer parameters compared to Cascaded.

## Acknowledgments

This work was primarily supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number 140D0423C0075. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

We thank Jing Wen, Alexander Raistrick, and other Princeton Vision & Learning Lab members for their insightful discussions and detailed comments on the manuscript.

## References

1. Bae, G., Budvytis, I., Cipolla, R.: Irondepth: Iterative refinement of single-view depth using surface normal and its uncertainty. In: BMVC (2022)
2. Bartolomei, L., Poggi, M., Conti, A., Tosi, F., Mattoccia, S.: Revisiting depth completion from a stereo matching perspective for cross-domain generalization. arXiv preprint arXiv:2312.09254 (2023)
3. Carranza-García, M., Galán-Sales, F.J., Luna-Romera, J.M., Riquelme, J.C.: Object detection using depth completion and camera-lidar fusion for autonomous driving. Integrated Computer-Aided Engineering **29**(3), 241–258 (2022)
4. Cheng, X., Wang, P., Guan, C., Yang, R.: Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In: AAAI (2020)
5. Cheng, X., Wang, P., Yang, R.: Learning depth with convolutional spatial propagation network. IEEE TPAMI (2019)
6. Cheng, X., Wang, P., Yang, R.: Learning depth with convolutional spatial propagation network. IEEE transactions on pattern analysis and machine intelligence **42**(10), 2361–2379 (2019)
7. Conti, A., Poggi, M., Mattoccia, S.: Sparsity agnostic depth completion. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 5871–5880 (2023)
8. Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., Gaidon, A.: 3d packing for self-supervised monocular depth estimation. In: CVPR (2020)
9. Häne, C., Heng, L., Lee, G.H., Fraundorfer, F., Furgale, P., Sattler, T., Pollefeys, M.: 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image and Vision Computing **68**, 14–27 (2017)
10. Hawe, S., Kleinsteuber, M., Diepold, K.: Dense disparity maps from sparse disparity measurements. In: 2011 International Conference on Computer Vision. pp. 2126–2133. IEEE (2011)
11. Hestenes, M.R., Stiefel, E., et al.: Methods of conjugate gradients for solving linear systems. Journal of research of the National Bureau of Standards **49**(6), 409–436 (1952)

12. Holynski, A., Kopf, J.: Fast depth densification for occlusion-aware augmented reality. ACM TOG **37**(6), 1–11 (2018)
13. Hu, M., Wang, S., Li, B., Ning, S., Fan, L., Gong, X.: Penet: Towards precise and efficient image guided depth completion. In: ICRA. pp. 13656–13662. IEEE (2021)
14. Hu, Y.T., Schwing, A.G., Yeh, R.A.: Surface snapping optimization layer for single image object shape reconstruction. In: ICML (2023)
15. Huang, J., Zhou, Y., Funkhouser, T., Guibas, L.J.: Framenet: Learning local canonical frames of 3d surfaces from a single rgb image. In: ICCV. pp. 8638–8647 (2019)
16. Imran, S., Liu, X., Morris, D.: Depth completion with twin surface extrapolation at occlusion boundaries. In: CVPR. pp. 2583–2592 (2021)
17. Khan, M.A.U., Nazir, D., Pagani, A., Mokayed, H., Liwicki, M., Stricker, D., Afzal, M.Z.: A comprehensive survey of depth completion approaches. Sensors **22**(18), 6969 (2022)
18. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: CVPR. pp. 2041–2050 (2018)
19. Liao, Y., Huang, L., Wang, Y., Kodagoda, S., Yu, Y., Liu, Y.: Parse geometry from a line: Monocular depth estimation with partial laser observation. In: ICRA. pp. 5059–5066. IEEE (2017)
20. Lin, Y., Cheng, T., Zhong, Q., Zhou, W., Yang, H.: Dynamic spatial propagation network for depth completion. In: AAAI (2022)
21. Lipson, L., Teed, Z., Deng, J.: Raft-stereo: Multilevel recurrent field transforms for stereo matching. In: 3DV. pp. 218–227. IEEE (2021)
22. Liu, C., Kumar, S., Gu, S., Timofte, R., Van Gool, L.: Va-depthnet: A variational approach to single image depth prediction. In: ICLR (2023)
23. Liu, L.K., Chan, S.H., Nguyen, T.Q.: Depth reconstruction from sparse samples: Representation, algorithm, and sampling. IEEE TIP **24**(6), 1983–1996 (2015)
24. Liu, L., Song, X., Lyu, X., Diao, J., Wang, M., Liu, Y., Zhang, L.: Fcfr-net: Feature fusion based coarse-to-fine residual learning for depth completion. In: AAAI (2021)
25. Liu, X., Shao, X., Wang, B., Li, Y., Wang, S.: Graphcspn: Geometry-aware depth completion via dynamic gcns. In: ECCV (2022)
26. Long, C., Zhang, W., Chen, Z., Wang, H., Liu, Y., Cao, Z., Dong, Z., Yang, B.: Sparsedc: Depth completion from sparse and non-uniform inputs. arXiv preprint arXiv:2312.00097 (2023)
27. Long, X., Lin, C., Liu, L., Li, W., Theobalt, C., Yang, R., Wang, W.: Adaptive surface normal constraint for depth estimation. In: ICCV. pp. 12849–12858 (2021)
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018)
29. Ma, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In: ICRA (2018)
30. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5), 1147–1163 (2015)
31. Mur-Artal, R., Tardós, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics **33**(5), 1255–1262 (2017)
32. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
33. Park, J., Joo, K., Hu, Z., Liu, C.K., So Kweon, I.: Non-local spatial propagation network for depth completion. In: ECCV (2020)
34. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS **32** (2019)

35. Qi, X., Liao, R., Liu, Z., Urtasun, R., Jia, J.: Geonet: Geometric neural network for joint depth and surface normal estimation. In: CVPR. pp. 283–291 (2018)
36. Qi, X., Liu, Z., Liao, R., Torr, P.H., Urtasun, R., Jia, J.: Geonet++: Iterative geometric neural network with edge-aware refinement for joint depth and surface normal estimation. IEEE TPAMI **44**(2), 969–984 (2020)
37. Qiu, J., Cui, Z., Zhang, Y., Zhang, X., Liu, S., Zeng, B., Pollefeys, M.: Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In: CVPR (2019)
38. Ramamonjisoa, M., Du, Y., Lepetit, V.: Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In: CVPR. pp. 14648–14657 (2020)
39. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3) (2022)
40. Shao, S., Pei, Z., Chen, W., Wu, X., Li, Z.: Nddepth: Normal-distance assisted monocular depth estimation. In: ICCV. pp. 7931–7940 (2023)
41. Tang, J., Tian, F.P., Feng, W., Li, J., Tan, P.: Learning guided convolutional network for depth completion. IEEE TIP (2020)
42. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: ECCV. pp. 402–419. Springer (2020)
43. Teed, Z., Deng, J.: Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. NeurIPS **34**, 16558–16569 (2021)
44. Teed, Z., Lipson, L., Deng, J.: Deep patch visual odometry. NeurIPS **36** (2024)
45. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. In: 3DV (2017)
46. Wang, Y., Li, B., Zhang, G., Liu, Q., Gao, T., Dai, Y.: Lrru: Long-short range recurrent updating networks for depth completion. In: CVPR. pp. 9422–9432 (2023)
47. Wong, A., Fei, X., Tsuei, S., Soatto, S.: Unsupervised depth completion from visual inertial odometry. IEEE Robotics and Automation Letters (2020)
48. Xu, Z., Yin, H., Yao, J.: Deformable spatial propagation networks for depth completion. In: ICIP (2020)
49. Yan, Z., Wang, K., Li, X., Zhang, Z., Xu, B., Li, J., Yang, J.: Rignet: Repetitive image guided network for depth completion. In: ECCV (2022)
50. Yeh, R.A., Hu, Y.T., Ren, Z., Schwing, A.G.: Total variation optimization layers for computer vision. In: CVPR. pp. 711–721 (2022)
51. Yin, W., Liu, Y., Shen, C.: Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction. IEEE TPAMI **44**(10), 7282–7295 (2021)
52. Zhang, Y., Funkhouser, T.: Deep depth completion of a single rgb-d image. In: CVPR. pp. 175–185 (2018)
53. Zhang, Y., Guo, X., Poggi, M., Zhu, Z., Huang, G., Mattoccia, S.: Completionformer: Depth completion with convolutions and vision transformers. In: CVPR (2023)
54. Zhao, S., Gong, M., Fu, H., Tao, D.: Adaptive context-aware multi-modal network for depth completion. IEEE TIP (2021)
55. Zhao, W., Liu, S., Wei, Y., Guo, H., Liu, Y.J.: A confidence-based iterative solver of depths and surface normals for deep multi-view stereo. In: ICCV. pp. 6168–6177 (2021)
56. Zhou, W., Yan, X., Liao, Y., Lin, Y., Huang, J., Zhao, G., Cui, S., Li, Z.: Bev@dc: Bird's-eye view assisted training for depth completion. In: CVPR. pp. 9233–9242 (2023)