# Plug-and-Play: Improve Depth Prediction via Sparse Data Propagation

Tsun-Hsuan Wang[1], Fu-En Wang[1], Juan-Ting Lin[1], Yi-Hsuan Tsai[2], Wei-Chen Chiu[3], Min Sun[1]

*Abstract*— We propose a novel plug-and-play (PnP) module for improving depth prediction with taking arbitrary patterns of sparse depths as input. Given any pre-trained depth prediction model, our PnP module updates the intermediate feature map such that the model outputs new depths consistent with the given sparse depths. Our method requires no additional training and can be applied to practical applications such as leveraging both RGB and sparse LiDAR points to robustly estimate dense depth map. Our approach achieves consistent improvements on various state-of-the-art methods on indoor (i.e., NYU-v2) and outdoor (i.e., KITTI) datasets. Various types of LiDARs are also synthesized in our experiments to verify the general applicability of our PnP module in practice.

## I. INTRODUCTION

Depth perception is of great importance in a wide range of applications, including autonomous driving, robotics, augmented reality, and scene reconstruction. While numerous depth sensing technologies are available, e.g., RGB-D camera, stereo camera rig, and LiDAR, they all have their own pros and cons in different scenarios. For instance, RGB-D or stereo cameras could provide depth estimation which is dense but typically fails on highly reflective and transparent surfaces, or distant objects from the camera. Oppositely, expensive depth sensors such as LiDAR are able to function well under challenging weather conditions and robustly estimate depth of distant objects, but only limit to sparse coverage (e.g., Velodyne 64-channel LiDAR only covers around 5% of the image space after re-projection [1]). Therefore, there is a clear need to take advantage of strengths from different sensing approaches to improve depth perception. In this paper, we particularly tackle the problem of improving dense depth prediction with sparse yet accurate data.

On the one hand, building upon monocular or binocular depth prediction methods, several existing methods (detailed in Sec. II) take sparse depth information as additional input [2][3][4] or propagate sparse data towards their neighborhood [5][6]. However, these approaches are either inflexible due to requirement of additional training, or being constrained to specific input types, patterns, or density. Hence, they are less practical for real-world applications.

On the other hand, network visualization [7] and adversarial attack [8] investigate how intermediate feature representations of a network influence the prediction and how such representations can be modified to achieve desirable outputs. Inspired by these techniques, we introduce a plug-and-play (PnP) module for improving dense depth prediction, which requires no additional training and can be adapted to various

[1] National Tsin Hua University
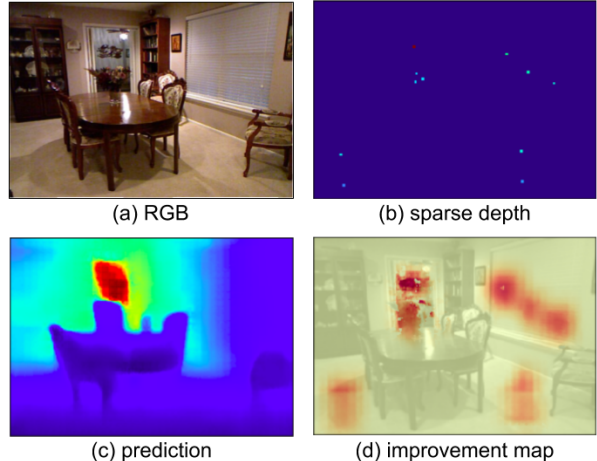[2] NEC Labs America
[3] National Chiao Tung University

Fig. 1: Given an (a) RGB image and a depth estimation/completion model, our PnP module propagates information from (b) sparse points and generates (c) improved depth prediction without any retraining. We show the (d) improvement denoted in red region.

input types. Given a pre-trained depth prediction model, our PnP module updates intermediate feature maps of the original dense depth prediction, enforcing the model to produce a new depth output which is consistent with the given sparse data. Thereby, the proposed PnP module is not only easily integrable to all differentiable depth prediction approaches but also training-free, simply performed during inference.

We investigate the integration of our PnP module to numerous depth prediction approaches and conduct experiments on both the indoor and outdoor benchmark datasets, i.e., NYU-v2 [9] and KITTI [10], respectively. For all the methods on both datasets, our PnP module provides consistent improvements for depth prediction. Moreover, we show the robustness of the proposed method under various settings, including different density and input types. Most importantly, we synthesized various types of LiDARs to verify the general applicability of our method to leverage RGB and sparse LiDAR depth to robustly estimate dense depths. Finally, we demonstrate how our module propagates information from sparse data and improves prediction in the local region, and provide a rule of thumb of using our PnP module via a thorough ablative analysis. The source code is available at https://zswang666.github.io/PnP-Depth-Project-Page/.

## II. RELATED WORK

**RGB-based Depth Estimation.** Monocular or binocular depth estimation has been studied widely in robotics and computer vision. Recent works are mostly built upon deep networks with differentiable architectures. For instance, [11][12] first propose a multi-scale network to predict depth

map in a coarse-to-fine manner. In order to increase the edge response on the estimated depth map, Liu *et al.* [13] develop a deep convolutional neural field model to incorporate continuous Conditional Random Field (CRF) technique into the network. The performance of depth estimation is further boosted by Laina *et al.* [14] via introducing a powerful residual network and up-projection layers in the deep architecture. More recently, Fu *et al.* [15] propose to formulate depth prediction as an ordinal regression problem and obtain impressive results on depth estimation. Despite that these depth estimation methods use various network architectures, our PnP module can be flexibly integrated into these approaches without re-training their models and requiring specific constraints on the network choice.

**Depth Reconstruction from Sparse Data.** The nature of obtained high-precision but sparse depth maps from active depth sensors (e.g., LiDAR, Kinect) recently brings the research interests in sparse-to-dense depth reconstruction and completion. [16][17] perform depth reconstruction upon hypothesizing the sparsity of disparity maps in the Wavelet domain. Ferstl *et al.* [18] propose to upsample low-resolution depth maps guided by anisotropic diffusion tensors. For specific sensors, Uhrig *et al.* [1] design a sparse convolution layer to handle missing data captured from LiDAR scans, and Liao *et al.* [19] construct a residual network to parse the 2D planar measurements obtained by a laser range finder. In order to design a more generalizable method not only for particular sensors, Ma *et al.* [2] and Jaritz *et al.* [3] respectively adopt early-fusion and late-fusion strategies to combine the RGB image and sparse depth for producing the final depth estimation. [20] further studies the connection between depth and surface normal maps for depth completion. In addition, Chodosh *et al.* [5] combine compressed sensing with deep learning into a single framework and perform depth completion to be consistent with the given sparse data. This model shares a similar idea to ours of optimizing intermediate representation to yield the depth prediction.

While the above-mentioned works propose stand-alone approaches for depth reconstruction, our goal is to design a module that can be flexibly integrated into any existing methods and improve their performance. From this modeling perspective, the target task is extended from depth reconstruction further to depth refinement, where the initial result of depth estimation or reconstruction is boosted by the given sparse data. To the best of our knowledge, the work closest to our scope is [6] that proposes a convolutional spatial propagation network by concatenating it to a depth estimation network thus achieves depth refinement. However, this approach [6] requires an additional training stage while our method simply needs to perform the inference stage. It is also worth noting that, with the given sparse data, our method can be comfortably applied to both depth reconstruction and depth refinement approaches.

## III. METHOD

Different from existing depth reconstruction methods that design a full model [2][3][19][20] or a refinement component

[5][6] that requires additional training, we propose a plug-and-play module that is used during inference, without updating model weights, and can be applied to any differentiable models, including the above-mentioned methods and existing depth estimation approaches.

Given the input data $x$ (e.g., an RGB image) and a depth estimation model $f$, our goal is to improve the dense depth prediction $f(x)$ with the help of the sparse depth $D_s$ during inference. Here, input $x$ could be RGB images, sparse depths, or both. In the following, we sequentially introduce the proposed PnP module, demonstrate its usefulness with mathematical insights, and describe a rule of thumb of utilizing our module.

### A. Preliminary for Depth Estimation

Without loss of generality, the objective of learning-based approaches for depth estimation can be described as follows. Given the input data $x$, we aim to minimize the error between the prediction $f(x)$ and the ground truth depth $D$, with respect to the network $f$ parametrized by $\theta$:

$$\theta^* = \arg\min_\theta \mathcal{L}(f(x; \theta), D), \qquad (1)$$

where $\mathcal{L}(\cdot, \cdot)$ is the loss function that could be mean absolute error ($L_1$), mean squared error ($L_2$), reversed Huber loss [14], inverse depth loss [3], or classification loss [15][19][21].

Although the predicted depth $f(x; \theta)$ can be affected by both the model parameters $\theta$ and the input $x$, typically only the parameters $\theta$ are updated in learning depth estimation network. In our method, we aim to design a plug-and-play module for improving the depth estimation without any additional training, such that the $\theta$ of network $f$ needs to remain fixed whereas updating $x$ seems the only feasible way in order to minimize the $\mathcal{L}$ loss. This idea is analogous to the one in adversarial attack [8] (which we are motivated from), where the network is unchanged and the perturbation is added to input $x$ for making the new prediction towards an assigned target output. In practice, instead of directly updating $x$, we choose to update an intermediate feature representation that is a function of $x$ for efficiency (details described as follows).

### B. Sparse Data Propagation

Since we aim to update the intermediate feature, which corresponds to the response of a certain layer in depth estimation network, the network $f$ can be treated as a cascade of two parts, $f_{front}$ and $f_{rear}$, such that $z = f_{front}(x)$ and $f(x) = f_{rear}(f_{front}(x))$. The essence of our method is to update $z$ in the direction that improves depth prediction $f(x)$ with partial ground truth $D_s$ and it can be formulated as:

$$z^* = \arg\min_z \mathcal{L}(f_{rear}(z; \theta_{rear}), D_s),$$
$$f(x) = f_{rear}(z^*), \qquad (2)$$

where the value of $z$ during minimization is initialized as $f_{front}(x)$. Similar to training a deep network, we adopt back-propagation [22] to solve the minimization problem over $z$.
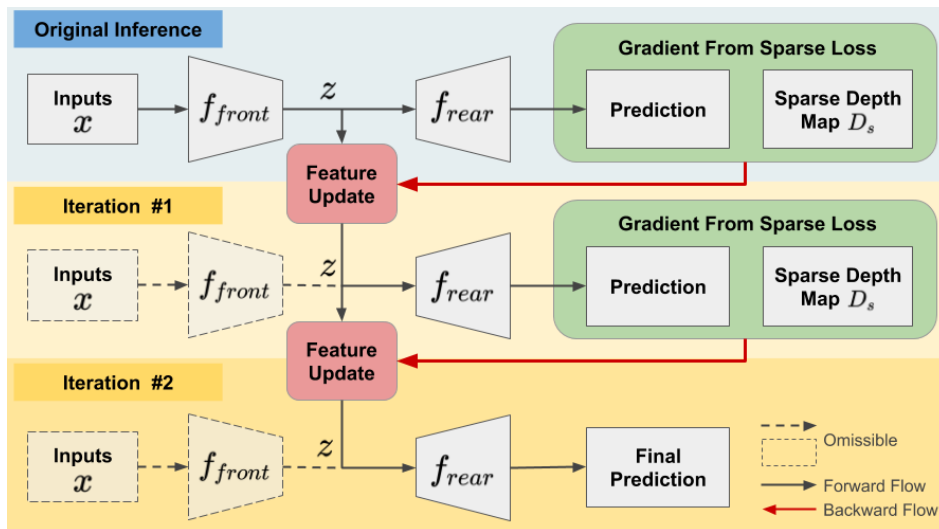
Fig. 2: **Illustration of the PnP module via backward-forward propagation.** Given a depth estimation model $f$ and sparse ground truth $D_s$, our PnP module iteratively updates the intermediate representation $z$ based on the gradient computed from sparse points and re-inference to obtain better depth prediction. Here, we illustrate an example for two iterations.

The iterative optimization process is written as:

$$z_0 = f_{front}(x),$$
$$z_{k+1} = z_k - \alpha U\left(\frac{\partial \mathcal{L}(f_{rear}(z_k), D_s)}{\partial z_k}\right), \quad (3)$$

where $U(\cdot)$ can be any variants of gradient descent methods such as Adam [23] and $\alpha$ is the learning rate. For simplicity, we adopt fast gradient sign method [8], where $U(\cdot) = sign(\cdot)$ that aligns well with the adversarial attack technique used in [8] and is found to be both efficient and empirically effective in experiments. Fig. 2 illustrates how our proposed PnP module works. By alternatingly performing back-propagation and re-inference on the depth estimation network, which we denote this procedure as *backward-forward propagation*, the information of the given sparse depth is propagated to their corresponding local regions. We also note that, the parameters of network $f$ is not updated but only $z$ is. In addition, the choice of which intermediate layer for $z$ depends on whether it contains sufficiently large area impacted in the output depth map, which we name this as *influential field* and we will discuss it in Sec. III-D.

## C. Theoretical Discussion

We have described the proposed PnP module to update the intermediate feature map $z$ based on the sparse points only. However, it still remains a question whether such updates guarantee improvement in dense depth map. In this section, we aim to demonstrate that our method provides sufficiently correct guidance for dense depth reconstruction. Here, we first consider the gradient with respect to $z$ if the dense ground truth $D$ is given:

$$g_{ij} = \frac{\partial \mathcal{L}(f_{rear}(z)_{ij}, D_{ij})}{\partial z}, \quad (4)$$

where $g_{ij}$ provides a gradient for $z$ toward correct prediction guided by the given ground truth at pixel $(i, j)$. Assuming

$M$ is a mask such that $D_s = M \cdot D$, we can compute the gradient based on the sparse ground truth $D_s$:

$$\hat{g}_{ij} = \frac{\partial M_{ij}\mathcal{L}(f_{rear}(z)_{ij}, D_{ij})}{\partial z}$$
$$= M_{ij}g_{ij} + \frac{\partial M_{ij}}{\partial z}\mathcal{L}(f_{rear}(z)_{ij}, D_{ij}), \quad (5)$$

where $M_{ij}$ is 1 if the depth is given and 0, otherwise. Comparing (4) and (5), $\hat{g}$ is a masked $g$ plus a residual term. For the residual term, when $D_s$ is included in the input $x$, $z$ is a function of $D_s$ as $z = f_{front}(x)$, resulting in $z$ also being a function of $M$, and thus this term is not always zero.

There are two cases for the residual term being a small value, which indicates that the gradient computed using sparse points still provides an approximately correct direction for the dense map: 1) $M_{ij} = 1$ when sparse ground truth is provided on pixel $(i, j)$, leading to small $\mathcal{L}(f_{rear}(z)_{ij}, D_{ij})$; and 2) $M_{ij} = 0$ when sparse ground truth is not provided on pixel $(i, j)$, in which $z$ should not be affected by $M_{ij}$, resulting in $\frac{\partial M_{ij}}{\partial z} \approx 0$.

## D. Property and Usage

In this section, we describe the property of the proposed PnP module and how it is applicable to existing models.

**Influential Field.** We define the local area propagated from the sparse points as the influential field, in which this field is confined by the backward-forward propagation process. The idea of influential field is similar to the one of receptive field, while the former indicates a region in the network output and the latter points to the input. For example, back-propagating through two consecutive $3 \times 3$ convolution layers with stride 1 yields a $5 \times 5$ influential field in our method. As such, in order to enlarge the propagated area, our module suggests to choose an intermediate feature map $z$ that covers sufficiently large influential field. Please note that [6] formulates depth refinement as an anisotropic diffusion process and the information of sparse points is iteratively

diffused to their neighboring regions, in which its concept is similar to our influential field but needs additional training.

**Number of Iteration.** During the process of backward-forward propagation, we can control how many iterations are performed. With more iterations, our module is able to gradually update the dense depth map from the sparse points but result in slower inference time. We will show in the experimental section that the improvement gradually saturates as iteration grows, and hence large iteration number is not preferable.

## IV. Experimental Results

In this section, we first introduce the baseline methods used to integrate with the proposed PnP module and the datasets for evaluation. Second, we present experimental results of our method with various types of input. Further analysis on the property of our PnP module is also presented.

### A. Baselines and Implementation Details

With the given sparse points of depth, our PnP module can further improve the depth estimation during inference. Here, we select numerous representative approaches for RGB-based depth estimation and depth reconstruction.

**RGB-based Depth Estimation Methods.** Two commonly used monocular depth estimation methods [11][14] and one unsupervised approach [24] are considered:

- **Eigen *et al.*** [11] utilize a coarse network of depth estimation followed by a refinement model that takes the coarse output and RGB image as inputs to produce the final depth estimation. Since the refinement network is only composed of two convolutional layers, we further back-propagate through the coarse network;
- **FCRN** [14] is built upon ResNet-50 followed by up-sample blocks to decode features for dense estimation. We back-propagate gradients to the layer before the first up-sample block;
- **Zhou *et al.*** [24] adopt an U-Net based model and we select the bottleneck feature map as the intermediate feature map $z$, while all the skip-connected features are not updated.

**Depth Reconstruction Methods.** For the approaches that already use the sparse depth as the input, our PnP module can still improve the estimation. Here, we consider two depth reconstruction methods [1][5] that take sparse depth only as the input and the other two algorithms [2][3] that input both the sparse depth and RGB image.

- **Uhrig *et al.*** [1] design a sparse convolutional operation to reconstruct dense depth from sparse LiDAR data. We choose the output of the first sparse convolutional layer as the intermediate feature map $z$ for sufficiently large influential field;
- **Chodosh *et al.*** [5] develop a feed-forward technique to obtain an optimal $x$ such that the predicted depth map $f(x)$ is consistent with the given sparse depth $D_s$. There are only three layers in their network $f$ and we select the optimal $x$ as the intermediate feature map $z$;

- **Ma *et al.*** [2] adopt a ResNet-based encoder-decoder network to predict dense depth map from the concatenation of an RGB image and a sparse depth map. We select the bottleneck feature map to be the intermediate representation $z$ for back-propagation;
- **Jaritz *et al.*** [3] introduce late fusion for sparse depth data and predict dense depth map through a skip-connected encoder-decoder network. We empirically found that the output is not sensitive to the bottleneck feature map (32x) and thus we select a higher resolution feature map (8x) as the intermediate representation $z$.

**Implementation Details.** In our PnP module, the learning rate $\alpha$, the number of iterations, and the loss function are set as 0.01, 5, and $L_1$ loss respectively for all experiments if not particularly mentioned. To evaluate the improvement introduced by our method with fair comparisons, all the baselines follow the original settings (e.g., number of sparse samples and sampling strategy) as mentioned in their original papers. For instance, as some baselines adopt uniform sampling to obtain sparse depth, we then use the same random seed as theirs to create sparse data for our PnP module.

### B. Datasets and Metrics

**NYU-Depth-v2 Dataset.** The NYU-Depth-v2 dataset [9] consists of RGB and depth images captured from a Microsoft Kinect in 464 different indoor scenes. The official split contains 249 training and 215 testing scenes. For benchmarking, we compute error measures on the commonly used test subset of 654 images, following [2][14]. The depth images in the dataset are reprojected to the RGB image coordinate and then is in-painted with an official toolbox as the ground truth. As used in [11][14], the original frames of size $640 \times 480$ is down-sampled by half and center-cropped to $304 \times 228$.

**KITTI Dataset.** Two KITTI datasets are used in this paper: 1) The *Odometry* dataset [10] contains 22 sequences with camera and LiDAR measurements. As described in [2][6], half of them is used for training and a random subset of 3200 images from the other half is used for evaluation. Following the setting in [2][3], the original image is bottom-cropped to $912 \times 228$ since there is no depth at the top area; 2) The *Depth Completion* dataset [1] focuses on recovering dense depth from a single LiDAR sensor. It contains over 93 thousands depth maps with corresponding raw LiDAR scans and RGB images, where the dense ground truth is collected by accumulating LiDAR measurements from nearby frames in the video sequences with outliers removed [1].

**Evaluation Metrics.** We adopt standard metrics in depth estimation. Given the ground truth depth and predicted depth, we use the following metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Relative error (MRE), and $\delta$ Threshold [11]. Please note that only for $\delta$ Threshold, it is larger the better.

### C. Overall Performance

In Table I and Table II, we demonstrate that the performance of both depth estimation and reconstruction baselines are improved by using the proposed PnP module without the

TABLE I: Evaluation on NYU-Depth-v2. Uniform sampling is adopted to obtain sparse samples following [2].

| Problem | Method | #Samples | %Samples | RMSE (m) ↓ | MAE (m) ↓ | MRE ↓ | $\delta_1$ ↑ | $\delta_2$ ↑ | $\delta_3$ ↑ |
|---|---|---|---|---|---|---|---|---|---|
| rgb | Eigen *et al.* [11] | - | - | 0.8933 | 0.7076 | 0.2862 | 52.1 | 82.3 | 93.8 |
| | Our + Eigen | 100 | 0.145 | 0.5021(+43.8%) | 0.3765(+46.8%) | 0.1497 | 80.5 | 94.8 | 98.8 |
| | FCRN [14] | - | - | 0.5111 | 0.3665 | 0.1369 | 81.5 | 95.4 | 98.8 |
| | Our + FCRN | 100 | 0.145 | 0.3720(+27.2%) | 0.2438(+33.5%) | 0.0890 | 91.5 | 98.3 | 99.6 |
| sd | Uhrig *et al.* [1] | 3466 | 5 | 0.1154 | 0.0500 | 0.0177 | 99.3 | 99.9 | 99.9 |
| | Our + Uhrig | 3466 | 5 | 0.1050(+8.9%) | 0.0441(+11.8%) | 0.0162 | 99.4 | 99.9 | 99.9 |
| rgb + sd | Ma *et al.* [2] | 500 | 0.725 | 0.1982 | 0.1125 | 0.0398 | 97.9 | 99.6 | 99.9 |
| | Our + Ma | 500 | 0.725 | 0.1782(+10.1%) | 0.0087 (+22.7%) | 0.0315 | 98.2 | 99.7 | 99.9 |
| | Jaritz *et al.* [3] | 500 | 0.725 | 0.1712 | 0.0996 | 0.0343 | 98.7 | 99.7 | 99.9 |
| | Our + Jaritz | 500 | 0.725 | 0.1598(+6.7%) | 0.0817(+17.9%) | 0.0320 | 98.7 | 99.7 | 99.9 |

TABLE II: Evaluation on KITTI. † denotes the KITTI *Depth Completion* dataset and the others are evaluated on the KITTI *Odometry* dataset. Uniform sampling is adopted to obtain sparse samples on KITTI *Odometry* following [2].

| Problem | Method | #Samples | %Samples | RMSE (m) ↓ | MAE (m) ↓ | MRE ↓ | $\delta_1$ ↑ | $\delta_2$ ↑ | $\delta_3$ ↑ |
|---|---|---|---|---|---|---|---|---|---|
| rgb | Eigen *et al.* [11] | - | - | 6.6128 | 3.9234 | 0.2249 | 58.9 | 86.4 | 94.7 |
| | Our + Eigen | 500 | 0.5 | 5.1395(+22.3%) | 2.8633(+27.0%) | 0.1790 | 75.1 | 91.7 | 97.0 |
| | Zhou *et al.* [24] | - | - | 7.2722 | 4.1411 | 0.2895 | 63.6 | 85.7 | 93.3 |
| | Our + Zhou | 500 | 0.1 | 6.0726(+16.5%) | 3.4042(+17.8%) | 0.2269 | 71.0 | 89.6 | 95.2 |
| sd | Uhrig *et al.* [1] | LiDAR data (∼ 5%)† | | 1.6616 | 0.5188 | 0.0311 | 39.0 | 66.9 | 81.6 |
| | Our + Uhrig | LiDAR data (∼ 5%)† | | 1.6244(+2.2%) | 0.4533(+12.6%) | 0.0275 | 50.7 | 75.7 | 86.1 |
| | Chodosh *et al.* [5] | LiDAR data (∼ 5%)† | | 1.3859 | 0.4813 | 0.0276 | 34.7 | 64.4 | 80.0 |
| | Our + Chodosh | LiDAR data (∼ 5%)† | | 1.3602(+1.9%) | 0.4192(+12.9%) | 0.0233 | 52.7 | 75.8 | 85.1 |
| rgb + sd | Ma *et al.* [2] | 500 | 0.1 | 3.0272 | 1.1879 | 0.0703 | 94.8 | 97.9 | 99.0 |
| | Our + Ma | 500 | 0.1 | 2.9754(+1.7%) | 1.0241(+13.79%) | 0.0579 | 94.9 | 98.0 | 99.0 |
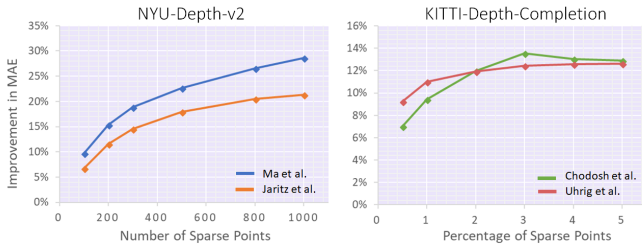


Fig. 3: Improvement with different numbers of sparse points.

need of model re-training on the NYU and KITTI datasets. In each table, we group the methods according to the input type of a model, denoted as RGB image only (**rgb**), sparse depth only (**sd**), and RGB image together with sparse depth (**rgb+sd**). We follow the settings, e.g., number/percentage of sparse samples, as used in each method. Among the evaluated methods, [2][11][24] are tested on the KITTI *Odometry* dataset while the others are evaluated on the KITTI *Depth Completion* dataset.

For the **rgb** case, since both methods [11][14] do not use the sparse depth as the guidance in the first place, the performance are thus improved significantly, e.g., larger than 25% on NYU-Depth-v2 in Table I. In addition, our method can be generalized to the unsupervised learning-based model [24], which achieves more than 15% improvement on KITTI in Table II. In this setting, we show the benefit of combining the RGB-based depth prediction with the proposed PnP module. For the other cases (**sd** and **rgb + sd**), although the sparse depth is already utilized in these baseline methods, we show that with the proposed PnP module, the performance can be further improved under various sparsity patterns. We also note that, due to the fact that the NYU and KITTI datasets have different scales of sparse data, we observe relatively larger improvement on NYU than that on KITTI. Some qualitative results are presented in Fig. 6.

## D. Sparsity Patterns

**Number/Percentage of Sparse Points.** In real-world tasks, different sensors produce various numbers of sparse points. In Fig. 3, we show such a study by applying the proposed PnP module to evaluated methods. On NYU-Depth-v2, we consider [2] and [3] as they are designed for using the sparse depth with sparse points, and we follow [2] to perform uniform sampling. For both baselines, our method provides consistent improvements as the number of sparse points increases. On KITTI *Depth Completion*, We follow [1] to sub-sample LiDAR data comparing two baselines [1][5]. Although the improvement degrades a bit for [5], it is of great interest to observe performance gain using LiDARs, where such measurements often contain missing data.

**Sparse Points from LiDARs.** In Table III, we further show that our method can be applied to practical applications, e.g., autonomous vehicles, which usually acquire the raw depth from LiDARs. We simulate four types of Velodyne LiDAR (i.e., **VLP-16**, **VLP-32C**, **HDL-32E**, and **HDL-64E**) which are distinct on their field of view (**FoV**) and vertical angular resolution (**vRes**), with additional random noise on rotation. We evaluate [2] on the SYNTHIA dataset [25] since it is the only outdoor dataset with dense ground truth that is required by LiDAR sampling. In Table III, we present the **MAE** without using our PnP module and its relative improvement **MAE+**. Among these four types, **VLP-32C** obtains the largest improvement since it has the largest FoV and number of sparse samples, followed by **HDL-32E** which has a similar FoV and lower vRes (lower number means higher resolution). Interesting, **HDL-32E** with fewer sparse points and less density (vRes) performs better than **HDL-64E**, suggesting that the spatial distribution of sparse points (FoV) is more important than the density of sparse points (vRes) for depth reconstruction.

TABLE III: Improvement with different types of LiDAR simulations on SYNTHIA. FoV and vRes are field of view and vertical resolution (lower number means higher resolution). MAE and MAE+ are the performance without PnP module and the relative improvement in MAE using our method, respectively.

| Source | %Samples | FoV | vRes | MAE ↓ | MAE+ ↑ |
|---|---|---|---|---|---|
| VLP-16 | 5.3 | 30 | 2.0 | 1.274 | 9.3% |
| VLP-32C | 28.8 | 40 | 0.3 | 1.075 | 11.8% |
| HDL-32E | 10.3 | 41 | 1.3 | 1.109 | 11.1% |
| HDL-64E | 18.7 | 27 | 0.4 | 1.236 | 8.9% |
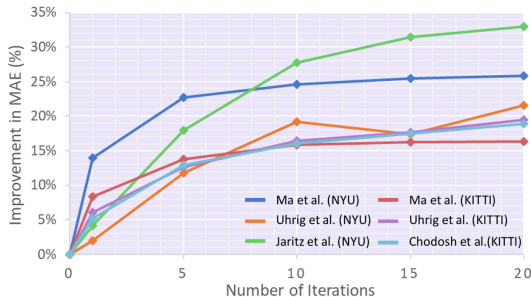


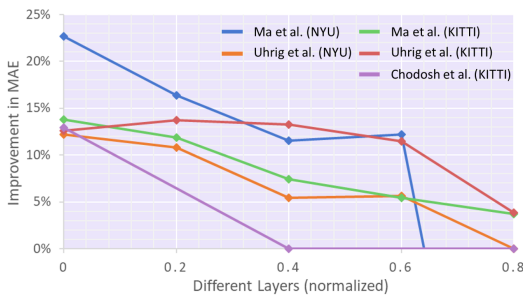Fig. 4: Improvement under multiple iterations.



Fig. 5: Improvement under different layers during backpropagation. The layer number is normalized for better illustration.

### E. Ablation Study

**Effects of Iterative Optimization.** In Fig. 4, we demonstrate that our method is capable of gradually propagating the information of sparse depth over multiple iterations, where the improvement are increased as the number of iterations grows. Nevertheless, since more iterations stand for longer inference time and the improvement gradually saturates when the number of iterations is sufficiently large, we would not prefer to have extremely many iterations in practical usage considering the trade-off between runtime and accuracy.

**Back-propagation to Different Layers.** Here, we investigate different choices of intermediate feature map $z$. In this experiment, we fix the number of iterations and learning rate $\alpha$ as 5 and 0.01, respectively. In Fig. 5, the horizontal axis starts from shallow to deep layers and is normalized since the number of layers in each model is different. For instance, we set the first layer of the decoder in Ma *et al.* as 0. For Uhrig *et al.* and Chodosh *et al.*, which are not the encoder-decoder architecture and only contain 6 and 3 layers, respectively, we start from the first layer of the entire network as the intermediate feature $z$ and proceed.

In Fig. 5, choosing $z$ further from the output layer (i.e., smaller number in the horizontal axis) yields better results since it produces a larger influential field (as described in Sec. III-D) and this observation holds for both NYU and
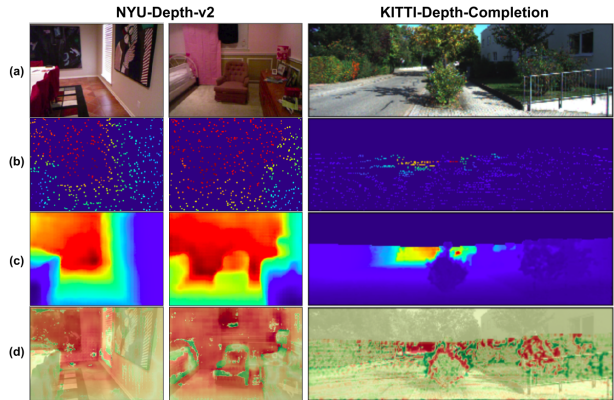


Fig. 6: Qualitative results on NYU and KITTI. Row (a), (b), (c), (d) are RGB image, sparse depth, final prediction with our PnP module, and corresponding improvement map. In (d), the red region indicates improvement and the green one indicates degradation.

TABLE IV: Inference time of applying PnP to different models.

| Dataset | NYU | | KITTI | |
|---|---|---|---|---|
| Model | Ma *et al.* | Jaritz *et al.* | Uhrig *et al.* | Chodosh *et al.* |
| Original | 0.0046 | 0.0484 | 0.0311 | 0.0320 |
| w/ PnP | 0.0233 | 0.1593 | 0.1614 | 0.0447 |

KITTI. Furthermore, predictions could be more sensitive when updating the feature map that is closer to the output (e.g., Ma *et al.* on NYU). In addition, it is worth mentioning that the improvement in Chodosh et al. [5] saturates quickly. The reason is that this method already shares a similarity to ours in a sense that both aim to optimize a latent representation that is consistent to the given sparse map.

### F. Computational Time

In Table IV, we show the computational time applying our PnP module to different models on both NYU and KITTI. We fix the number of iterations as 5. The inference time mainly depends on the model architecture and the selection of intermediate representation $z$. For instance, performing our PnP module on Chodosh *et al.* brings only slightly computational time overhead, because the model only contains three layers and $z$ is chosen much further from the output layer.

## V. CONCLUSIONS

In this paper, we propose a plug-and-play (PnP) module for improving depth prediction given arbitrary patterns of sparse depths, that can be applied to any differentiable model without re-training. We validate our PnP module using several state-of-the-art depth prediction methods with various input types on indoor (NYU) and outdoor (KITTI) datasets, achieving consistent improvements. In addition, we synthesize various types of LiDARs and verify the general applicability of our method. Finally, comprehensive analysis is carried out to provide a simple rule of thumb for applying our PnP module.

## REFERENCES

[1] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017. 1, 2, 4, 5

[2] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *International Conference on Robotics and Automation (ICRA)*, 2018. 1, 2, 4, 5

[3] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with cnns: Depth completion and semantic segmentation," *arXiv preprint arXiv:1808.00769*, 2018. 1, 2, 4, 5

[4] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from rgb and sparse sensing," *arXiv preprint arXiv:1804.02771*, 2018. 1

[5] N. Chodosh, C. Wang, and S. Lucey, "Deep convolutional compressed sensing for lidar depth completion," *arXiv preprint arXiv:1803.08949*, 2018. 1, 2, 4, 5, 6

[6] R. Y. Xinjing Cheng, Peng Wang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3, 4

[7] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015. 1

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016. 1, 2, 3

[9] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision (ECCV)*, 2012. 1, 4

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 4

[11] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374. 1, 4, 5

[12] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658. 1

[13] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170. 2

[14] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248. 2, 4, 5

[15] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011. 2

[16] S. Hawe, M. Kleinsteuber, and K. Diepold, "Dense disparity maps from sparse disparity measurements," in *13th International Conference on Computer Vision*, 2011. 2

[17] L.-K. Liu, S. H. Chan, and T. Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1983–1996, 2015. 2

[18] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 993–1000. 2

[19] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu, "Parse geometry from a line: Monocular depth estimation with partial laser observation," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5059–5066. 2

[20] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[21] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 2

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 1986. 2

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2014. 3

[24] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017. 4, 5

[25] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 5