

# GAUSSIAN BELIEF PROPAGATION NETWORK FOR DEPTH COMPLETION

000  
001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
Anonymous authors  
Paper under double-blind review

## ABSTRACT

Depth completion aims to predict a dense depth map from a color image with sparse depth measurements. Although deep learning methods have achieved state-of-the-art (SOTA), effectively handling the sparse and irregular nature of input depth data in deep networks remains a significant challenge, often limiting performance, especially under high sparsity. To overcome this limitation, we introduce the Gaussian Belief Propagation Network (GBPN), a novel hybrid framework synergistically integrating deep learning with probabilistic graphical models for end-to-end depth completion. Specifically, a scene-specific Markov Random Field (MRF) is dynamically constructed by the Graphical Model Construction Network (GMCN), and then inferred via Gaussian Belief Propagation (GBP) to yield the dense depth distribution. Crucially, the GMCN learns to construct not only the data-dependent potentials of MRF but also its structure by predicting adaptive non-local edges, enabling the capture of complex, long-range spatial dependencies. Furthermore, we enhance GBP with a serial & parallel message passing scheme, designed for effective information propagation, particularly from sparse measurements. Extensive experiments demonstrate that GBPN achieves SOTA performance on the NYUv2 and KITTI benchmarks. **Evaluations across varying sparsity levels, sparsity patterns, and datasets highlight GBPN's superior performance, notable robustness, and generalizable capability.**

## 1 INTRODUCTION

Dense depth estimation is critical for various computer vision and robotics tasks. While dedicated sensors or methods provide sparse, irregular depth measurements (Schonberger & Frahm, 2016), acquiring dense depth directly is often challenging or costly. Depth completion (DC) bridges this gap through inferring a dense depth map from sparse depth guided by a synchronized color image. Sparse depth provides essential scale and absolute constraints, while the high-resolution color image offers rich structural and semantic information necessary for propagating depth and filling missing regions. Consequently, DC is a vital technique attracting increasing interest for enhancing downstream applications like 3D object detection (Wu et al., 2022), novel view synthesis (Roessle et al., 2022), and robotic manipulation (Li et al., 2023).

Traditional DC methods mostly relied on hand-crafted pipelines (Ku et al., 2018) or fixed graphical models like Markov Random Fields (MRF) (Diebel & Thrun, 2005). Although offering some robustness, their rigid, pre-defined priors struggled to capture complex geometry and fine details. Recently, learning-based methods, primarily deep neural networks, have achieved impressive accuracy improvements by directly regressing dense depth from extracted features (Ma & Karaman, 2018). However, effectively processing sparse, irregular input within standard deep architectures remains a significant challenge (Uhrig et al., 2017; Huang et al., 2019; Tang et al., 2024), often degrading performance and robustness, particularly under high sparsity.

In this paper, we introduce the Gaussian Belief Propagation Network (GBPN), a novel hybrid framework that synergistically combines deep learning's representational power with the structured inference of probabilistic graphical models. Unlike methods that directly regress depth, GBPN trains a deep network (the Graphical Model Construction Network, GMCN) to dynamically construct a scene-specific MRF over dense depth variables. Dense depth is then efficiently inferred via Gaussian Belief Propagation (GBP) on the learned MRF.

054 This formulation offers several key advantages. Firstly, learning to construct a scene-specific MRF  
 055 overcomes the limitations of fixed, hand-crafted models, enabling adaptation to diverse geometries.  
 056 Secondly, sparse depth measurements are naturally integrated as principled data terms within the  
 057 globally consistent MRF framework, inherently addressing input sparsity and irregularity by propa-  
 058 gating depth across the entire image. Finally, GBPN yields a depth distribution, providing valuable  
 059 confidence estimates for risk-aware downstream tasks, such as planning (Burns & Brock, 2007).

060 To realize this, our GMCN infers not only potentials of the MRF but also its structure by predicting  
 061 non-local edges, allowing the model to adaptively capture complex, long-range spatial dependencies  
 062 guided by image content. We also propose a novel serial & parallel message passing scheme for  
 063 GBP to enhance information flow, particularly from sparse measurements to distant unmeasured  
 064 pixels. The entire GBPN is trained end-to-end using a probability-based loss function leveraging the  
 065 estimated mean and precision from GBP, promoting the learning of reliable depth predictions along  
 066 with an estimate of their confidence.

067 We validate the efficacy of GBPN through extensive experiments on two leading DC benchmarks:  
 068 NYUv2 for indoor scenes and KITTI for outdoor scenes. Our method achieves state-of-the-art perfor-  
 069 mance on these datasets at the time of submission. Comprehensive ablation studies demonstrate the  
 070 effectiveness of each component within GBPN, including the dynamic MRF construction, propagation  
 071 scheme, and probability-based loss function, *etc.* Furthermore, evaluations across varying sparsity  
 072 levels, sparsity patterns, and datasets highlight GBPN’s superior performance, notable robustness,  
 073 and generalizable capability. Code and trained models will be available at *omitted for blind review*.

## 074 2 RELATED WORK

075 Research on depth completion (DC) has undergone a significant evolution, transitioning from tra-  
 076 ditional hand-crafted techniques to data-driven learning methods. Notably, concepts and principles  
 077 from traditional approaches have significantly influenced the design of recent learning-based methods.  
 078 This section briefly reviews these two lines of work and the combination of their respective strengths.

079 **Traditional Hand-crafted Techniques.** Early DC approaches largely relied on explicitly defined  
 080 priors and assumptions about scene geometry and texture, often borrowing techniques from traditional  
 081 image processing and inpainting. These methods constructed hand-crafted pipelines to infer dense  
 082 depth from sparse measurements. For example, Kopf et al. (2007) proposed joint bilateral filtering  
 083 guided by color information to upsample low-resolution depth maps. Ku et al. (2018) employed a  
 084 sequence of classical image processing operators, such as dilation and hole filling, for depth map  
 085 densification. Zhao et al. (2021b) leveraged local surface smoothness assumptions to estimate depth  
 086 by computing surface normals in a spherical coordinate system. Hawe et al. (2011) reconstructed  
 087 dense disparity from sparse measurements by minimizing an energy function formulated by Com-  
 088 pressive Sensing. Diebel & Thrun (2005) modeled depth completion as a multi-resolution Markov  
 089 Random Field (MRF) with simple smoothness potentials, and then solved using conjugate gradient  
 090 optimization. Chen & Koltun (2014) developed a global optimization approach to reconstruct dense  
 091 depth modeled by MRF. While demonstrating effectiveness in certain scenarios, these hand-crafted  
 092 methods often struggle to capture complex geometric structures and fine-grained details.

093 **Data-driven Learning Methods.** In contrast, learning-based methods typically formulate DC as  
 094 an end-to-end regression task, taking color images and sparse depth maps as input and directly  
 095 predicting the target dense depth map. A primary focus in this line of work has been on effectively  
 096 handling the sparse input data and fusing information from different modalities (color and depth).  
 097 For explicitly processing sparse data, pioneering work by Uhrig et al. (2017) introduced sparsity-  
 098 invariant convolutions, designed to handle missing data by maintaining validity masks. Huang et al.  
 099 (2019) integrated the sparsity-invariant convolutions into an encoder-decoder architecture. Eldesokey  
 100 et al. (2019) extended this concept by propagating a continuous confidence measure. However,  
 101 general-purpose network architectures employing standard convolutional layers, often combined  
 102 with sophisticated multi-modal fusion strategies, have frequently demonstrated strong performance.  
 103 For instance, Tang et al. (2020) proposed to learn content-dependent and spatially-variant kernels to  
 104 guide the fusion of color and depth features. Zhang et al. (2023) leveraged Transformer architectures  
 105 to capture long-range dependencies for feature extraction in depth completion. Chen et al. (2019)  
 106 used 2D convolutions for image features and continuous convolutions for 3D point cloud features,  
 107 followed by fusion. While achieving impressive results and learning complex mappings from data,

108 purely data-driven methods often exhibit limited generalization performance, particularly when faced  
 109 with data of different sparsity levels.  
 110

111 **Combination of Models and Learning.** More recently, a significant trend has emerged towards  
 112 approaches that combine the strengths of both traditional modeling and data-driven learning. These  
 113 methods aim to leverage the benefits of learned features and powerful network architectures while  
 114 incorporating explicit priors or structured inference mechanisms. A famous pioneering work is Liu  
 115 et al. (2015), combining deep network with closed-form solver for monocular depth estimation. For  
 116 depth completion, a prominent line of work, exemplified by CSPN-based methods (Cheng et al.,  
 117 2019; 2020; Lin et al., 2022; Park et al., 2020), employs deep networks to predict parameters for a  
 118 learned anisotropic diffusion process, refining the regressed depth from deep network. Wang et al.  
 119 (2023a) integrated traditional image processing techniques to generate an initial depth map before  
 120 applying a learned refinement module. Tang et al. (2024) learned a bilateral filter-like propagation  
 121 process to effectively spread information from sparse measurements, Qu et al. (2020) combined  
 122 deep learning with a least-squares solver to estimate depth with constraints derived from sparse  
 123 measurements. Zuo & Deng (2024) formulated depth completion as a learned optimization problem,  
 124 where a network predicts local depth differences used in an energy function iteratively minimized via  
 125 conjugate gradient. However, these methods still struggle with sparse data processing and limited  
 126 propagation range. Our method falls within this hybrid category, by learning a MRF and inferring it  
 127 via GBP, inherently addressing the issues of input sparsity and irregularity.  
 128

### 129 3 THE PROPOSED METHOD

#### 130 3.1 OVERVIEW

131 Given a color image  $I \in \mathbb{R}^{H \times W \times 3}$ , regardless of how sparse depth is measured—whether via active  
 132 sensor, like LiDAR (Geiger et al., 2012), or passive method, like SfM (Schonberger & Frahm, 2016),  
 133 or even interactive user guidance (Ron et al., 2018)—we can project these depth measurements onto  
 134 the image plane to yield a sparse depth map  $S \in \mathbb{R}^{H \times W}$ , with the same resolution  $(H, W)$  as  $I$ .  
 135 The valid pixels in  $S$  are typically irregularly distributed, and may vary significantly in number and  
 136 location. Unlike most learning-based approaches that employ dedicated neural network layers to  
 137 process  $S$  (Eldesokey et al., 2019; Huang et al., 2019) and directly regress the dense depth map  
 138  $X \in \mathbb{R}^{H \times W}$ , as illustrated in Fig. 1, we formulate the dense depth estimation task as inference in a  
 139 Markov Random Field (MRF) (in Section 3.2), and infer  $X$  via Gaussian Belief Propagation (GBP)  
 140 (in Section 3.3). In this formulation,  $S$  serves as data term within the global optimization framework  
 141 of MRF, which eliminates the need for designing specific neural network architectures tailored to  
 142 processing sparse input data (Tang et al., 2024). Specifically, the MRF structure, particularly its edges,  
 143 is dynamically generated by a graphical model construction network (in Section 3.4), depending on  
 144 the input color image  $I$  and optionally on intermediate estimates of the dense depth distribution. The  
 145 entire framework is end-to-end trained with a probability-based loss function (in Section 3.5).  
 146

#### 147 3.2 PROBLEM FORMULATION

148 We formulate the dense depth estimation problem using a Markov Random Field (MRF), *i.e.* a type  
 149 of undirected graphical model  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  is the set of nodes representing the image pixels  
 150 and  $\mathcal{E}$  is the set of pairwise edges connecting neighboring pixels. As shown in Fig. 1, each node  
 151  $i \in \mathcal{V}$  is associated with a random variable  $x_i$ , representing the depth at pixel  $i$ . The joint probability  
 152 distribution over the depth variables is defined according to the MRF structure:

$$153 \quad p(X|I, S) \propto \prod_{i \in \mathcal{V}_v} \phi_i \prod_{(i,j) \in \mathcal{E}} \psi_{ij}, \quad (1)$$

154 where  $\phi_i$  and  $\psi_{ij}$  are the abbreviations for  $\phi_i(x_i)$  and  $\psi_{ij}(x_i, x_j)$ , indicating unary and pairwise po-  
 155 tentials respectively, and  $\mathcal{V}_v$  is the set of nodes corresponding to pixels with a valid depth measurement  
 156 in the sparse map  $S$ .  
 157

158 The *unary potential*  $\phi_i$  defined for pixel  $i \in \mathcal{V}_v$ , constrains the quadratic distance between estimated  
 159 depth  $x_i$  and the depth measurement  $s_i$ , written as  
 160

$$161 \quad \phi_i = \exp\left(-\frac{w_i(x_i - s_i)^2}{2}\right). \quad (2)$$

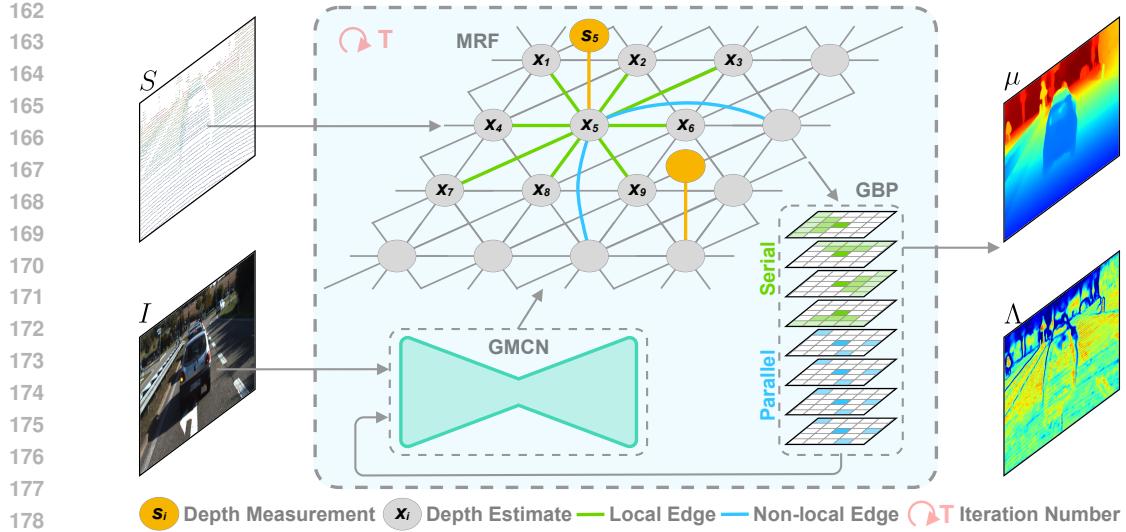


Figure 1: **Overview of the proposed approach.** Markov Random Field (MRF) is constructed depending on parameters dynamically generated from the Graphical Model Construction Network (GMCN), and then optimized via Gaussian Belief Propagation (GBP) for the distribution of dense depth map.

Here,  $w_i > 0$  is a weight indicating the confidence for the depth measurement, and  $\phi_i$  is only available if depth measurement  $s_i$  is valid.

The *pairwise potential*  $\psi_{ij}$  defined for edge  $(i, j) \in \mathcal{E}$ , encourages spatial coherence between the estimated depths of neighboring pixels  $i$  and  $j$ , expressed as

$$\psi_{ij} = \exp\left(-\frac{w_{ij}(x_i - x_j - r_{ij})^2}{2}\right). \quad (3)$$

Here,  $w_{ij} > 0$  is a weight controlling the strength of the spatial constraint, and  $r_{ij}$  is the expected depth difference between  $x_i$  and  $x_j$ .

In traditional MRF-based approaches, the parameters  $w$ , and  $r$  are typically hand-crafted based on simple assumptions or features. For instance,  $r_{ij}$  is often set to 0 to enforce simple smoothness (Diebel & Thrun, 2005), and  $w_{ij}$  is commonly derived from local image features like color differences. These approaches, while providing some robustness, often limited the model's ability to capture complex scene geometry and fine details, leading to suboptimal results. In contrast, we employ an end-to-end trained deep network to dynamically construct the MRF based on the color image and optional optimized intermediate dense depth map distribution. This allows the MRF to adaptively model the scene, via placing stronger data constraints where measurements seem reliable, and applying sophisticated smoothness constraints that vary based on image content. Moreover, we further enhance the MRF's expressive power with *dynamic parameters* and *dynamic edges*.

**MRF with Dynamic Parameters.** The constructed MRF is approximately inferred in an iterative manner. Unlike traditional methods that define the MRF parameters a priori and keep them fixed during optimization, our approach updates the MRF parameters dynamically as the iterative inference progresses. This adaptive parameterization allows the MRF to evolve alongside the solution, making it more responsive to the current state of the variables and potentially improving both the solution quality and the convergence behavior of the optimization.

**MRF with Dynamic Edges.** Traditionally, the graph structure of an MRF, specifically the edges connecting a variable to its neighbors, is fixed and predefined, *e.g.* 8-connected local pixels for each variable. As illustrated in Fig. 1, alongside these fixed local edges (depicted in green), our framework also dynamically estimates and establishes non-local connections for each pixel (shown in blue). This dynamic graph structure allows the MRF to capture dependencies beyond immediate spatial neighbors, increasing its flexibility and capacity to model complex scene structures. Consequently, during inference, each variable's state is conditioned on both its predefined local neighbors and its dynamically generated non-local connections.

216 3.3 APPROXIMATE INFERENCE TECHNOLOGY  
217218 3.3.1 GAUSSIAN BELIEF PROPAGATION  
219

220 For a high-resolution image, computing the exact posterior for the above MRF is computationally  
221 prohibitive, due to the large number of variables and complex dependencies. Belief Propagation  
222 (BP) (Pearl, 1988) is a widely used algorithm for performing approximate probabilistic inference  
223 in graphical models. By iteratively operating on belief updating and message passing, BP enables  
224 parallel computation and often exhibits reasonably fast empirical convergence compared to other  
225 iterative inference methods (Weiss & Freeman, 1999). Though lacking formal convergence guarantees  
226 for general graphs, it has been successfully applied to various graphs, even with cycles (Freeman et al.,  
227 2000; Sun et al., 2003), known as loopy belief propagation. As introduced below and Section 3.3.2,  
228 we adopts damping tricks and graph decomposition to improve the stability and convergence.

229 In BP, the belief on variable  $x_i$  is proportional to the product of its local evidence  $\phi_i$  and incoming  
230 messages from all neighboring variables, written as

$$231 \quad b_i \propto \phi_i \prod_{(i,j) \in \mathcal{E}} m_{j \rightarrow i}. \quad (4)$$

232 Messages are computed by marginalizing over the variable of the sending node, considering its local  
233 evidence and incoming messages from its other neighbors, expressed as

$$234 \quad m_{j \rightarrow i} \propto \int_{x_j} \psi_{ij} \phi_j \prod_{(k,j) \in \mathcal{E} \setminus (i,j)} m_{k \rightarrow j} dx_j. \quad (5)$$

235 The formulated MRF in Section 3.2 is a Gaussian graph model, for which we can use Gaussian Belief  
236 Propagation (GBP) for inference. GBP (Bickson, 2008) assumes all beliefs  $b$  and messages  $m$  are  
237 under Gaussian distribution. This assumption allows the complex integration and product operations  
238 in eq. (5) to be simplified into algebraic updates on the parameters of the Gaussian distributions. We  
239 can take moment form  $\mathcal{N}(\mu, \Lambda^{-1})$  or canonical form  $\mathcal{N}^{-1}(\eta, \Lambda)$  as Gaussian parametrization for  
240 convenience, where  $\eta = \mu\Lambda$ . Then, in GBP, the belief updating corresponding to eq. (4) is given by:

$$241 \quad \eta_i = w_i s_i + \sum_{(i,j) \in \mathcal{E}} \hat{\eta}_{j \rightarrow i},$$

$$242 \quad \Lambda_i = w_i + \sum_{(i,j) \in \mathcal{E}} \hat{\Lambda}_{j \rightarrow i}. \quad (6)$$

243 And message passing corresponding to eq. (5) is given by:

$$244 \quad \mu_{j \rightarrow i} = \mu_{j \setminus i} + r_{ij},$$

$$245 \quad \Lambda_{j \rightarrow i}^{-1} = \Lambda_{j \setminus i}^{-1} + w_{ij}^{-1}. \quad (7)$$

246 Here,  $j \setminus i$  means the set of all messages sent to  $j$  except the message from  $i$ . We leave the detailed  
247 derivation of Gaussian belief propagation in the Section A.1.1. To improve convergence and stability,  
248 we adopt the *damping* trick for message passing (Murphy et al., 2013) by applying weighted average  
249 of the message from the previous iteration and the message computed in the current iteration, i.e.  
250  $\hat{m}_{j \rightarrow i, t} = \hat{m}_{j \rightarrow i, t-1}^{\beta_i} m_{j \rightarrow i, t}^{1-\beta_i}$ . Once again corresponding to Gaussian parameters update, given by:

$$251 \quad \hat{\eta}_{j \rightarrow i, t} = \beta_i \hat{\eta}_{j \rightarrow i, t-1} + (1 - \beta_i) \eta_{j \rightarrow i, t},$$

$$252 \quad \hat{\Lambda}_{j \rightarrow i, t} = \beta_i \hat{\Lambda}_{j \rightarrow i, t-1} + (1 - \beta_i) \Lambda_{j \rightarrow i, t}. \quad (8)$$

253 3.3.2 PROPAGATION SCHEME  
254

255 A crucial aspect of Belief Propagation is the definition of its message propagation scheme (Davison  
256 & Ortiz, 2019). Both serial and parallel propagation schemes offer distinct advantages. Serial  
257 propagation is effective at propagating information across the entire graph structure, allowing local  
258 evidence to influence distant nodes. This is particularly important for depth completion, ensuring  
259 that information from depth measurements propagates broadly, resulting in a dense depth estimation  
where every variable receives sufficient incoming messages to form a valid belief. In contrast, parallel

270 propagation, propagating messages over short ranges, is significantly more efficient by effectively  
 271 leveraging modern hardware parallelism.  
 272

273 To combine both advantages, we design a hybrid serial & parallel propagation scheme. As illustrated  
 274 in Fig. 1, our scheme splits the message passing into updates performed serially on directional  
 275 local connections and updates performed in parallel on non-local connections, decomposing the  
 276 loopy graph into loop-free sub-graphs. Specifically, we categorize edges  $\mathcal{E}$  in MRF into four sets  
 277 corresponding to local directional sweeps: left-to-right (LR), top-to-bottom (TB), right-to-left (RL),  
 278 and bottom-to-top (BT), denoted  $\mathcal{E}_{LR}$ ,  $\mathcal{E}_{TB}$ ,  $\mathcal{E}_{RL}$ , and  $\mathcal{E}_{BT}$  respectively, and a set for non-local  
 279 connections,  $\mathcal{E}_{NL}$ . With this decomposition, each set of edges is with directionality and loop-free,  
 280 which is more stable for convergence.

281 For serial propagation, we sequentially perform message passing sweeps utilizing these four local  
 282 directional edge sets. Under each sweep direction, message and belief updates are processed serially  
 283 according to a defined order (e.g., column by column for horizontal sweeps). For instance, during  
 284 the sweep using edges  $\mathcal{E}_{LR}$  (left-to-right), message updates for variables in column  $n$  are computed  
 285 only after updates related to column  $n - 1$  have been completed. And the update for a variable  
 286 at pixel  $(m, n)$  incorporates messages received from specific neighbors in column  $n - 1$ , such as  
 287 those connected via edges from  $\mathcal{E}_{LR}$  linking  $(m - 1, n - 1)$ ,  $(m, n - 1)$ , and  $(m + 1, n - 1)$  to  
 288  $(m, n)$ . For non-local propagation, message updates for all pixels based on connections in  $\mathcal{E}_{NL}$  are  
 289 computed simultaneously. These non-local connections are dynamically generated to link pixels that  
 290 are relevant but not necessarily spatially adjacent.

291 The whole propagation scheme is detailed in Algorithm 1. We initialize the  $\eta$  and  $\Lambda$  parameters for  
 292 all messages on all relevant edges to 0. We empirically set a fixed total number of iterations  $T$ . In  
 293 each iteration, we first sequentially perform the four directional local serial propagation sweeps (LR,  
 294 TB, RL, BT). Following the serial sweeps, we execute  $T_n$  steps of parallel non-local propagation.  
 295 Finally, after  $T$  iterations, we yield the marginal beliefs for each pixel  $i$ . The estimated depth is the  
 296 mean  $\mu_i = \eta_i / \Lambda_i$  with the precision  $\Lambda_i$  serving as a measure of confidence. The visualization of  
 297 propagation and learned edges is introduced in Section A.3.

---

**Algorithm 1** Serial & Parallel Propagation Scheme
 

---

299 **Require:** Graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . Edge subsets  $\mathcal{E}_{LR}, \mathcal{E}_{TB}, \mathcal{E}_{RL}, \mathcal{E}_{BT}, \mathcal{E}_{NL}$ .

300 **Ensure:** Estimated marginal belief represented by  $\eta, \Lambda$ .

301 1: **Initialization:**  
 302 2: **for** all edges  $(i, j) \in \mathcal{E}$  **do** ▷ Initialize messages  
 303 3:      $\eta_{j \rightarrow i} = 0$   
 304 4:      $\Lambda_{j \rightarrow i} = 0$   
 305 5: **end for**  
 306 6: **Iterations:**  
 307 7: **for**  $t = 1$  to  $T$  **do**  
 308 8:     **for** all  $\mathcal{E}_{LR}, \mathcal{E}_{TB}, \mathcal{E}_{RL}, \mathcal{E}_{BT}$  **do** ▷ Serial Propagation  
 309 9:         Serial message passing(eqs. (7) and (8)) and belief updating(eq. (6))  
 310 10:     **end for**  
 311 11:     **for**  $t_n = 1$  to  $T_n$  **do** ▷ Parallel Propagation  
 312 12:         message passing (eqs. (7) and (8)) for all  $(i, j) \in \mathcal{E}_{NL}$  in parallel  
 313 13:         belief updating (eq. (6))  
 314 14:     **end for**  
 315 15: **end for**  
 16: **return**  $\eta, \Lambda$

---

 316  
 317 3.4 GRAPHICAL MODEL CONSTRUCTION NETWORK
 

---

318 As previously mentioned, the Markov Random Field (MRF) is dynamically constructed and subse-  
 319 quently optimized using Gaussian Belief Propagation (GBP). Our Graphical Model Construction  
 320 Network is designed to learn the parameters and structures of this MRF from input data.  
 321

322 We employ a U-Net architecture (Ronneberger et al., 2015), comprising encoder and decoder layers,  
 323 to extract multi-scale features essential for MRF construction. Within the U-Net, we introduce a novel

324 global-local processing unit. Each unit combines a dilated neighborhood attention layer (Hassani &  
 325 Shi, 2022) and a ResNet block (He et al., 2016). The dilated neighborhood attention layer is utilized  
 326 to capture long-range dependencies, effectively expanding the receptive field without increasing  
 327 computational cost. As a complement, the ResNet block focuses on extracting and refining local  
 328 features. More details about the network architecture is introduced in Section A.1.2.

329 We apply convolutional layers on the aggregated features to estimate the MRF parameters, such as  $r$   
 330 and  $w$ . Additionally, the network estimates the damping rate  $\beta$  used in the GBP and the offsets for  
 331 constructing non-local neighboring pixels. Inspired by Deformable Convolutional Networks (Dai  
 332 et al., 2017), we employ bilinear interpolation to sample Gaussian parameters at these non-local  
 333 neighbor locations defined by the estimated float offsets. This approach ensures that gradients can be  
 334 effectively backpropagated during the training phase. Considering that GBP typically converges to an  
 335 accurate mean  $\mu$  but not the exact precision  $\Lambda$  (Weiss & Freeman, 1999), we also utilize convolution  
 336 layers to estimate a residual term. This residual is added to the estimated  $\Lambda$  from GBP, and a sigmoid  
 337 activation function is then applied to yield the updated precision.

338 We provide two approaches to construct the MRF: a color image-only approach (termed GBPN-1)  
 339 and a multi-modal fusion approach (termed GBPN-2) incorporating depth information. In GBPN-1,  
 340 the U-Net architecture receives a color image as its sole input to construct the MRF. The GBPN-2  
 341 utilizes a second U-Net that takes both the color image and the optimized depth distribution from the  
 342 GBPN-1 (*i.e.*  $\mu$  and  $\Lambda$ ) as input. For GBPN-2, cross-attention is leveraged within the dilated attention  
 343 layers, where the query is generated from multi-modal features, while the keys and values are derived  
 344 from the color image features in the first U-Net.

### 345 3.5 PROBABILITY-BASED LOSS FUNCTION

347 We adopt the combination of  $L_1$  and  $L_2$  loss as the loss on depth. For a pixel  $i$ , the loss on depth is  
 348

$$349 \quad L_i^X = \frac{\|\mu_i - x_i^g\|_2^2 + \alpha \|\mu_i - x_i^g\|_1}{\max(\|\mu - x^g\|_1)}, \quad (9)$$

352 where  $x_i^g$  is the ground-truth depth and  $\alpha$  is the hyperparameter to balance the  $L_1$  and  $L_2$  loss. The  
 353 combined  $L_1$  and  $L_2$  loss is normalized with the maximum  $L_1$  loss across the image, which makes  
 354 the convergence more stable. As the output of our framework is the distribution of dense depth  
 355 map, which is in Gaussian and can be represented with  $\mu_i$  and  $\Lambda_i$  for each pixel, we adopt the  
 356 probability-based loss (Kendall & Gal, 2017) as the final loss:

$$357 \quad L = \frac{1}{|\mathcal{V}_g|} \sum_{i \in \mathcal{V}_g} \Lambda_i L_i^X - \log(\Lambda_i). \quad (10)$$

360 Here,  $\mathcal{V}_g$  denotes the set of pixels with valid ground-truth depth. In this way, the precision  $\Lambda$  can also  
 361 be directly supervised, without the need for its ground-truth value.

## 363 4 EXPERIMENTS

365 We evaluate our proposed method, GBPN, on two leading depth completion benchmarks: NYUv2  
 366 (Silberman et al., 2012) for indoor scenes and KITTI (Geiger et al., 2012) for outdoor scenes. To  
 367 demonstrate its effectiveness, we provide comprehensive comparisons against state-of-the-art (SOTA)  
 368 methods in Section 4.1. We then perform thorough ablation studies in Section 4.2 to analyze the  
 369 contribution of core components. We also assess the robustness of GBPN by varying input sparsity  
 370 levels in Section 4.3, a critical factor for real-world deployment. Finally, cross-dataset evaluations  
 371 in Section 4.4 verify the model’s generalization capability. The appendix contains further analysis  
 372 on the experimental setup (Section A.2), noise sensitivity (Section A.6), and runtime efficiency  
 373 (Section A.7).

### 375 4.1 COMPARISON WITH STATE-OF-THE-ART METHODS

377 We evaluate GBPN (the GBPN-2) on the official test sets of the NYUv2 (Silberman et al., 2012)  
 378 dataset and the KITTI Depth Completion (DC) (Geiger et al., 2012) dataset. Quantitative comparisons

378 between GBPN and other top-performing published methods are presented in Table 1. On the KITTI  
 379 DC benchmark<sup>1</sup>, our method achieves the best iRMSE among all submissions at the time of writing,  
 380 and demonstrates highly competitive performance across other evaluation metrics. Specifically, our  
 381 method ranks second under RMSE among all published papers, where DMD3C (Liang et al., 2025)  
 382 obtains the lowest RMSE. It's worth noting that DMD3C utilizes exact BP-Net (Tang et al., 2024) but  
 383 incorporates additional supervision derived from a foundation model during training. In contrast, our  
 384 method is trained solely on the standard KITTI training set from scratch, similar to BP-Net, while  
 385 surpassing BP-Net in all evaluation metrics. Training with more data or distilling from a widely  
 386 trained foundation model to improve our method is an interesting direction left for future work. On  
 387 the NYUv2 dataset, our method achieves the best RMSE. As the commonly used  $\delta_{1.25}$  metric is  
 388 nearing saturation (typically  $\geq 99.6\%$ ), we provide results using the stricter  $\delta_{1.02}$  and  $\delta_{1.05}$  metrics  
 389 in Table 1 to better highlight the superiority of GBPN compared to other methods with publicly  
 390 available models on the NYUv2 dataset. More comparisons can be found in Section A.4.  
 391

392 **Table 1: Performance on KITTI and NYUv2 datasets.** For the KITTI dataset, results are evaluated  
 393 by the KITTI testing server. For the NYUv2 dataset, authors report their results in their papers. The  
 394 best result under each criterion is in **bold**. The second best is with underline.

		KITTI				NYUv2		
		RMSE $\downarrow$ (mm)	MAE $\downarrow$ (mm)	iRMSE $\downarrow$ (1/km)	iMAE $\downarrow$ (1/km)	RMSE $\downarrow$ (m)	REL $\downarrow$	$\delta_{1.02} \uparrow$ (%)
S2D (Ma & Karaman, 2018)	814.73	249.95	2.80	1.21	0.230	0.044	—	—
DeepLiDAR (Qiu et al., 2019)	758.38	226.50	2.56	1.15	0.115	0.022	—	—
GuideNet (Tang et al., 2020)	736.24	218.83	2.25	0.99	0.101	0.015	82.0	93.9
NLSPN (Park et al., 2020)	741.68	199.59	1.99	0.84	0.092	0.012	88.0	95.4
ACMNet (Zhao et al., 2021a)	744.91	206.09	2.08	0.90	0.105	0.015	—	—
DySPN (Lin et al., 2022)	709.12	192.71	1.88	<u>0.82</u>	0.090	0.012	—	—
BEV@DC (Zhou et al., 2023)	697.44	189.44	1.83	<u>0.82</u>	0.089	0.012	—	—
CFormer (Zhang et al., 2023)	708.87	203.45	2.01	0.88	0.090	0.012	87.5	95.3
LRRU (Wang et al., 2023a)	696.51	189.96	1.87	<b>0.81</b>	0.091	<u>0.011</u>	—	—
TPVD (Yan et al., 2024)	693.97	<u>188.60</u>	<u>1.82</u>	<b>0.81</b>	<u>0.086</u>	<b>0.010</b>	—	—
ImprovingDC (Wang et al., 2024)	686.46	<b>187.95</b>	1.83	<b>0.81</b>	0.091	<u>0.011</u>	—	—
OGNI-DC (Zuo & Deng, 2024)	708.38	193.20	1.86	0.83	0.087	<u>0.011</u>	88.3	95.6
BP-Net (Tang et al., 2024)	684.90	194.69	<u>1.82</u>	0.84	0.089	0.012	87.2	95.3
DMD3C (Liang et al., 2025)	<b>678.12</b>	194.46	<u>1.82</u>	0.85	<b>0.085</b>	<u>0.011</u>	—	—
GBPN	<u>682.20</u>	192.14	<b>1.78</b>	<u>0.82</u>	<b>0.085</b>	<u>0.011</u>	<b>89.1</b>	<b>95.9</b>

## 4.2 ABLATION STUDIES

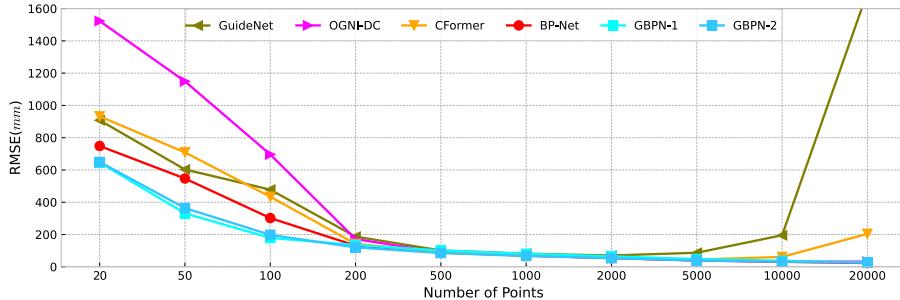
411 We conduct ablation studies on the NYUv2 dataset to evaluate the contribution of core components in  
 412 our GBPN. Starting from a simple optimization-based baseline  $V_1$ , we incrementally add components  
 413 to arrive at  $V_2$  to  $V_9$ . All models are trained with half amount epoches of the full training schedule,  
 414 due to resource constraints. Quantitative results, including RMSE and  $\delta_{1.02}$ , are presented in Table 2.

416 **Table 2: Ablation studies on NYUv2 dataset.**

	Basic Block	Local Edges	Dynamic MRF	GBP Iter.	Loss	Criteria									
						Conv.	Attn.	4	8	Param.	Edge	3	5	Probability	RMSE $\downarrow$ (mm)
$V_1$	✓													342.70	21.58
$V_2$	✓													340.84	25.23
$V_3$	✓	✓												108.29	83.71
$V_4$		✓	✓											109.54	83.27
$V_5$	✓	✓	✓											107.01	84.01
$V_6$	✓	✓	✓	✓										103.92	84.69
$V_7$	✓	✓	✓		✓			✓						101.42	85.19
$V_8$	✓	✓	✓		✓			✓						100.95	85.20
$V_9$	✓	✓	✓		✓			✓						100.69	85.27

426 The baseline model  $V_1$  employs a convolutional U-Net to estimate an initial depth and confidence map  
 427 from RGB image, and then optimizes the parameters of an affine transformation using constraints  
 428 from sparse depth measurements. Similar to Conti et al. (2023), this process involves solving a  
 429 weighted least squares problem based solely on depth observation constraints, without any pair-wise  
 430 terms. The model is trained using only an L2 loss on depth, following Tang et al. (2020; 2024).  
 431

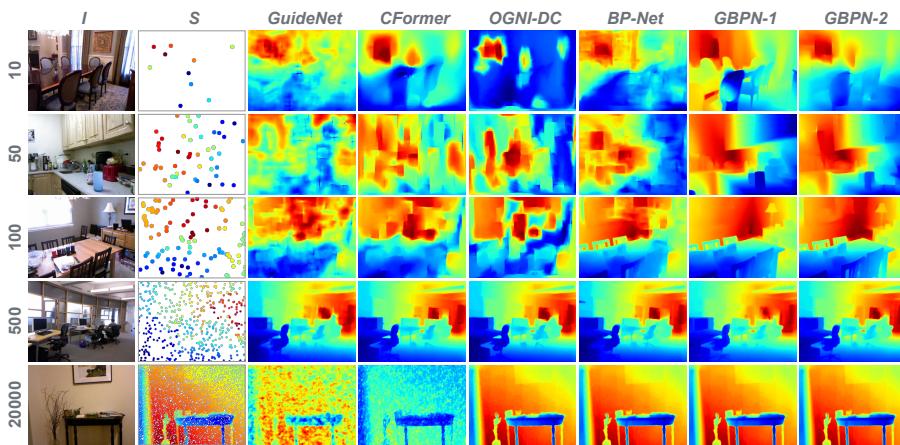
<sup>1</sup>[http://www.cvlabs.net/datasets/kitti/eval\\_depth.php?benchmark](http://www.cvlabs.net/datasets/kitti/eval_depth.php?benchmark)

Figure 2: RMSE( $mm$ ) on NYUv2 testset under various sparsity.

In our ablation study, we first replace the L2 loss in  $V_1$  with our proposed optimization-based loss function, resulting in  $V_2$  with slight performance improvement. Building on  $V_2$ , we develop  $V_3$ ,  $V_4$ , and  $V_5$ , which generate depth by solving a fixed MRF with fixed local edges, but using features from different backbones. All three variants outperform  $V_2$  by a large margin, with  $V_5$  achieving the best RMSE. This confirms the effectiveness of our problem formulation and the proposed global-local processing unit, which combines local convolutions with long-range attention. We then extend  $V_5$  by incorporating an MRF with dynamic parameters ( $V_6$ ) and dynamic non-local edges ( $V_7$ ). The performance gains of  $V_6$  and  $V_7$  demonstrate the advantage of a more expressive graph model. Subsequently, we increase the number of local edges in  $V_7$  to construct  $V_8$ , and then the number of iterations to form  $V_9$ . The performance improvements seen in  $V_8$  and  $V_9$  highlight the benefits of more dense constraints and a greater number of optimization iterations. We attempt further increases in iteration numbers but observing little performance improvement, and finally choose  $V_9$  as GBPN-1.

### 4.3 SPARSITY ROBUSTNESS ANALYSIS

In real-world applications, the sparsity level of the input depth map may vary significantly depending on the sensor and environment. To evaluate the robustness of our method on input sparsity, we conduct experiments on the NYUv2 dataset, comparing our approach against other SOTA methods with publicly available code and models. For this analysis, all methods for comparison are from released models trained with 500 depth points on NYUv2. These models are then directly evaluated on sparse inputs generated at various sparsity levels, ranging from 20 to 20,000 points.

Figure 3: **Qualitative comparisons under different sparsity.** All comparing methods are trained with 500 depth points and directly tested with depth input from various sparsity levels.

As demonstrated in Fig. 2, both GBPN-1 and GBPN-2 exhibit consistently better robustness across the entire range of tested sparsity levels than others. For GBPN, the RMSE consistently decreases

486 as the number of points increases. In contrast, the RMSE of methods like GuideNet (Tang et al.,  
 487 2020) and CFormer (Zhang et al., 2023) increases notably when the number of depth points becomes  
 488 significantly denser (beyond approximately 5000 points) than the training sparsity (500 points). More  
 489 about sparsity robustness analysis is in Section A.5.

490 We also visualize qualitative results from these compared methods under various sparsity levels in  
 491 Fig. 3. In very sparse scenarios, such as the 10 points case visualized in the first row, depth maps  
 492 from our method maintain relatively sharp boundaries, whereas the results from other methods appear  
 493 ambiguous or messy. For the 50 and 100 points cases shown in the second and third rows, our method  
 494 can produce relatively clear depth maps with structural details, while results from others tend towards  
 495 being unclear and oversmooth. In very dense scenarios, *i.e.* 20,000 points case in the last row, our  
 496 method consistently produces very clear depth maps, while results from some comparison methods  
 497 tend to be noisy. We attribute this superior robustness to our problem formulation, which incorporates  
 498 sparse input data as a data term within a globally optimized MRF framework, making it inherently  
 499 robust to the irregularity and varying density of the input sparse measurements.

#### 501 4.4 GENERALIZATION CAPABILITY

503 To demonstrate the generalization of GBPN, we evaluate state-of-the-art depth completion methods  
 504 on the VOID (Wong et al., 2020) benchmark, whose sparse depth is from visual odometry. All  
 505 methods are trained on NYUv2 with 500 random points and are evaluated zero-shot on the VOID  
 506 validation set under 150, 500, and 1500 point settings. Thus, this setup tests generalization across  
 507 different sparsity levels, sparsity patterns, and scene domains.

510 Table 3: Depth completion results on the VOID dataset at different sparsity levels (1500, 500, 150  
 511 points). RMSE and MAE are reported in meters.

512 Method	513 VOID 1500		514 VOID 500		515 VOID 150	
	516 RMSE (m)	517 MAE (m)	518 RMSE (m)	519 MAE (m)	520 RMSE (m)	521 MAE (m)
522 OGNI (Zuo & Deng, 2024)	523 0.92	524 0.39	525 1.10	526 0.61	527 1.25	528 0.75
529 NLSPN (Park et al., 2020)	530 0.69	531 <b>0.22</b>	532 0.76	533 <b>0.30</b>	534 0.93	535 0.43
536 CFormer (Zhang et al., 2023)	537 0.73	538 0.26	539 0.82	540 0.38	541 0.96	542 0.48
543 BP-Net (Tang et al., 2024)	544 0.74	545 0.27	546 0.80	547 0.37	548 0.94	549 0.47
550 GuideNet (Tang et al., 2020)	551 2.14	552 1.05	553 2.06	554 0.85	555 2.05	556 0.70
557 GBPN	558 <b>0.68</b>	559 <b>0.22</b>	560 <b>0.74</b>	561 <b>0.30</b>	562 <b>0.90</b>	563 <b>0.41</b>

564 As reported in Table 3, GBPN consistently surpasses other methods across all sparsity levels in both  
 565 RMSE and MAE. We believe this robustness stems from our model’s formulation, which integrates  
 566 sparse depth as principled observation terms within an MRF. By avoiding specialized neural layers  
 567 for sparse input, our method achieves stronger generalization. In contrast, GuideNet (Tang et al.,  
 568 2020) performs poorly on this dataset, perhaps due to its direct application of standard convolutional  
 569 layers on sparse inputs.

## 570 5 CONCLUSION

571 This paper presents the Gaussian Belief Propagation Network (GBPN), a novel framework that  
 572 seamlessly integrates deep learning with probabilistic graphical models for depth completion. GBPN  
 573 employs a Graphical Model Construction Network (GMCN) to dynamically build a scene-specific  
 574 Markov Random Field (MRF). This learned MRF formulation naturally incorporates sparse data and  
 575 models complex spatial dependencies through adaptive non-local edges and dynamic parameters.  
 576 Dense depth inference is efficiently performed using Gaussian Belief Propagation, enhanced by  
 577 a serial & parallel message passing scheme. Extensive experiments on the NYUv2, KITTI, and  
 578 VOID benchmarks demonstrate that GBPN achieves state-of-the-art accuracy and exhibits superior  
 579 robustness and generalizable capability. Training on larger, more diverse datasets or leveraging  
 580 supervision from pre-trained foundation models and optimize the GBP implementation are promising  
 581 avenues for future works to enhance performance and efficiency for applications in broader scenarios.

540 REFERENCES  
541

542 Danny Bickson. Gaussian belief propagation: Theory and application. *arXiv preprint arXiv:0811.2518*,  
543 2008.

544 Brendan Burns and Oliver Brock. Sampling-based motion planning with sensing uncertainty. In  
545 *ICRA*, pp. 3313–3318. IEEE, 2007.

546 Qifeng Chen and Vladlen Koltun. Fast mrf optimization with application to depth reconstruction. In  
547 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3914–3921,  
548 2014.

549 Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for  
550 depth completion. In *ICCV*, pp. 10023–10032, 2019.

551 Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation  
552 network. *IEEE TPAMI*, 42(10):2361–2379, 2019.

553 Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and re-  
554 source aware convolutional spatial propagation networks for depth completion. In *AAAI*, volume 34,  
555 pp. 10615–10622, 2020.

556 Andrea Conti, Matteo Poggi, and Stefano Mattoccia. Sparsity agnostic depth completion. In *WACV*,  
557 pp. 5871–5880, 2023.

558 Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable  
559 convolutional networks. In *ICCV*, pp. 764–773, 2017.

560 Andrew J Davison and Joseph Ortiz. Futuremapping 2: Gaussian belief propagation for spatial ai.  
561 *arXiv preprint arXiv:1910.14139*, 2019.

562 James Diebel and Sebastian Thrun. An application of markov random fields to range sensing.  
563 *NeurIPS*, 18, 2005.

564 Abdelrahman Elde索key, Michael Felsberg, and Fahad Shahbaz Khan. Confidence propagation  
565 through cnns for guided sparse depth regression. *IEEE TPAMI*, 42(10):2423–2436, 2019.

566 William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *IJCV*, 40:  
567 25–47, 2000.

568 Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti  
569 vision benchmark suite. In *CVPR*, 2012.

570 Ali Hassani and Humphrey Shi. Dilated neighborhood attention transformer. *arXiv preprint*  
571 *arXiv:2209.15001*, 2022.

572 Simon Hawe, Martin Kleinsteuber, and Klaus Diepold. Dense disparity maps from sparse disparity  
573 measurements. In *ICCV*, pp. 2126–2133. IEEE, 2011.

574 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image  
575 recognition. In *CVPR*, pp. 770–778, 2016.

576 Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and  
577 efficient image guided depth completion. In *ICRA*, pp. 13656–13662. IEEE, 2021.

578 Zixuan Huang, Junming Fan, Shenggan Cheng, Shuai Yi, Xiaogang Wang, and Hongsheng Li.  
579 Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *IEEE*  
580 *TIP*, 29:3429–3441, 2019.

581 Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer  
582 vision? *NeurIPS*, 30, 2017.

583 Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling.  
584 *ACM TOG*, 26(3):96–es, 2007.

594 Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth  
 595 completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 16–22.  
 596 IEEE, 2018.

597 Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks  
 598 without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

600 Tianan Li, Zhehan Chen, Huan Liu, and Chen Wang. Fdct: Fast depth completion for transparent  
 601 objects. *IEEE Robotics and Automation Letters*, 8(9):5823–5830, 2023.

602 Yingping Liang, Yutao Hu, Wenqi Shao, and Ying Fu. Distilling monocular foundation model for  
 603 fine-grained depth completion. *CVPR*, 2025.

605 Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation  
 606 network for depth completion. In *AAAI*, volume 36, pp. 1638–1646, 2022.

607 Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation  
 608 from a single image. In *Proceedings of the IEEE conference on computer vision and pattern  
 609 recognition*, pp. 5162–5170, 2015.

611 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018.

612 Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples  
 613 and a single image. In *ICRA*, pp. 4796–4803. IEEE, 2018.

615 Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference:  
 616 An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

617 Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation  
 618 network for depth completion. In *ECCV*, pp. 120–136. Springer, 2020.

620 Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan  
 621 Kaufmann, 1988.

622 Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc  
 623 Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse  
 624 lidar data and single color image. In *CVPR*, pp. 3313–3322, 2019.

626 Chao Qu, Ty Nguyen, and Camillo Taylor. Depth completion via deep basis fitting. In *WACV*, pp.  
 627 71–80, 2020.

628 Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner.  
 629 Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, pp. 12892–12901,  
 630 2022.

632 Daniel Ron, Kun Duan, Chongyang Ma, Ning Xu, Shenlong Wang, Suman Hanumante, and  
 633 Dhritiman Sagar. Monocular depth estimation via deep structured models with ordinal constraints.  
 634 In *2018 International Conference on 3D Vision (3DV)*, pp. 570–577. IEEE, 2018.

635 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical  
 636 image segmentation. In *International Conference on Medical image computing and computer-  
 637 assisted intervention*, pp. 234–241. Springer, 2015.

638 Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pp.  
 639 4104–4113, 2016.

641 Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support  
 642 inference from rgbd images. In *ECCV*, pp. 746–760, 2012.

643 Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using  
 644 large learning rates. In *Artificial intelligence and machine learning for multi-domain operations  
 645 applications*, volume 11006, pp. 369–386. SPIE, 2019.

647 Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. 25  
 (7):787–800, 2003.

648 Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. Learning guided convolutional network for  
 649 depth completion. *IEEE TIP*, 30:1116–1129, 2020.  
 650

651 Jie Tang, Fei-Peng Tian, Boshi An, Jian Li, and Ping Tan. Bilateral propagation network for depth  
 652 completion. In *CVPR*, pp. 9763–9772, 2024.

653 Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger.  
 654 Sparsity invariant cnns. In *IEEE International Conference on 3D Vision (3DV)*, pp. 11–20, 2017.

655

656 Yufei Wang, Bo Li, Ge Zhang, Qi Liu, Tao Gao, and Yuchao Dai. Lrru: Long-short range recurrent  
 657 updating networks for depth completion. In *ICCV*, pp. 9422–9432, 2023a.

658

659 Yufei Wang, Yuxin Mao, Qi Liu, and Yuchao Dai. Decomposed guided dynamic filters for efficient  
 660 rgb-guided depth completion. *IEEE TCSVT*, 34(2):1186–1198, 2023b.

661

662 Yufei Wang, Ge Zhang, Shaoqian Wang, Bo Li, Qi Liu, Le Hui, and Yuchao Dai. Improving depth  
 663 completion via depth feature upsampling. In *CVPR*, pp. 21104–21113, 2024.

664

665 Yair Weiss and William Freeman. Correctness of belief propagation in gaussian graphical models of  
 666 arbitrary topology. *NeurIPS*, 12, 1999.

667

668 Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from  
 669 visual inertial odometry. *IEEE Robotics and Automation Letters*, 5(2):1899–1906, 2020.

670

671 Xiaopei Wu, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu,  
 672 and Deng Cai. Sparse fuse dense: Towards high quality 3d detection with depth completion. In  
 673 *CVPR*, pp. 5418–5427, 2022.

674

675 Zhiqiang Yan, Kun Wang, Xiang Li, Zhenyu Zhang, Jun Li, and Jian Yang. Rignet: Repetitive image  
 676 guided network for depth completion. In *ECCV*, pp. 214–230. Springer, 2022.

677

678 Zhiqiang Yan, Yuankai Lin, Kun Wang, Yupeng Zheng, Yufei Wang, Zhenyu Zhang, Jun Li, and Jian  
 679 Yang. Tri-perspective view decomposition for geometry-aware depth completion. In *CVPR*, pp.  
 680 4874–4884, 2024.

681

682 Youmin Zhang, Xianda Guo, Matteo Poggi, Zheng Zhu, Guan Huang, and Stefano Mattoccia.  
 683 Completionformer: Depth completion with convolutions and vision transformers. In *CVPR*, pp.  
 684 18527–18536, 2023.

685

686 Shanshan Zhao, Mingming Gong, Huan Fu, and Dacheng Tao. Adaptive context-aware multi-modal  
 687 network for depth completion. *IEEE TIP*, 30:5264–5276, 2021a.

688

689 Yiming Zhao, Lin Bai, Ziming Zhang, and Xinming Huang. A surface geometry model for lidar  
 690 depth completion. *IEEE Robotics and Automation Letters*, 6(3):4457–4464, 2021b.

691

692 Wending Zhou, Xu Yan, Yinghong Liao, Yuankai Lin, Jin Huang, Gangming Zhao, Shuguang Cui,  
 693 and Zhen Li. Bev@ dc: Bird’s-eye view assisted training for depth completion. In *CVPR*, pp.  
 694 9233–9242, 2023.

695

696 Yiming Zuo and Jia Deng. Ogni-dc: Robust depth completion with optimization-guided neural  
 697 iterations. In *ECCV*, pp. 78–95. Springer, 2024.

698

699

700

701

## A APPENDIX

### A.1 ADDITIONAL METHOD DETAILS

#### A.1.1 GAUSSIAN BELIEF PROPAGATION

This section provides a detailed derivation of the Gaussian Belief Propagation (GBP) equations, which are summarized in the main paper due to page limitations. Following the formulation in the

main paper, we model the joint probability distribution over the depth variables  $X$  using a Markov Random Field (MRF):

$$p(X|I, S) \propto \prod_{i \in \mathcal{V}_v} \phi_i \prod_{(i,j) \in \mathcal{E}} \psi_{ij}, \quad (11)$$

where  $\phi_i$  is the unary potential for node  $i$ , and  $\psi_{ij}$  is the pairwise potential between node  $i$  and  $j$ . Specifically, we define the unary potential as:

$$\phi_i = \exp\left(-\frac{w_i(x_i - s_i)^2}{2}\right), \quad (12)$$

and the pairwise potential as:

$$\psi_{ij} = \exp\left(-\frac{w_{ij}(x_i - x_j - r_{ij})^2}{2}\right). \quad (13)$$

In Belief Propagation, the belief at each node  $i$  is calculated as the product of the local unary potential and all incoming messages from neighboring nodes

$$b_i \propto \phi_i \prod_{(i,j) \in \mathcal{E}} m_{j \rightarrow i}. \quad (14)$$

And the messages are updated iteratively according to

$$m_{j \rightarrow i} \propto \int_{x_j} \psi_{ij} \phi_j \prod_{(k,j) \in \mathcal{E} \setminus (i,j)} m_{k \rightarrow j} dx_j. \quad (15)$$

All the above eqs. (11) to (15) are same as eqs. (1) to (5) in the main paper. We repeat them here for reading convenience.

In GBP, all beliefs and messages are assumed to be Gaussian distributions. Then, a message proportional to a Gaussian, can be written as

$$m_{j \rightarrow i} \propto \exp\left(-\frac{\hat{\Lambda}_{j \rightarrow i}}{2} x_i^2 + \hat{\eta}_{j \rightarrow i} x_i\right). \quad (16)$$

Substituting the unary potential eq. (12) and the incoming messages eq. (16) into the belief equation eq. (14), we derive the belief

$$\begin{aligned} b_i &\propto \phi_i \prod_{(i,j) \in \mathcal{E}} m_{j \rightarrow i} \\ &\propto \exp\left(-\frac{w_i(x_i - s_i)^2}{2}\right) \prod_{(i,j) \in \mathcal{E}} \exp\left(-\frac{\hat{\Lambda}_{j \rightarrow i}}{2} x_i^2 + \hat{\eta}_{j \rightarrow i} x_i\right) \\ &\propto \exp\left(-\frac{1}{2}(w_i + \sum_{(i,j) \in \mathcal{E}} \hat{\Lambda}_{j \rightarrow i}) x_i^2 + (w_i s_i + \sum_{(i,j) \in \mathcal{E}} \hat{\eta}_{j \rightarrow i}) x_i\right). \end{aligned} \quad (17)$$

Thus, the resulting belief  $b_i$  is also Gaussian, expressed as  $\mathcal{N}^{-1}(\eta_i, \Lambda_i)$ , with

$$\begin{aligned} \eta_i &= w_i s_i + \sum_{(i,j) \in \mathcal{E}} \hat{\eta}_{j \rightarrow i}, \\ \Lambda_i &= w_i + \sum_{(i,j) \in \mathcal{E}} \hat{\Lambda}_{j \rightarrow i}. \end{aligned} \quad (18)$$

The belief parameters in eq. (18) are identical to the eq. (6) of the main paper.

We can also define  $b_{j \setminus i}$  as the belief on node  $j$  taking all messages, except the message from  $i$ . Thus, similar to eq. (17), we have

$$b_{j \setminus i} \propto \phi_j \prod_{(k,j) \in \mathcal{E} \setminus (i,j)} m_{k \rightarrow j}. \quad (19)$$

756 And similar to eq. (18),  $b_{j \setminus i}$  is in Gaussian  $\mathcal{N}^{-1}(\eta_{j \setminus i}, \Lambda_{j \setminus i})$ , with  
 757

$$\begin{aligned} 758 \quad \eta_{j \setminus i} &= w_j s_j + \sum_{(k,j) \in \mathcal{E} \setminus (i,j)} \hat{\eta}_{k \rightarrow j}, \\ 759 \quad \Lambda_{j \setminus i} &= w_j + \sum_{(k,j) \in \mathcal{E} \setminus (i,j)} \hat{\Lambda}_{k \rightarrow j}. \end{aligned} \quad (20)$$

760  
 761

762 Now, substituting the pairwise potential eq. (13) and the expression for  $b_{j \setminus i}$  (from combining eq. (19)  
 763 and eq. (20)) into the message update equation eq. (15), we get:  
 764

$$\begin{aligned} 765 \quad m_{j \rightarrow i} &\propto \int_{x_j} \psi_{ij} \phi_j \prod_{(k,j) \in \mathcal{E} \setminus (i,j)} m_{k \rightarrow j} dx_j \\ 766 \\ 767 \quad &\propto \int_{x_j} \psi_{ij} b_{j \setminus i} dx_j \\ 768 \\ 769 \quad &\propto \int_{x_j} \exp\left(-\frac{w_{ij}(x_i - x_j - r_{ij})^2}{2}\right) \exp\left(-\frac{\Lambda_{j \setminus i}}{2} x_j^2 + \eta_{j \setminus i} x_j\right) dx_j \\ 770 \\ 771 \quad &\propto \exp\left(-\frac{w_{ij}}{2} x_i^2 + w_{ij} r_{ij} x_i\right) \int_{x_j} \exp\left(-\frac{w_{ij}}{2} x_j^2 + (x_i - r_{ij}) w_{ij} x_j\right) \exp\left(-\frac{\Lambda_{j \setminus i}}{2} x_j^2 + \eta_{j \setminus i} x_j\right) dx_j \\ 772 \\ 773 \quad &\propto \exp\left(-\frac{w_{ij}}{2} x_i^2 + w_{ij} r_{ij} x_i\right) \int_{x_j} \exp\left(-\frac{w_{ij} + \Lambda_{j \setminus i}}{2} x_j^2 + (x_i w_{ij} - r_{ij} w_{ij} + \eta_{j \setminus i}) x_j\right) dx_j \\ 774 \\ 775 \quad &\propto \exp\left(-\frac{w_{ij}}{2} x_i^2 + w_{ij} r_{ij} x_i\right) \exp\left(\frac{(x_i w_{ij} - r_{ij} w_{ij} + \eta_{j \setminus i})^2}{2(w_{ij} + \Lambda_{j \setminus i})}\right) \\ 776 \\ 777 \quad &\propto \exp\left(-\frac{w_{ij}}{2} x_i^2 + w_{ij} r_{ij} x_i\right) \exp\left(\frac{(x_i w_{ij} - r_{ij} w_{ij} + \Lambda_{j \setminus i} \mu_{j \setminus i})^2}{2(w_{ij} + \Lambda_{j \setminus i})}\right) \\ 778 \\ 779 \quad &\propto \exp\left(-\frac{w_{ij} \Lambda_{j \setminus i}}{2(w_{ij} + \Lambda_{j \setminus i})} x_i^2 + \frac{w_{ij} \Lambda_{j \setminus i}}{w_{ij} + \Lambda_{j \setminus i}} (\mu_{j \setminus i} + r_{ij}) x_i\right) \\ 780 \\ 781 \quad &\propto \exp\left(-\frac{(x_i - (\mu_{j \setminus i} + r_{ij}))^2}{2(w_{ij}^{-1} + \Lambda_{j \setminus i}^{-1})}\right) \\ 782 \\ 783 \quad &\propto \exp\left(-\frac{w_{ij} \Lambda_{j \setminus i}}{2(w_{ij} + \Lambda_{j \setminus i})} x_i^2 + \frac{w_{ij} \Lambda_{j \setminus i}}{w_{ij} + \Lambda_{j \setminus i}} (\mu_{j \setminus i} + r_{ij}) x_i\right) \\ 784 \\ 785 \quad &\propto \exp\left(-\frac{(x_i - (\mu_{j \setminus i} + r_{ij}))^2}{2(w_{ij}^{-1} + \Lambda_{j \setminus i}^{-1})}\right) \\ 786 \\ 787 \end{aligned} \quad (21)$$

788 Finally, the message  $m_{j \rightarrow i}$  is in Gaussian  $\mathcal{N}(\mu_{j \rightarrow i}, \Lambda_{j \rightarrow i}^{-1})$ , with  
 789

$$\begin{aligned} 790 \quad \mu_{j \rightarrow i} &= \mu_{j \setminus i} + r_{ij}, \\ 791 \\ 792 \quad \Lambda_{j \rightarrow i}^{-1} &= \Lambda_{j \setminus i}^{-1} + w_{ij}^{-1}. \end{aligned} \quad (22)$$

793

794 The eq. (22) here is identical to the eq. (7) of the main paper.  
 795

### 796 A.1.2 GRAPHICAL MODEL CONSTRUCTION NETWORK

797

798 The Graphical Model Construction Network (GMCN) is designed to learn the parameters and  
 799 structures of the MRF from input data. We employ a U-Net architecture (Ronneberger et al., 2015),  
 800 comprising encoder and decoder layers, as the GMCN. The detailed network architecture of GMCN  
 801 with  $256 \times 320$  input images is provided in Table 4. Note that for clarity, only the main operators  
 802 are listed here, with trivial operations like normalization, activation, and skip connections omitted.  
 803 The encoder of the U-Net extracts features at six different scales ( $\mathbf{E}^0, \dots, \mathbf{E}^5$ ). From scale 1 to 5,  
 804 feature extraction is performed using global-local units, which combine a dilated neighbor attention  
 805 layer (Hassani & Shi, 2022) to capture long-range dependencies with ResBlocks (He et al., 2016) for  
 806 robust local feature learning. The decoder then aggregates these multi-resolution features back to the  
 807 original input resolution using deconvolution layers and concatenation operations. This multi-scale  
 808 approach ensures that the network captures both fine-grained local details and broader contextual  
 809 information necessary for robust MRF construction. We explore two variations of the GMCN. The  
 810 first takes only color images as input, constructing the graphical model solely based on visual cues  
 811 using a single U-Net. The second incorporates intermediate optimized dense depth distributions as

810

811

Table 4: Detailed Architecture of Graphical Model Construction Network (GMCN).

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

additional input to refine the constructed graphical model. This is achieved by introducing a second multi-modal fusion U-Net, which utilizes a cross-attention mechanism to effectively integrate features from the color-based U-Net and the depth information.

	Output	Input	Operator	Output Size
GMCN	$\mathbf{E}_1^0$	$I$	Conv. + ResBlock	( 32, 256, 320)
	$\mathbf{E}_1^1$	$\mathbf{E}_1^0$	Conv. + (Self-Attn. + ResBlock) $\times 2$	( 64, 128, 160)
	$\mathbf{E}_1^2$	$\mathbf{E}_1^1$	Conv. + (Self-Attn. + ResBlock) $\times 2$	(128, 64, 80)
	$\mathbf{E}_1^3$	$\mathbf{E}_1^2$	Conv. + (Self-Attn. + ResBlock) $\times 2$	(256, 32, 40)
	$\mathbf{E}_1^4$	$\mathbf{E}_1^3$	Conv. + (Self-Attn. + ResBlock) $\times 2$	(256, 16, 20)
	$\mathbf{E}_1^5$	$\mathbf{E}_1^4$	Conv. + (Self-Attn. + ResBlock) $\times 2$	(256, 8, 10)
	$\mathbf{D}_1^4$	$\mathbf{E}_1^5, \mathbf{E}_1^4$	Deconv. + Concat. + Conv.	(256, 16, 20)
	$\mathbf{D}_1^3$	$\mathbf{D}_1^4, \mathbf{E}_1^3$	Deconv. + Concat. + Conv.	(256, 32, 40)
	$\mathbf{D}_1^2$	$\mathbf{D}_1^3, \mathbf{E}_1^2$	Deconv. + Concat. + Conv.	(256, 64, 80)
	$\mathbf{D}_1^1$	$\mathbf{D}_1^2, \mathbf{E}_1^1$	Deconv. + Concat. + Conv.	( 64, 128, 160)
GMCN	$\mathbf{D}_1^0$	$\mathbf{D}_1^1, \mathbf{E}_1^0$	Deconv. + Concat. + Conv.	( 32, 256, 320)
	$\beta_1, r_1, w_1, o_1$	$\mathbf{D}_1^0$	Conv. + Conv.	( *, 256, 320)
	GBP	$\mu_1, \Lambda_1$	$S, \beta_1, r_1, w_1, o_1$	Serial & Parallel Propagation
	$\mathbf{E}_2^0$	$I, \mu_1, \Lambda_1$	Conv. + ResBlock	( 32, 256, 320)
	$\mathbf{E}_2^1$	$\mathbf{E}_2^0, \mathbf{E}_1^1$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	( 64, 128, 160)
	$\mathbf{E}_2^2$	$\mathbf{E}_2^1, \mathbf{E}_1^2$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(128, 64, 80)
	$\mathbf{E}_2^3$	$\mathbf{E}_2^2, \mathbf{E}_1^3$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 32, 40)
	$\mathbf{E}_2^4$	$\mathbf{E}_2^3, \mathbf{E}_1^4$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 16, 20)
	$\mathbf{E}_2^5$	$\mathbf{E}_2^4, \mathbf{E}_1^5$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 8, 10)
	$\mathbf{D}_2^4$	$\mathbf{E}_2^5, \mathbf{E}_2^4$	Deconv. + Concat. + Conv.	(256, 16, 20)
GMCN	$\mathbf{D}_2^3$	$\mathbf{D}_2^4, \mathbf{E}_2^3$	Deconv. + Concat. + Conv.	(256, 32, 40)
	$\mathbf{D}_2^2$	$\mathbf{D}_2^3, \mathbf{E}_2^2$	Deconv. + Concat. + Conv.	(256, 64, 80)
	$\mathbf{D}_2^1$	$\mathbf{D}_2^2, \mathbf{E}_2^1$	Deconv. + Concat. + Conv.	( 64, 128, 160)
	$\mathbf{D}_2^0$	$\mathbf{D}_2^1, \mathbf{E}_2^0$	Deconv. + Concat. + Conv.	( 32, 256, 320)
	$\beta_2, r_2, w_2, o_2$	$\mathbf{D}_2^0$	Conv. + Conv.	( *, 256, 320)
	GBP	$\mu_2, \Lambda_2$	$S, \beta_2, r_2, w_2, o_2$	Serial & Parallel Propagation
	$\mathbf{E}_2^0$	$I, \mu_2, \Lambda_2$	Conv. + ResBlock	( 32, 256, 320)
	$\mathbf{E}_2^1$	$\mathbf{E}_2^0, \mathbf{E}_1^1$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	( 64, 128, 160)
	$\mathbf{E}_2^2$	$\mathbf{E}_2^1, \mathbf{E}_1^2$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(128, 64, 80)
	$\mathbf{E}_2^3$	$\mathbf{E}_2^2, \mathbf{E}_1^3$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 32, 40)
	$\mathbf{E}_2^4$	$\mathbf{E}_2^3, \mathbf{E}_1^4$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 16, 20)
	$\mathbf{E}_2^5$	$\mathbf{E}_2^4, \mathbf{E}_1^5$	Conv. + (Self/Cross-Attn. + ResBlock) $\times 4$	(256, 8, 10)
	$\mathbf{D}_2^4$	$\mathbf{E}_2^5, \mathbf{E}_2^4$	Deconv. + Concat. + Conv.	(256, 16, 20)
	$\mathbf{D}_2^3$	$\mathbf{D}_2^4, \mathbf{E}_2^3$	Deconv. + Concat. + Conv.	(256, 32, 40)
	$\mathbf{D}_2^2$	$\mathbf{D}_2^3, \mathbf{E}_2^2$	Deconv. + Concat. + Conv.	(256, 64, 80)
	$\mathbf{D}_2^1$	$\mathbf{D}_2^2, \mathbf{E}_2^1$	Deconv. + Concat. + Conv.	( 64, 128, 160)
	$\mathbf{D}_2^0$	$\mathbf{D}_2^1, \mathbf{E}_2^0$	Deconv. + Concat. + Conv.	( 32, 256, 320)
	$\beta_2, r_2, w_2, o_2$	$\mathbf{D}_2^0$	Conv. + Conv.	( *, 256, 320)
	GBP	$\mu_2, \Lambda_2$	$S, \beta_2, r_2, w_2, o_2$	Serial & Parallel Propagation

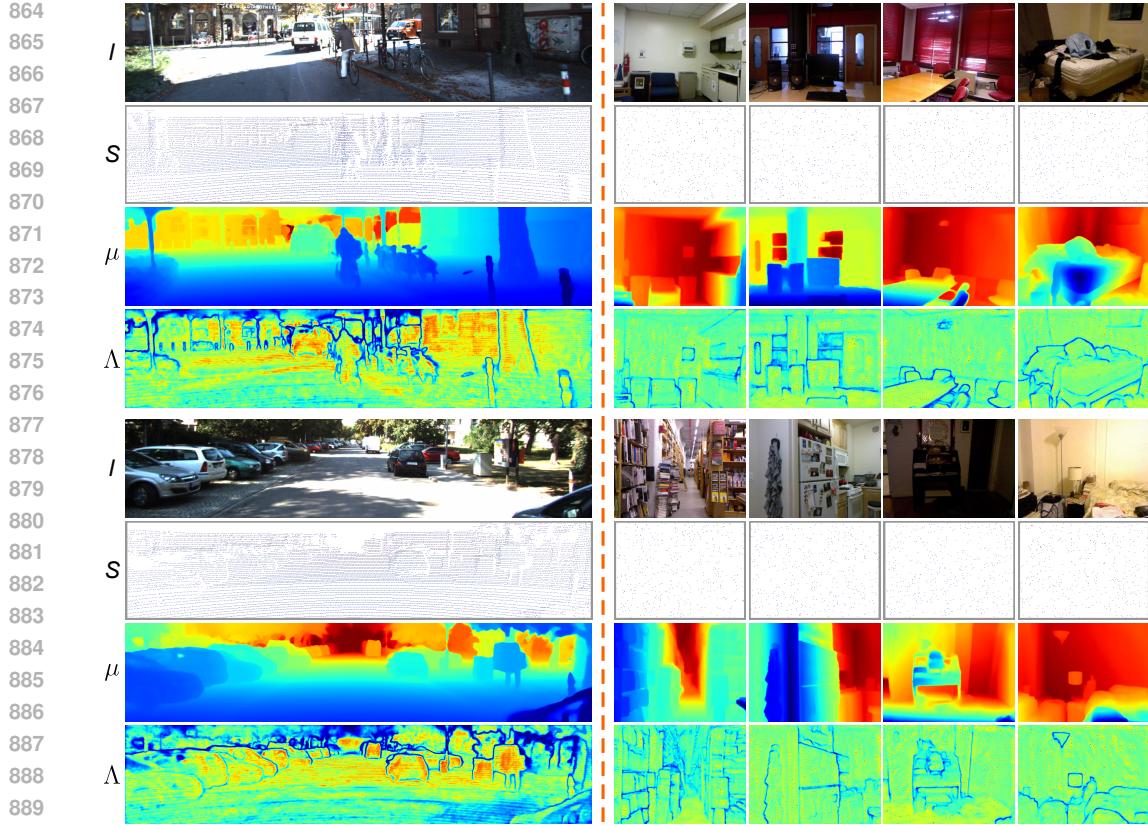


Figure 4: Visualization of estimated depth in Gaussian with  $\mu$  and  $\Lambda$ . The results on KITTI are on the left, and results on NYUv2 are on the right.

## A.2 EXPERIMENTS SETUP

### A.2.1 DATASETS

We adopt the standard depth completion datasets, *i.e.* NYUv2 for indoor scenes and KITTI for outdoor scenes, to train and evaluate our method. We also evaluate our method on the VOID (Wong et al., 2020) dataset to validate the robustness and generalization.

The NYUv2 dataset (Silberman et al., 2012) comprises 464 scenes captured by a Kinect sensor. For training, we take the data proposed by Ma & Karaman (2018), utilizing 50,000 frames sampled from 249 scenes. Evaluation is performed on the official test set, which contains 654 samples from 215 distinct scenes. We follow common practice (Tang et al., 2020; Park et al., 2020; Zhang et al., 2023; Tang et al., 2024) to process data. Input images are initially down-sampled to  $240 \times 320$  and then center-cropped to a resolution of  $228 \times 304$ . Sparse depth maps are generated for each frame by randomly sampling 500 points from the ground truth depth map.

The KITTI depth completion (DC) dataset (Uhrig et al., 2017) was collected using an autonomous driving platform. Ground truth depth is derived from temporally aggregated LiDAR scans and further refined using stereo image pairs. The dataset provides 86,898 frames for training and 1000 frames for validation. An additional 1000 test frames are evaluated on a remote server with a public leaderboard<sup>2</sup>. During training, we randomly crop frames to  $256 \times 1216$ . For testing, full-resolution frames are used.

The VOID dataset (Wong et al., 2020) was collected using an Intel RealSense D435i camera, with sparse measurements by a visual odometry system at 3 different sparsity levels, *i.e.*, 1500, 500, and 150 points, corresponding to 0.5%, 0.15%, and 0.05% density. Each test split contains 800 images at  $480 \times 640$  resolution. Though the dataset is collected for indoor scenes, the scenes, depth

<sup>2</sup>[http://www.cvlabs.net/datasets/kitti/eval\\_depth.php?benchmark](http://www.cvlabs.net/datasets/kitti/eval_depth.php?benchmark)

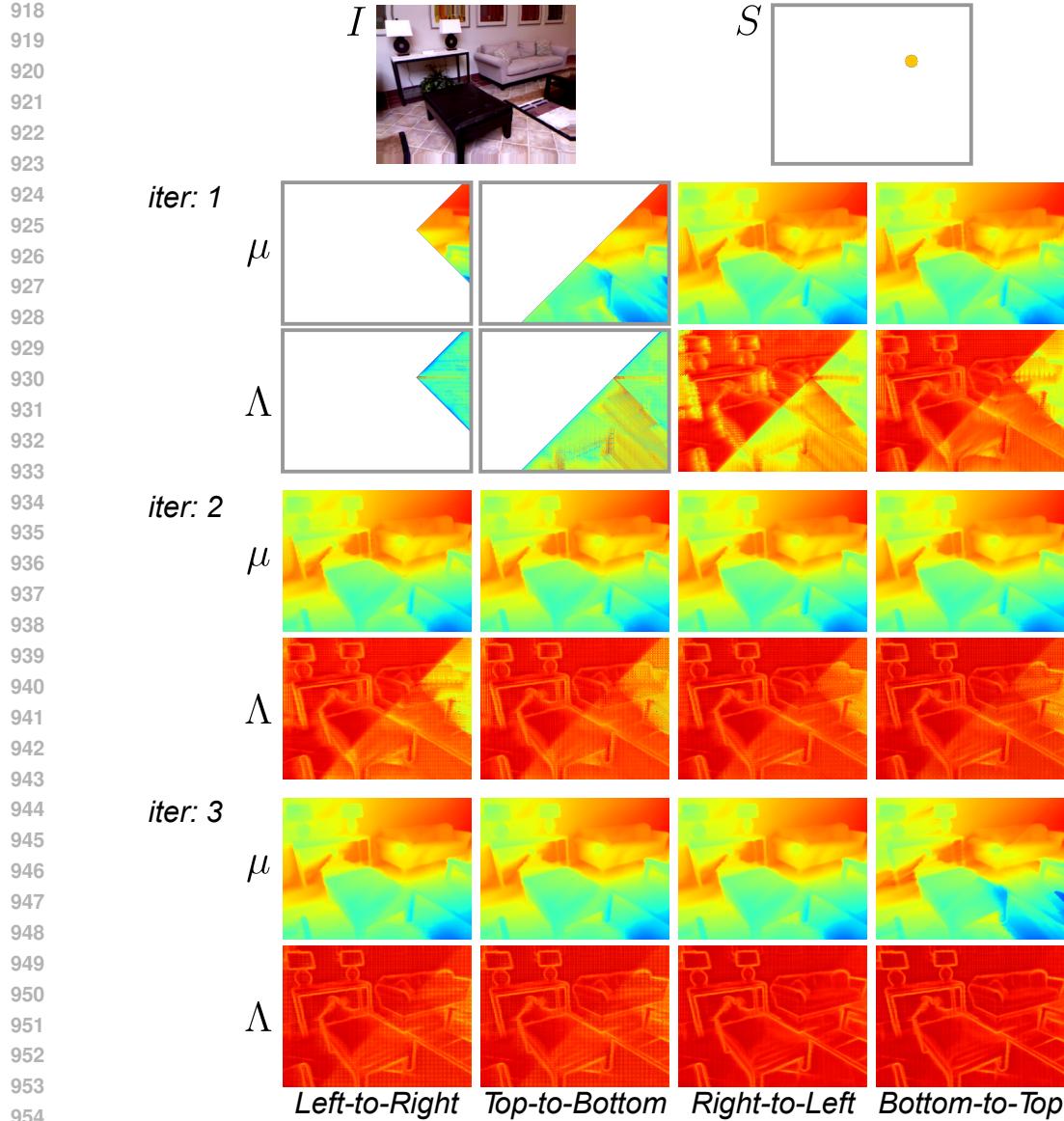


Figure 5: Visualization of intermediate results under the serial propagation of Gaussian Belief Propagation. Messages are propagated iteratively corresponding to local directional sweeps: left-to-right, top-to-bottom, right-to-left, and bottom-to-top.

measurement manner and depth sparsity in this dataset are distinct from NYUv2. As the collected ground truth data is noisy, this dataset is not used for training, but only for evaluation to validate the zero-shot generalization of DC methods to different sparsity patterns and scenes.

## A.2.2 TRAINING DETAILS

Our method is implemented in PyTorch and trained on a workstation with 4 NVIDIA RTX 4090 GPUs. We incorporate DropPath (Larsson et al., 2016) before residual connections as a regularization technique. We use the AdamW optimizer (Loshchilov & Hutter, 2018) with a weight decay of 0.05 and apply gradient clipping with an L2-norm threshold of 0.1. Models are trained from scratch for approximately 300,000 iterations. We employ the OneCycle learning rate policy (Smith & Topin, 2019), where the learning rate is annealed to 25% of its peak value during the cycle. For KITTI, the batch size is 8 and the peak learning rate is 0.001. For NYUv2 dataset, we use a batch size of 16

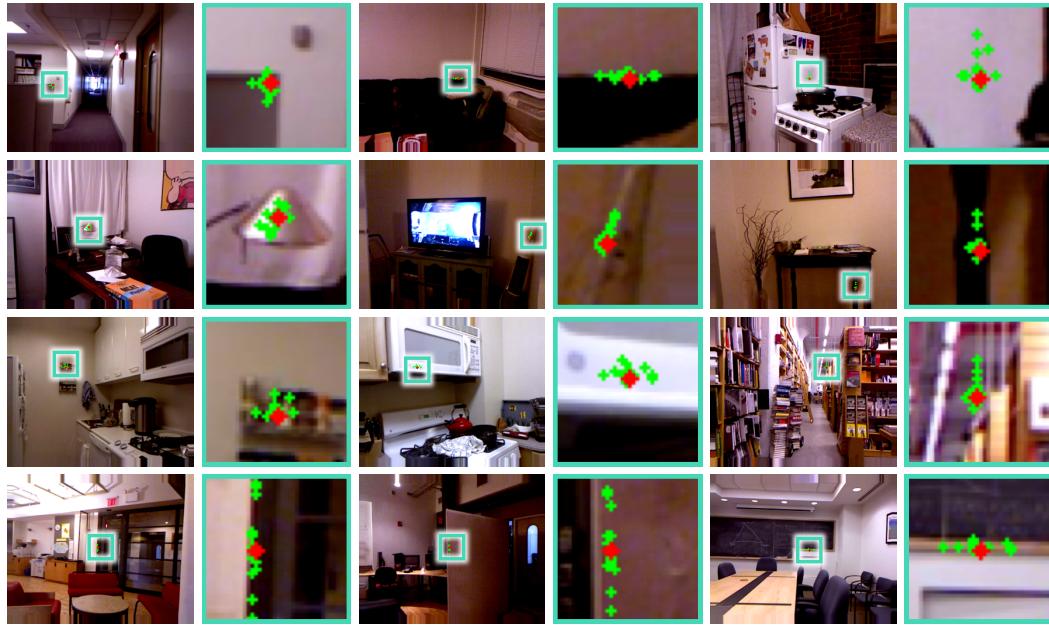


Figure 6: Visualization of dynamically constructed non-local edges in MRF. The red point is the target node, and the green points are the dynamically constructed neighbor nodes.

and a peak learning rate of 0.002. The final model weights are obtained using Exponential Moving Average (EMA).

### A.2.3 DETAILS ON EVALUATION METRICS

We verify our method on both indoor and outdoor scenes with standard evaluation metrics. For indoor scenes, root mean squared error (RMSE), mean absolute relative error (REL), and  $\delta_\theta$  are chosen as evaluation metrics. For outdoor scenes, the standard evaluation metrics are root mean squared error (RMSE), mean absolute error (MAE), root mean squared error of the inverse depth (iRMSE), and mean absolute error of the inverse depth (iMAE). These evaluation metrics are firstly calculated on each sample and then averaged among samples. And for each sample, they can be written as:

$$\begin{aligned}
 RMSE &= \left( \frac{1}{n} \sum_{i \in \mathcal{V}_g} (x_i^g - x_i)^2 \right)^{\frac{1}{2}}, \\
 iRMSE &= \left( \frac{1}{n} \sum_{i \in \mathcal{V}_g} \left( \frac{1}{x_i^g} - \frac{1}{x_i} \right)^2 \right)^{\frac{1}{2}}, \\
 MAE &= \frac{1}{n} \sum_{i \in \mathcal{V}_g} |x_i^g - x_i|, \\
 iMAE &= \frac{1}{n} \sum_{i \in \mathcal{V}_g} \left| \frac{1}{x_i^{gt}} - \frac{1}{x_i} \right|, \\
 REL &= \frac{1}{n} \sum_{i \in \mathcal{V}_g} \frac{|x_i^{gt} - x_i|}{x_i^g}, \\
 \delta_\theta &= \frac{1}{n} \sum_{i \in \mathcal{V}_g} \left| \left\{ \max\left(\frac{x_i^g}{x_i}, \frac{x_i}{x_i^g}\right) < \theta \right\} \right|.
 \end{aligned} \tag{23}$$

Here,  $\mathcal{V}_g$  is the set of pixels with valid ground truth, and  $n = |\mathcal{V}_g|$  is the size of the set.

### A.3 VISUALIZATION

The output of our Gaussian Belief Propagation Network (GBPN) is a dense depth distribution, modeled as a Gaussian and represented by its mean  $\mu$  and precision  $\Lambda$ . This precision  $\Lambda$  can serve as a confidence measure for the estimated depth. As shown in Fig. 4,  $\Lambda$  is typically lower on object boundaries—indicating higher uncertainty—and higher near sparse point measurements, where confidence is greater.

To efficiently compute the depth distribution, GBPN employs a Serial & Parallel propagation scheme. The ablation study (see Section 4.2 and Table 2) validates the efficacy of this design. For instance,

1026 introducing serial propagation (comparing  $V_2$  and  $V_3$ ) resulted in a substantial performance gain,  
 1027 reducing RMSE from  $340.84\text{mm}$  to  $108.29\text{mm}$ . Furthermore, augmenting the model with parallel  
 1028 propagation over non-local edges (comparing  $V_6$  and  $V_7$ ) yielded a further improvement, lowering  
 1029 RMSE from  $103.92\text{mm}$  to  $101.42\text{mm}$ . In addition to these quantitative results, we provide the  
 1030 qualitative analysis through visualizations to illustrate the impact of the propagation scheme.

1031 To validate the information propagation capability of our serial propagation scheme, we modified  
 1032 GBPN by removing dynamic parameters and edges, resulting in a model that uses only four-directional  
 1033 (left-to-right, top-to-bottom, right-to-left, bottom-to-top) serial propagation. When trained on the  
 1034 NYUv2 dataset with 500 sparse points and tested on an extreme case with only a single depth  
 1035 measurement, the model successfully propagates information across the entire image within a single  
 1036 iteration. The intermediate results, illustrated in Fig. 5, demonstrate rapid convergence to a meaningful  
 1037 depth map, confirming the scheme’s effectiveness.

1038 Unlike traditional MRFs with static, predefined graph edges (e.g., an 8-connected grid), our framework  
 1039 enhances the graph structure by inferring dynamic, non-local edges for each pixel, as illustrated in  
 1040 Fig. 6. These edges connect contextually relevant pixels beyond immediate spatial neighborhoods,  
 1041 enabling the MRF to capture long-range dependencies and model complex scene structures more  
 1042 effectively. GBPN uses a parallel propagation scheme to pass messages simultaneously along these  
 1043 non-local connections, thereby conditioning each variable on both its local neighbors and its learned,  
 1044 non-local context.

1045 It is worth noting that while the theoretical computational complexity of parallel propagation is  
 1046 similar to that of serial propagation (and significantly less than the cost of generating the MRF in  
 1047 GMCN, as detailed in Section A.7), its practical implementation is far more efficient. The parallel  
 1048 scheme readily leverages modern hardware parallelism, resulting in faster execution.

1049  
 1050

#### 1051 A.4 MORE COMPARISON WITH STATE-OF-THE-ART (SOTA) METHODS

1052 The comprehensive comparison of various depth completion methods on the KITTI and NYUv2  
 1053 datasets is presented in Table 5. The results reveal that the proposed GBPN method achieves  
 1054 competitive performance across all metrics. On KITTI, it achieves the best iRMSE and second-best  
 1055 RMSE, indicating strong depth accuracy and consistency. On NYUv2, it matches the lowest RMSE  
 1056 and achieves the highest  $\delta_{1.02}$  and  $\delta_{1.05}$ , reflecting excellent depth accuracy.

1057  
 1058

1059 Visual comparisons with other SOTA open-source methods on the validation set of KITTI DC  
 1060 and the test set of NYUv2 are shown in Fig. 7 and Fig. 8. The official implementations and the  
 1061 best-performing models released by the authors are used to ensure fair comparisons, with the same  
 1062 sparse depth maps across all methods. Our results, presented in the last row, demonstrate sharper  
 1063 object boundaries and more detailed structures, while other methods tend to underperform in these  
 1064 challenging areas, leading to less accurate depth estimates.

1065  
 1066

#### 1067 A.5 MORE COMPARISON ON SPARSITY ROBUSTNESS

1068  
 1069

##### 1070 A.5.1 SIMULATION WITH REDUCED LiDAR SCAN LINES

1071 To further evaluate our model’s performance on sparser inputs, we tested all compared methods using  
 1072 sub-sampled LiDAR data corresponding to 8, 16, 32, and 64 scan lines. A detailed comparison on  
 1073 the KITTI validation set is shown in Table 6. Across all levels of sparsity, our method consistently  
 1074 delivers the best performance for RMSE, REL, and MAE. Notably, under the most challenging 8-line  
 1075 setting, our model demonstrates a clear advantage over other methods. For example, compared to  
 1076 BP-Net, GBPN reduces the RMSE from 4541.9 mm to 2750.4 mm, yielding a substantial 1791.5 mm  
 1077 absolute improvement. This underscores the robustness and adaptability of our method, particularly  
 1078 in scenarios with extremely sparse depth input. Visual results in Fig. 9 further support these findings.  
 1079 Our model effectively preserves structural details even with limited LiDAR information, whereas  
 competing methods tend to produce smoother or more ambiguous results with fewer LiDAR lines.

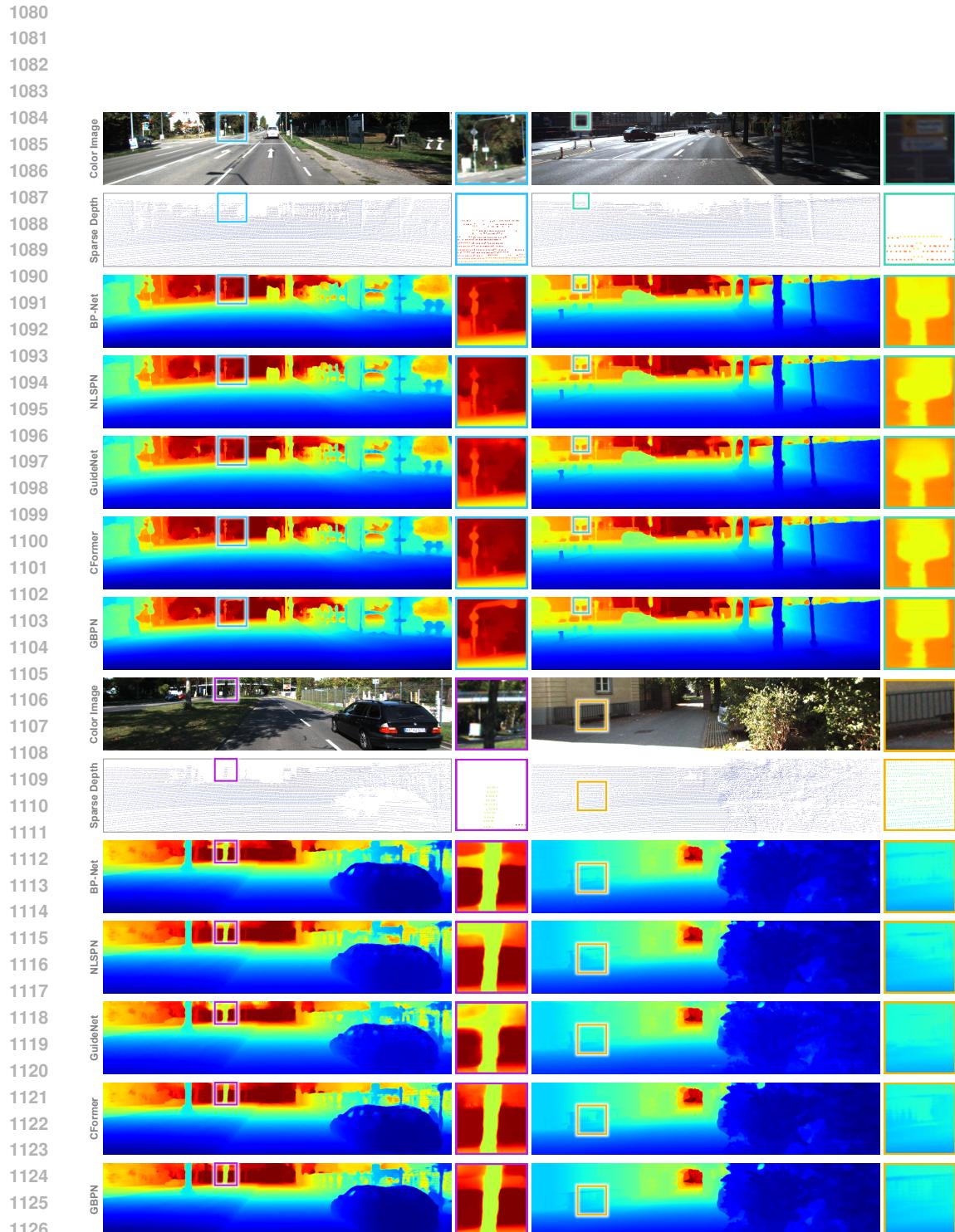


Figure 7: **Qualitative comparison on KITTI validation set.** Comparing with GuideNet (Tang et al., 2020), NLSPN (Park et al., 2020), CFormer (Zhang et al., 2023) and BP-Net (Tang et al., 2024). Our method is presented in the last row, with key regions highlighted by rectangles for easy comparison

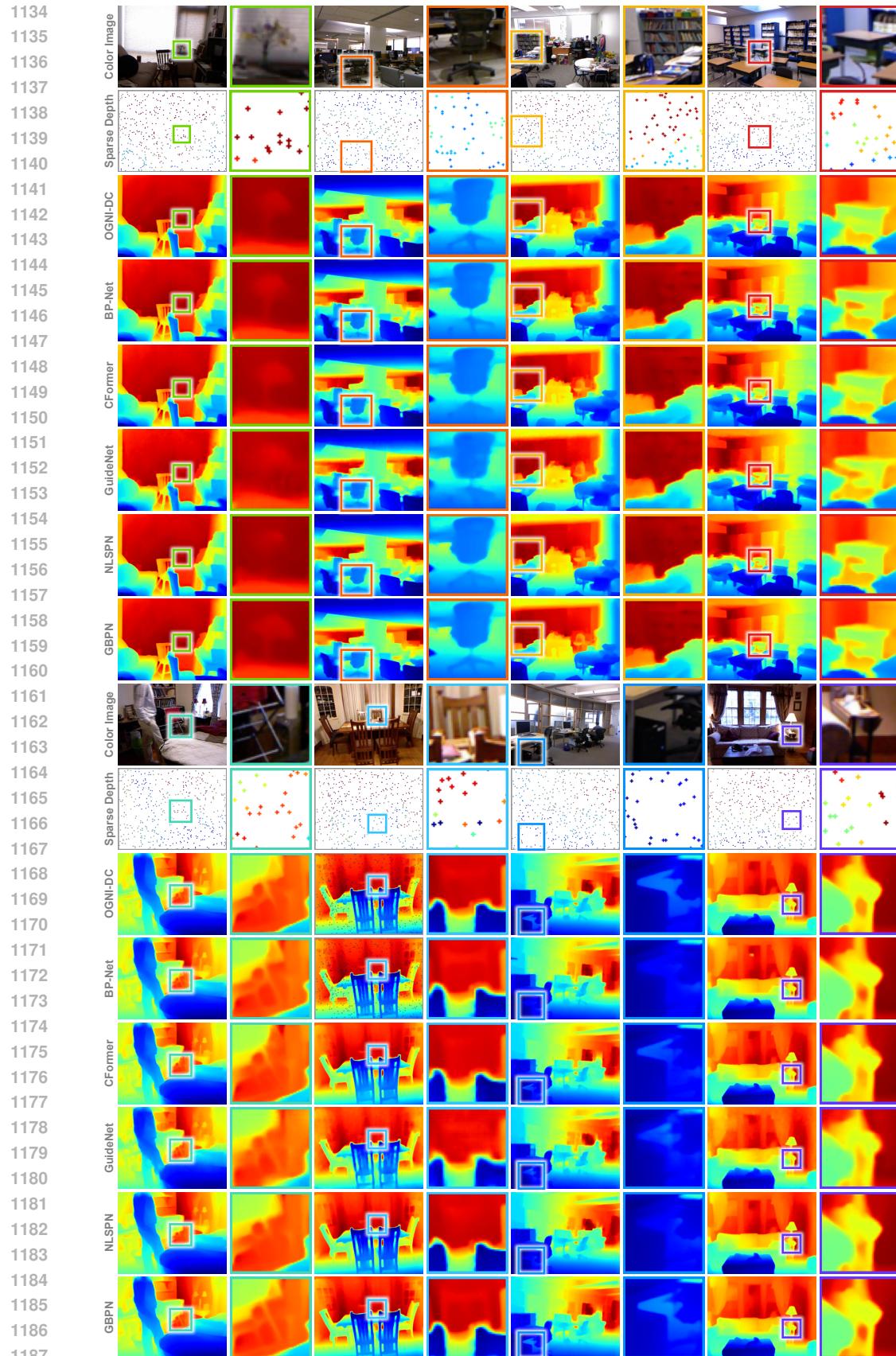


Figure 8: **Qualitative comparison on NYUv2 test set.** Our method is compared with BP-Net (Tang et al., 2024), NLSPPN (Park et al., 2020), GuideNet (Tang et al., 2020), CFormer (Zhang et al., 2023) and OGNI-DC (Zuo & Deng, 2024). For clearer visualization, sparse depth points are enlarged. Our method is presented in the last row, with key regions highlighted by rectangles to facilitate comparison.

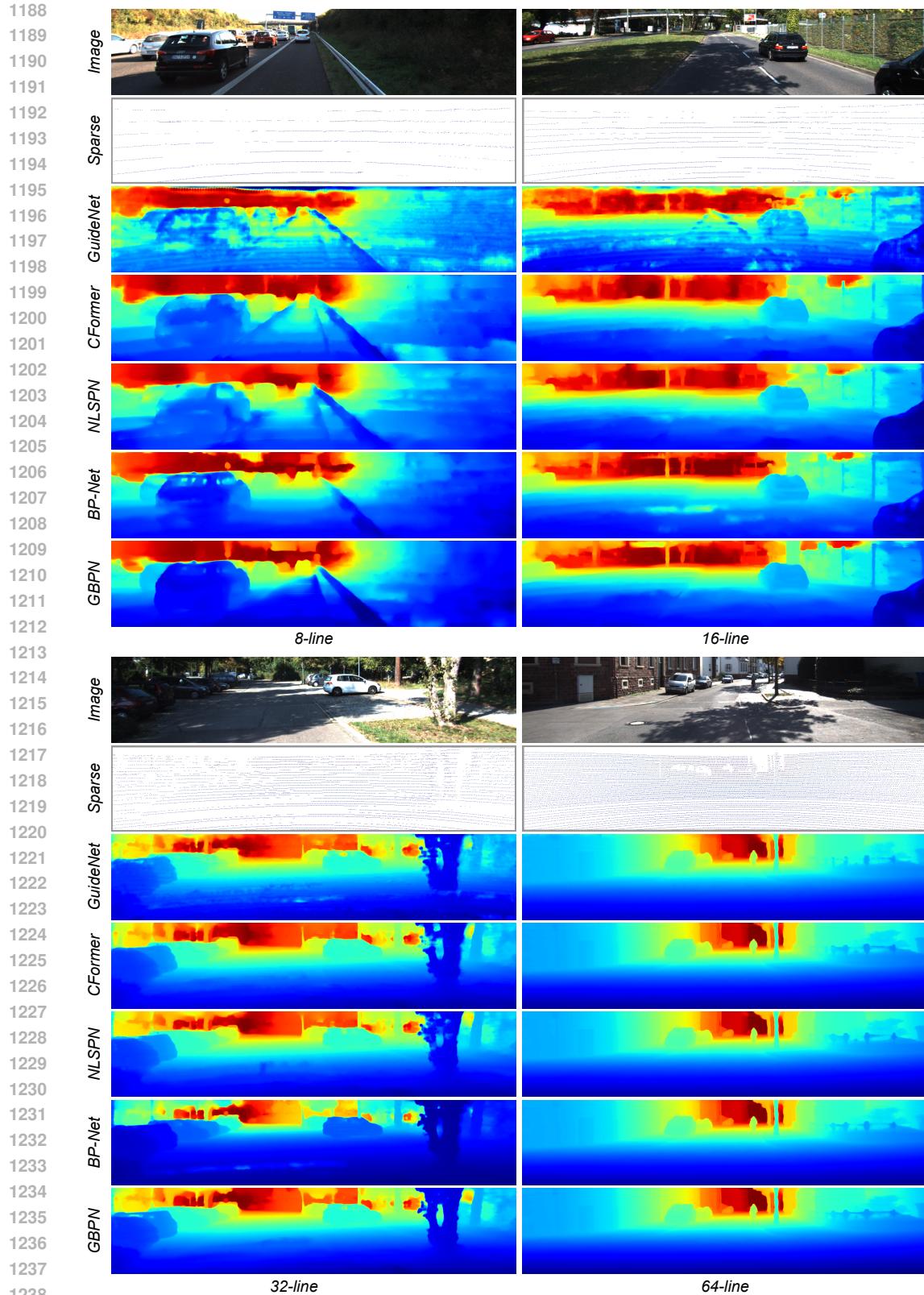


Figure 9: **Qualitative comparison on KITTI validation set with different LiDAR line.** Our method is compared with BP-Net (Tang et al., 2024), NLSPN (Park et al., 2020), GuideNet (Tang et al., 2020) and CFormer (Zhang et al., 2023).

1242  
 1243 **Table 5: Performance on KITTI and NYUv2 datasets.** For the KITTI dataset, results are evaluated  
 1244 by the KITTI testing server. For the NYUv2 dataset, authors report their results in their papers. The  
 1245 best result under each criterion is in **bold**. The second best is with underline.

	KITTI				NYUv2			
	RMSE $\downarrow$ (mm)	MAE $\downarrow$ (mm)	iRMSE $\downarrow$ (1/km)	iMAE $\downarrow$ (1/km)	RMSE $\downarrow$ (m)	REL $\downarrow$	$\delta_{1.02} \uparrow$ (%)	$\delta_{1.05} \uparrow$ (%)
S2D (Ma & Karaman, 2018)	814.73	249.95	2.80	1.21	0.230	0.044	—	—
DeepLiDAR (Qiu et al., 2019)	<u>758.38</u>	226.50	2.56	1.15	0.115	0.022	—	—
CSPN++ (Cheng et al., 2020)	743.69	209.28	2.07	0.90	0.115	—	—	—
GuideNet (Tang et al., 2020)	736.24	218.83	2.25	0.99	0.101	0.015	82.0	93.9
FCFR Hu et al. (2021)	735.81	217.15	2.20	0.98	0.106	0.015	—	—
NLSPN (Park et al., 2020)	741.68	199.59	1.99	0.84	0.092	0.012	88.0	95.4
ACMNet (Zhao et al., 2021a)	744.91	206.09	2.08	0.90	0.105	0.015	—	—
RigNet Yan et al. (2022)	712.66	203.25	2.08	0.90	0.090	0.013	—	—
DySPN (Lin et al., 2022)	709.12	192.71	1.88	<u>0.82</u>	0.090	0.012	—	—
BEV@DC (Zhou et al., 2023)	697.44	189.44	1.83	<u>0.82</u>	0.089	0.012	—	—
DGDF (Wang et al., 2023b)	707.93	205.11	2.05	0.91	0.098	0.014	—	—
CFormer (Zhang et al., 2023)	708.87	203.45	2.01	0.88	0.090	0.012	87.5	95.3
LRRU (Wang et al., 2023a)	696.51	189.96	1.87	<b>0.81</b>	0.091	<u>0.011</u>	—	—
TPVD (Yan et al., 2024)	693.97	<u>188.60</u>	<u>1.82</u>	<b>0.81</b>	<u>0.086</u>	<b>0.010</b>	—	—
ImprovingDC (Wang et al., 2024)	686.46	<b>187.95</b>	1.83	<b>0.81</b>	0.091	<u>0.011</u>	—	—
OGNI-DC (Zuo & Deng, 2024)	708.38	193.20	1.86	0.83	0.087	<u>0.011</u>	88.3	<b>95.6</b>
BP-Net (Tang et al., 2024)	684.90	194.69	<u>1.82</u>	0.84	0.089	0.012	87.2	95.3
DMD3C (Liang et al., 2025)	<b>678.12</b>	194.46	<u>1.82</u>	0.85	<b>0.085</b>	<u>0.011</u>	—	—
GBPN	<u>682.20</u>	192.14	<b>1.78</b>	<u>0.82</u>	<b>0.085</b>	<u>0.011</u>	<b>89.1</b>	<b>95.9</b>

1265  
 1266 **Table 6: Performance on KITTI’s validation set with different simulated lines.** All compared methods  
 1267 are tested using sub-sampled LiDAR data with 8, 16, 32, and 64 scan lines.

LiDAR Scans	8-Line			16-Line			32-Line			64-Line		
	Methods	RMSE	REL	MAE	RMSE	REL	MAE	RMSE	REL	MAE	RMSE	REL
CFormer (Zhang et al., 2023)	3660.5	106.4	1720.3	2196.2	46.4	822.1	1242.2	21.9	380.9	745.6	10.7	197.3
NLSPN (Park et al., 2020)	3244.2	92.3	1512.2	1949.8	38.4	676.2	1195.2	20.5	353.5	771.8	10.5	197.3
BP-Net (Tang et al., 2024)	4541.9	143.8	1822.8	2362.7	52.0	1822.8	1425.2	29.6	399.6	715.2	10.4	194.0
GuideNet (Tang et al., 2020)	5805.8	216.2	3109.2	3291.6	93.5	1455.8	1357.5	28.5	452.7	770.8	12.3	222.2
GBPN	<b>2750.4</b>	<b>79.5</b>	<b>1233.6</b>	<b>1744.1</b>	<b>31.6</b>	<b>560.6</b>	<b>1073.1</b>	<b>17.6</b>	<b>311.4</b>	<b>712.4</b>	<b>10.3</b>	<b>191.8</b>

### 1277 A.5.2 SIMULATION WITH VARIOUS DEPTH DENSITY

1279 The metrics on the NYUv2 validation set under various levels of input depth sparsity are presented in  
 1280 Table 7, ranging from 20 to 20,000 sparse points. For a thorough evaluation, given a sparsity level,  
 1281 each test image is sampled 100 times with different random seeds to generate the input sparse depth  
 1282 map. The performance of each method is averaged on these 100 randomly sampled inputs to reduce  
 1283 the potential bias due to random sampling, especially for high sparsity.

1285 **Table 7: RMSE (m) on NYUv2 with depth input from various sparsity.** The best metric under each  
 1286 sparsity level is in **bold**.

Points	20	50	100	200	500	1000	2000	5000	10000	20000
CFormer (Zhang et al., 2023)	0.932	0.709	0.434	0.142	0.091	0.071	0.056	0.045	0.066	0.204
NLSPN (Park et al., 2020)	0.691	0.429	0.248	0.136	0.092	0.072	0.056	0.042	0.035	0.035
OGNI-DC (Zuo & Deng, 2024)	1.523	1.149	0.696	0.172	0.088	0.069	0.054	0.040	0.032	0.030
BP-Net (Tang et al., 2024)	0.749	0.547	0.302	0.131	0.090	0.070	0.054	<b>0.039</b>	<b>0.031</b>	<b>0.023</b>
GuideNet (Tang et al., 2020)	0.908	0.603	0.478	0.187	0.101	0.081	0.070	0.087	0.195	—
GBPN-1	<b>0.647</b>	<b>0.331</b>	<b>0.179</b>	0.135	0.101	0.081	0.065	0.048	0.037	0.027
GBPN-2	0.649	0.364	0.198	<b>0.120</b>	<b>0.085</b>	<b>0.067</b>	<b>0.053</b>	<b>0.039</b>	0.032	0.026

Our method is listed in the last two rows, GBPN (with variants GBPN-1 and GBPN-2), consistently achieves the highly competitive performance across all sparsity levels. Under extremely sparse input, 20 and 50 points, GBPN-1 achieves the lowest RMSE, significantly outperforming other methods. As the number of input points increases, GBPN-2 starts to excel, obtaining the best performance at intermediate sparsity levels more than 100 points. Notably, both GBPN variants remain highly effective even at dense input levels (5000 and 20,000 points). In contrast, GuideNet (Tang et al., 2020) and Cformer (Zhang et al., 2023) has a worse performance when the input depth is significantly denser (beyond approximately 5000 points) than the training sparsity (500 points). Similar phenomenon has also been observed by (Zuo & Deng, 2024), and we attribute this to the lack of robustness to changes in input sparsity. Both GuideNet (Tang et al., 2020) and CFormer (Zhang et al., 2023) directly process the sparse depth map using convolutional layers, which are not optimal for handling sparse data. In addition, these methods were trained exclusively with 500 valid points. When presented with a significantly denser input at test time, the input distributions passed to the initial convolutional layers differ drastically from what the network was trained on. This domain shift may cause the network to produce worse results, regardless of whether the input density is increased or decreased. Our method alleviate this issue by treating sparse depth measurements as principled observation terms within a globally optimized MRF framework.

The qualitative comparison in Fig. 10 further confirms these findings. Despite varying sparsity levels, our method reconstructs sharper structures and more accurate depth boundaries compared to existing approaches. In particular, it maintains strong performance under sparse input conditions where other methods tend to blur edges or produce overly smooth estimations. These results demonstrate that GBPN is both robust and generalizable across a wide range of input sparsity, making it well-suited for real-world scenarios with depth measurements of various sparsity, e.g. SfM (Schonberger & Frahm, 2016).

## A.6 NOISE SENSITIVITY

In the real world, depth measurements may tend to be noisy due to sensor errors or environmental factors like fog. To evaluate the noise sensitivity of DC methods, we evaluate the compared methods with noisy depth input. Specifically, each sparse depth measurement is perturbed by a uniformly distributed relative error,  $\theta$ . For instance,  $\theta = 1\%$  indicates an error uniformly sampled between  $-1\%$  and  $1\%$  of the true depth value. All models were trained on the clean NYUv2 dataset and evaluated on these corrupted inputs. As shown in Table 8, GBPN consistently achieves the best RMSE across noise levels ranging from  $\theta = 1\%$  to  $\theta = 5\%$ , demonstrating its robustness against noise and its ability to maintain stable and accurate depth predictions even under corrupted measurements.

Table 8: RMSE (mm) Comparison across different levels of noise in sparse depth measurements.

Method	$\theta = 1\%$	$\theta = 2\%$	$\theta = 3\%$	$\theta = 4\%$	$\theta = 5\%$
OGNI (Zuo & Deng, 2024)	89.82	93.31	98.77	107.11	116.23
NLSPN (Park et al., 2020)	93.48	96.86	101.93	110.38	119.72
CFormer (Zhang et al., 2023)	92.81	96.27	101.88	110.12	119.98
BP-Net (Tang et al., 2024)	90.63	93.54	98.43	106.12	113.85
GuideNet (Tang et al., 2020)	101.91	104.12	108.32	114.92	122.51
GBPN	<b>87.17</b>	<b>90.41</b>	<b>95.60</b>	<b>104.01</b>	<b>113.10</b>

## A.7 EFFICIENCY COMPARISON

We compare the runtime efficiency of DC methods on samples from NYU and KITTI dataset with input resolution of  $256 \times 320$  and  $352 \times 1216$ , respectively. As shown in Table 9, GBPN achieves a competitive balance of runtime, memory usage, and parameter count compared with these SOTA methods. Here, GuideNet demonstrates the fastest inference speed, while CFormer and OGNI-DC are notably slower. Our GBPN exhibits a moderate inference time compared to these SOTA methods. Comparing with BP-Net(Tang et al., 2024), GBPN has a higher runtime but requires fewer parameters. The lower parameter count is because the Markov Random Field (MRF) is dynamically constructed by a relatively small network compared to other methods.

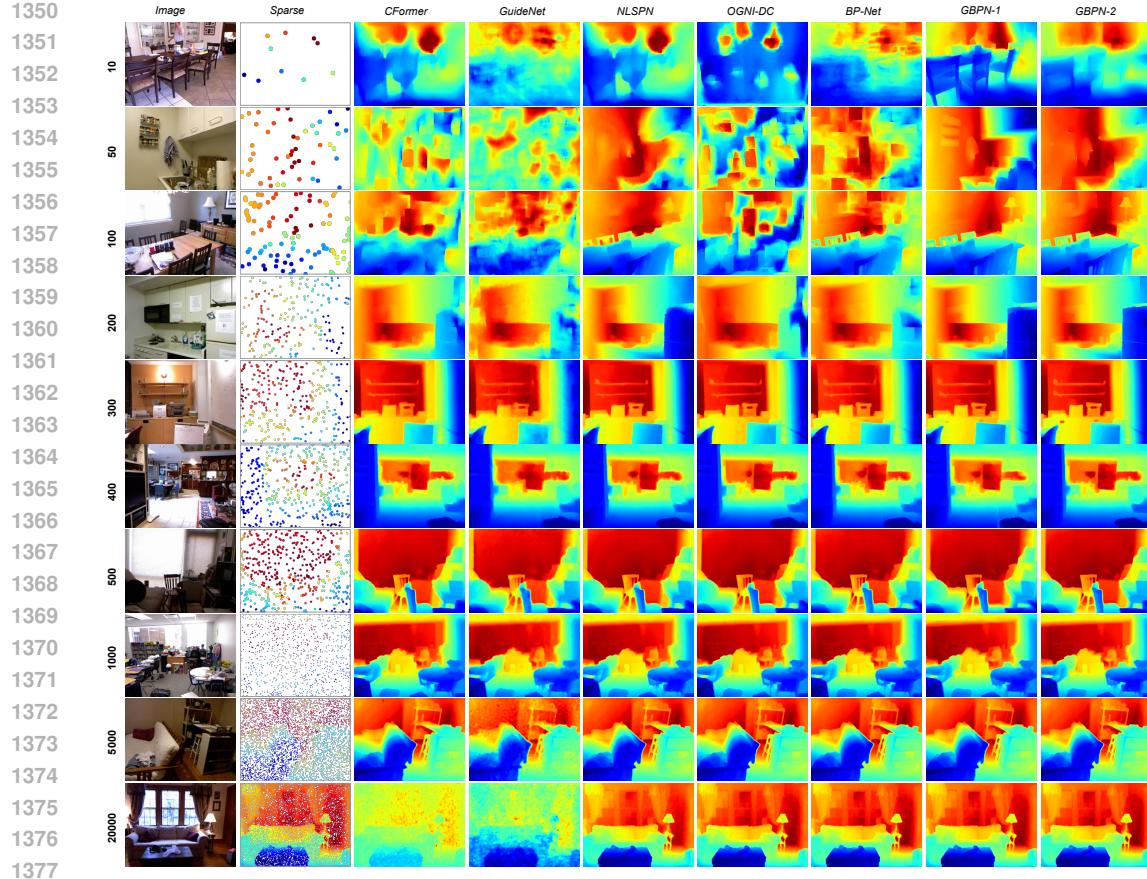


Figure 10: **Qualitative comparison on NYUv2 test set with different sparsity.** Our method is compared with BP-Net (Tang et al., 2024), NLSPN (Park et al., 2020), GuideNet (Tang et al., 2020), CFormer (Zhang et al., 2023) and OGNI-DC (Zuo & Deng, 2024). For each sparsity level, the first row is input image, the second row is sparse map.

Table 9: Comparison of inference time (ms), GPU memory (GB), and number of parameters (M).

Method	256 × 320		352 × 1216		Params (M)
	Runtime (ms)	GPU Memory (GB)	Runtime (ms)	GPU Memory (GB)	
GuideNet (Tang et al., 2020)	3.52	4.65	12.95	8.85	62.62
NLSPN (Park et al., 2020)	8.50	1.26	39.34	5.69	26.23
BP-Net (Tang et al., 2024)	19.87	5.31	77.64	9.07	89.87
CFormer (Zhang et al., 2023)	115.50	1.98	150.76	7.85	83.51
OGNI-DC (Zuo & Deng, 2024)	177.02	2.77	266.06	9.25	84.37
GBPN	44.57	4.47	137.49	8.70	39.03

Our GBPN consists of a Graphical Model Construction Network (GMCN) to dynamically construct a scene-specific Markov Random Field (MRF), and a Gaussian Belief Propagation (GBP) module for depth distribution inference. We analyzed the computation and runtime of these two components at an input resolution of 256 × 320. The GMCN is built using highly-optimized PyTorch layers, whereas our GBP module is a custom implementation. As shown in Table 10, the computation of the GBP module are roughly 200 times lower than those of the GMCN, yet it requires 3 times more computation time. This inefficiency, where less computation takes more time, is primarily due to the limitations of our custom GBP implementation. Although our current implementation is parallelized

1404 on GPUs, it lacks the extensive optimization like the standard PyTorch layers and thus has significant  
 1405 potential for acceleration in future work.  
 1406

1407  
 1408 Table 10: Comparison of computation and runtime between GMCN and GBP.  
 1409

Components	Computation (GFLOPs)	Runtime (ms)
GMCN	62.58	11.03
GBP	0.26	33.54

1413 Our GBPN also has the advantage of *flexible latency tuning*. As the target depth is optimized gradually  
 1414 in an iterative manner, we can trade off the model’s latency and performance by simply adjusting the  
 1415 number of iterations. We list the trade-off in Table 11 by starting from our GBPN model with a total of  
 1416 13 iterations and then gradually reducing the number of iterations. This property is highly beneficial  
 1417 for real applications’ deployment, as GBPN can be easily tailored to specific latency requirements  
 1418 by simply adjusting the number of optimization iterations. In contrast, feed-forward networks like  
 1419 CFormer (Zhang et al., 2023) must complete their full inference pass to produce the result, making  
 1420 them difficult to adapt to varying latency budgets.  
 1421

1422 Table 11: Trade-off between latency and accuracy on the NYUv2 validation set under different  
 1423 number of iterations.  
 1424

Iteration	5	6	7	8	9	10	11	12	13
RMSE under 500 points (m)	0.146	0.141	0.117	0.106	0.099	0.098	0.090	0.087	0.085
RMSE under 400 points (m)	0.157	0.151	0.126	0.114	0.107	0.104	0.096	0.093	0.091
RMSE under 300 points (m)	0.178	0.171	0.140	0.128	0.119	0.113	0.104	0.101	0.100
RMSE under 200 points (m)	0.241	0.228	0.184	0.173	0.154	0.137	0.124	0.121	0.119
RMSE under 100 points (m)	0.471	0.439	0.356	0.344	0.252	0.298	0.238	0.233	0.214
Runtime (ms)	22.17	24.98	27.97	31.35	33.41	36.55	39.27	42.00	44.57

1431  
 1432 A.8 DECLARATION OF LLM USAGE  
 1433

1434 This is an original research paper. The core method development in this research does not involve  
 1435 LLMs as any important, original, or non-standard components. LLM is used only for editing and  
 1436 formatting purposes.  
 1437

1438  
 1439  
 1440  
 1441  
 1442  
 1443  
 1444  
 1445  
 1446  
 1447  
 1448  
 1449  
 1450  
 1451  
 1452  
 1453  
 1454  
 1455  
 1456  
 1457