Unsupervised Depth Completion with Calibrated Backprojection Layers

Alex Wong
UCLA Vision Lab
alexw@cs.ucla.edu

Stefano Soatto
UCLA Vision Lab
soatto@cs.ucla.edu

Abstract

We propose a deep neural network architecture to infer dense depth from an image and a sparse point cloud. It is trained using a video stream and corresponding synchronized sparse point cloud, as obtained from a LIDAR or other range sensor, along with the intrinsic calibration parameters of the camera. At inference time, the calibration of the camera, which can be different than the one used for training, is fed as an input to the network along with the sparse point cloud and a single image. A Calibrated Backprojection Layer backprojects each pixel in the image to three-dimensional space using the calibration matrix and a depth feature descriptor. The resulting 3D positional encoding is concatenated with the image descriptor and the previous layer output to yield the input to the next layer of the encoder. A decoder, exploiting skip-connections, produces a dense depth map. The resulting Calibrated Backprojection Network, or KBNet, is trained without supervision by minimizing the photometric reprojection error. KBNet imputes missing depth value based on the training set, rather than on generic regularization. We test KBNet on public depth completion benchmarks, where it outperforms the state of the art by 30% indoor and 8% outdoor when the same camera is used for training and testing. When the test camera is different, the improvement reaches 62%.

1. Introduction

Sensor platforms designed to enable interaction with physical space often include optical as well as range sensors. From cars to phones, cameras are paired with active sensors such as LIDARs, Sonars or Radars. We address the case of a single camera and a single sensor that returns the three-dimensional (3D) coordinates of a number of points far fewer than the number of pixels in the RGB image. The range sensor alone provides a sparse estimate of the Euclidean geometry of the surrounding environment, but often insufficient for planning in applications such as autonomous navigation or manipulation. We wish to leverage the complementarity of the optical and range modalities to provide a *dense* depth map, whereby a range value[1] is associated to every pixel in the image (in the millions) as opposed to just the LIDAR or radar returns (in the thousands).

Depth completion consists of mapping a single RGB image and a sparse 3D point cloud onto a dense depth map, which requires inferring a depth value where missing. This can be done by means of regularization, or inductively using previously observed data for scenes other than the present. We assume we have available a *training set* consisting of monocular videos, corresponding sparse 3D point cloud, and intrinsic calibration matrix of the camera used for capture,[2] but without any manual annotation or ground-truth dense depth i.e. *unsupervised*.

Our goal is to use the training set to learn a function that, for a scene and camera not used for training, can map a sparse point cloud registered to an image, along with the matrix of intrinsic calibration parameters of the camera, and produce a dense depth map associated with the test image.

We propose a novel deep neural network architecture that leverages a *sparse-to-dense* (S2D) module and *calibrated backprojection* (KB) layers. S2D is comprised of various pooling and convolutional layers to yield a dense representation of the sparse points. A KB layer then maps camera intrinsics, input image, and current depth estimate onto the 3D scene. This can be thought of as a form of *spatial (Euclidean) positional encoding* of the image. Unlike previous architectures, camera intrinsics are an *input* to our model, as opposed to a fixed set of parameters in the training loss. This allows us more flexibility to transfer the trained model to sensor platforms other than that used for training.

Our network is trained unsupervised with the standard Photometric Euclidean Reprojection Loss (PERL) i.e. the absolute difference between a reconstructed image and the actual image measured at a time instant. We also penalize the reconstruction error of the input sparse points and Total

[1] The depth associated with the pixel is the Euclidean distance of the closest point in the scene along the projection ray through that pixel and the optical center. We assume the sensors to be calibrated and synchronized, and in particular the intrinsic calibration matrix of the camera is known so that pixel coordinates can be converted to Euclidean 3D coordinates.

[2] Typically, range and optical sensors are calibrated mechanically and pre-registered, so extrinsic calibration is not needed.

Variation of the estimated depth map, a standard sparsity-inducing prior to reduce the penalty for large depth changes at adjacent pixels that straddle occluding boundaries. At test time, no video is necessary and inference is performed on each image and sparse point cloud independently.

These innovations allow us to improve the baseline [41] and state of the art [39] by an average of 13% and 8%, respectively, on outdoors (KITTI [35]), and 51.7% and 30.5% indoors (VOID [41]), when calibration is the same for training and testing. When different calibrations are used, our method generalizes better than the baseline and state of the art by 83% and 62%, respectively, in relative error. All of this is achieved with a smaller computational footprint thanks to the inductive bias induced by KB layers, which allows us to use a smaller network than current methods.

1.1. Related Work and Contributions

Depth completion is a form of imputation, which requires regularization that hinges the assumption that *"nearby points"* should be assigned "similar" (depth) values. Methods differ in the choice of topology i.e. what points should be considered "nearby," and how to combine the values of such points to impute the missing depth value.

Generic Image-Based Regularization. In image topology, nearby points correspond to adjacent pixels. This is not a good choice, for their depths can be arbitrarily different at occluding boundaries. In image segmentation, the RGB values are used to define a topology to partition the image domain into connected regions of nearby points, putatively corresponding to "objects." The topology induced by (color) values can be exploited by minimizing Total Variation (TV [31] and "Color TV" [1]) while trying to reproduce the image itself. We adopt TV as a generic regularizer since the statistics of natural range images are very similar to that of natural (intensity) images [25], whereby the gradient distribution is highly kurtotic, corresponding to homogeneous smooth regions separated by sharp boundaries.

Data-driven Regularization. "Closeness" among pixels can be defined not just within the same image, but across different images in the training set. In this case, the regularity criterion is not explicit, but implicit in the inductive bias used for training. Before training starts, the bias is encoded in the training loss (L^1 prediction error), the generic regularizers (TV), the training set, and the choice of architecture and optimization. After training is completed, all these biases are burnished in the parameters (weights) of the trained model, which inform the prediction of our depth map and therefore act as a regularizing mechanism.

Among data-driven methods for depth completion, many are **supervised**. Early works cast depth completion as compressive sensing [4] and as morphological operators [5]. Recent works focused on network operations [7, 15] and architectures [2, 22, 35, 43] to effectively deal with the sparse inputs. [22] proposed an early fusion architecture while [16, 43] used late fusion to process each data modality separately. [15] performed joint concatenation and convolution to upsample the sparse depth. [2] proposed a 2D-3D fusion network while [20] used a cascade hourglass network. [3] used a convolutional spatial propagation network and [26] leveraged non-local spatial propagation. [6, 7] used valid sparse depth locations as confidence. Whereas, [36] learned confidence maps, and [28, 29, 42, 44] used surface normals for guidance. Like us, [24, 32, 45] proposed light-weight networks that can be deployed onto SLAM/VIO systems.

All of these methods require ground truth for training, which is often unavailable and, when available, prohibitively expensive [35]. Hence, these methods are limited to offline training. But even if ground truth were available online, most of these methods employ complex architectures with many layers and parameters, e.g. 25.84M for [26], 53.4M [28], and 28.99M [42], and thus are not suitable for learning online. Instead, we propose to learn dense depth from the virtually limitless amount of un-annotated images and sparse point clouds via a predictive cross-modal validation criterion. Our proposed architecture only uses 6.9M parameters and our choice of supervision allows us to continuously learn even after the system is deployed.

Unsupervised/Self-supervised depth completion assumes stereo images or monocular videos to be available during training. Both stereo [33, 43] and monocular [22, 39, 40, 41] training paradigms leverage sparse depth reconstruction and photometric reprojection error as a training signal by minimizing photometric discrepancies between the input image and its reconstruction from other views. [22] used Perspective-n-Point [19] and RANSAC [10] to align consecutive video frames. However, [22] does not generalize well to indoor scenes with many textureless surfaces. [43] learned a depth prior conditioned on the image by pretraining a separate network on ground truth from an additional dataset. As mentioned earlier, this is not scalable; also, using a network trained on a specific domain (e.g. outdoors) as supervision will not generalize (e.g. indoors). Unlike [43], our method does not require ground truth and is not limited to a specific domain. [21, 39] leverage additional synthetic datasets, which require dealing with sim-to-real; our method is able to achieve the state-of-the-art *without* needing access to additional data.

The challenge of depth completion is precisely the sparsity, which renders convolutions ineffective as the activations of early layers tend to be zeros as well. To obtain a denser representation, early layers must propagate (or densify) the signal. As a result, [22, 33, 43] employed very deep networks with many layers and parameters in order to learn the map from sparse depth and image to dense depth. To handle this problem, [41] approximated the scene with a hand-crafted mesh, but it is not differentiable and prone to

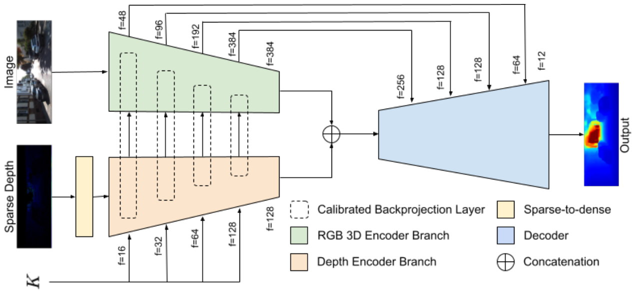

Figure 1: *KBNet architecture.* Our architecture takes, as input, an RGB image, the corresponding sparse depth map and camera calibration matrix. We first learn a dense representation of the sparse points with our sparse-to-dense module. The result of which and the calibration matrix are used for calibrated lifting, which allows us to backproject image features onto 3D space (akin to spatial positional encodings) to yield a RGB 3D representation. Our network is very light-weight and fast, yet achieves the state of the art.

errors in regions with very few points or complex structures. [39] proposed spatial pyramid pooling (SPP), but their max pooling layers decimated details on closer objects. Instead, we propose a fully differentiable sparse-to-dense module that learns the trade-off between density and detail to retain both near and far structures.

Our work goes counter to the trend of forgoing inductive bias, i.e. learning everything with generic architectures like Transformers [37], including what we already know such as basic Euclidean geometry. Our model has a strong inductive bias in our calibrated backprojection layer, which incorporates the calibration matrix directly into the architecture to yield an RGB representation lifted into scene topology via 3D positional encoding. This may seem futile as we could just add intrinsics to the long list of parameters to be learned [11]. However, unlike semantic retrieval, spatial inference requires *identifiability*: There is *one* true scene in front of us, and unless information about calibration is available and properly exploited, inference yields one of infinitely many depth maps that are equally good at predicting the next frame in the training set. Since there is no supervision, calibration mediates the relation between the prediction error and the *true* depth. Because existing methods use calibration in the computation of the loss, which the intrinsics are encoded in the weights, hampering transferability. In our architecture, calibration is an input, which can be changed at inference time. While one could pre-process the images to a canonical calibration, this introduces latency, cost and artifacts that can affect the reconstruction quality. We note that [12, 30] proposed backprojection as a layer and [8] used calibration as input, but we are the first to consider an RGB 3D representation for depth completion.

Our contributions include (a) a sparse-to-dense module that learns a dense representation of the sparse point cloud, (b) an unsupervised depth completion method that takes calibration information as input to the model, and (c) incorporates it directly in the architecture through a novel *calibrated backprojection* module, which represents spatial positional encoding that is transferred laterally across different branches of the encoder. The resulting inductive bias helps select, among all depth, maps compatible with the prediction loss, those that result in a Euclidean (calibrated) reconstruction. The strong inductive bias allows us to (d) reduce the computational footprint, increase generalization and achieve performance beyond the state of the art despite having fewer parameters.

2. Method Formulation

Our goal is to recover a 3D scene from an RGB image $I : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}^3_+$ and the associated sparse point cloud projected onto the image plane $z : \Omega_z \subset \Omega \mapsto \mathbb{R}_+$, without access to ground-truth depth annotations.

We propose a sparse-to-dense module (Fig. 2) f_ω, parameterized by ω, that captures local and global structure of the sparse inputs by combining min and max pooling at different scales. The result is a dense or quasi-dense depth representation $f_\omega(z)$, depending on the sparsity of the input, which frees the rest of network to utilize its early convolutional layers to learn scene structure rather than to densify the input – making the overall architecture more efficient.

The sparse-to-dense module (Sec. 2.1) is part of an overall encoder-decoder architecture f_θ, parameterized by θ, called KBNet (Sec. 2.2), that includes a Calibrated Backprojection layer which explicitly backprojects pixels onto 3D space using intrinsic camera calibration and depth encoding from f_ω . Unlike previous works [22, 33, 39, 41, 43] that encode depth and image in two separate branches, we leverage camera calibration and our depth encoding to lift the image representation to 3D and passed it to the decoder via skip connections. KBNet (Fig. 1) produces dense depth $\hat{d} := f_\theta(f_\omega(z), I, K)$, where $K \in \mathbb{R}^{3\times3}$ is the upper-triangular matrix of intrinsic calibration parameters. To train our model, we use monocular videos to compose a loss function from temporally adjacent frames (Sec. 2.3).

2.1. Sparse-to-Dense Module (S2D)

Our S2D module f_ω (Fig. 2) performs multi-scale densification on the input sparse depth map z using a series of min and max pooling layers with various kernel sizes, which are chosen based on the sparsity of the point cloud e.g. from LIDAR returns or sparse points tracked by VIO [9] (see Supp. Mat. for kernel sizes). The outputs of the pooling layers are concatenated and fed into three 1×1 convolutions to learn the trade-offs between pooling types and kernel sizes. The result of which is fused with the z via a 3×3 convolutional layer, yielding a dense or quasi-dense depth representation that is fed to the rest of the network f_θ.

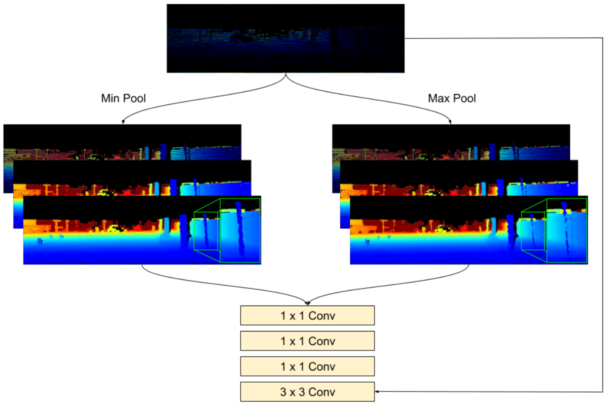

Figure 2: *Sparse-to-dense module.* We perform min and max pooling with various kernel sizes to produce a dense representation. There exists trade-offs between density and detail (large vs. small kernel sizes) and preservation of near and far structures (min vs. max pooling, as highlighted in green). We balance these trade-offs with 1×1 convolutions and fuse the result with the input via a 3×3 convolution.

Because the depth inputs are sparse, we design our min pooling layers to avoid pooling zeros or invalid (negative) depth values. We set all values $z(x)$ less than zero to be infinity for $x \in \Omega$:

$$z'(x) = \begin{cases} z(x) & \text{if } z(x) > 0 \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

z' is fed to a min pooling layer with $k \times k$ kernel size,

$$p = \texttt{minpool}(z', k). \tag{2}$$

Finally, for all x, any infinity values pooled due to large empty regions are set to zero:

$$p_{min}(x) = \begin{cases} p(x) & \text{if } p(x) \neq \infty \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Our approach involves two main trade-offs: (i) density versus detail and (ii) preservation of near versus far structures.

Density versus details. For the purpose of densification, one may perform pooling with large kernel sizes, but it comes at the expense of details of local structures. In contrast, pooling with small kernel sizes in an attempt to retain detail will result in very few neuron activations, which hinders learning. Hence, to retain local details while obtaining a dense representation, we propose to perform pooling with both small and large kernel sizes.

Near versus far. When pooled solely with max pooling, farther structures are preserved, but details of the closer ones are decimated as the kernel size grows larger. For instance in Fig. 2, thin structures close to the camera i.e. the highlighted pole "disappears" due to large max pooling kernel size. On the other hand, when only using min pooling, the closer structures become more prominent, but in turn, the farther regions are corrupted. Moreover, in cluttered scenes, min pooling causes adjacent structures to "bleed" into each other. Hence, to preserve close and far structures, we employ both min and max pooling layers.

To optimize for both trade-offs, we concatenate the outputs of min and max pooling together and feed them into 1×1 convolutional layers. Finally, we use a 3×3 convolution to fuse the multi-scale depth features back into the original sparse depth via a residual connection, yielding a dense representation $f_\omega(z)$ to be fed to f_θ.

We note that our S2D bares some resemblance to spatial pyramid pooling (SPP) [14]; however, SPP was designed to ensure the same size feature maps are maintained when different size of inputs. It is also intended to operate on dense inputs. While [39] also proposed an SPP for sparse inputs, its use of max pooling decimated details for nearby structures. Neither are substitutes for our S2D module.

2.2. KBNet Architecture

Motivation. Unsupervised methods [22, 39, 40, 41] use the photometric reprojection error ℓ_{perl} as a training signal. The input image I_t is reconstructed from temporally adjacent frames I_τ for $\tau \in T \doteq \{t-1, t+1\}$ to yield $\hat{I}_\tau$,

$$\hat{I}_\tau(x, \hat{d}) = I_\tau\big(\pi g_{\tau t} K^{-1} \bar{x} \hat{d}(x)\big), \tag{4}$$

and the per pixel photometric reprojection error is measured by $\ell_{perl} = |\hat{I}_\tau(x, \hat{d}) - I_t(x)|$. Here $\bar{x} = [x^\top, 1]^\top$ are the homogeneous coordinates of $x \in \Omega$. Using the notation in [23], $g_{\tau t} \in SE(3)$ is the relative pose (rotation and translation) of the camera from time t to τ, K denotes the calibration matrix, and π is a canonical perspective projection.

Inferring Euclidean structure and motion in the absence of calibration information is notoriously difficult and dependent on conditions rarely satisfied in ordinary training videos, such as rotation around three independent axes [23]. Minimizing any form of ℓ_{perl} forces the network to implicitly learn the calibration matrix K, as all prior work does. As pretrained models are commonly deployed on sensor platforms different than those used during training, this hinders generalization as the network becomes overfitted to the camera used for to collect training data. In contrast, our network, KBNet, takes it as input; this allows us to use different calibrations in training and test, which significantly improves generalization (Table 5).

Calibrated Backprojection Layers take, as input, the depth and RGB image encodings, and the camera calibration matrix K and output not only the corresponding encodings of the depth map and of the RGB image, but also an encoding of the RGB image backprojected onto 3D space. Once we have formed this RGB 3D representation, it is fed as input to subsequent Calibrated Backprojection (KB) layers and as skip connection to the decoder and once we have form this representation (Fig. 3).

To realize a KB layer, first, we use the calibration matrix to lift the coordinates of each pixel $x \in \Omega$ to three dimensional space $x \rightarrow K^{-1}\bar{x}$. Then, the feature map of the depth encoder $\phi(x) \in \mathbb{R}^M$, with M ranging from 16 in the first layer to 128 in the last one, is collapsed to a scalar by a trainable projection or "compression" module q, $d(x) = q^\top \phi(x)$. The imputed depth $d(x)$ is used to backproject the lifted coordinate $\bar{x}$ to yield a 3D positional encoding for each pixel $x_{3D} = K^{-1}\bar{x}d(x)$.

Here $\Omega \subset \mathbb{R}^2$ is discretized into a lattice of $H \times W$ pixels in the first layer, corresponding to the resolution of the original image, that decreases by a factor of 2 in each subsequent layer until the 5-th or last layer at $H/32 \times W/32$. Hence, the intrinsics parameters, focal lengths and principal point, must also be scaled by the same factor according to the resolution reduction in each layer.

The 3D positional encoding is concatenated with the image encoding $\psi(x) \in \mathbb{R}^N$, and, if available, the output of the previous KB layer $\psi_{3D}(x) \in \mathbb{R}^N$ where N ranges from 48 in the first layer to 386 in the last. This is fused together by a 1×1 convolution to yield the output RGB 3D encoding. This encoding is fed to the next layer and also replaces the typical RGB skip connection to the decoder. Finally, the output depth and image encodings of the KB layer are produced by convolving separate 3×3 kernels. After which, both are also passed to the next layer as input.

In addition to benefits of generalization (Table 5), KB layers also produce depth estimates that better respect object boundaries. Because each layer encodes "closeness" based on the scene topology via 3D positional encoding rather than the 2D image topology (as in previous works), adjacent pixels in the image that are often confused to be close are now well separated (Fig. 4) and hence distinct adjacent objects are better delineated and points belonging to the same surface are better regularized. This reduces the common bleed effect observed when a depth map is backprojected to a point cloud in 3D. Moreover, by instilling 3D structure as an architectural inductive bias, we enable a faster and slimmer network with fewer layers and parameters to achieve better performance (see Table 2, 4).

We note that our S2D module complements our KB layers as it provides us with dense or quasi-dense depth representation. Without it, we are left with sparse geometry, which limits the potential performance gain. Yet, as demonstrated in Table 3, there are still benefits to using calibrated backprojection with a sparse representation.

2.3. Loss Function

Similar to previous works [22, 39, 41], our loss function is the linear combination of three terms:

$$\mathcal{L} = w_{ph}\ell_{ph} + w_{sz}\ell_{sz} + w_{sm}\ell_{sm} \tag{5}$$

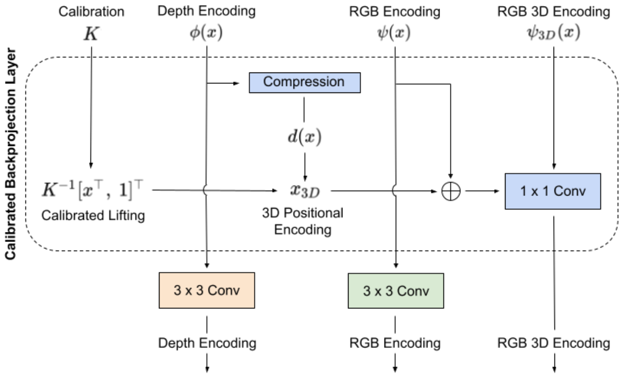

Figure 3: *Calibrated Backprojection (KB) Layer.* The standard depth and color image encoding layers [41] are combined using the calibration matrix as additional input. Calibration is used to lift pixel coordinates to three dimensions, which are backprojected by a compressed depth descriptor into a 3D positional encoding. The result is concatenated with the image encoding and the output of the previous KB layer, and fused with a 1×1 convolution. This yields an RGB 3D representation, which is used as a skip connection to the decoder and input to subsequent layers.

where ℓ_{ph} denotes photometric consistency, ℓ_{sz} sparse depth consistency, and ℓ_{sm} local smoothness. Each term is weighted by their associated w (see Sec. 3.1).

Photometric Consistency. As mentioned in Sec. 2.2, unsupervised methods leverage photometric reprojection error as a supervisory signal by reconstructing I_t from I_τ for $\tau \in T \doteq \{t-1, t+1\}$ via Eqn. 4. To accomplish this, one can obtain pose from a VIO [9] or employ a pose network to estimate the relative pose between I_t and I_τ (see full system diagram in Supp. Mat.). We note that pose is only needed for training and is not used at test time.

From the reconstructions, the photometric consistency loss measures the average photometric reprojection error using a combination of L^1 penalty and $\texttt{SSIM}$ [38]:

$$\begin{aligned}\ell_{ph} = \frac{1}{|\Omega|}\sum_{\tau \in T}\sum_{x \in \Omega} & w_{co}|\hat{I}_\tau(x, \hat{d}) - I_t(x)| + \\ & w_{st}\big(1 - \texttt{SSIM}(\hat{I}_\tau(x, \hat{d}), I_t(x))\big),\end{aligned} \tag{6}$$

w_{co} and w_{st} are weights for each term and are discussed in Sec. 3.1. We note that if $g_{\tau t}$ is estimated via a pose network, instead of a VIO, it can be jointly learned with KBNet (Fig. 1) as a by product from minimizing Eqn. 6 and 7, and hence does not require any extra supervision.

Sparse Depth Consistency. Minimizing the reprojection error will reconstruct the scene structure up to an unknown scale. To ground the predictions to *metric* scale, we minimize the L^1 difference between our predictions $\hat{d}$ and

Metric	Definition
MAE	$\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert\hat{d}(x)-d_{gt}(x)\rvert$
RMSE	$\big(\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert\hat{d}(x)-d_{gt}(x)\rvert^2\big)^{1/2}$
iMAE	$\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert 1/\hat{d}(x)-1/d_{gt}(x)\rvert$
iRMSE	$\big(\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert 1/\hat{d}(x)-1/d_{gt}(x)\rvert^2\big)^{1/2}$

Table 1: *Error metrics.* d_{gt} denotes the ground-truth depth.

the sparse depth inputs over its domain (Ω_z):

$$\ell_{sz} = \frac{1}{|\Omega_z|}\sum_{x\in\Omega_z}|\hat{d}(x) - z(x)|. \quad (7)$$

Local Smoothness. We enforce local smoothness and connectivity over $\hat{d}$ by minimizing the L^1 penalty on its gradients in the $x-$ (∂_X) and $y-$ (∂_X) directions. We also weight each term using its respective image gradients, $\lambda_X = e^{-|\partial_X I_t(x)|}$ and $\lambda_Y = e^{-|\partial_Y I_t(x)|}$, to allow discontinuities along object boundaries:

$$\ell_{sm} = \frac{1}{|\Omega|}\sum_{x\in\Omega}\lambda_X(x)|\partial_X\hat{d}(x)| + \lambda_Y(x)|\partial_Y\hat{d}(x)|. \quad (8)$$

3. Experiments and Results

We evaluate our method on benchmark datasets, KITTI [35] for outdoors settings, and VOID [41] for indoors, using metrics describes in Table 1. We also demonstrate that our approach generalizes well to scenes captures by camera setup different than that used to collect the training set by training our model on VOID and testing it on NYUv2 [34].

3.1. Implementation Details

We implemented our method in PyTorch [27]. End-to-end inference takes 16ms per frame. We used Adam [17] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimize our network. Training on KITTI [35] takes 58 hours for 50 epochs, VOID [41] 16 hours for 15 epochs, and NYUv2 [34] 13 hours for 15 epochs on an Nvidia GTX 1080Ti GPU. We use a batch size of 8 with 768 × 320 crops for KITTI, 640 × 480 for VOID and 576 × 416 for NYUv2. For KITTI, we choose $w_{ph} = 1$, $w_{co} = 0.15$, $w_{st} = 0.95$, $w_{sz} = 0.6$, and $w_{sm} = 0.04$; for VOID and NYUv2, we set $w_{sz} = 2$ and $w_{sm} = 2$. For detailed learning rate schedule, augmentations and S2D kernel sizes used for each dataset, please see Supp. Mat.

3.2. Datasets

KITTI [35] provides ≈80,000 raw image frames and associated sparse depth maps. The sparse depth maps are the raw output from the Velodyne lidar sensor, each with a density of ≈5%. Ground-truth depth is obtained by accumulating 11 neighbouring raw lidar scans. Semi-dense depth is available for the lower 30% of the image space. We use the official 1,000 samples for validation and test on 1,000 designated samples (evaluated on their online test server).

Method	# Param	Time	MAE	RMSE	iMAE	iRMSE
SS-S2D [22]	27.8M	80ms	350.32	1299.85	1.57	4.07
IP-Basic [18]	0	11ms	302.60	1288.46	1.29	3.78
DFuseNet [33]	n/a	80ms	429.93	1206.66	1.79	3.62
DDP* [43]	18.8M	80ms	343.46	1263.19	1.32	3.58
VOICED [41]	9.7M	44ms	299.41	1169.97	1.20	3.56
AdaFrame [40]	6.4M	40ms	291.62	1125.67	1.16	3.32
SynthProj* [21]	2.6M	60ms	280.42	1095.26	1.19	3.53
ScaffNet* [39]	7.8M	32ms	280.76	1121.93	1.15	3.30
Ours	6.9M	16ms	**258.36**	**1068.07**	**1.03**	**3.01**

Table 2: *Quantitative results on the KITTI test set.* Our method outperforms all unsupervised methods across all metrics on the KITTI leaderboard. Compared to the the baseline [41], we improve by an average of 13% across all metrics while using 29% fewer parameters. * denotes methods that use additional synthetic data for training.

VOID [41] contains synchronized 640 × 480 RGB images and sparse depth maps of indoor (laboratories, classrooms) and outdoor (gardens) scenes. ≈ 1500 sparse depth points (covering ≈ 0.5% of the image) are the set of features tracked by XIVO [9], a VIO system. The ground-truth depth maps are dense and are acquired by active stereo. The entire dataset contains 56 sequences with challenging motion. Of the 56 sequences, 48 sequences (≈ 40,000) are designated for training and 8 for testing. The testing set contains 800 frames. We follow the evaluation protocol of [41] and cap the depths between 0.2 and 5 meters.

NYUv2 [34] consists of 372K synchronized 640 × 480 RGB images and depth maps for 464 indoors scenes (household, offices, commercial), captured with a Microsoft Kinect. The official split consisting in 249 training and 215 test scenes. For training, we evenly sample a subset of the training split to yield 46K frames. We use the official validation set of 795 images and test set of 654 images. Because there are no sparse depth maps provided, we sampled ≈ 1500 points from the depth map via Harris corner detector [13] to mimic the sparse depth produced by SLAM/VIO.

3.3. KITTI Depth Completion Benchmark

We compare our method against recent unsupervised depth completion methods on the KITTI test set in Table 2 (results taken from online leaderboard). Compared to the baseline [41], we improve by an average of 13% across metrics and by as much as 15.5% in iRMSE while reducing model size by 29%. Overall, we beat the best performing method [39] by an average of 8% and up to 10.4% on the iMAE metric with a 11.5% reduction in model size. We note that top methods [21, 39] use *additional* synthetic data for training; whereas, we do not. Also, for inference, our method takes 16ms per image (62 FPS), which is 2.75×

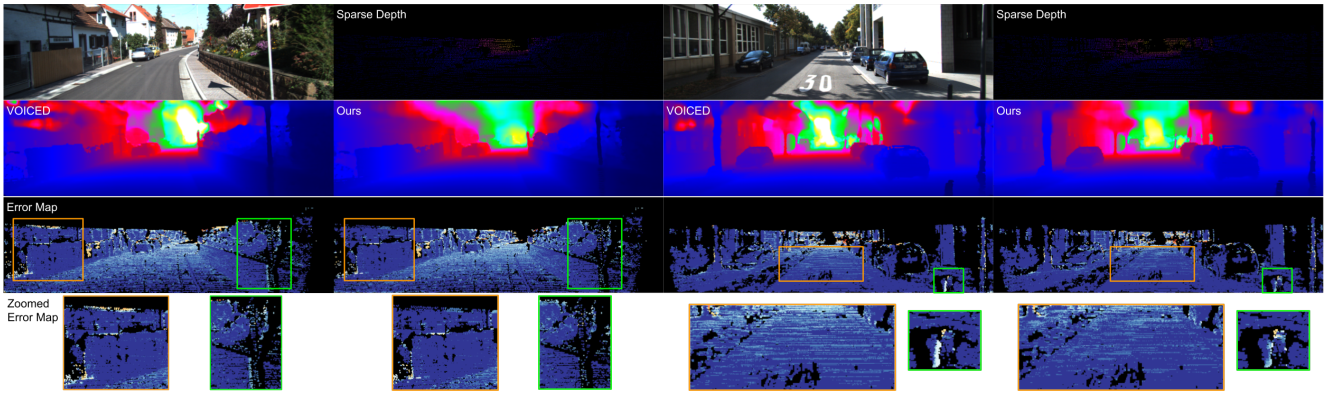

Figure 4: *Qualitative results on KITTI test set.* Head-to-head comparison against [41]. Thanks to our 3D positional encoding, our method performs well on regions where adjacent structures in 2D image space are far apart in the 3D scene e.g. street sign and wall (left panel, highlighted in green) and far region of the road (right panel, in orange).

Method	MAE	RMSE	iMAE	iRMSE
VOICED [41] w/o Scaffolding	347.14	1330.88	1.46	4.22
VOICED [41]	305.06	1239.06	1.21	3.71
Ours w/o S2D	287.76	1184.24	1.12	3.48
Ours w/o KB layers	285.97	1171.88	1.11	3.40
Ours w/ Scaffolding [41]	275.56	1183.57	1.08	3.39
Ours w/ SPP [14, 39]	273.08	1177.69	1.07	3.35
Ours	**262.01**	**1128.01**	**1.04**	**3.26**

Table 3: *Ablation study on KITTI validation set.* Without S2D (row 3), our performance degrade because our 3D positional features will only encode sparse geometry, but we still beat [41] in rows 1, 2 ("w/o Scaffolding" is [41] with sparse representation). We observe similar degradation without KB layers (row 6, replaced with VGG block used by [41]). Substituting our S2D with Scaffolding [41] or SPP [14, 39] also hurts performance (rows 7, 8).

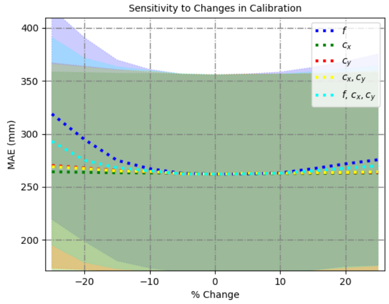

Figure 5: *Sensitivity to changes in calibration on KITTI.* Focal length and principal point are altered to test sensitivity to changes in intrinsics parameters. Our method is robust to change up to ≈ 10%. After which, performance degrades.

faster than [41][3] and 2× faster than the state of the art [39].

To show the improvements from our contributions, we show head-to-head qualitative comparisons against the baseline [41] in Fig. 4. Our method performs better in regions where depth discontinuities occur in image topology i.e. street sign and wall (left panel, highlighted in green) and far regions of the road (right panel, in orange). This is thanks to our KB layers, which imposes inductive bias (although points in 2D image topology are "close", they can be far in 3D scene topology) by incorporating the camera intrinsics into 3D positional encoding.

Table 3 shows an ablation study on the KITTI validation set. As mentioned in Sec. 2.2, our sparse-to-dense module (S2D) provides dense depth representation which in turn enables dense 3D topology in our calibrated backprojection (KB) layers. Hence, removing it ("w/o S2D) will hurt performance because it results in a *sparse* 3D positional encoding. Nonetheless, sparse geometry is still helpful as we outperform [41] in rows 1, 2. Similarly, replacing our KB ("w/o KB layers") with VGG blocks used by [41] also hurts performance as the model now lacks 3D spatial position. We show in rows 5 and 6 that one cannot simply substitute S2D with scaffolding [41] or SPP [14, 39].

In Fig. 5, we perform a sensitivity study of our model to calibration on the KITTI validation set. To this end, we altered the calibration by increasing or decreasing focal length (f) and/or principal point (c_x, c_y) and feed it as input. Our model is robust to changes up to $\approx$ 10%; after which, performance degrades. While changes in c_x, c_y have minor effects (which is scene-dependent), we observe a sharp decrease in performance when we decrease f by 20 to 25%. This is because, geometrically, decreasing f backprojects points to a larger field of view, distorting surfaces and sending points of the same surface far from each other. Increasing f conversely "packs" them tighter; this is okay for small increases, but for larger values, points will get "squashed together" – thus hurting performance. Also,

[3] The reported run time of [41] on the KITTI leaderboard did not include their scaffolding step; whereas, the number in Table 2 accounts for it.

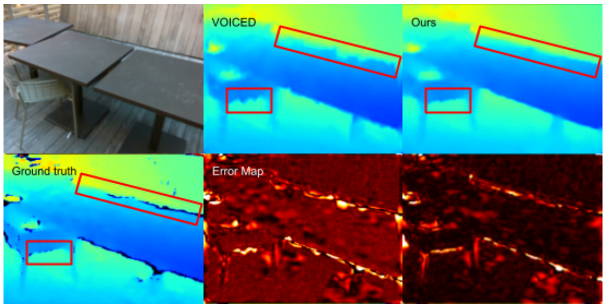

Figure 6: *Qualitative results on VOID test set.* Comparison against [41]. Our method performs better overall.

Method	# Param	Time	MAE	RMSE	iMAE	iRMSE
SS-S2D [22]	27.8M	59ms	178.85	243.84	80.12	107.69
DDP [43]	18.8M	54ms	151.86	222.36	74.59	112.36
VOICED [41]	9.7M	29ms	85.05	169.79	48.92	104.02
ScaffNet [39]	7.8M	25ms	59.53	119.14	35.72	68.36
Ours	6.9M	13ms	**39.80**	**95.86**	**21.16**	**49.72**

Table 4: *Quantitative results on VOID test set.* We outperform all competing methods across all metrics. Compared to [39], we improve by an average of 30.5%.

to quantify the effect of sparsity, we provide a sensitivity study on various density levels in Supp. Mat.

3.4. VOID Depth Completion Benchmark

As there exists many complex scene layouts for indoor scene, understanding 3D topology becomes even more important. This is shown in Table 4 where we outperform [22, 39, 41, 43] across all metrics to achieve the state of the art on VOID. A key comparison is between our method and [41]. Even though [41] creates a hand-crafted scaffolding of the scene to obtain a dense representation, because there are very few points, it is prone to error i.e. forming surfaces between discontinuous objects. This is where our method shines. By optimizing for the trade-off between density and detail, our S2D module learns to exploit the natural statistics of the dataset to obtain a dense representation more compatible with the scene. Also, our KB layers introduces 3D topology as an inductive bias, allowing the network to delineate points that are close in image topology, but are far in scene topology – culminating in 51.7% and 30.5% improvement over [41] and the state of the art [39], respectively.

In Table 5, we show that our method generalizes well to sensor platforms not used in the training set by training our method on VOID (captured on Intel RealSense) and testing it on NYUv2 (Microsoft Kinect). Similarly, we test models pretrained on VOID released by [39, 41] on NYUv2. We also train our method and [39, 41] from scratch on NYUv2 to show the paragon performance (rows 1, 3, 5). Rows 1 shows that [41] does not generalize well to NYUv2 where error increases by 56% (as much as 94% in iRMSE). While [39] does better, there is still a sharp decrease of 25.1% in performance. This is in part due to the change in sensor platform as well scene distribution in NYUv2. While we do not achieve paragon performance, our method generalizes better with a reasonable 9.5% increase in error – improving over [41] by 83% and [39] by 62% in relative error. For qualitative comparisons, please see Fig. 7 in Supp. Mat.

Method	Trained on	MAE	RMSE	iMAE	iRMSE
VOICED [41]	NYUv2	127.61	228.38	28.89	54.70
VOICED [41]	VOID	178.87	329.28	42.57	105.93
ScaffNet [39]	NYUv2	117.49	199.31	24.89	44.06
ScaffNet [39]	VOID	155.20	241.42	31.77	52.62
Ours	NYUv2	105.76	197.77	21.37	42.74
Ours	VOID	117.18	218.67	23.01	47.96

Table 5: *Quantitative results on the NYUv2 test set.* Column titled "Trained on" denotes the dataset each method is trained on. [39, 41] degrade much more than our method when tested on a dataset captured by a different sensor platform than the one used for gathering its training data.

4. Discussion

We present an approach to unsupervised depth completion that imposes strong inductive biases on Euclidean reconstruction in the architecture, rather than learning from data with a generic model such as a Transformer. This presents some advantages. First, it allows feeding calibration as an input, which means that we can easily use a model trained with a certain sensor platform with a different one at inference time. Second, the calibrated backprojection layer explicitly incorporates a basic geometric image formation model based on Euclidean transformations in 3D and central perspective projection onto 2D. This allows us to reduce the model size while still achieving the state of the art.

However, imposing strong inductive biases also presents some risks and limitations. First, if the camera is miscalibrated, inputing the wrong calibration can backfire, yielding distorted depth maps. Second, only a very rudimentary calibration model is used, so if a sensor platform has fancy optics such as omnidirectional lenses, one cannot use one of our pre-trained models but rather has to modify the core backprojection module. Third, even with these ad-hoc architectural choices, our model suffers the limitations of all imputations, which is that where there is insufficient evidence to constrain the solution, the regularizer dominates, which is a form of hallucination and can yield wildly wrong inferences. This would be mitigated by having an accurate measure of uncertainty associated to the depth map, this is an open problem well beyond our focus here.

Acknowledgements. This work was supported by ARL W911NF-20-1-0158 and ONR N00014-17-1-2072.

References

[1] Peter Blomgren and Tony F Chan. Color tv: total variation methods for restoration of vector-valued images. *IEEE transactions on image processing*, 7(3):304–309, 1998. 2

[2] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10023–10032, 2019. 2

[3] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10615–10622, 2020. 2

[4] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. In *Asian Conference on Computer Vision*, pages 499–513. Springer, 2018. 2

[5] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Learning morphological operators for depth completion. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 450–461. Springer, 2018. 2

[6] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12014–12023, 2020. 2

[7] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Propagating confidences through cnns for sparse data regression. In *Proceedings of British Machine Vision Conference (BMVC)*, 2018. 2

[8] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Cam-convs: Camera-aware multi-scale convolutions for single-view depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11826–11835, 2019. 3

[9] Xiaohan Fei, Alex Wong, and Stefano Soatto. Geo-supervised visual depth prediction. *IEEE Robotics and Automation Letters*, 4(2):1661–1668, 2019. 3, 5, 6

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[11] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 3

[12] Ankur Handa, Michael Bloesch, Viorica Pătrăucean, Simon Stent, John McCormac, and Andrew Davison. gvnn: Neural network library for geometric computer vision. In *European Conference on Computer Vision*, pages 67–82. Springer, 2016. 3

[13] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 4, 7

[15] Zixuan Huang, Junming Fan, Shenggan Cheng, Shuai Yi, Xiaogang Wang, and Hongsheng Li. Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *IEEE Transactions on Image Processing*, 29:3429–3441, 2019. 2

[16] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018. 2

[17] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, pages 1–15, 2015. 6

[18] Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22. IEEE, 2018. 6

[19] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 2

[20] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 32–40, 2020. 2

[21] Adrian Lopez-Rodriguez, Benjamin Busam, and Krystian Mikolajczyk. Project to adapt: Domain adaptation for depth completion from noisy and sparse sensor data. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2, 6

[22] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019. 2, 3, 4, 5, 6, 8

[23] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012. 4

[24] Nathaniel Merrill, Patrick Geneva, and Guoquan Huang. Robust monocular visual-inertial depth completion for embedded systems. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. 2

[25] David Bryant Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 1989. 2

[26] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In-So Kweon. Non-local spatial propagation network for depth completion. In *European Conference on Computer Vision, ECCV 2020*. European Conference on Computer Vision, 2020. 2

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming

Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019. 6

[28] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019. 2

[29] Chao Qu, Wenxin Liu, and Camillo J Taylor. Bayesian deep basis fitting for depth completion with uncertainty. *arXiv preprint arXiv:2103.15254*, 2021. 2

[30] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020. 3

[31] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992. 2

[32] Kourosh Sartipi, Tien Do, Tong Ke, Khiem Vuong, and Stergios I Roumeliotis. Deep depth estimation from visual-inertial slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10038–10045. IEEE, 2020. 2

[33] Shreyas S Shivakumar, Ty Nguyen, Ian D Miller, Steven W Chen, Vijay Kumar, and Camillo J Taylor. Dfusenet: Deep fusion of rgb and sparse depth information for image guided dense depth completion. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 13–20. IEEE, 2019. 2, 3, 6

[34] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 6

[35] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017. 2, 6

[36] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6. IEEE, 2019. 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3

[38] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[39] Alex Wong, Safa Cicek, and Stefano Soatto. Learning topology from synthetic data for unsupervised depth completion. *IEEE Robotics and Automation Letters*, 6(2):1495–1502, 2021. 2, 3, 4, 5, 6, 7, 8

[40] Alex Wong, Xiaohan Fei, Byung-Woo Hong, and Stefano Soatto. An adaptive framework for learning unsupervised depth completion. *IEEE Robotics and Automation Letters*, 6(2):3120–3127, 2021. 2, 4, 6

[41] Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from visual inertial odometry. *IEEE Robotics and Automation Letters*, 2020. 2, 3, 4, 5, 6, 7, 8

[42] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2811–2820, 2019. 2

[43] Yanchao Yang, Alex Wong, and Stefano Soatto. Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3353–3362, 2019. 2, 3, 6, 8

[44] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018. 2

[45] Xingxing Zuo, Nathaniel Merrill, Wei Li, Yong Liu, Marc Pollefeys, and Guoquan Huang. Codevio: Visual-inertial odometry with learned optimizable dense depth. 2021. 2