Estimating Depth from RGB and Sparse Sensing

Zhao Chen[0000−0002−6681−4053], Vijay Badrinarayanan[0000−0002−3297−7495], Gilad Drozdov[0000−0002−6660−6481], and Andrew Rabinovich[0000−0003−3078−6705]

Magic Leap, Sunnyvale CA 94089, USA
{zchen, vbadrinarayanan, gdrozdov, arabinovich}@magicleap.com

Abstract. We present a deep model that can accurately produce dense depth maps given an RGB image with known depth at a very sparse set of pixels. The model works *simultaneously* for both indoor/outdoor scenes and produces state-of-the-art dense depth maps at nearly real-time speeds on both the NYUv2 and KITTI datasets. We surpass the state-of-the-art for monocular depth estimation even with depth values for only 1 out of every $\sim$ 10000 image pixels, and we outperform other sparse-to-dense depth methods at all sparsity levels. With depth values for 1/256 of the image pixels, we achieve a mean error of less than 1% of actual depth on indoor scenes, comparable to the performance of consumer-grade depth sensor hardware. Our experiments demonstrate that it would indeed be possible to efficiently transform sparse depth measurements obtained using e.g. lower-power depth sensors or SLAM systems into high-quality dense depth maps.

Keywords: Sparse-to-Dense Depth, Depth Estimation, Deep Learning.

1 Introduction

Efficient, accurate and real-time depth estimation is essential for a wide variety of scene understanding applications in domains such as virtual/mixed reality, autonomous vehicles, and robotics. Currently, a consumer-grade Kinect v2 depth sensor consumes $\sim$ 15W of power, only works indoors at a limited range of $\sim 4.5m$, and degrades under increased ambient light [8]. For reference, a future VR/MR head mounted depth camera would need to consume 1/100th the power and have a range of 1-80m (indoors and outdoors) at the full FOV and resolution of an RGB camera. Such requirements present an opportunity to jointly develop energy-efficient depth hardware and depth estimation models. Our work begins to address depth estimation from this perspective.

Due to its intrinsic scale ambiguity, monocular depth estimation is a challenging problem, with state-of-the-art models [4, 17] still producing $> 12\%$ mean absolute relative error on the popular large-scale NYUv2 indoor dataset [24]. Such errors are prohibitive for applications such as 3D reconstruction or tracking, and fall very short of depth sensors such as the Kinect that boast relative depth error on the order of $\sim 1\%$ [14, 25] indoors.

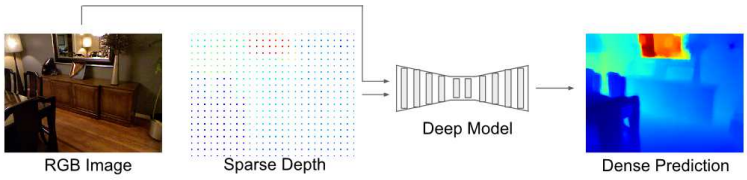


Fig. 1: **From Sparse to Dense Depth.** An RGB Image and very sparse depth map are input into a deep neural network. We obtain a high-quality dense depth prediction as our final output.

Acknowledging the limitations of monocular depth estimation, we provide our depth model with a sparse amount of measured depth along with an RGB image (See Fig. 1) in order to estimate the full depth map. Such sparse depth resolves the depth scale ambiguity, and could be obtained from e.g. a sparser illumination pattern in Time-of-Flight sensors [8], confident stereo matches, LiDAR-like sensors, or a custom-designed sparse sensor. We show that the resultant model can provide comparable performance to a modern depth sensor, despite only observing a small fraction of the depth map. We believe our results can thus motivate the design of smaller and more energy-efficient depth sensor hardware. As the objective is now to *densify* a sparse depth map (with additional cues from an RGB image), we call our model Deep Depth Densification, or D^3.

One advantage of our D^3 model is that it accommodates for arbitrary sparse depth input patterns, each of which may correspond to a relevant physical system. A regular grid of sparse depth may come from a lower-power depth sensor, while certain interest point sparse patterns such as ORB [27] or SIFT [21] could be output from modern SLAM systems [23]. In the main body of this work, we will focus on regular grid patterns due to their ease of interpretation and immediate relevance to existing depth sensor hardware, although we detail experiments on ORB sparse patterns in the Supplementary Materials.

Our contributions to the field of depth estimation are as follows:

1. A deep network model for dense scene depth estimation that achieves accuracies comparable to conventional depth sensors.
2. A depth estimation model which works *simultaneously* for indoors and outdoors scenes and is robust to common measurement errors.
3. A flexible, invertible method of parameterizing sparse depth inputs that can accommodate arbitrary sparse input patterns during training and testing.

2 Related Work

Depth estimation has been tackled in computer vision well before the advent of deep learning [28, 29]; however, the popularization of encoder-decoder deep net architectures [1, 20], which produce full-resolution pixel-wise prediction maps, make deep neural networks particularly well-suited for this task. Such advances

have spurred a flurry of research into deep methods for depth estimation, whether through fusing CRFs with deep nets [37], leveraging geometry and stereo consistency [5, 16], or exploring novel deep architectures [17].

Depth in computer vision is often used as a component for performing other perception tasks. One of the first approaches to deep depth estimation also simultaneously estimates surface normals and segmentation in a multitask architecture [4]. Other multitask vision networks [3, 12, 34] also commonly use depth as a complementary output to benefit overall network performance. Using depth as an explicit input is also common in computer vision, with plentiful applications in tracking [30, 33], SLAM systems [13, 36] and 3d reconstruction/detection [7, 19]. There is clearly a pressing demand for high-quality depth maps, but current depth hardware solutions are power-hungry, have severe range limitations [8], and the current traditional depth estimation methods [4, 17] fail to achieve the accuracies necessary to supersede such hardware.

Such challenges naturally lead to depth densification, a middle ground that combines the power of deep learning with energy-efficient sparse depth sensors. Depth densification is related to depth superresolution [10, 31], but superresolution generally uses a bilinear or bicubic downsampled depth map as input, and thus still implicitly contains information from all pixels in the low-resolution map. This additional information would not be accessible to a true sparse sensor, and tends to make the estimation problem easier (see Supplementary Material). Work in [22] and [23] follows the more difficult densification paradigm where only a few pixels of measured depth are provided. We will show that our densification network outperforms the methods in both [22] and [23].

3 Methodology

3.1 Input Parametrization for Sparse Depth Inputs

We desire a parametrization of the sparse depth input that can accommodate arbitrary sparse input patterns. This should allow for varying such patterns not only across different deep models but even within the same model during training and testing. Therefore, rather than directly feeding a highly discontinuous sparse depth map into our deep depth densification (D^3) model (as in Fig. 1), we propose a more flexible parametrization of the sparse depth inputs.

At each training step, the inputs to our parametrization are:

1. $I(x, y)$ and $D(x, y)$: RGB vector-valued image I and ground truth depth D. Both maps have dimensions H×W. Invalid values in D are encoded as zero.
2. $M(x, y)$: Binary pattern mask of dimensions H×W, where $M(x, y) = 1$ defines (x, y) locations of our desired depth samples. $M(x, y)$ is preprocessed so that all points where $M(x, y) = 1$ must correspond to valid depth points ($D(x, y) > 0$). (see Algorithm 1).

From I, D, and M, we form *two maps* for the sparse depth input, $\mathcal{S}_1(x, y)$ and $\mathcal{S}_2(x, y)$. Both maps have dimension H×W (see Fig. 2 for examples).

Algorithm 1 Sparse Inputs for the Deep Depth Densification (D^3) Model

INPUT image $I(x, y)$, depth $D(x, y)$, and pattern mask $M(x, y)$.
INITIALIZE $\mathcal{S}_1(x, y) = 0$, $\mathcal{S}_2(x, y) = 0$ for all (x, y).
FOR $\mathbf{r} := (x, y)$ s.t. $D(\mathbf{r}) = 0$ AND $M(\mathbf{r}) = 1$:
  $\mathbf{r}_{new} = \text{argmin}_{\mathbf{r}'} ||\mathbf{r} - \mathbf{r}'||_2$ s.t. $D(\mathbf{r}') > 0$; ($||\cdot||_2$ *denotes the* L_2 *norm.*)
  $M(\mathbf{r}) = 0$; $M(\mathbf{r}_{new}) = 1$;
ENDFOR (*All depth locations are now valid.*)
FOR $\mathbf{r} := (x, y)$:
  $\mathbf{r}_{\text{nearest}} = \text{argmin}_{\mathbf{r}'} ||\mathbf{r}' - \mathbf{r}||_2$ s.t. $M(\mathbf{r}') = 1$;
  $\mathcal{S}_1(x, y) = D(\mathbf{r}_{nearest})$; $\mathcal{S}_2(x, y) = \sqrt{||\mathbf{r}_{nearest} - \mathbf{r}||_2}$;
ENDFOR
OUTPUT **concatenate**$(\mathcal{S}_1, \mathcal{S}_2)$

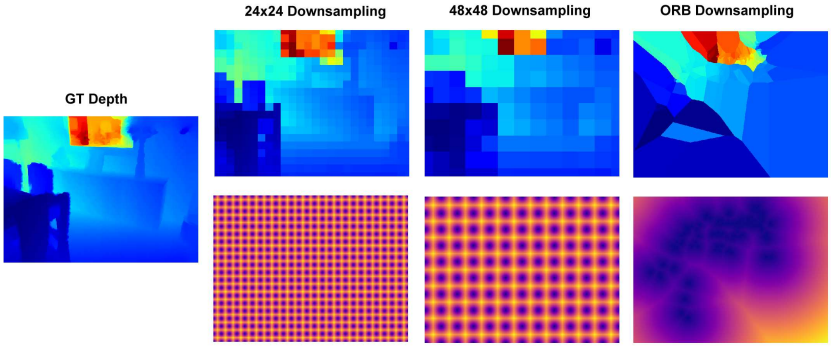


Fig. 2: **Various Sparse Patterns.** NN fill maps $\mathcal{S}_1$ (top row) and the sampling pattern Euclidean Distance transforms $\mathcal{S}_2$ (bottom row) are shown for both regular and irregular sparse patterns. Dark points in $\mathcal{S}_2$ correspond to the pixels where we have access to depth information.

- $\mathcal{S}_1(x, y)$ is a NN (nearest neighbor) fill of the sparse depth $M(x, y) * D(x, y)$.
- $\mathcal{S}_2(x, y)$ is the Euclidean Distance Transform of $M(x, y)$, i.e. the L_2 distance between (x,y) and the closest point (x',y') where $M(x', y') = 1$.

The final parametrization of the sparse depth input is the concatenation of $\mathcal{S}_1(x, y)$ and $\mathcal{S}_2(x, y)$, with total dimension H×W×2. This process is described in Algorithm 1. The parametrization is fast and involves at most two Euclidean Transforms. The resultant NN map $\mathcal{S}_1$ is nonzero everywhere, allowing us to treat the densification problem as a *residual prediction* with respect to $\mathcal{S}_1$. The distance map $\mathcal{S}_2$ informs the model about the pattern mask $M(x, y)$ and acts as a prior on the residual magnitudes the model should output (i.e. points farther from a pixel with known depth tend to incur higher residuals). Inclusion of $\mathcal{S}_2$ can substantially improve model performance and training stability, especially when multiple sparse patterns are used during training (see Section 5.3).

In this work, we primarily focus on regular grid patterns, as they are high-coverage sparse maps that enable straightforward comparisons to prior work (as in [22]) which often assume a grid-like sparse pattern, but our methods fully generalize to other patterns like ORB (see Supplementary Materials).

3.2 Sparse Pattern Selection

For regular grid patterns, we try to ensure minimal spatial bias when choosing the pattern mask $M(x, y)$ by enforcing equal spacing between subsequent pattern points in both the x and y directions. This results in a checkerboard pattern of square regions in the sparse depth map $\mathcal{S}_1$ (see Fig. 2). Such a strategy is convenient when one deep model must accommodate images of different resolutions, as we can simply extend the square pattern in $M(x, y)$ from one resolution to the next. For ease of interpretation, we will always use sparse patterns close to an integer level of downsampling; for a downsampling factor of $A \times A$, we take $\sim H * W/A^2$ depth values as the sparse input. For example, for 24×24 downsampling on a 480×640 image, this would be 0.18% of the total pixels.

Empirically we observed that it is beneficial to vary the sparse pattern $M(x, y)$ during training. For a desired final pattern of N sparse points, we employ a slow decay learning schedule following $N_{\text{sparse}}(t) = \lfloor 5Ne^{-0.0003t} + N \rfloor$ for training step $0 \leq t \leq 80000$. Such a schedule begins training at six times the desired sparse pattern density and smoothly decays towards the final density as training progresses. Compared to a static sparse pattern, we see a relative decrease of $\sim 3\%$ in the training L_2 loss and also in the mean relative error when using this decay schedule. We can also train with randomly varying sampling densities at each training step. This we show in Section 5.3 results in a deep model which performs well *simultaneously* at different sampling densities.

4 Experimental Setup

4.1 Architecture

We base our network architecture (see Fig. 3) on the network used in [2] but with DenseNet [9] blocks in place of Inception [32] blocks. We empirically found it critical for our proposed model to carry the sparse depth information throughout the deep network, and the residual nature of DenseNet is well-suited for this requirement. For optimal results, our architecture retains feature maps at multiple resolutions for addition back into the network during the decoding phase.

Each block in Fig. 3 represents a DenseNet Module (see Fig. 3 inset for a precise module schematic) except for the first and last blocks, which are simple 3x3 stride-2 convolutional layers. A copy of the sparse input $[\mathcal{S}_1, \mathcal{S}_2]$ is presented as an additional input to each module, downsampled to the appropriate resolution. Each DenseNet module consists of $2L$ layers and k feature maps per layer; we use $L = 5$ and $k = 12$. At downsample/upsample blocks, the final convolution has stride 2. The (residual) output of the network is added to the sparse input map $\mathcal{S}_1$ to obtain the final depth map estimate.

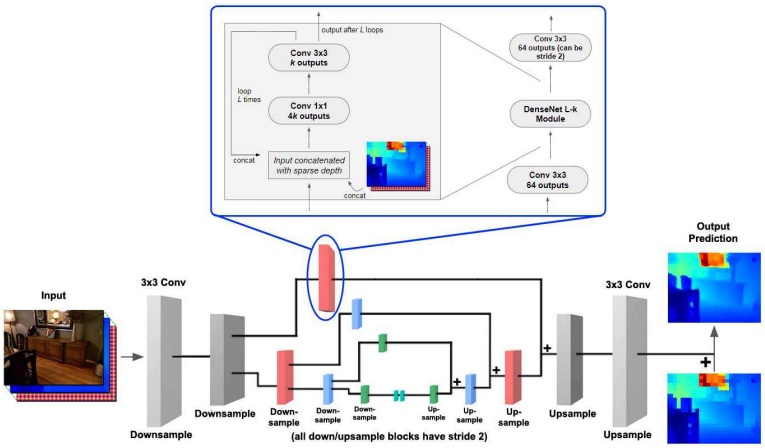


Fig. 3: **D^3 Network Architecture.** Our proposed multi-scale deep network takes an RGB image concatenated with $\mathcal{S}_1$ and $\mathcal{S}_2$ as inputs. The first and last computational blocks are simple 3x3 stride-2 convolutions, but all other blocks are DenseNet modules [9] (see inset). All convolutional layers in the network are batch normalized [11] and ReLU activated. The network outputs a residual that is added to the sparse depth map $\mathcal{S}_1$ to produce the final dense depth prediction.

4.2 Datasets

We experiment extensively with both indoor and outdoor scenes. For indoor scenes, we use the NYUv2 [24] dataset, which provides high-quality 480×640 depth data taken with a Kinect V1 sensor with a range of up to 10m. Missing depth values are filled using a standard approach [18]. We use the official split of 249/215 train/validation scenes, and sample 26331 images from the training scenes. We further augment the training set with horizontal flips. We test on the standard validation set of 654 images to compare against other methods.

For outdoor scenes, we use the KITTI road scenes dataset [35], which has a depth range up to ∼85m. KITTI provides over 80000 images for training, which we further augment with horizontal flips. We test on the full validation set (∼10% the size of the training set). KITTI images have resolution 1392×512, but we take random 480×640 crops during training to enable joint training with NYUv2 data. The 640 horizontal pixels are sampled randomly while the 480 vertical pixels are the 480 bottom pixels of the image (as KITTI only provides LiDAR GT depth towards ground level). The LiDAR projections used in KITTI result in very sparse depth maps (with only ∼10% of depths labeled per image), and we only evaluate our models on points with GT depth.

4.3 General Training Characteristics and Performance Metrics

In all our experiments we train with a batch size of 8 across 4 Maxwell Titan X GTX GPUs using Tensorflow 1.2.1. We train for 80000 batches and start with

a learning rate of 1e-3, decaying the learning rate by 0.2 every 25000 steps. We use Adam [15] as our optimizer, and standard pixel-wise L_2 loss to train.

Standard metrics are used [4, 23] to evaluate our depth estimation model against valid GT depth values. Let $\hat{y}$ be the predicted depth and y the GT depth for N pixels in the dataset. We measure: (1) Root Mean Square Error (RMSE): $\sqrt{\frac{1}{N}\sum[\hat{y}-y]^2}$, (2) Mean Absolute Relative Error (MRE): $\frac{100}{N}\sum\left(\frac{|\hat{y}-y|}{y}\right)$, and (3) Delta Thresholds (δ_i): $\frac{|\{\hat{y}|\max(\frac{y}{\hat{y}},\frac{\hat{y}}{y})<1.25^i\}|}{|\{\hat{y}\}|}$. δ_i is the percentage of pixels with relative error under a threshold controlled by the constant i.

5 Results and Analysis

Here we present results and analysis of the D^3 model for both indoor (NYUv2) and outdoor (KITTI) datasets. We further demonstrate that D^3 is robust to input errors and also generalizes to multiple sparse input patterns.

5.1 Indoor scenes from NYUv2

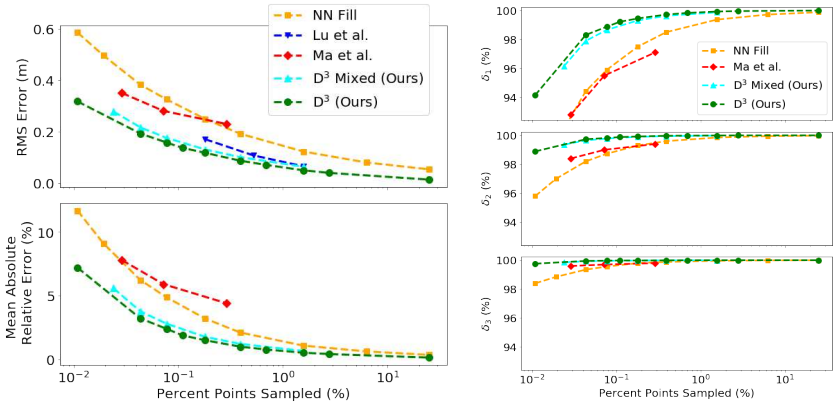


Fig. 4: **Performance on the NYUv2 Dataset.** RMSE and MRE are plotted on the left (lower is better), while the δ_i are plotted on the right (higher is better). Our D^3 models achieve the best performance at all sparsities, while joint training on outdoor data (D^3 mixed) only incurs a minor performance loss.

From Table 1 we see that, at all pattern sparsities, the D^3 network offers superior performance for all metrics[1] compared to the results in [23] and in [22][2].

[1] A model trained with 0.18% sparsity performs very well on a larger NYUv2 test set of 37K images: RMSE 0.116m/ MRE 1.34%/δ_1 99.52%/δ_2 99.93%/δ_3 99.986%.

[2] As results in [22] were computed on a small subset of the NYUv2 val set, metrics were normalized to each work's reported NN fill RMSE to ensure fair comparison.

Table 1: **D^3 Performance on NYUv2**. Lower RMSE and MRE is better, while higher δ_i is better. NN Fill corresponds to using the sparse map $\mathcal{S}_1$ as our final prediction. If no sparse depth is provided, the D^3 model falls short of [4] and [17], but even at 0.01% points sampled the D^3 model offers significant improvements over state-of-the-art non-sparse methods. D^3 additionally performs the best compared to other sparse depth methods at all input sparsities.

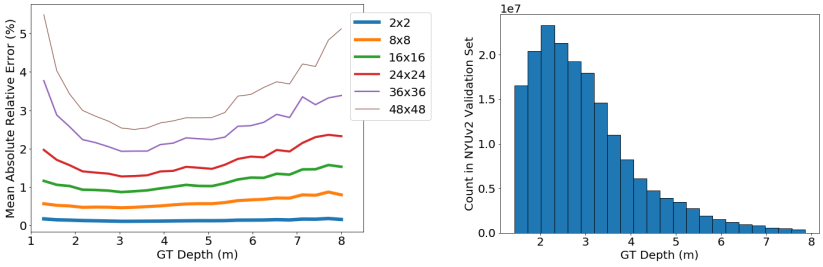
Model	% Points Sampled	Downsampling Factor	RMSE (m)	MRE (%)	δ_1 (%)	δ_2 (%)	δ_3 (%)
Eigen et al. [4]	0	N/A	0.641	15.8	76.9	95.0	98.8
Laina et al. [17]	0	N/A	**0.573**	**12.7**	**81.1**	**95.3**	**98.8**
D^3 No Sparse	0	N/A	0.711	22.37	67.32	89.68	96.73
NN Fill	0.011	96×96	0.586	11.69	86.8	95.8	98.4
D^3 (Ours)	0.011	96×96	**0.318**	**7.20**	**94.2**	**98.9**	**99.8**
Ma et al. [23]	0.029	∼59×59	0.351	7.8	92.8	98.4	99.6
NN Fill	0.043	48×48	0.383	6.23	94.42	98.20	99.35
D^3 Mixed (Ours)	0.043	48×48	0.217	3.77	97.90	99.65	99.93
D^3 (Ours)	0.043	48×48	**0.193**	**3.21**	**98.31**	**99.73**	**99.95**
NN Fill	0.174	24×24	0.250	3.20	97.5	99.3	99.8
Lu et al. [22]	-	24×24	0.171	-	-	-	-
D^3 Mixed (Ours)	0.174	24×24	0.131	1.76	99.31	99.90	99.98
D^3 (Ours)	0.174	24×24	**0.118**	**1.49**	**99.45**	**99.92**	**99.98**
Ma et al. [23]	0.289	∼19×19	0.23	4.4	97.1	99.4	99.8
NN Fill	0.391	16×16	0.192	2.10	98.5	99.6	99.88
Lu et al. [22]	-	16×16	0.108	-	-	-	-
D^3 (Ours)	0.391	16×16	**0.087**	**0.99**	**99.72**	**99.97**	**99.99**

The accuracy metrics for the D^3 *mixed* network represent the NYUv2 results for a network that has been simultaneously trained on the NYUv2 (indoors) and KITTI (outdoors) datasets (more details in Section 5.4). We see that incorporating an outdoors dataset with significantly different semantics only incurs a mild degradation in accuracy. Fig. 4 has comparative results for additional sparsities, and once again demonstrates that our trained models are more accurate than other recent approaches.

At 16×16 downsampling our absolute mean relative error falls below 1% (at 0.99%). At this point, the error of our D^3 model becomes comparable to the error in consumer-grade depth sensors [8]. Fig. 5(a) presents a more detailed plot of relative error at different values of GT depth. Our model performs well at most common indoor depths (around 2-4m), as can be assessed from the histogram in Fig. 5(b). At farther depths the MRE deteriorates, but these depth values are rarer in the dataset. This suggests that using a more balanced dataset can improve those MRE values as well.

Table 2: **Timing for D^3 and other architectures.** Models are evaluated assuming 0.18% sparsity and using 1 Maxwell Titan X. The D^3 network achieves the lowest RMSE compared to other well known efficient network architectures. A slim version of D^3 runs at a near real-time 16fps for VGA resolution.

Model	L	k	RMSE (m)	FPS	Forward Pass (s)	Model	L	k	RMSE (m)	FPS	Forward Pass (s)
D^3	5	12	**0.118**	10	0.11	SegNet [1]	-	-	0.150	5	0.20
D^3	3	8	0.127	13	0.08	ENet [26]	-	-	0.237	25	0.04
D^3	2	6	0.131	16	0.06						



(a) (b)

Fig. 5: **NYUv2 MRE performance at different depths.** (a) MRE at different depths for varying levels of sparsity. At 0.39% sparsity the average MRE is less than 1% which is comparable to depth sensors. (b) Histogram of GT depths in the validation dataset; higher relative errors correspond to rarer depth values.

Visualizations of our network predictions on the NYUv2 dataset are shown in Fig. 6. At a highly sparse 48×48 downsampling, our D^3 network already shows a dramatic improvement over a vanilla network without any sparse input. We note here that although network outputs are added as residuals to a sparse map with many first order discontinuities, the final predictions appear smooth and relatively free of sharp edge artifacts. Indeed, in the final column of Fig. 6, we can see how the direct residual predictions produced by our networks also contain sharp features which cancels out the non-smoothness in the sparse maps.

5.2 Computational Analysis

In Table 2 we show the forward pass time and accuracy for a variety of models at 0.18% points sampled. Our standard D^3 model with $L = 5$ and $k = 12$ achieves the lowest error and takes 0.11s per VGA frame per forward pass. Slimmer versions of the D^3 network incur mild accuracy degradation but still outperform other well known efficient architectures [1, 26]. The baseline speed for our D^3

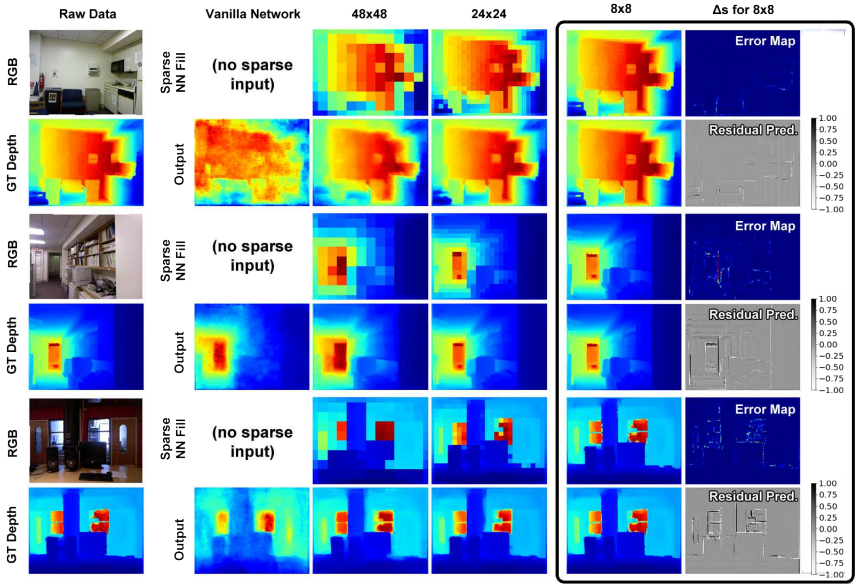
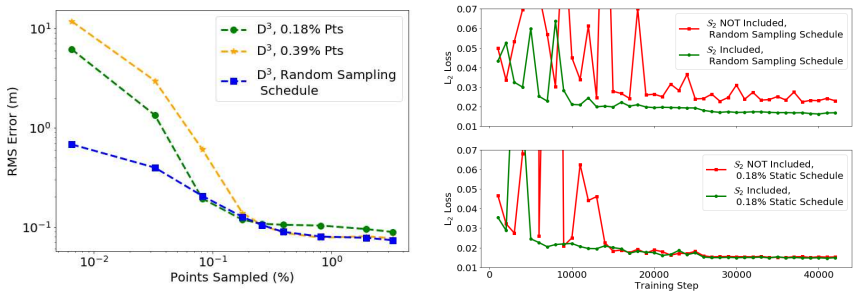


Fig. 6: **Visualization of D^3 Predictions on NYUv2.** Left column: Sample RGB and GT depths. Middle columns: sparse $\mathcal{S}_1$ map on top and D^3 network prediction on bottom for different sparsities. Vanilla network denotes the case with no sparse input (monocular depth estimation). Final column: D^3 residual predictions (summed with $\mathcal{S}_1$ to obtain the final prediction) and error maps of the final estimate with respect to GT. Errors are larger at farther distances. Residuals are plotted in grayscale and capped at $|\delta| \leq 1$ for better visualization; they exhibit similar sharp features as $\mathcal{S}_1$, showing how a D^3 model cancels out the nonsmoothness of $\mathcal{S}_1$.

networks can thus approach real-time speeds for full-resolution 480×640 inputs, and we expect these speeds can be further improved by weight quantization and other optimization methods for deep networks [6]. Trivially, operating at half resolution would result in our slimmer D^3 networks operating at a real-time speed of >60fps. This speed is important for many application areas where depth is a critical component for scene understanding.

5.3 Generalizing D^3 to Multiple Patterns and the Effect of $\mathcal{S}_2$

We train a D^3 network with a different input sparsity (sampled uniformly between 0.065% and 0.98% points) for each batch. Fig. 7(a) shows how this multi-sparsity D^3 network performs relative to the 0.18% and 0.39% sparsity models. The single-sparsity trained D^3 networks predictably perform the best near the sparsity they were tuned for. However, the multi-sparsity D^3 network only performs slightly worse at those sparsities, and dramatically outperforms the single-sparsity networks away from their training sparsity value. Evidently, a random



(a) (b)

Fig. 7: **Multi-sparsity D^3 models.** (a) The Random Sampling network was trained with a different sparse pattern (between 0.065% and 0.98% points sampled) on every iteration, and performs well at all sparsity levels while being only mildly surpassed by single-density networks at their specialized sparsities. (b) Validation loss curves (for 0.18% points sampled) for a D^3 models trained with and without inclusion of distance map $\mathcal{S}_2$. $\mathcal{S}_2$ is clearly crucial for stability and performance, especially when training with complex pattern schedules.

sampling schedule effectively regularizes our model to work *simultaneously at all sparsities within a wide range*. This robustness may be useful in scenarios where different measurement modes are used in the same device.

Inclusion of the distance map $\mathcal{S}_2$ gives our network spatial information of the sparse pattern, which is especially important when the sparse pattern changes during training. Fig. 7(b) shows validation L_2 loss curves for D^3 networks trained with and without $\mathcal{S}_2$. $\mathcal{S}_2$ improves relative L_2 validation loss by 34.4% and greatly stabilizies training when the sparse pattern is varied randomly during training. For a slow decay sampling schedule (i.e. what is used for the majority of our D^3 networks), the improvement is 8.8%, and even for a static sampling schedule (bottom of Fig. 7(b)) there is a 2.8% improvement. The inclusion of the distance map is thus clearly essential to train our model well.

5.4 Generalizing D^3 to Outdoor Scenes

We extend our model to the challenging outdoor KITTI dataset [35]. All our KITTI D^3 models are initialized to a pre-trained NYUv2 model. We then train either with only KITTI data (KITTI-exclusive) or with a 50/50 mix of NYUv2 and KITTI data for each batch (mixed model). Since NYUv2 images have a max depth of 10m, *depth values are scaled by 0.1* for the KITTI-exclusive model. For the mixed model we use a scene-agnostic scaling rule; we scale all images down to have a max depth of $\leq$10m, and invert this scaling at inference. Our state-of-the-art results are shown in Table 3. Importantly, as for NYUv2, our mixed model only performs slightly worse than the KITTI-exclusive network. More results for additional sparsities are presented in the Supplementary Material.

Table 3: **D^3 Model Performance on the KITTI Dataset.** Lower values of RMSE and MRE are better, while higher values of δ_i are better. For competing methods we show results at the closest sparsity. The performance of our models, including the mixed models, is superior by a large margin.

Model	% Points Sampled	Downsample Factor	RMSE (m)	MRE (%)	δ_1 (%)	δ_2 (%)	δ_3 (%)
NN Fill	0.077	36×36	4.441	9.306	91.88	97.75	99.04
D^3 Mixed (Ours)	0.077	36×36	1.906	3.14	98.62	99.65	99.88
D^3 (Ours)	0.077	36×36	**1.600**	**2.50**	**99.12**	**99.76**	**99.91**
Ma et al. [23]	0.096	∼32×32	3.851	8.3	91.9	97.0	98.6
NN Fill	0.174	24×24	3.203	5.81	96.62	99.03	99.57
D^3 Mixed (Ours)	0.174	24×24	1.472	2.22	99.30	99.83	99.94
D^3 (Ours)	0.174	24×24	**1.387**	**2.09**	**99.40**	**99.85**	**99.95**
Ma et al. [23]	0.240	∼20×20	3.378	7.3	93.5	97.6	98.9
NN Fill	0.391	16×16	2.245	3.73	98.67	99.60	99.81
D^3 Mixed (Ours)	0.391	16×16	1.120	1.62	99.67	99.92	99.97
D^3 (Ours)	0.391	16×16	**1.008**	**1.42**	**99.76**	**99.94**	**99.98**

Visualizations of the our model outputs are shown in Fig. 8. The highlight here is that the mixed model produces high-quality depth maps for both NYUv2 and KITTI. Interestingly, even the KITTI-exclusive model (bottom row of Fig. 8) produces good qualitative results on the NYUv2 dataset. Perhaps more strikingly, even an NYUv2 pretrained model with no KITTI data training (third-to-last row of Fig. 8) produces reasonable results on KITTI. This suggests that our D^3 models intrinsically possess some level of cross-domain generalizability.

5.5 Robustness Tests

Thus far, we have sampled depth from high-quality Kinect and LiDAR depth maps, but in practice sparse depth inputs may come from less reliable sources. We now demonstrate how our D^3 network performs given the following common errors within the sparse depth input:

1. Spatial misregistration between the RGB camera and depth sensor.
2. Random gaussian error.
3. Random holes (dropout), e.g. due to shadows, specular reflection, etc.

In Fig. 9 we show examples of each of these potential sources of error, and in Fig. 10 we show how D^3 performs when trained with such errors in the sparse depth input (see Supplementary Material for for tabulated metrics). The D^3 network degrades gracefully under all sources of error, with most models still

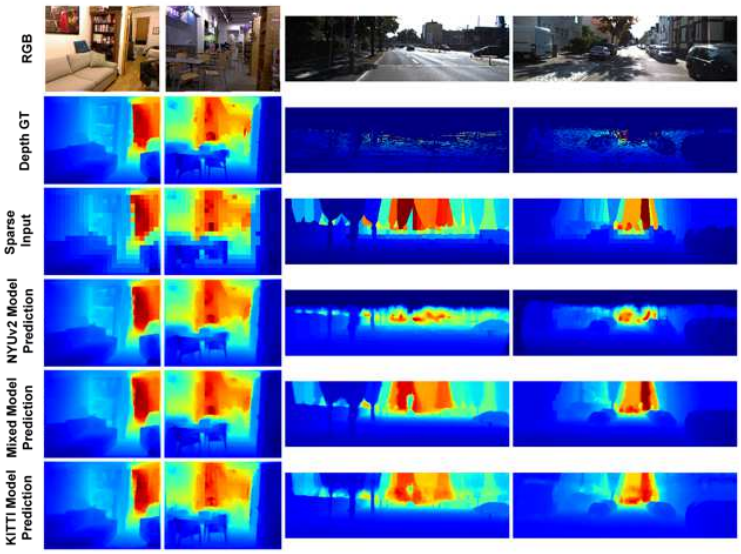


Fig. 8: **Joint Predictions on NYUv2 and KITTI.** The RGB, Depth GT, and Sparse Input $\mathcal{S}_1$ are given in the first three rows. Predictions by three models on both indoors and outdoors scenes are given in the final three rows, with the second-to-last row showing the mixed model trained on both datasets simultaneously. All sparse maps have a density of 0.18% (∼24×24 downsampling).

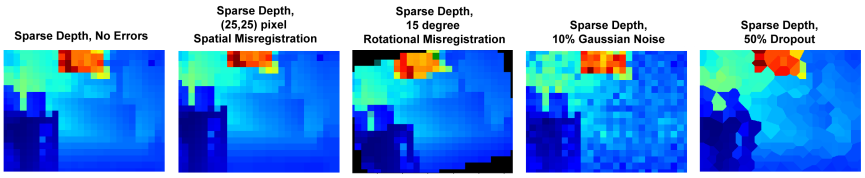


Fig. 9: **Potential Errors in Sparse Depth**. The three sparse depth maps on the right all exhibit significant errors that are common in real sensors.

outperforming the other baselines in Table 1 (none of which were subject to input errors). It is especially encouraging that network performs robustly under constant mis-registration error, a very common issue when multiple imaging sensors are active in the same visual system. The network effectively learns to fix the calibration between the different visual inputs. Predictably, the error is much higher when the mis-registration is randomly varying per image.

5.6 Discussion

Through our experiments, we've shown how the D^3 model performs very well at taking a sparse depth measurement in a variety of settings and turning it into

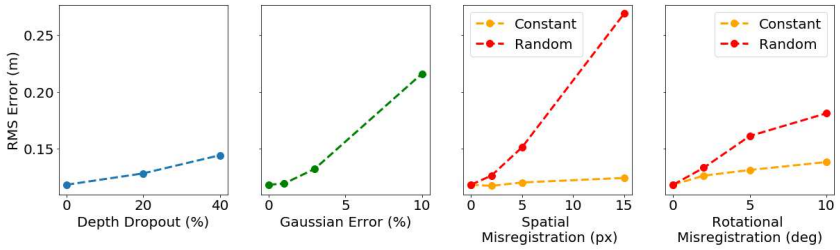


Fig. 10: **Accuracy of D^3 Networks under Various Sparse Depth Errors**. Under all potential error sources (with the exception of the unlikely random spatial mis-registration error), the D^3 network exhibits graceful error degradation. This error degradation is almost negligible for constant spatial mis-registration.

a dense depth map. Most notably, our model can simultaneously perform well both on indoor and outdoor scenes. We attribute the overall performance of the model to a number of factors. As can be gathered from Table 2, the design of our multi-scale architecture, in which the sparse inputs are ingested at various scales and outputs are treated as residuals with respect to $\mathcal{S}_1$, is important for optimizing performance. Our proposed sparse input parameterization clearly allows for better and more stable training as seen in Fig. 7. Finally, the design of the training curriculum, in which we use varying sparsities in the depth input during training, also plays an important role. Such a strategy makes the model robust to test time variations in sparsity (see Fig. 7) and reduces overall errors.

6 Conclusions

We have demonstrated that a trained deep depth densification (D^3) network can use sparse depth information and a registered RGB image to produce high-quality, dense depth maps. Our flexible parametrization of the sparse depth information leads to models that generalize readily to multiple scene types (working simultaneously on indoor and outdoor images, from depths of 1m to 80m) and to diverse sparse input patterns. Even at fairly aggressive sparsities for indoor scenes, we achieve a mean absolute relative error of under 1%, comparable to the performance of consumer-grade depth sensor hardware. We also found that our model is fairly robust to various input errors.

We have thus shown that sparse depth measurements can be sufficient for applications that require an RGBD input, whether indoors or outdoors. A natural next step in our line of inquiry would be to evaluate how densified depth maps perform in 3d-reconstruction algorithms, tracking systems, or perception models for related vision tasks such as surface normal prediction. We hope that our work motivates additional research into uses for sparse depth from both the software and hardware perspectives.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495 (2017)
2. Chen, W., Fu, Z., Yang, D., Deng, J.: Single-image depth perception in the wild. In: Advances in Neural Information Processing Systems. pp. 730–738 (2016)
3. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. arXiv preprint arXiv:1711.02257 (2017)
4. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2650–2658 (2015)
5. Garg, R., BG, V.K., Carneiro, G., Reid, I.: Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: European Conference on Computer Vision. pp. 740–756. Springer (2016)
6. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
7. Hermans, A., Floros, G., Leibe, B.: Dense 3d semantic mapping of indoor scenes from rgb-d images. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on. pp. 2631–2638. IEEE (2014)
8. Horaud, R., Hansard, M., Evangelidis, G., Ménier, C.: An overview of depth cameras and range scanners based on time-of-flight technologies. Machine Vision and Applications **27**(7), 1005–1020 (2016)
9. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. vol. 1, p. 3 (2017)
10. Hui, T.W., Loy, C.C., Tang, X.: Depth map super-resolution by deep multi-scale guidance. In: European Conference on Computer Vision. pp. 353–369. Springer (2016)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456 (2015)
12. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. arXiv preprint arXiv:1705.07115 (2017)
13. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for rgb-d cameras. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. pp. 2100–2106. IEEE (2013)
14. Khoshelham, K., Elberink, S.O.: Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors **12**(2), 1437–1454 (2012)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Kuznietsov, Y., Stückler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6647–6655 (2017)
17. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3D Vision (3DV), 2016 Fourth International Conference on. pp. 239–248. IEEE (2016)

18. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: ACM Transactions on Graphics (ToG). vol. 23, pp. 689–694. ACM (2004)
19. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3d object detection with rgbd cameras. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 1417–1424. IEEE (2013)
20. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
21. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
22. Lu, J., Forsyth, D.A., et al.: Sparse depth super resolution. In: CVPR. vol. 6 (2015)
23. Ma, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. arXiv preprint arXiv:1709.07492 (2017)
24. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
25. Nguyen, C.V., Izadi, S., Lovell, D.: Modeling kinect sensor noise for improved 3d reconstruction and tracking. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on. pp. 524–530. IEEE (2012)
26. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
27. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE international conference on. pp. 2564–2571. IEEE (2011)
28. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: Advances in neural information processing systems. pp. 1161–1168 (2006)
29. Sinz, F.H., Candela, J.Q., Bakır, G.H., Rasmussen, C.E., Franz, M.O.: Learning depth from stereo. In: Joint Pattern Recognition Symposium. pp. 245–252. Springer (2004)
30. Song, S., Xiao, J.: Tracking revisited using rgbd camera: Unified benchmark and baselines. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 233–240. IEEE (2013)
31. Song, X., Dai, Y., Qin, X.: Deep depth super-resolution: Learning depth super-resolution using deep convolutional neural network. In: Asian Conference on Computer Vision. pp. 360–376. Springer (2016)
32. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions. Cvpr (2015)
33. Teichman, A., Lussier, J.T., Thrun, S.: Learning to segment and track in rgbd. IEEE Transactions on Automation Science and Engineering **10**(4), 841–852 (2013)
34. Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., Urtasun, R.: Multinet: Real-time joint semantic reasoning for autonomous driving. arXiv preprint arXiv:1612.07695 (2016)
35. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. In: International Conference on 3D Vision (3DV) (2017)
36. Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J.J., McDonald, J.: Real-time large-scale dense rgb-d slam with volumetric fusion. The International Journal of Robotics Research **34**(4-5), 598–626 (2015)
37. Xu, D., Ricci, E., Ouyang, W., Wang, X., Sebe, N.: Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In: Proceedings of CVPR (2017)