Depth Completion via Deep Basis Fitting

Chao Qu Ty Nguyen Camillo J. Taylor
University of Pennsylvania
{quchao, tynguyen, cjtaylor}@seas.upenn.edu

Abstract

In this paper we consider the task of image-guided depth completion where our system must infer the depth at every pixel of an input image based on the image content and a sparse set of depth measurements. We propose a novel approach that builds upon the strengths of modern deep learning techniques and classical optimization algorithms and significantly improves performance. The proposed method replaces the final 1×1 *convolutional layer employed in most depth completion networks with a least squares fitting module which computes weights by fitting the implicit depth bases to the given sparse depth measurements. In addition, we show how our proposed method can be naturally extended to a multi-scale formulation for improved self-supervised training. We demonstrate through extensive experiments on various datasets that our approach achieves consistent improvements over state-of-the-art baseline methods with small computational overhead.*

1. Introduction

Deep convolutional networks have proven to be effective tools for solving deep regression problems like depth prediction and depth completion [7]. Most networks proposed for this regression task share a common structure where the penultimate features are reduced to single channel by a final convolutional layer. This final convolutional output is then passed through a nonlinear function to map it onto the range of acceptable depth values.

This observation motivates the main contribution of this paper: Instead of using a fixed set of weights in the final layer, we perform a least squares fit from the penultimate features to the sparse depths to get a set of *data-dependent* weights. The rest of the network parameters are still shared across input data and learned using stochastic gradient descent. From a regression point of view, the network that produces the penultimate layer of features is an adaptive basis function [2] and we refer to the features before the final layer as *depth bases*. We argue that explicitly carrying out a regression from the depth bases to the sparse depths allows the network to learn a different representation that better enforce its predictions to be consistent with the measurements, which manifests as significant performance gain.

To this end, we first demonstrate how one could circumvent the nonlinearity from the depth activation function by solving a linear least squares problem with transformed target sparse depths. We then address the full robustified nonlinear least squares problem in order to deal with noisy measurements and outliers in real-world data. Finally, to make our module truly a drop-in replacement for the final convolutional layer, we show how to adapt it to output predictions at multiple scales with progressively increased detail, which is a feature required by self-supervised training schemes.

2. Related Work

2.1. Depth Estimation

Supervised Learning. Estimating dense depths from a single image is a fundamentally ill-posed problem. Recent learning-based approaches try to solve this by leveraging the predictive power of deep convolutional neural networks (CNN) with strong regularization [7, 24, 9]. These works require dense or semi-dense ground truth annotations, which are costly to obtain in large quantities in practice. Synthetic data [33, 10, 35], on the other hand, can be generated more easily from current graphics systems. However, it is non-trivial to generate synthetic data that closely matches the appearance and structure of the real-world, thus the resulting networks may require an extra step of fine-tuning or domain adaptation [1].

Self-Supervised Learning. When ground truth depths are not available, one could instead seek to use view synthesis as a supervisory signal [39]. This so-called self-supervised training has gained popularity in recent years [27, 31, 44]. The network still takes a single image as input and predicts depths, but the loss is computed on a set of images. This is achieved by warping pixels from a set of source images to the target image using the predicted depths, along with estimated camera poses and camera intrinsics. Under various constancy assumptions [30], errors between target and synthesized images are computed and back-propagated through the network for learning.

Another version of self-supervision utilizes synchronized stereo pairs [12] during training. In this setting, the network predicts the depth for the left view and uses the known focal length and baseline to reconstruct the right view, and vice versa. A more complex form utilizes the motion in monocular videos [52]. In these approaches the network also needs to predict the transformation between frames. The biggest challenge faced by monocular self-supervision is handling moving objects. Many authors try to address this issue by predicting an explanability mask [52], motion segmentation [43], and joint optical-flow estimation [50]. We refer readers to [15] for a more detailed review.

2.2. Depth Completion

Depth completion is an extension to the depth estimation task where sparse depths are available as input. Uhrig *et al*. [42] propose a sparse convolution layer that explicitly handles missing data, which renders it invariant to different levels of sparsity. Ma *et al*. [26] adopt an early-fusion strategy to combine color and sparse depths inputs in a self-supervised training framework. On the other hand, Jaritz *et al*. [22] and Shivakumar *et al*. [37] advocate a late-fusion approach to transform both inputs into a common feature space. Zhang *et al*. [51] and Qiu *et al*. [32] estimate surface normals as a secondary task to help densify the sparse depths. Irman *et al*. [20] identify the cause of artifacts caused by convolution on sparse data and propose a novel scheme, Depth Coefficients, to address this problem. Eldesokey *et al*. [8] and Gansbeke [11] propose to use a confidence mask to handle noise and uncertainty in sparse data. Yang *et al*. [49] infer the posterior distribution of depth given an image and sparse depths by a Conditional Prior Network. While most of the above works deal with data from LiDARs or depth cameras, Wong *et al*. [48] design a system that works with very sparse data from a visual-inertial odometry system. Weeraskera *et al*. [47] attach a fully-connected Conditional Random Field to the output of a depth prediction network, which can also handle any input sparsity pattern.

Cheng *et al*. [4] propose a convolutional spatial propagation network that learns the affinity matrix to complete sparse depths. This is similar to a diffusion process and uses several iterations to update the depth map. Another iterative approach is described by Wang *et al*. [45], in which they design a module that can be integrated into many existing methods to improve performance of a pre-trained network without re-training. This is done by iteratively updating the intermediate feature map to make the model output consistent with the given sparse depths. Like [45], our approach could be readily integrated into many of the previously proposed depth completion networks.

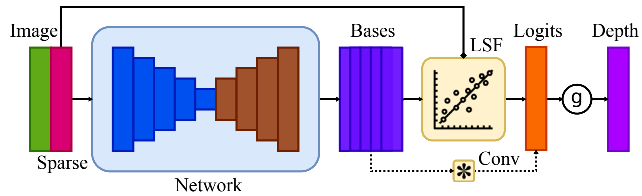

Figure 1: An overview of our proposed method. Solid lines indicate the data flow of our module, while dotted lines indicate that of the baseline method, which is simply a convolutional layer. Our LSF module can replace the convolutional layer with no change to the rest of the network.

In other related work Tang *et al*. [40]. propose to parameterize depth map with a set of basis depth maps and optimize weights to minimize a feature-metric distance. In contrast, our bases are multi-scale by construction and are fit directly to the sparse depths.

3. Method

In this section, we describe our proposed method for the task of monocular image-guided depth completion[1]. Given an image X and a sparse depth map S, we wish to predict a dense depth image D' from a depth estimation function f that minimizes some loss function $\mathcal{L}$ with respect to the ground truth depth D. Typically, X is a color image, S the sparse depth map where invalid pixels are encoded by 0, and f a fully convolutional neural network whose parameters are denoted by θ. When ground-truth depth D is available, the learning problem is to determine θ^* according to

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(X, S; \theta), D) \tag{1}$$

For supervised training we choose $\mathcal{L}$ to be the L1 norm on depth and for self-supervised training we use a combination of L1+SSIM on the intensity values [46] coupled with an edge-aware smoothness term [15].

3.1. Linear Least-Squares Fitting (LSF) Module

Existing depth prediction networks usually employ a final convolutional layer to convert an M-channel set of basis features, B, to a single-channel result, L, which is sometimes referred to as the logits layer. The inputs to this final layer are allowed to range freely between $-\infty$ and $+\infty$ while the logit outputs are mapped to positive depth values by a nonlinear activation function, g. Following common practice in the depth completion literature [15] we choose g as follows:

$$g(x) = a/\sigma(x) = a(1 + e^{-x}) \tag{2}$$

[1] From now on we will refer to this task as depth completion.

where a is a scaling factor that controls the minimum depth and $\sigma(\cdot)$ the sigmoid function. In this work, we set $a = 1$.

For simplicity we assume that the final convolution filter that maps the basis features, B, onto the logits, L, has a kernel size of 1×1 with bias w_0, but one could easily extend our result to arbitrary kernel size. L is, therefore, an affine combination of channels in B and the predicted depth at pixel i is

$$D'[i] = g\left(L[i]\right) = g\left(\sum_{j=0}^{M} w_j \cdot B_j[i]\right) = g(\mathbf{w}^\top \mathbf{b}_i) \quad (3)$$

where $\mathbf{w} = (w_0, \cdots, w_M)^\top$ represents the combined filter weights and bias, and $\mathbf{b}_i$ the basis (feature) vector at pixel i with $B_0[i] = 1$, and $[\cdot]$ the pixel index operator. To simplify notations, we use lower case letters, *e.g.* $\mathbf{b}_i = B[i]$, to denote values at a particular pixel location. The weights $\mathbf{w}$ are updated via *back-propagation* [25] using stochastic gradient descent [3]. Once learned they are typically fixed at inference time.

When enough sparse depth measurements are available the weights $\mathbf{w}$ can instead be directly computed from data. Specifically, our weights are obtained through a least squares fit from the bases B to the sparse depths S at valid pixels, which can then be used in place of the final convolutional layer. An overview of our proposed method is shown in Figure 1.

The objective function we wish to minimize for the least squares problem is

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} r^2\left(\mathbf{w}, \mathbf{b}_i, s_i\right) \quad (4)$$

with residual function

$$r(\mathbf{w}, \mathbf{b}_i, s_i) = g\left(\sum_{j=0}^{M} \mathbf{w}_j \mathbf{b}_{ij}\right) - s_i = g\left(\mathbf{w}^\top \mathbf{b}_i\right) - s_i \quad (5)$$

where s_i denotes an individual sparse depth measurement, N is the number of valid pixels in S, M the number of channels in B, and $g(\cdot)$ a nonlinear activation function.

The residual function $r(\cdot)$ is obviously nonlinear w.r.t. the weights $\mathbf{w}$ due to the nonlinearity in $g(\cdot)$. A simple workaround is to transform the target variable s by $g^{-1}(\cdot)$ to arrive at a new linear residual function

$$\tilde{r}(\mathbf{w}, \mathbf{b}_i, s_i) = \mathbf{w}^\top \mathbf{b}_i - g^{-1}(s_i) = \mathbf{w}^\top \mathbf{b}_i - t_i \quad (6)$$

We can then rewrite the new objective function (4) in matrix form to obtain a linear least squares problem

$$\min_{\mathbf{w}} \frac{1}{2} \left\|\mathbf{B}\mathbf{w} - \mathbf{t}\right\|^2 \quad (7)$$

where $\mathbf{B}$ denotes the $N \times (M+1)$ matrix of stacked features $\mathbf{b}_i$ at valid pixel locations and $\mathbf{t}$ the corresponding transformed sparse depths vector. The solution to (7) is the well-known *Moore-Penrose pseudo-inverse* which can be further regularized with parameter λ [2].

$$\mathbf{w}^* = \left(\lambda \mathbf{I} + \mathbf{B}^\top \mathbf{B}\right)^{-1} \mathbf{B}^\top \mathbf{t} \quad (8)$$

Notice here that our weights $\mathbf{w}^*$ are calculated deterministically as a function of the bases B and the sparse depth S, while the original convolution filter is independent of both. In practice, this problem is usually solved via LU or Cholesky decomposition both of which are differentiable [28]. Thus, the entire training process including our LSF module is differentiable which means that it can be trained in an end-to-end manner. This is an important point since we have found that retraining the network with this fitting module produces much better results than simply adding the fitting procedure to a pretrained network without retraining. Effectively the retraining allows the network to make best use of the new adaptive fitting layer.

3.2. Robustified Nonlinear Fitting

The linear LSF module is readily usable as a replacement for the final convolution layer in many depth prediction networks. One problem remains to be addressed, which is the fact that the original objective function in Equation 5 is nonlinear w.r.t. the weights $\mathbf{w}$. Although applying the inverse transformation $g^{-1}(\cdot)$ to the sparse depths is a simple yet effective solution, we show that performing a full robustified nonlinear least squares fitting provides further performance improvements and outlier rejection at the cost of extra computation time.

Real-world data often contain noise and outliers that are hard to model or eliminate. Cheng *et al*. [5] point out that there exist many different types of noise in LiDAR data from the well-known KITTI [13] dataset. They propose a novel feedback loop that utilizes stereo matching from the network to clean erroneous data points in the sparse depths. Gansbeke *et al*. [11] let the network predict a confidence map to weight information from different input branches. To handle these cases, we employ M-estimators [18], which fit well within our least squares framework.

Recall the objective function in equation (4), taking the derivative with respect to $\mathbf{w}$, setting it to zero and ignoring higher-order terms yields the following linear equation (Gauss-Newton approximation)

$$\mathbf{J}^\top \mathbf{J} \Delta \mathbf{w} = -\mathbf{J}^\top \mathbf{r} \quad (9)$$

where $\mathbf{J}$ is the Jacobian matrix that is formed by stacking Jacobians $\mathbf{J}_i(\mathbf{w}, \mathbf{b}_i, s_i) = \partial r(\mathbf{w}, \mathbf{b}_i, s_i)/\partial \mathbf{w}$, and $\mathbf{r}$ is the residual vector formed by stacking $r_i(\mathbf{w}, \mathbf{b}_i, s_i)$. Following standard practice in Triggs *et al*. [41], we minimize the

effective squared error where the cost function is statistically weighted and robustified, which is equivalent to solving for $\Delta\mathbf{w}$ in

$$\bar{\mathbf{J}}^\top \mathbf{W} \bar{\mathbf{J}} \Delta\mathbf{w} = -\bar{\mathbf{J}}^\top \mathbf{W} \bar{\mathbf{r}} \tag{10}$$

$$\text{with} \quad \bar{\mathbf{J}}_i = \sqrt{\rho'_i} \mathbf{J}_i \quad \text{and} \quad \bar{\mathbf{r}}_i = \sqrt{\rho'_i} \mathbf{r}_i \tag{11}$$

where $\mathbf{W} = \mathbf{L}^\top \mathbf{L}$ a diagonal matrix with terms inverse-proportional to the noise in each measurement, which we assume to be Gaussian for LiDARs , $\rho(x)$ is the Huber loss [19] and ρ' its first derivative

$$\rho(x) = \begin{cases} x^2, & |x| \leq 1 \\ 2|x| - 1, & |x| > 1 \end{cases} \tag{12}$$

We iteratively calculate $\Delta\mathbf{w}$ by solving (10) and update $\mathbf{w}$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w} \tag{13}$$

with $\mathbf{w}$ initialized from the linear fitting in Section 3.1.

Theoretically, one should repeat this until convergence, but to alleviate the problem of vanishing or exploding gradients [17], we adopt the fixed-iteration approach used in [40], which is also known as an *incomplete optimization* [6]. Despite its limitations, it has the advantage of having a fixed training/inference time and reduced memory consumption, which is often desirable in robotic systems with limited computational resources. As discussed in earlier Section 3.1, solving a linear system like equation (10) via Cholesky decomposition is differentiable, thus optimizing this non-linear objective function by performing a fixed number of Gauss-Newton steps maintains the differentiability of the entire system.

3.3. Multi-scale Prediction for Self-supervision

Self-supervised training formulates the learning problem as novel view synthesis, where the network predicted depth is used to synthesize a target image from other viewpoints. To overcome the gradient locality problem of the bilinear sampler [21] during image warping, previous works [14, 52] adopt a multi-scale prediction and image reconstruction scheme by predicting a depth map at each decoder layer's resolution. According to Godard *et al*. [15], this has the side effect of creating artifacts in large texture-less regions in the lower resolution depth maps due to ambiguities in photometric errors. They later improved upon this multi-scale formulation by upsampling all the lower resolution depth maps to the input image resolution.

This technique greatly reduces various artifacts in the final depth prediction, but it still has one undesired property, namely, depth maps predicted at each scale are largely independent. Lower resolution depth maps are used in training phase, but are discarded during inference, resulting in a waste of parameters.

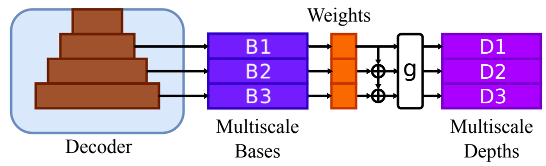

Figure 2: Our proposed multi-scale depth prediction. The full resolution depth D_3 is reconstructed using all bases prediction.

Rather than predicting a depth map D'_k at each scale k separately, we propose to predict a set of bases B_k, as shown in Figure 2. Each of the basis vectors is obtained by upsampling features from corresponding scales in the decoder as shown in Figure 2 so the resulting basis images are *band-limited* by construction with coarser basis images corresponding to earlier layers in the decoder. The depth prediction at a particular scale s is then reconstructed using bases up to that scale.

$$d'_{is} = g\left(\sum_{k=0}^{s} \mathbf{w}_k^\top \mathbf{b}_{ik}\right) \tag{14}$$

The final depth prediction at highest scale K is

$$d'_i := d'_{iK} = g\left(\sum_{k=0}^{K} \mathbf{w}_k^\top \mathbf{b}_{ik}\right) = g\left(\mathbf{w}^\top \mathbf{b}_i\right) \tag{15}$$

where $\mathbf{b}_i = (\mathbf{b}_{i0}^\top, \dots, \mathbf{b}_{iK}^\top)^\top$ and $\mathbf{w} = (\mathbf{w}_0^\top, \dots, \mathbf{w}_K^\top)^\top$.

With this formulation, predictions at different scales will work towards the same goal, which is to reconstruct the full resolution depth map. This approach is analogous to wavelet or Fourier encodings of an image where the basis maps are organized into band-limited components to represent the signal at various scales.

Our LSF module handles this multi-scale approach quite naturally since we can simply allocate the basis maps in B amongst the desired scales, then upsample and group them back together to perform the fitting step. Henceforth we use this new multi-scale prediction scheme in all our experiments, even for supervised training where only the full resolution depth prediction is required.

4. Experiments

4.1. Implementation Details

Network Architecture. All networks and training are implemented in PyTorch To investigate the effectiveness of the proposed LSF module, we adopt the network used in Ma *et al*. [26] as our main baseline. The network is a symmetric encoder-decoder [34] with skip connections. We make the following modifications for better training: 1) transposed

Dataset	Resolution	# Train	# Val	Cap [m]
KITTI [13, 42]	375 × 1242	38412	3347	80
V-KITTI [10]	188 × 621	5156	837	130
Synthia [35]	304 × 512	3634	901	130
NYU-V2 [38]	480 × 640	1086	363	-

Table 1: A summary of all datasets used. **Cap** indicates the maximum depth being used for sampling sparse depths as well as in computing various error metrics. **Resolution** is the image resolution that we use in our experiments, which we downsample from the original one if necessary.

convolutions are replaced with resize convolutions [29] for better upsampling, 2) the extra convolution layer between the encoder and the decoder are removed, 3) the encoder is based on ResNet18, as opposed to ResNet34 [16] and is initialized with parameters pretrained on ImageNet [36].

We let the decoder output 4, 8, 16, and 32-dimensional bases at each scale. These are then upsampled to the image resolution and concatenated together to form a 60-dimensional basis. For the baseline network, it is fed directly into a final convolution layer while for ours, it is passed into the LSF module together with the sparse depths. Therefore, these two methods are exactly the same in terms of network parameters, up to the last convolution layer.

Training Parameters. Following [26], we use the Adam optimizer [23] with an initial learning rate of 1e-4 and reduce it by half every 5 epochs. Training is carried out on a single Tesla V100 GPU with 15 epochs and the best validation result is reported. Batch sizes may vary across datasets due to GPU memory constraints, but are kept the same for experiments of the same dataset. Only random horizontal flips are used to augment the data for supervised training, no data augmentation is performed for self-supervised training. The above settings are used across **all** experiments in this work (unless explicitly stated) with the same random seed to ensure controlled experiments with fair and meaningful comparisons.

4.2. Datasets

A summary of all datasets we evaluate on is shown in Table 1.

KITTI Depth Completion. We evaluate on the newly introduced KITTI depth completion dataset [42] and follow the official training/validation split. The ground truth depth is generated by merging several consecutive LiDAR scans around a given frame and refined using a stereo matching algorithm. The sparse depth map is generated by projecting LiDAR measurements onto the closest image, which occupies on average 4% of the image resolution. We use all categories from the KITTI raw dataset [13] except for **Person** as it contains mostly static scenes with moving objects, which is not suitable for self-supervised training.

Virtual KITTI. The Virtual KITTI (V-KITTI) dataset is a synthetic video dataset [10], which contains 50 monocular videos generated with various simulated lighting and weather conditions with dense ground truth annotations. We adopt an out-of-distribution testing scheme for this dataset. Specifically, we use sequences 1, 2, 6, 18 with variations **clone**, **morning**, **overcast** and **sunset** for training, and sequence 20 with variation **clone** for validation. Thus the testing sequence is never seen during training. The sparse depths are generated by randomly sampling pixels that have a depth value less than 130 meters. We intentionally increase the depth cap to 130 meters for all synthetic datasets since recent LiDAR units can easily achieve this range.

Synthia. Synthia [35] is another synthetic dataset in urban settings with dense ground truth. We use the **SYNTHIA-Seqs** version which simulates four video sequences acquired from a virtual car across different seasons. Following the training protocol in V-KITTI, we use sequences 1,2,5,6 for training and sequence 4 for validation, all under the **summer** variation. We include this dataset because it has simulated stereo images, which serves as a complement to the monocular only V-KITTI. Again ground truth and sparse depths are capped at 130 meters.

NYU Depth V2. In addition to all the outdoor datasets, we also validate our approach on NYU Depth V2 (NYU-V2) [38], which is an indoor dataset. We use the 1449 densely labeled pairs of aligned RGB and depth images instead of the full dataset which is comprised of raw image and depth data as provided by the Kinect sensor. The dataset is split into approximately 75% training and 25% validation. We use the same strategy as above for sampling sparse depths but put no cap on the maximum depth.

4.3. Results

We evaluate performance using standard metrics in the depth estimation literature. Note that for accuracy (δ threshold) [7] we only report $\delta_1 < 1.25$, due to space limitations and the fact that the δ_2 and δ_3 are typically 99% for our experiments and thus provide limited insights. Following [45], we group results based on input modalities, where **rgb** denotes a network that only takes a color image as input. In contrast **rgbd** indicates a network that takes both the color image and the sparse depths as inputs.

Performance of Linear Fitting. Table 2 shows quantitative comparisons between our proposed linear LSF module from Section 3.1 and the baseline under supervised training. We see consistent improvements of our linear LSF module

Supervised Training			NYU-V2			V-KITTI			Synthia			KITTI		
Input	Method	Sparse	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1
rgb	conv	-	0.6244	0.8693	58.44	6.9998	14.653	66.43	2.3911	6.3915	76.09	1.8915	4.1164	86.24
rgb	pnp	0.2%	0.5517	0.7976	64.23	6.4701	13.990	70.18	2.1716	6.0084	81.37	1.6581	3.8019	88.67
rgb	lsf-	0.2%	0.4081	0.6124	77.86	5.8379	12.712	71.62	2.4089	6.2520	78.49	1.7033	3.5986	91.80
rgb	lsf	0.2%	**0.1826**	**0.3165**	**96.11**	**4.5122**	**9.7933**	**77.18**	**2.0104**	**5.6285**	**84.37**	**0.7716**	**2.0808**	**97.69**
(conv-lsf) / conv			+71%	+64%		+36%	+33%		+16%	+12%		+59%	+50%	
rgbd	conv	4%	0.1089	0.1679	99.20	1.5683	4.8982	94.71	0.7506	3.3322	96.50	0.3033	1.1392	99.57
rgbd	pnp	4%	0.1008	0.1604	99.24	1.5301	4.8798	94.81	0.7311	3.3217	96.60	0.2993	1.1343	99.57
rgbd	lsf-	4%	0.1127	0.1853	99.34	2.1049	6.1901	95.30	1.3220	4.6594	94.27	0.6319	2.2895	98.46
rgbd	lsf	4%	**0.0300**	**0.0735**	**99.83**	**1.2598**	**4.6227**	**97.43**	**0.5317**	**3.1146**	**97.85**	**0.2266**	**0.9988**	**99.67**
(conv-lsf) / conv			+72%	+56%		+20%	+6%		+29%	+7%		+25%	+12%	

Table 2: Quantitative results of supervised training on various datasets. **conv** denotes the baseline network, **pnp** denotes running the PnP [45] module on the trained **conv** network without re-training, **lsf-** indicates adding a linear LSF module to the pre-trained **conv** network without re-training for 5 iterations, and **lsf** is our linear fitting module (re-trained). Percentage values listed under the **Sparse** column indicates sparse depths percentage of image resolution. Best results in each category are in **bold**.

Noise and Outliers			NYU-V2			V-KITTI			Synthia			KITTI		
Input	Method	Sparse	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1
rgb	pnp	0.2%	0.5587	0.8019	63.66	6.5099	14.018	69.86	2.2044	6.0268	80.89	1.6571	3.8019	88.67
rgb	lsf	0.2%	0.2439	0.3815	92.93	5.2670	10.696	65.00	2.2197	5.9136	78.34	0.7716	2.0808	97.69
rgb	lsf2	0.2%	0.2304	0.3519	92.70	6.0025	10.768	51.01	3.2160	7.2096	59.68	1.0111	2.4547	95.88
rgb	lsf2+	0.2%	**0.1880**	**0.3217**	**94.97**	**4.6786**	**9.7402**	**70.16**	**2.1032**	**5.7685**	**79.00**	**0.6775**	**1.9651**	**98.28**
(lsf - lsf2+) / lsf			+23%	+16%		+11%	+9%		+5%	+2%		+12%	+6%	
rgbd	conv	4%	0.1173	0.1788	99.07	1.8748	5.1880	94.17	0.8774	3.4660	96.03	0.3033	1.1392	99.57
rgbd	pnp	4%	0.1061	0.1688	99.15	1.8067	5.1342	94.46	0.8452	3.4511	96.19	0.2993	1.1343	99.57
rgbd	lsf	4%	0.0606	0.1102	99.73	1.8599	5.1987	**95.90**	**0.7082**	**3.2426**	**97.41**	0.2266	0.9988	99.67
rgbd	lsf2	4%	0.0577	0.1080	99.72	1.8008	**5.0008**	94.58	0.7890	3.4142	96.78	0.2305	1.0417	99.67
rgbd	lsf2+	4%	**0.0493**	**0.1003**	**99.73**	**1.7273**	5.0422	95.50	0.7188	3.2579	97.31	**0.2208**	**0.9758**	**99.71**
(conv - lsf2+) / conv			+58%	+44%		+8%	+3%		+18%	+6%		+27%	+14%	

Table 3: Quantitative results of supervised training with noisy data and outliers. For all datasets except KITTI, noise is additive Gaussian with standard deviation of 0.05m. We randomly sample 30% of sparse depths to be outliers. **conv** denotes the baseline network, **pnp** denotes running the PnP [45] module on the trained **conv** network without re-training, **lsf** is our linear fitting module, **lsf2** is our nonlinear fitting module with 2 iterations, and **lsf2+** is **lsf2** with robust norm (Huber). Best results in each category are in **bold**.

Self-Supervised Training			V-KITTI Mono			Synthia Mono			Synthia Stereo			KITTI Stereo		
Input	Method	Sparse	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1	MAE	RMSE	δ_1
rgbd	conv-ms	4%	2.9904	7.4517	86.87	3.0191	9.1076	66.43	1.3498	5.8643	92.73	0.6295	2.0950	**99.00**
rgbd	lsf	4%	**2.3804**	**6.7326**	**93.76**	**1.4564**	**4.6260**	**91.76**	**0.8619**	**3.9523**	**96.30**	**0.5820**	**1.7370**	98.79
(conv-ms - lsf) / conv-ms			+20%	+10%		+52%	+49%		+36%	+33%		+8%	+17%	

Table 4: Quantitative results of self-supervised training on various datasets. The densely labeled NYU-V2 is random and monocular, thus is excluded from this experiment. Here **conv-ms** is the baseline multi-scale prediction, **lsf** is the our proposed method with linear fitting and multi-scale basis. Best results in each category are in **bold**.

over the baseline in all metrics across all datasets. Note that for **rgb** input only, the baseline doesn't use any sparse depth information at all. Thus the large improvement achieved by our fitting method using depth measurements for only 0.2% of the pixels is quite significant. For the **rgbd** case, although the sparse depth map is already used as the input to the baseline network, adding our fitting module better constrains the final prediction to be in accordance with the measurements and improves the baseline network. Since we use the L1 norm as our loss function, the improvement in MAE is bigger than that in RMSE. Examples of depth prediction are shown in Figure 3 for qualitative comparisons.

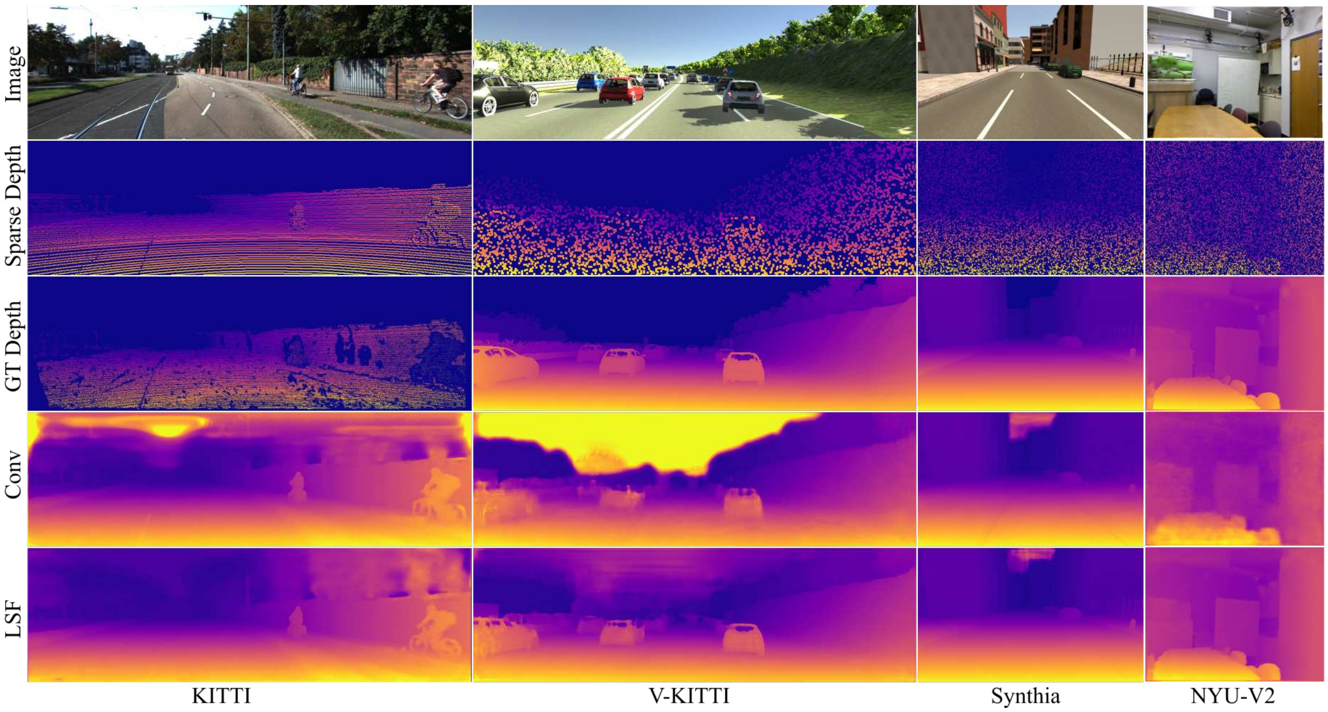

Figure 3: Qualitative results of supervised learning on various datasets. Sparse depths are dilated for visualization purpose (4% of image resolution). Artifacts in the upper part of depth prediction from outdoor datasets are due to lack of supervision.

We also perform experiments in which we take a pre-trained baseline method, replace the final convolutional layer with our LSF module and evaluate without re-training. This is denoted by **lsf-**. Results show that re-training a baseline network with the LSF module allows it to achieve significantly better performance.

Additionally, we compare with PnP [45], which is a similar method that can be used on many existing networks to improve performance (see Table 2 and 3). The main difference is that PnP does not require re-training. We use the author's official implementation on our baseline network by updating the output of the encoder and run for 5 iterations with update rate 0.01 as suggested in the paper. We found that although PnP has the advantage no re-training, it takes much longer to run, uses a large amount of memory and yields a smaller improvement compared to ours. Comparisons of runtime are provided in the supplementary material.

Table 5 compares our results to those achieved with CSPN[4]. The numbers for the CSPN system are taken directly taken from their paper and the official KITTI depth completion benchmark. For NYU-V2 we use the same data split they used and sample 500 sparse depths. These results show the improvement afforded by our method.

Dealing with Noise and Outliers. To verify the effectiveness of our proposed robustified nonlinear fitting module, we inject additive Gaussian noise with a standard deviation

		NYU-V2		KITTI		
Input	Method	RMSE	δ_1	MAE	RMSE	iRMSE
rgbd	cspn	0.136	99.0	0.2795	1.0196	**2.93**
rgbd	lsf2+	**0.134**	**99.3**	**0.2552**	**0.8850**	3.40

Table 5: Comparison results on both NYU-V2 and KITTI between CSPN[4] and our method. **lsf2+** is our LSF module with 2 iterations and Huber loss.

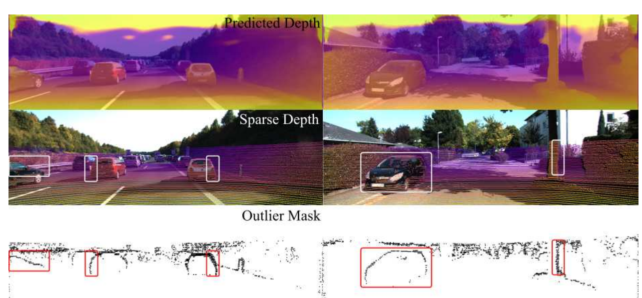

Figure 4: When using a robust norm, outliers from the input sparse depths can be identified. For KITTI dataset, these outliers usually occur at object boundaries, which we highlight a few in rectangles. Best view when zoomed in.

of 0.05 meters to sparse depths from NYU-V2, V-KITTI, and Synthia. We then randomly select 30% of the available sparse depths to be outliers and set them to random values drawn uniformly from a range between $0.5\times$ to $1.5\times$ of the true depth value. We left KITTI untouched as it already contains noise and outliers [5]. All nonlinear variants of LSF runs for 2 iterations, which we empirically found to achieve a good balance between performance and efficiency. We refer the reader to our supplementary material for further discussion on the number of iterations. We then train various models with different configurations using the corrupted data, which are also grouped by input modalities. Quantitative results are shown in Table 3.

For the **rgb** case, we ignore the baseline **conv** as it doesn't use sparse depths and is, therefore, unaffected by noise. We again see consistent improvements in all metrics across all datasets. Notice that for our nonlinear fitting without Huber loss (**lsf2**), we get worse numbers on some datasets compared to our linear variant (**lsf**). This is because least squares fitting is sensitive to outliers without a robust norm. There are also some models in the **rgbd** case where the robustified version (**lsf2+**) doesn't outperform the linear and nonlinear ones. We hypothesize this to be caused by using the corrupted sparse depths as network input which degrades the networks performance early on. We show in Figure 4 that our proposed method is able to identify outliers in the sparse depths and downplay them during fitting.

These results can also be cross-compared with those in Table 2, which are all trained on clean data. Clearly, models trained with clean data outperforms those trained with corrupted ones with the same configuration. But ours with nonlinear fitting and Huber loss (**lsf2+**) can sometimes reach similar performance to those trained with clean data even when significant noise and outliers are present.

Self-supervised Training with Multi-scale Prediction. Table 4 shows quantitative comparisons between our linear LSF module with multi-scale basis and the baseline network under both monocular and stereo self-supervised training. In this case, the baseline network has more parameters because it needs to predict depths at different scales independently. We again witness consistent improvement in all metrics across all datasets except for δ_1 in KITTI. Qualitative results are shown in Figure 5. For all self-supervised training, we use the same hyper-parameters on photometric and smoothness loss as in [15], where $\lambda_p = 1.0$ and $\lambda_s = 0.001$. Note in monocular training, we use the ground truth poses directly, as opposed to having a dedicated pose network.

5. Conclusions

In this paper we propose a novel approach to the depth completion problem that augments deep convolutional networks with a least squares fitting procedure. This method allows us to combine some of the best features of modern deep networks and classical regression algorithms. This scheme could be applied to a number of proposed depth completion networks or other regression problems to improve performance. Our proposed module is differentiable which means the modified networks can still be trained from end to end. This is important because retraining the networks allows them to make better use of the new fitting layer and allows them to produce better depth bases from the input data. We then describe how a linear least squares fitting scheme could be extended to incorporate robust estimation to improve resilience to noise and outliers which are common in real world data. We also show the method can be employed in self-supervised settings where no ground truth is available. We validate our fitting module on a state-of-the-art depth completion network with various input modalities, training frameworks, and datasets. One limitation of our approach is that it is unable to handle extremely sparse points, which creates an underdetermined linear system and can only be solved by adding strong regularization. In future work, we propose to handle this case by adopting a full bayesian approach.

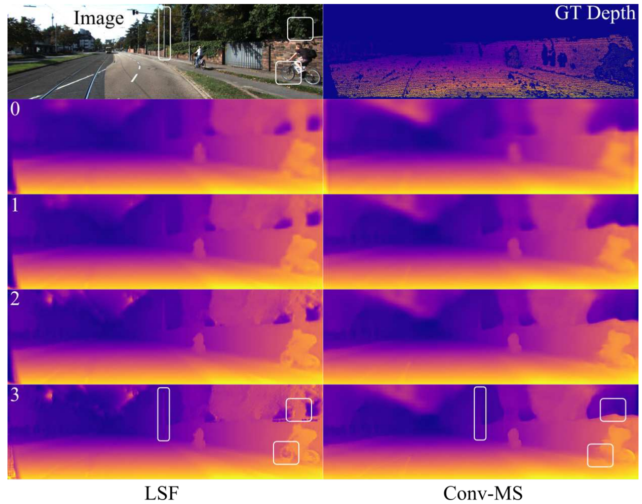

Figure 5: Qualitative results of our proposed multi-scale prediction versus the baseline using stereo self-supervision on KITTI dataset. All intermediate depth maps are upsampled to the image resolution as suggested in [15]. Our multi-scale bases are able to learn a much more detailed depth map compared to the baseline method. Numbers at top left corner of each image indicate the scale level, where 3 is the full resolution depth map. Best view when zoomed in.

Acknowledgement

We gratefully acknowledge the support of Novateur Research Solutions, and NVIDIA through the NVAIL grant.

References

[1] A. A. Abarghouei and T. P. Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2800–2810, 2018.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. 2010.

[4] X. Cheng, P. Wang, and R. Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *ECCV*, 2018.

[5] X. Cheng, Y. Zhong, Y. Dai, P. Ji, and H. Li. Noise-aware unsupervised deep lidar-stereo fusion. *ArXiv*, abs/1904.03868, 2019.

[6] J. Domke. Generic methods for optimization-based modeling. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 318–326, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[8] A. Eldesokey, M. Felsberg, and F. S. Khan. Propagating confidences through cnns for sparse data regression. In *BMVC*, 2018.

[9] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[10] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016.

[11] W. V. Gansbeke, D. Neven, B. D. Brabandere, and L. V. Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6, 2019.

[12] R. Garg, B. G. V. Kumar, G. Carneiro, and I. D. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *ArXiv*, abs/1603.04992, 2016.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[14] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, 2016.

[15] C. Godard, O. M. Aodha, and G. J. Brostow. Digging into self-supervised monocular depth estimation. *ArXiv*, abs/1806.01260, 2018.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[17] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[18] P. Huber, J. Wiley, and W. InterScience. *Robust statistics*. Wiley New York, 1981.

[19] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, Mar. 1964.

[20] S. Imran, Y. Long, X. Liu, and D. Morris. Depth coefficients for depth completion. *ArXiv*, abs/1903.05421, 2019.

[21] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *ArXiv*, abs/1506.02025, 2015.

[22] M. Jaritz, R. de Charette, É. Wirbel, X. Perrotton, and F. Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. *2018 International Conference on 3D Vision (3DV)*, pages 52–60, 2018.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[24] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248, 2016.

[25] Y. Lecun. A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, pages 21–28. Morgan Kaufmann, 1988.

[26] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. *ArXiv*, abs/1807.00275, 2018.

[27] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

[28] I. Murray. Differentiation of the cholesky decomposition. *ArXiv*, abs/1602.07527, 2016.

[29] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[30] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67:141–158, 2005.

[31] S. Pillai, R. Ambrus, and A. Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. *ArXiv*, abs/1810.01849, 2018.

[32] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. C. Liu, B. Zeng, and M. Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. *ArXiv*, abs/1812.00488, 2018.

[33] W. Qiu and A. L. Yuille. Unrealcv: Connecting computer vision to unreal engine. *ArXiv*, abs/1609.01326, 2016.

[34] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*

(MICCAI), volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[35] G. Ros, L. Sellart, J. Materzynska, D. Vázquez, and A. M. López. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2014.

[37] S. S. Shivakumar, T. Nguyen, S. W. Chen, and C. J. Taylor. Dfusenet: Deep fusion of rgb and sparse depth information for image guided dense depth completion. *ArXiv*, abs/1902.00761, 2019.

[38] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[39] R. Szeliski. Prediction error as a quality metric for motion and stereo. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:781–788 vol.2, 1999.

[40] C. Tang and P. Tan. BA-net: Dense bundle adjustment networks. In *International Conference on Learning Representations*, 2019.

[41] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms*, 1999.

[42] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *2017 International Conference on 3D Vision (3DV)*, pages 11–20, 2017.

[43] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *ArXiv*, abs/1704.07804, 2017.

[44] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey. Learning depth from monocular videos using direct methods. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2017.

[45] T.-H. Wang, F.-E. Wang, J.-T. Lin, Y.-H. Tsai, W.-C. Chiu, and M. Sun. Plug-and-play: Improve depth estimation via sparse data propagation. *arXiv preprint arXiv:1812.08350*, 2018.

[46] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarty. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[47] C. S. Weerasekera, T. Dharmasiri, R. Garg, T. Drummond, and I. D. Reid. Just-in-time reconstruction: Inpainting sparse maps using single view depth predictors as priors. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[48] A. Y. F. Wong, X. Fei, and S. Soatto. Voiced: Depth completion from inertial odometry and vision. *ArXiv*, abs/1905.08616, 2019.

[49] Y. Yang, A. Wong, and S. Soatto. Dense depth posterior (ddp) from single image and sparse range. *ArXiv*, abs/1901.10034, 2019.

[50] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[51] Y. Zhang and T. A. Funkhouser. Deep depth completion of a single rgb-d image. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018.

[52] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, 2017.