Deep Depth Completion of a Single RGB-D Image

Yinda Zhang
Princeton University

Thomas Funkhouser
Princeton University

Abstract

The goal of our work is to complete the depth channel of an RGB-D image. Commodity-grade depth cameras often fail to sense depth for shiny, bright, transparent, and distant surfaces. To address this problem, we train a deep network that takes an RGB image as input and predicts dense surface normals and occlusion boundaries. Those predictions are then combined with raw depth observations provided by the RGB-D camera to solve for depths for all pixels, including those missing in the original observation. This method was chosen over others (e.g., inpainting depths directly) as the result of extensive experiments with a new depth completion benchmark dataset, where holes are filled in training data through the rendering of surface reconstructions created from multiview RGB-D scans. Experiments with different network inputs, depth representations, loss functions, optimization methods, inpainting methods, and deep depth estimation networks show that our proposed approach provides better depth completions than these alternatives.

1. Introduction

Depth sensing has become pervasive in applications as diverse as autonomous driving, augmented reality, and scene reconstruction. Despite recent advances in depth sensing technology, commodity-level RGB-D cameras like Microsoft Kinect, Intel RealSense, and Google Tango still produce depth images with missing data when surfaces are too glossy, bright, thin, close, or far from the camera. These problems appear when rooms are large, surfaces are shiny, and strong lighting is abundant – e.g., in museums, hospitals, classrooms, stores, etc. Even in homes, depth images often are missing more than 50% of the pixels (Figure 1).

The goal of our work is to complete the depth channel of an RGB-D image captured with a commodity camera (i.e., fill all the holes). Though depth inpainting has received a lot of attention over the past two decades, it has generally been addressed with hand-tuned methods that fill holes by extrapolating boundary surfaces [51] or with Markovian image synthesis [16]. Newer methods have been proposed to estimate depth de novo from color using deep networks

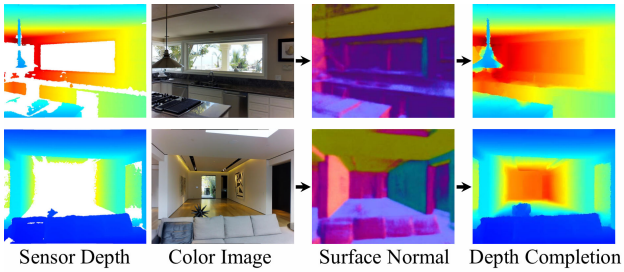


Figure 1. **Depth Completion.** We fill in large missing areas in the depth channel of an RGB-D image by predicting normals from color and then solving for completed depths.

[19]. However, they have not been used for depth completion, which has its own unique challenges:

Training data: Large-scale training sets are not readily available for captured RGB-D images paired with "completed" depth images (e.g., where ground-truth depth is provided for holes). As a result, most methods for depth estimation are trained and evaluated only for pixels that are captured by commodity RGB-D cameras [64]. From this data, they can at-best learn to reproduce observed depths, but not complete depths that are unobserved, which have significantly different characteristics. To address this issue, we introduce a new dataset with 105,432 RGB-D images aligned with completed depth images computed from large-scale surface reconstructions in 72 real-world environments.

Depth representation: The obvious approach to address our problem is to use the new dataset as supervision to train a fully convolutional network to regress depth directly from RGB-D. However, that approach does not work very well, especially for large holes like the one shown in the bottom row of Figure 1. Estimating absolute depths from a monocular color image is difficult even for people [53]. Rather, we train the network to predict only local differential properties of depth (surface normals and occlusion boundaries), which are much easier to estimate [35]. We then solve for the absolute depths with a global optimization.

Deep network design: There is no previous work on studying how best to design and train an end-to-end deep network for completing depth images from RGB-D inputs. At first glance, it seems straight-forward to extend previous net-

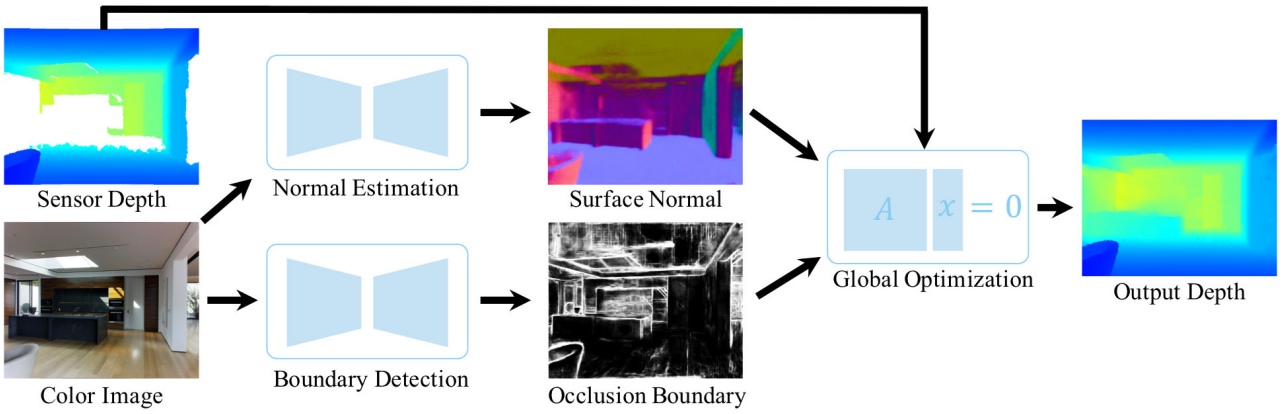


Figure 2. **System pipeline.** Given an input RGB-D image, we predict surface normals and occlusion boundaries from color, and then solve for the output depths with a global linear optimization regularized by the input depth.

works trained for color-to-depth (e.g., by providing them an extra depth channel as input). However, we found it difficult to train the networks to fill large holes from depth inputs – they generally learn only to copy and interpolate the input depth. It is also challenging for the network to learn how to adapt for misalignments of color and depth. Our solution is to provide the network with only color images as input (Figure 2). We train it to predict local surface normals and occlusion boundaries with supervision. We later combine those predictions with the input depths in a global optimization to solve back to the completed depth. In this way, the network predicts only local features from color, a task where it excels. The coarse-scale structure of the scene is reconstructed through global optimization with regularization from the input depth.

Overall, our main algorithmic insight is that it is best to decompose RGB-D depth completion into two stages: 1) prediction of surface normals and occlusion boundaries only from color, and 2) optimization of global surface structure from those predictions with soft constraints provided by observed depths. During experiments we find with this proposed approach has significantly smaller relative error than alternative approaches. It has the extra benefit that the trained network is independent of the observed depths and so does not need to be retrained for new depth sensors.

2. Related Work

There has been a large amount of prior work on depth estimation, inpainting, and processing.

Depth estimation. Depth estimation from a monocular color image is a long-standing problem in computer vision. Classic methods include shape-from-shading [78] and shape-from-defocus [67]. Other early methods were based on hand-tuned models and/or assumptions about surface orientations [31, 60, 61]. Newer methods treat depth estimation as a machine learning problem, most recently using deep networks [19, 73]. For example, Eigen et al. first used a multiscale convolutional network to regress from color images to depths [19, 18]. Laina et al. used a fully convolutional network architecture based on ResNet [37]. Liu et al. proposed a deep convolutional neural field model combining deep networks with Markov random fields [40]. Roy et al. combined shallow convolutional networks with regression forests to reduce the need for large training sets [59]. All of these methods are trained only to reproduce the raw depth acquired with commodity RGB-D cameras. In contrast, we focus on depth completion, where the explicit goal is to make novel predictions for pixels where the depth sensor has no return. Since these pixels are often missing in the raw depth, methods trained only on raw depth as supervision do not predict them well.

Depth inpainting. Many methods have been proposed for filling holes in depth channels of RGB-D images, including ones that employ smoothness priors [30], fast marching methods [25, 42], Navier-Stokes [6], anisotropic diffusion [41], background surface extrapolation [51, 54, 68], color-depth edge alignment [10, 77, 81], low-rank matrix completion [75], tensor voting [36], Mumford-Shah functional optimization [44], joint optimization with other properties of intrinsic images [4], and patch-based image synthesis [11, 16, 24]. Recently, methods have been proposed for inpainting *color* images with auto-encoders [70] and GAN architectures [58]. However, prior work has not investigated how to use those methods for inpainting of depth images. This problem is more difficult due to the absence of strong features in depth images and the lack of large training datasets, an issue addressed in this paper.

Depth super-resolution. Several methods have been proposed to improve the spatial resolution of depth images using high-resolution color. They have exploited a variety of approaches, including Markov random fields [48, 15, 46, 56, 63], shape-from-shading [27, 76], segmentation [45], and dictionary methods [21, 34, 49, 69]. Although some of these techniques may be used for depth completion, the challenges of super-resolution are quite different – there

the focus is on improving spatial resolution, where low-resolution measurements are assumed to be complete and regularly sampled. In contrast, our focus is on filling holes, which can be quite large and complex and thus require synthesis of large-scale content.

Depth reconstruction from sparse samples. Other work has investigated depth reconstruction from color images augmented with sparse sets of depth measurements. Hawe et al. investigated using a Wavelet basis for reconstruction [29]. Liu et al. combined wavelet and contourlet dictionaries [43]. Ma et al. showed that providing ∼100 well-spaced depth samples improves depth estimation over color-only methods by two-fold for NYUv2 [47], yet still with relatively low-quality results. These methods share some ideas with our work. However, their motivation is to reduce the cost of sensing in specialized settings (e.g., to save power on a robot), not to complete data typically missed in readily available depth cameras.

3. Method

In this paper, we investigate how to use a deep network to complete the depth channel of a single RGB-D image. Our investigation focuses on the following questions: "how can we get training data for depth completion?," "what depth representation should we use?," and "how should cues from color and depth be combined?."

3.1. Dataset

The first issue we address is to create a dataset of RGB-D images paired with completed depth images.

A straight-forward approach to this task would be to capture images with a low-cost RGB-D camera and align them to images captured simultaneously with a higher cost depth sensor. This approach is costly and time-consuming – the largest public datasets of this type cover a handful of indoor scenes (e.g., [57, 62, 75]).

Instead, to create our dataset, we utilize existing surface meshes reconstructed from multi-view RGB-D scans of large environments. There are several datasets of this type, including Matterport3D [8], ScanNet [12], SceneNN [32], and SUN3D [26, 72], to name a few. We use Matterport3D. For each scene, we extract a triangle mesh M with ∼1-6 million triangles per room from a global surface reconstruction using screened Poisson surface reconstruction [33]. Then, for a sampling of RGB-D images in the scene, we render the reconstructed mesh M from the camera pose of the image viewpoint to acquire a completed depth image D*. This process provides us with a set of RGB-D → D* image pairs without having to collect new data.

Figure 3 shows some examples of depth image completions from our dataset. Though the completions are not always perfect, they have several favorable properties for

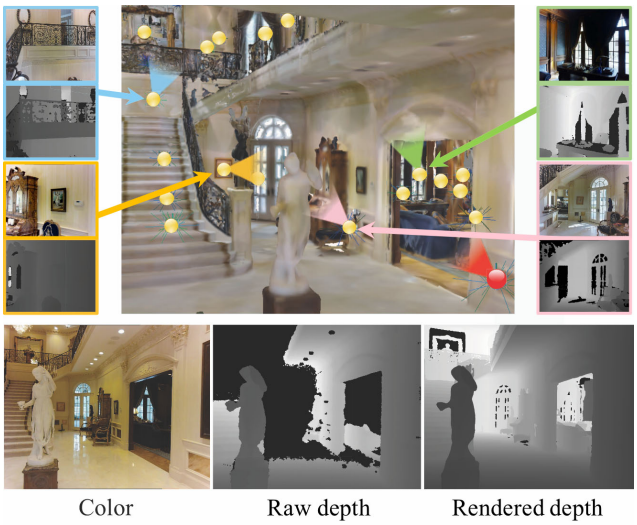


Figure 3. **Depth Completion Dataset.** Depth completions are computed from multi-view surface reconstructions of large indoor environments. In this example, the bottom shows the raw color and depth channels with the rendered depth for the viewpoint marked as the red dot. The rendered mesh (colored by vertex in large image) is created by combining RGB-D images from a variety of other views spread throughout the scene (yellow dots), which collaborate to fill holes when rendered to the red dot view.

training a deep network for our problem [52]. First, the completed depth images generally have fewer holes. That's because it is not limited by the observation of one camera viewpoint (e.g., the red dot in Figure 3), but instead by the union of all observations of all cameras viewpoints contributing to the surface reconstruction (yellow dots in Figure 3). As a result, surfaces distant to one view, but within range of another, will be included in the completed depth image. Similarly, glossy surfaces that provide no depth data when viewed at a grazing angle usually can be filled in with data from other cameras viewing the surface more directly (note the completion of the shiny floor in rendered depth). On average, 64.6% of the pixels missing from the raw depth images are filled in by our reconstruction process.

Second, the completed depth images generally replicate the resolution of the originals for close-up surfaces, but provide far better resolution for distant surfaces. Since the surface reconstructions are constructed at a 3D grid size comparable to the resolution of a depth camera, there is usually no loss of resolution in completed depth images. However, that same 3D resolution provides an effectively higher pixel resolution for surfaces further from the camera when projected onto the view plane. As a result, completed depth images can leverage subpixel antialiasing when rendering high resolution meshes to get finer resolution than the originals (note the detail in the furniture in Figure 3).

Finally, the completed depth images generally have far less noise than the originals. Since the surface reconstruc-

tion algorithm combines noisy depth samples from many camera views by filtering and averaging, it essentially denoises the surfaces. This is especially important for distant observations (e.g., >4 meters), where raw depth measurements are quantized and noisy.

In all, our dataset contains 117,516 RGB-D images with rendered completions, which we split into a training set with 105,432 images and a test set with 12,084 images.

3.2. Depth Representation

A second interesting question is "what geometric representation is best for deep depth completion?"

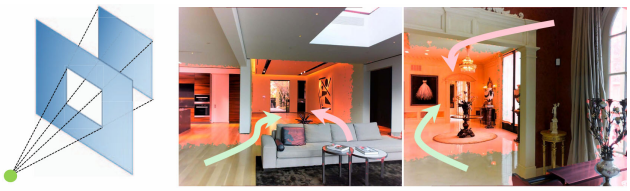
A straight-forward approach is to design a network that regresses completed depth from raw depth and color. However, absolute depth can be difficult to predict from monocular images, as it may require knowledge of object sizes, scene categories, etc. Instead, we train the network to predict local properties of the visible surface at each pixel and then solve back for the depth from those predictions.

Previous work has considered a number of indirect representations of depth. For example, Chen et al. investigated relative depths [9]. Charkrabarti et al. proposed depth derivatives [7]. Li et al. used depth derivatives in conjunction with depths [39]. We have experimented with methods based on predicted derivatives. However, we find that they do not perform the best in our experiments (see Section 4).

Instead, we focus on predicting surface normals and occlusion boundaries. Since normals are differential surface properties, they depend only on local neighborhoods of pixels. Moreover, they relate strongly to local lighting variations directly observable in a color image. For these reasons, previous works on dense prediction of surface normals from color images produce excellent results [3, 18, 38, 71, 80]. Similarly, occlusion boundaries produce local patterns in pixels (e.g., edges), and so they usually can be robustly detected with a deep network [17, 80].

A critical question, though, is how we can use predicted surface normals and occlusion boundaries to complete depth images. Several researchers have used predicted normals to refine details on observed 3D surfaces [28, 55, 74], and Galliani *et al*. [22] used surface normals to recover missing geometry in multi-view reconstruction for table-top objects. However, nobody has ever used surface normals before for depth estimation or completion from monocular RGB-D images in complex environments.

Unfortunately, it is theoretically not possible to solve for depths from only surface normals and occlusion boundaries. There can be pathological situations where the depth relationships between different parts of the image cannot be inferred only from normals. For example, in Figure 4(a), it is impossible to infer the depth of the wall seen through the window based on only the given surface normals. In this case, the visible region of the wall is enclosed completely by occlusion boundaries (contours) from the perspective of the camera, leaving its depth indeterminate with respect to the rest of the image.



(a) Ambiguity (b) Dense connected path in real scene

Figure 4. **Using surface normals to solve for depth completion.** (a) An example of where depth cannot be solved from surface normal. (b) The area missing depth is marked in red. The red arrow shows paths on which depth cannot be integrated from surface normals. However in real-world images, there are usually many paths through connected neighboring pixels (along floors, ceilings, etc.) over which depths can be integrated (green arrows).

In practice, however, for real-world scenes it is very unlikely that a region of an image will both be surrounded by occlusion boundaries AND contain no raw depth observations at all (Figure 4(b)). Therefore, we find it practical to complete even large holes in depth images using predicted surface normals with coherence weighted by predicted occlusion boundaries and regularization constrained by observed raw depths. During experiments, we find that solving depth from predicted surface normals and occlusion boundaries results in better depth completions than predicting absolute depths directory, or even solving from depth derivatives (see Section 4).

3.3. Network Architecture and Training

A third interesting question is "what is the best way to train a deep network to predict surface normals and occlusion boundaries for depth completion?"

For our study, we pick the deep network architecture proposed in Zhang et.al because it has shown competitive performance on both normal estimation and boundary detection [80]. The model is a fully convolutional neural network built on the back-bone of VGG-16 with symmetry encoder and decoder. It is also equipped with short-cut connections and shared pooling masks for corresponding max pooling and unpooling layers, which are critical for learning local image features. We train the network with "ground truth" surface normals and silhouette boundaries computed from the reconstructed mesh.

After choosing this network, there are still several interesting questions regarding how to training it for depth completion. The following paragraphs consider these questions with a focus on normal estimation, but the issues and conclusions apply similarly for occlusion boundary detection.

What loss should be used to train the network? Unlike past work on surface normal estimation, our primary goal is

to train a network to predict normals *only for pixels inside holes* of raw observed depth images. Since the color appearance characteristics of those pixels are likely different than the others (shiny, far from the camera, etc.), one might think that the network should be supervised to regress normals only for these pixels. Yet, there are fewer pixels in holes than not, and so training data of that type is limited. It was not obvious whether it is best to train only on holes vs. all pixels. So, we tested both and compared.

We define the observed pixels as the ones with depth data from both the raw sensor and the rendered mesh, and the unobserved pixels as the ones with depth from the rendered mesh but not the raw sensor. For any given set of pixels (observed, unobserved, or both), we train models with a loss for only those pixels by masking out the gradients on other pixels during the back-propagation.

Qualitative and quantitative results comparing the results for different trained models are shown in supplemental material. The results suggest that the models trained with all pixels perform better than the ones using only observed or only unobserved pixels, and ones trained with rendered normals perform better than with raw normals.

What image channels should be input to the network? One might think that the best way to train the network to predict surface normals from a raw RGB-D image is to provide all four channels (RGBD) and train it to regress the three normal channels. However, surprisingly, we find that our networks performed poorly at predicting normals for pixels without observed depth when trained that way. They are excellent at predicting normals for pixels with observed depth, but not for the ones in holes – i.e., the ones required for depth completion. This result holds regardless of what pixels are included in the loss.

We conjecture that the network trained with raw depth mainly learns to compute normals from depth directly – it fails to learn how to predict normals from color when depth is not present, which is the key skill for depth completion. In general, we find that the network learns to predict normals better from color than depth, even if the network is given an extra channel containing a binary mask indicating which pixels have observed depth [79]. For example, in Figure 5, we see that the normals predicted in large holes from color alone are better than from depth, and just as good as from both color and depth. Quantitative experiments support this finding in Table 1.

This result is very interesting because it suggests that we can train a network to predict surface normals from color alone and use the observed depth *only as regularization* when solving back for depth from normals (next section). This strategy of separating "prediction without depth" from "optimization with depth" is compelling for two reasons. First, the prediction network does not have to be retrained for different depth sensors. Second, the optimization can be generalized to take a variety of depth observations as regularization, including perhaps sparse depth samples [47]. This is investigated experimentally in Section 4.

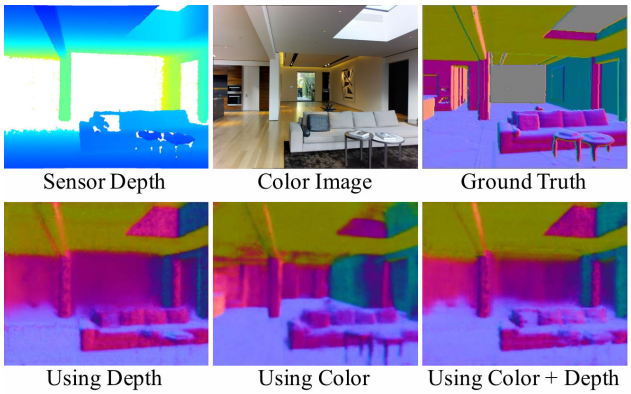


Figure 5. **Surface normal estimation for different inputs.** The top row shows an input color image, raw depth, and the rendered normal. The bottom row shows surface normal predictions when the inputs are depth only, color only, and both. The middle one performs the best for the missing area, while comparable elsewhere with the other two models even without depth as input.

3.4. Optimization

After predicting the surface normal image N and occlusion boundary image B, we solve a system of equations to complete the depth image D. The objective function is defined as the weighted sum of squared errors with four terms:

$$\begin{aligned} E &= \lambda_D E_D + \lambda_S E_S + \lambda_N E_N B \\ E_D &= \sum_{p \in T_{obs}} ||D(p) - D_0(p)||^2 \\ E_N &= \sum_{p,q \in N} || < v(p,q), N(p) > ||^2 \\ E_S &= \sum_{p,q \in N} ||D(p) - D(q))|^2 \end{aligned} \tag{1}$$

where E_D measures the distance between the estimated depth $D(p)$ and the observed raw depth $D_0(p)$ at pixel p, E_N measures the consistency between the estimated depth and the predicted surface normal $N(p)$, E_S encourages adjacent pixels to have the same depths. $B \in [0, 1]$ down-weights the normal terms based on the predicted probability a pixel is on an occlusion boundary (B(p)).

In its simplest form, this objective function is non-linear, due to the normalization of the tangent vector $v(p, q)$ required for the dot product with the surface normal in E_N. However, we can approximate this error term with a linear formation by foregoing the vector normalization, as suggested in [55]. In other settings, this approximation would add sensitivity to scaling errors, since smaller depths result in shorter tangents and potentially smaller E_N terms. However, in a depth completion setting, the data term E_D forces

	Depth Completion							Surface Normal Estimation				
Input	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑	Mean↓	Median↓	11.25↑	22.5↑	30↑
Depth	0.107	0.165	38.89	48.54	61.12	73.57	80.98	35.08	23.07	27.6	49.1	58.6
Both	0.090	0.124	40.13	51.26	64.84	76.46	83.05	35.30	23.59	26.7	48.5	58.1
Color	0.089	0.116	40.63	51.21	65.35	76.64	82.98	31.13	17.28	37.7	58.3	67.1

Table 1. **Effect of different inputs to our deep network.** We train models taking depth, color, and both respectively for surface normal estimation and depth completion. Using only color as input achieves similar performance as the case with both.

the global solution to maintain the correct scale by enforcing consistency with the observed raw depth, and thus this is not a significant problem.

Since the matrix form of the system of equations is sparse and symmetric positive definite, we can solve it efficiently with a sparse Cholesky factorization (as implemented in cs_cholsol in CSparse [13]). The final solution is a global minimum to the approximated objective function.

This linearization approach is critical to the success of the proposed method. Surface normals and occlusion boundaries (and optionally depth derivatives) capture only local properties of the surface geometry, which makes them relatively easy to estimate. Only through global optimization can we combine them to complete the depths for all pixels in a consistent solution.

4. Experimental Results

We ran a series of experiments to test the proposed methods. Unless otherwise specified, networks were pretrained on the SUNCG dataset [66, 80] and fine-tuned on the training split of the our new dataset using only color as input and a loss computed for all rendered pixels. Optimizations were performed with $\lambda_D = 10^3$, $\lambda_N = 1$, and $\lambda_S = 10^{-3}$. Evaluations were performed on the test split of our new dataset.

We find that predicting surface normals and occlusion boundaries from color at 320x256 takes ∼0.3 seconds on a NVIDIA TITAN X GPU. Solving the linear equations for depths takes ∼1.5 seconds on a Intel Xeon 2.4GHz CPU.

4.1. Ablation Studies

The first set of experiments investigates how different test inputs, training data, loss functions, depth representations, and optimization methods affect the depth prediction results (further results can be found in the supplemental material).

Since the focus of our work is predicting depth where it is unobserved by a depth sensor, our evaluations measure errors in depth predictions only for pixels of test images *unobserved* in the test depth image (but present in the rendered image). This is the opposite of most previous work on depth estimation, where error is measured only for pixels that are observed by a depth camera.

When evaluating depth predictions, we report the median error relative to the rendered depth (Rel), the root mean squared error in meters (RMSE), and percentages of pixels with predicted depths falling within an interval ($[\delta = |predicted - true|/true]$), where δ is 1.05, 1.10, 1.25, 1.25^2, or 1.25^3. These metrics are standard among previous work on depth prediction, except that we add thresholds of 1.05 and 1.10 to enable finer-grained evaluation.

When evaluating surface normal predictions, we report the mean and median errors (in degrees), plus the percentages of pixels with predicted normals less than thresholds of 11.25, 22.5, and 30 degrees.

What data should be input to the network? Table 1 shows results of an experiment to test what type of inputs are best for our normal prediction network: color only, raw depth only, or both. Intuitively, it would seem that inputting both would be best. However, we find that the network learns to predict surface normals better when given only color (median error = 17.28° for color vs. 23.07° for both), which results in depth estimates that are also slightly better (Rel = 0.089 vs. 0.090). This difference persists whether we train with depths for all pixels, only observed pixels, or only unobserved pixels (results in supplemental material). We expect the reason is that the network quickly learns to interpolate from observed depth if it is available, which hinders it from learning to synthesize new depth in large holes.

The impact of this result is quite significant, as it motivates our two-stage system design that separates normal/boundary prediction only from color and optimization with raw depth.

What depth representation is best? Table 2 shows results of an experiment to test which depth representations are best for our network to predict. We train networks separately to predict absolute depths (D), surface normals (N), and depth derivatives in 8 directions (DD), and then use different combinations to complete the depth by optimizing Equation 1. The results indicate that solving for depths from predicted normals (N) provides the best results (Rel = 0.089 for normals (N) as compared to 0.167 for depth (D), 0.100 for derivatives (DD), 0.092 for normals and derivatives (N+DD). We expect that this is because normals represent only the orientation of surfaces, which is relatively easy to predict [35]. Moreover, normals do not scale with depth, unlike depths or depth derivatives, and thus are more consistent across a range of views.

Does prediction of occlusion boundaries help? The last six rows of Table 2 show results of an experiment to test

B	Rep	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑
-	D	0.167	0.241	16.43	31.13	57.62	75.63	84.01
No	DD	0.123	0.176	35.39	45.88	60.41	73.26	80.73
	N+DD	0.112	0.163	37.85	47.22	61.27	73.70	80.83
	N	0.110	0.161	38.12	47.96	61.42	73.77	80.85
Yes	DD	0.100	0.131	37.95	49.14	64.26	76.14	82.63
	N+DD	0.092	0.122	39.93	50.73	65.33	**77.04**	**83.25**
	N	**0.089**	**0.116**	**40.63**	**51.21**	**65.35**	76.74	82.98

Table 2. **Effect of predicted representation on depth accuracy.** "DD" represents depth derivative, and "N" represents surface normal. We also evaluate the effect of using boundary weight. The first row shows the performance of directly estimating depth. Overall, solving back depth with surface normal and occlusion boundary gives the best performance.

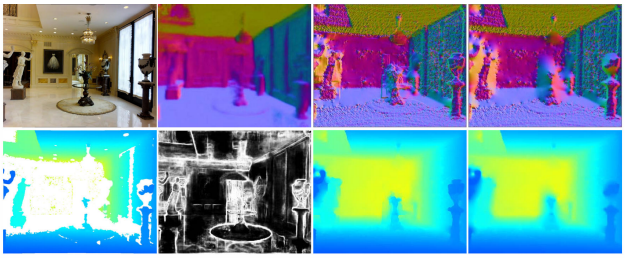


Figure 6. **Effect of occlusion boundary prediction on normals.** The 2nd column shows the estimated surface normal and occlusion boundary. The 3rd and 4th column shows the output of the optimization with/without occlusion boundary weight. To help understand the 3D geometry and local detail, we also visualize the surface normal computed from the output depth. The occlusion boundary provides information for depth discontinuity, which help to maintain boundary sharpness.

whether down-weighting the effect of surface normals near predicted occlusion boundaries helps the optimizer solve for better depths. Rows 2-4 are without boundary prediction ("No" in the first column), and Rows 5-7 are with ("Yes"). The results indicate that boundary predictions improve the results by ∼19% (Rel = 0.089 vs. 0.110). This suggests that the network is on average correctly predicting pixels where surface normals are noisy or incorrect, as shown qualitatively in Figure 6.

How much observed depth is necessary? Figure 7 shows results of an experiment to test how much our depth completion method depends on the quantity of input depth. To investigate this question, we degraded the input depth images by randomly masking different numbers of pixels before giving them to the optimizer to solve for completed depths from predicted normals and boundaries. The two plots shows curves indicating depth accuracy solved for pixels that are observed (left) and unobserved (right) in the original raw depth images. From these results, we see that the optimizer is able to solve for depth almost as accurately when given only a small fraction of the pixels in the raw depth image. As expected, the performance is much worse on pixels unobserved by the raw depth (they are harder). However, the depth estimations are still quite good when

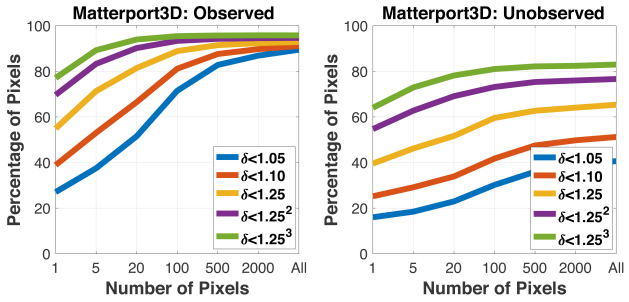


Figure 7. **Effect of sparse raw depth inputs on depth accuracy.** The depth completion performance of our method w.r.t number of input pixels with depth. The plot shows that depth estimation on unobserved pixels is harder than the observed. It also shows that our method works well with only a small number of sparse pixels, which is desirable to many applications.

only a small fraction of the raw pixels are provided (the rightmost point on the curve at 2000 pixels represents only 2.5% of all pixels). This results suggests that our method could be useful for other depth sensor designs with sparse measurements. In this setting, our deep network would not have to be retrained for each new dense sensor (since it depends only on color), a benefit of our two-stage approach.

4.2. Comparison to Baseline Methods

The second set of experiments investigates how the proposed approach compares to baseline depth inpainting and depth estimation methods.

Comparison to Inpainting Methods Table 8 shows results of a study comparing our proposed method to typical non-data-driven alternatives for depth inpainting. The focus of this study is to establish how well-known methods perform to provide a baseline on how hard the problem is for this new dataset. As such, the methods we consider include: a) joint bilinear filtering [64] (Bilateral), b) fast bilateral solver [5] (Fast), and c) global edge-aware energy optimization [20] (TGV). The results in Table 8 show that our method significantly outperforms these methods (Rel=0.089 vs. 0.103-0.151 for the others). By training to predict surface normals with a deep network, our method learns to complete depth with data-driven priors, which are stronger than simple geometric heuristics. The difference to the best of the tested hand-tuned approaches (Bilateral) can be seen in Figure 8.

Method	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑
Smooth	0.151	0.187	32.80	42.71	57.61	72.29	80.15
Bilateral [64]	0.118	0.152	34.39	46.50	61.92	75.26	81.84
Fast [5]	0.127	0.154	33.65	45.08	60.36	74.52	81.79
TGV [20]	0.103	0.146	37.40	48.75	62.97	75.00	81.71
Ours	**0.089**	**0.116**	**40.63**	**51.21**	**65.35**	**76.74**	**82.98**

Table 3. **Comparison to baseline inpainting methods.** Our method significantly outperforms baseline inpainting methods.

Comparison to Depth Estimation Methods Table 4 shows results for a study comparing our proposed method to pre-

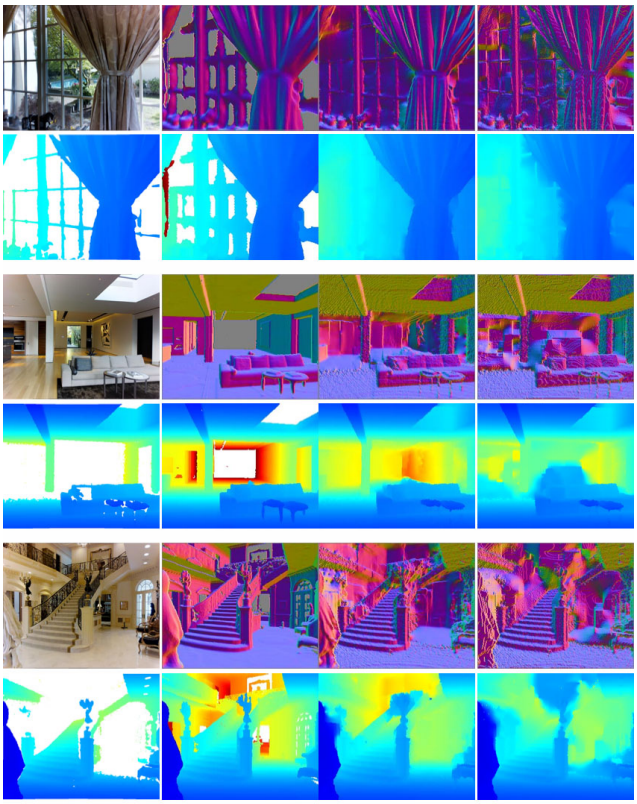


Figure 8. **Comparison to inpainting with a joint bilateral filter.** Our method learns better guidance from color and produce comparatively sharper and more accurate results.

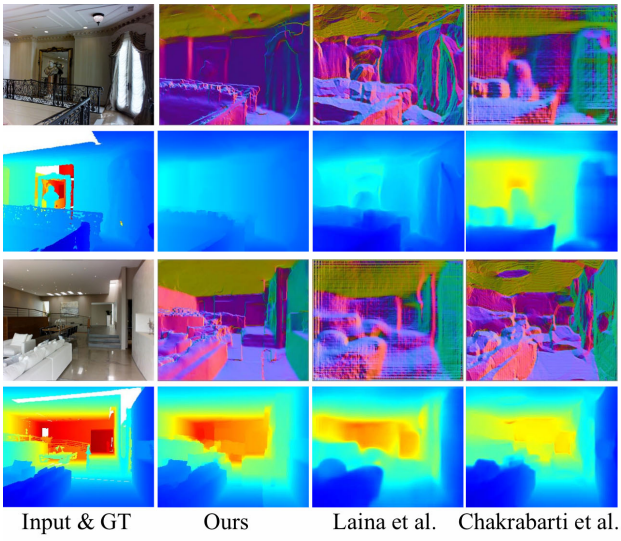


Figure 9. **Comparison to deep depth estimation methods.** We compare with the state of the art methods under the depth estimation setting. Our method produces not only accurate depth value but also large scale geometry as reflected in the surface normal.

vious methods that estimate depth only from color. We consider comparisons to Chakrabarti et al. [7], whose approach is most similar to ours (it uses predicted derivatives), and to Laina et al. [37], who recently reported state-of-the-art results in experiments with NYUv2 [64]. We finetune [7] on our dataset, but use pretrained model on NYUv2 for [37] as their training code is not provided.

Obs	Meth	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑
Y	[37]	0.190	0.374	17.90	31.03	54.80	75.97	85.69
	[7]	0.161	0.320	21.52	35.5	58.75	77.48	85.65
	Ours	**0.130**	**0.274**	**30.60**	**43.65**	**61.14**	**75.69**	**82.65**
N	[37]	0.384	0.572	8.86	16.67	34.64	55.60	69.21
	[7]	0.352	0.610	11.16	20.50	37.73	57.77	70.10
	Ours	**0.283**	**0.537**	**17.27**	**27.42**	**44.19**	**61.80**	**70.90**

Table 4. **Comparison to deep depth estimation methods.** We compare with Laina et al. [37] and Chakrabarti et al.[7]. All the methods perform worse on unobserved pixels than the observed pixels, which indicates unobserved pixels are harder. Our method significantly outperform other methods.

Of course, these depth estimation methods solve a different problem than ours (no input depth), and alternative methods have different sensitivities to the scale of depth values, and so we make our best attempt to adapt both their and our methods to the same setting for fair comparison. To do that, we run all methods with only color images as input and then uniformly scale their depth image outputs to align perfectly with the true depth at one random pixel (selected the same for all methods). In our case, since Equation 1 is under-constrained without any depth data, we arbitrarily set the middle pixel to a depth of 3 meters during our optimization and then later apply the same scaling as the other methods. This method focuses the comparison on predicting the "shape" of the computed depth image rather than its global scale.

Results of the comparison are shown in Figure 9 and Table 4. From the qualitative results in Figure 9, we see that our method reproduces both the structure of the scene and the fine details best – even when given only one pixel of raw depth. According to the quantitative results shown in Table 4, our method is 23-40% better than the others, regardless of whether evaluation pixels have observed depth (Y) or not (N). These results suggest that predicting surface normals is a promising approach to depth estimation as well.

5. Conclusion

This paper describes a deep learning framework for completing the depth channel of an RGB-D image acquired with a commodity RGB-D camera. It provides two main research contributions. First, it proposes to complete depth with a two stage process where surface normals and occlusion boundaries are predicted from color, and then completed depths are solved from those predictions. Second, it learns to complete depth images by supervised training on data rendered from large-scale surface reconstructions. During tests with a new benchmark, we find the proposed approach outperforms previous baseline approaches for depth inpainting and estimation.

References

[1] Kinect for windows. https://developer.microsoft.com/en-us/windows/kinect. 14

[2] Structure sensor. https://structure.io/. 14

[3] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016. 4

[4] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013. 2

[5] J. T. Barron and B. Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016. 7, 15

[6] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 2

[7] A. Chakrabarti, J. Shao, and G. Shakhnarovich. Depth from a single image by harmonizing overcomplete local network predictions. In *Advances in Neural Information Processing Systems*, pages 2658–2666, 2016. 4, 8, 13

[8] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 3

[9] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. In *Advances in Neural Information Processing Systems*, pages 730–738, 2016. 4

[10] W. Chen, H. Yue, J. Wang, and X. Wu. An improved edge detection algorithm for depth map inpainting. *Optics and Lasers in Engineering*, 55:69–77, 2014. 2

[11] M. Ciotta and D. Androutsos. Depth guided image completion for structure and texture synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 1199–1203. IEEE, 2016. 2

[12] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3, 14

[13] T. Davis. Csparse. *Society for Industrial and Applied Mathematics, Philadephia, PA*, 2006. 6

[14] J. D'Errico. Inpaint nans, 2017. www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans. 15

[15] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, pages 291–298, 2006. 2

[16] D. Doria and R. J. Radke. Filling large holes in lidar data by inpainting depth gradients. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 65–72. IEEE, 2012. 1, 2

[17] K. A. Ehinger, W. J. Adams, E. W. Graf, J. H. Elder, K. Vaiapury, B. Purushothaman, A. Pal, S. Agarwal, B. Bhowmick, I. Rafegas, et al. Local depth edge detection in humans and deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2681–2689, 2017. 4

[18] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 2, 4

[19] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 1, 2

[20] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 993–1000. IEEE, 2013. 7, 15

[21] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002. 2

[22] S. Galliani and K. Schindler. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5479–5487, 2016. 4

[23] D. Garcia. Robust smoothing of gridded data in one and higher dimensions with missing values. *Computational statistics & data analysis*, 54(4):1167–1178, 2010. 15

[24] J. Gautier, O. Le Meur, and C. Guillemot. Depth-based image completion for view synthesis. In *3DTV Conference: The True Vision-capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1–4. IEEE, 2011. 2

[25] X. Gong, J. Liu, W. Zhou, and J. Liu. Guided depth enhancement via a fast marching method. *Image and Vision Computing*, 31(10):695–703, 2013. 2

[26] M. Halber and T. Funkhouser. Fine-to-coarse global registration of rgb-d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[27] Y. Han, J.-Y. Lee, and I. So Kweon. High quality shape from a single rgb-d image under uncalibrated natural illumination. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1617–1624, 2013. 2

[28] C. Hane, L. Ladicky, and M. Pollefeys. Direction matters: Depth estimation with a surface normal classifier. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 381–389, 2015. 4

[29] S. Hawe, M. Kleinsteuber, and K. Diepold. Dense disparity maps from sparse disparity measurements. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2126–2133. IEEE, 2011. 3

[30] D. Herrera, J. Kannala, J. Heikkilä, et al. Depth map inpainting under a second-order smoothness prior. In *Scandinavian Conference on Image Analysis*, pages 555–566. Springer, 2013. 2

[31] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM transactions on graphics (TOG)*, 24(3):577–584, 2005. 2

[32] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 92–101. IEEE, 2016. 3

[33] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013. 3, 12

[34] M. Kiechle, S. Hawe, and M. Kleinsteuber. A joint intensity and depth co-sparse analysis model for depth map superresolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1545–1552, 2013. 2

[35] J. J. Koenderink, A. J. Van Doorn, and A. M. Kappers. Surface perception in pictures. *Attention, Perception, & Psychophysics*, 52(5):487–496, 1992. 1, 6

[36] M. Kulkarni and A. N. Rajagopalan. Depth inpainting by tensor voting. *JOSA A*, 30(6):1155–1165, 2013. 2

[37] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. 2, 8

[38] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015. 4

[39] J. Li, R. Klein, and A. Yao. A two-streamed network for estimating fine-scaled depth maps from single rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3372–3380, 2017. 4

[40] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016. 2

[41] J. Liu and X. Gong. Guided depth enhancement via anisotropic diffusion. In *Pacific-Rim Conference on Multimedia*, pages 408–417. Springer, 2013. 2

[42] J. Liu, X. Gong, and J. Liu. Guided inpainting and filtering for kinect depth maps. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2055–2058. IEEE, 2012. 2

[43] L.-K. Liu, S. H. Chan, and T. Q. Nguyen. Depth reconstruction from sparse samples: Representation, algorithm, and sampling. *IEEE Transactions on Image Processing*, 24(6):1983–1996, 2015. 3

[44] M. Liu, X. He, and M. Salzmann. Building scene models by completing and hallucinating depth and semantics. In *European Conference on Computer Vision*, pages 258–274. Springer, 2016. 2

[45] J. Lu and D. Forsyth. Sparse depth super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2245–2253, 2015. 2

[46] J. Lu, D. Min, R. S. Pahwa, and M. N. Do. A revisit to mrf-based depth map super-resolution and enhancement. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 985–988. IEEE, 2011. 2

[47] F. Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *arXiv preprint arXiv:1709.07492*, 2017. 3, 5

[48] O. Mac Aodha, N. D. Campbell, A. Nair, and G. J. Brostow. Patch based synthesis for single depth image superresolution. In *European Conference on Computer Vision*, pages 71–84. Springer, 2012. 2

[49] M. Mahmoudi and G. Sapiro. Sparse representations for range data restoration. *IEEE Transactions on Image Processing*, 21(5):2909–2915, 2012. 2

[50] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in Neural Information Processing Systems*, pages 2802–2810, 2016. 15

[51] K. Matsuo and Y. Aoki. Depth image enhancement using local tangent plane approximations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3574–3583, 2015. 1, 2

[52] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann. When can we use kinectfusion for ground truth acquisition. In *Proc. Workshop on Color-Depth Camera Fusion in Robotics*, volume 2, 2012. 3

[53] E. Mingolla and J. T. Todd. Perception of solid shape from shading. *Biological cybernetics*, 53(3):137–151, 1986. 1

[54] S. M. Muddala, M. Sjostrom, and R. Olsson. Depth-based inpainting for disocclusion filling. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014*, pages 1–4. IEEE, 2014. 2

[55] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM transactions on graphics (TOG)*, 24(3):536–543, 2005. 4, 5

[56] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1623–1630. IEEE, 2011. 2

[57] J. Park, Q.-Y. Zhou, and V. Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 143–152, 2017. 3

[58] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 2, 15

[59] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5506–5514, 2016. 2

[60] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006. 2

[61] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009. 2

[62] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo

datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014. 3

[63] E. Shabaninia, A. R. Naghsh-Nilchi, and S. Kasaei. High-order markov random field for single depth image super-resolution. *IET Computer Vision*, 2017. 2

[64] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. *Computer Vision–ECCV 2012*, pages 746–760, 2012. 1, 7, 8, 15

[65] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 14

[66] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[67] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. Depth from focus with your mobile phone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3497–3506, 2015. 2

[68] A. K. Thabet, J. Lahoud, D. Asmar, and B. Ghanem. 3d aware correction and completion of depth maps in piecewise planar scenes. In *Asian Conference on Computer Vision*, pages 226–241. Springer, 2014. 2

[69] I. Tosic and S. Drewes. Learning joint intensity-depth sparse representations. *IEEE Transactions on Image Processing*, 23(5):2122–2132, 2014. 2

[70] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[71] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015. 4

[72] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013. 3

[73] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016. 2

[74] W. Xie, M. Wang, X. Qi, and L. Zhang. 3d surface detail enhancement from a single normal map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2325–2333, 2017. 4

[75] H. Xue, S. Zhang, and D. Cai. Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26(9):4311–4320, 2017. 2, 3

[76] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin. Shading-based shape refinement of rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1415–1422, 2013. 2

[77] H.-T. Zhang, J. Yu, and Z.-F. Wang. Probability contour guided depth map inpainting and superresolution using non-local total generalized variation. *Multimedia Tools and Applications*, pages 1–18, 2017. 2

[78] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999. 2

[79] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017. 5, 12

[80] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 4, 6, 12, 15

[81] Y. Zuo, Q. Wu, J. Zhang, and P. An. Explicit edge inconsistency evaluation model for color-guided depth map enhancement. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016. 2

Supplementary Material

This document contains further implementation details and results of ablation studies, cross-dataset experiments, and comparisons to other inpainting methods that would not fit in the main paper.

A. Further Implementation Details

This section provides extra implementation details for our methods. All data and code will be released upon the acceptance to ensure reproducibility.

A.1. Mesh reconstruction and rendering

For every scene in the Matterport3D dataset, meshes were reconstructed and rendered to provide "completed depth images" using the following process. First, each house was manually partitioned into regions roughly corresponding to rooms using an interactive floorplan drawing interface. Second, a dense point cloud was extracted containing RGB-D points (pixels) within each region, excluding pixels whose depth is beyond 4 meters from the camera (to avoid noise in the reconstructed mesh). Third, a mesh was reconstructed from the points of each region using Screened Poisson Surface Reconstruction [33] with octree depth 11. The meshes for all regions were then merged to form the final reconstructed mesh M for each scene. "Completed depth images" were then created for each of the original RGB-D camera views by rendering M from that view using OpenGL and reading back the depth buffer.

Figure 10 shows images of a mesh produced with this process. The top row shows exterior views covering the entire house (vertex colors on the left, flat shading on the right). The bottom row shows a close-up image of the mesh from an interior view. Though the mesh is not perfect, it has 12.2M triangles reproducing most surface details. Please note that the mesh is complete where holes typically occur in RGB-D images (windows, shiny table tops, thin structures of chairs, glossy surfaces of cabinet, etc.). Please also note the high level of detail for surfaces distant to the camera (e.g., furniture in the next room visible through the doorway).

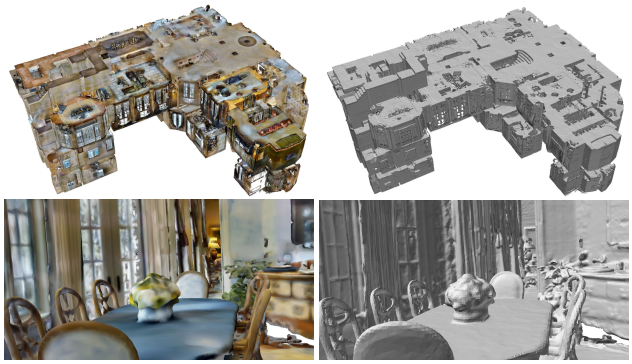


Figure 10. **Reconstructed mesh for one scene.** The mesh used to render completed depth images is shown from an outside view (top) and inside view (bottom), rendered with vertex colors (left) and flat shading (right).

A.2. Network architecture

All the networks used for this project are derived from the surface normal estimation model proposed in Zhang et.al [80] with the following modifications.

Input Depending on what is the input, the network takes data with different channels at the first convolution layer.

- Color. The color is a 3-channel tensor with R,G,B for each. The intensity values are normalized to [-0.5 0.5]. We use a bi-linear interpolation to resize color image if necessary.
- Depth. The absolute values of depth in meter are used as input. The pixels with no depth signal from sensor are assigned a value of zero. To resolve the ambiguity between "missing" and "0 meter", a binary mask indicating the pixels that have depth from sensor is added as an additional channel as suggested in Zhang et.al [79]. Overall, the depth input contains 2 channels (absolute depth and binary valid mask) in total. To prevent inaccurate smoothing, we use the nearest neighbor search to resize depth image.
- Color+Depth. The input in this case is the concatenation of the color and depth as introduced above. This results in a 5-channel tensor as the input.

Output The network for absolute depth, surface normal, and depth derivative outputs results with 1, 3, and 8 channels respectively. The occlusion boundary detection network generates 3 channel outputs representing the probability of each pixel belonging to "no edge", "depth crease", and "occlusion boundary".

Loss Depth, surface normal, and derivative are predicted as regression tasks. The SmoothL1 loss[1] is used for training depth and derivative, and the cosine embedding loss[2] is used for training surface normal. The occlusion boundary detection is formulated into a classification task, and cross entropy loss[3] is used. The last two batch normalization lay-

[1] https://github.com/torch/nn/blob/master/doc/criterion.md#nn.SmoothL1Criterion

[2] https://github.com/torch/nn/blob/master/doc/criterion.md#nn.CosineEmbeddingCriterion

[3] https://github.com/torch/nn/blob/master/doc/criterion.md#nn.CrossEntropyCriterion

Input	Rep	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑
C	D	0.408	0.500	6.49	12.80	30.01	54.44	72.88
C	1/D	0.412	0.492	6.86	12.88	28.99	54.51	73.13
D	D	0.167	0.241	16.43	31.13	57.62	75.63	**84.01**
D	1/D	0.199	0.255	14.06	27.32	53.70	74.19	83.85
Ours		**0.089**	**0.116**	**40.63**	**51.21**	**65.35**	**76.74**	82.98

Table 5. **Comparison of different depth representations.** Predicting either depth (D) or disparity (1/D) provides worse results than predicting surface normals and solving for depth (Ours) for either color or depth inputs.

ers are removed because this results in better performance in practice.

A.3. Training schema

The neural network training and testing are implemented in Torch. For all the training tasks, RMSprop optimization algorithm is used. The momentum is set to 0.9, and the batch size is 1. The learning rate is set to 0.001 initially and reduce to half every 100K iterations. All the models converge within 300K iterations.

B. Further Experimental Results

This section provides extra experimental results, including ablation studies, cross-dataset experiments, and comparisons to other depth completion methods.

B.1. Ablation Studies

Section 4.1 of the paper provides results of ablation studies aimed at investigating how different test inputs, training data, loss functions, depth representations, and optimization methods affect our depth prediction results. This section provides further results of that type.

More qualitative results about surface normal estimation model trained from different setting are shown in Figure 11. Comparatively, training the surface normal estimation model with our setting (i.e. using only color image as input, all available pixels with rendered depth as supervision, the 4-th column in the figure) achieves the best quality of prediction, and hence benefits the global optimization for depth completion.

What kind of ground truth is better? This test studies what normals should be used as supervision for the loss when training the surface prediction network. We experimented with normals computed from raw depth images and with normals computed from the rendered mesh. The result in the top two rows of Table 6 (Comparison:Target) shows that the model trained on rendered depth performs better the the one from raw depth. The improvement seems to come partly from having training pixels for unobserved regions and partly from more accurate depths (less noise).

What loss should be used to train the network? This test studies which pixels should be included in the loss when training the surface prediction network. We experimented with using only the unobserved pixels, using only the observed pixels, and both as supervision. The three models were trained separately in the training split of our new dataset and then evaluated versus the rendered normals in the test set. The quantitative results in the last three rows of Table 6 (Comparison:Pixel) show that models trained with supervision from both observed and unobserved pixels (bottom row) works slightly better than the one trained with only the observed pixels or only the unobserved pixels. This shows that the unobserved pixels indeed provide additional information.

What kind of depth representation is best? Several depth representations were considered in the paper (normals, derivatives, depths, etc.). This section provides further results regarding direct prediction of depth and disparity (i.e. one over depth) to augment/fix results in Table 2 of the paper.

Actually, the top row of Table 2 of the paper (where the Rep in column 2 is 'D') is mischaracterized as direct prediction of depth from color – it is actually direct prediction of complete depth from input depth. That was a mistake. Sorry for the confusion. The correct result is in the top line of Table 5 of this document (Input=C, Rep=D). The result is quite similar and does not change any conclusions: predicting surface normals and then solving for depth is better than predicting depth directly (Rel = 0.089 vs. 0.408).

We also consider prediction of disparity rather than depth, as suggested in Chakrabarti et.al and other papers [7]. We train models to estimate disparity directly from color and raw depth respectively. The results can be seen in Table 5. We find that estimating disparity results in performance that is not better than estimating depth when given either color or depth as input for our depth completion application.

B.2. Cross-Dataset Experiments

This test investigates whether it is possible to train our method on one dataset and then use it effectively for another.

Matterport3D and ScanNet We first conduct experiments between Matterport3D and ScanNet datasets. Both have 3D surface reconstructions for large sets of environments (∼1000 rooms each) and thus provide suitable training data for training and test our method with rendered meshes. We train a surface normal estimation model separately on each dataset, and then use it without fine tuning to perform depth completion for the test set of the other. The

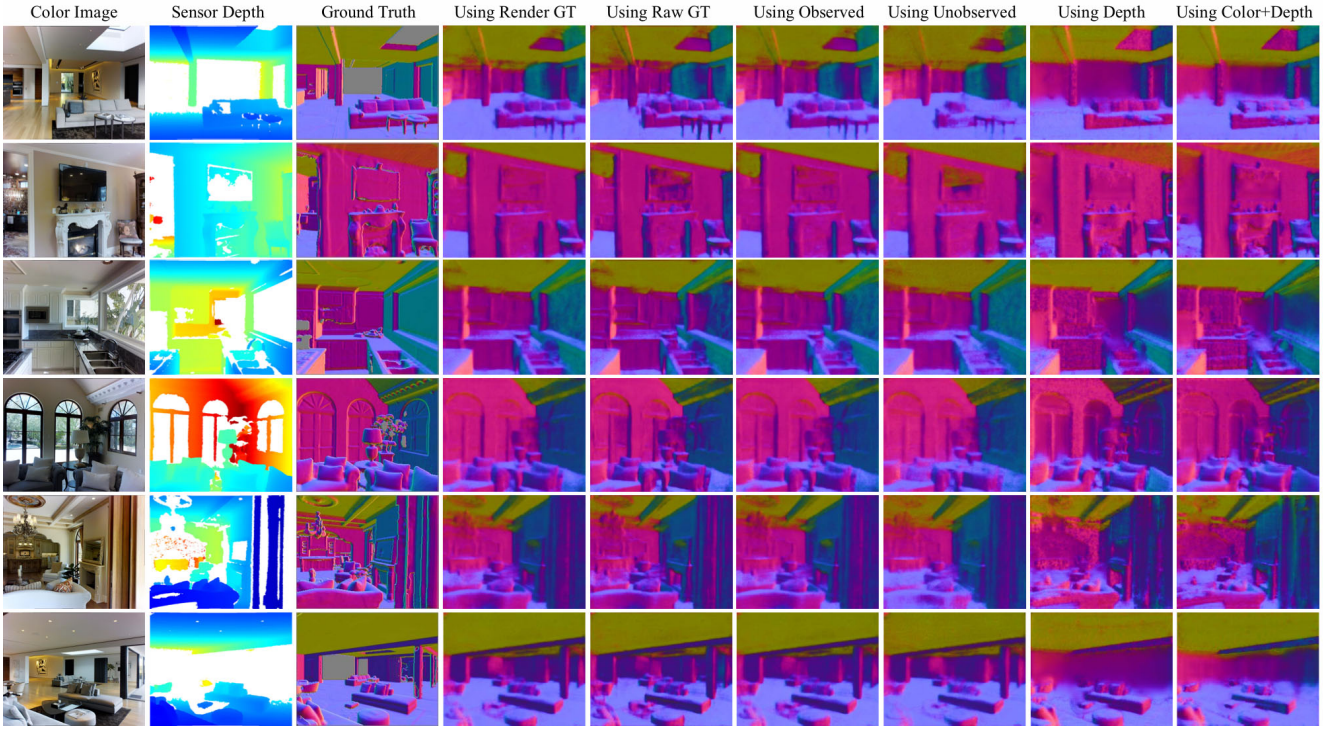


Figure 11. **Comparison of normal estimation with different training settings.** The 4-th column shows the output of the model trained using only color as input and the rendered depth from all pixels as supervision, which is the setting we chose for our system. Comparatively, it generates better surface normal than other alternative training settings.

Comparison	Setting			Depth Completion							Surface Normal Estimation				
	Input	Target	Pixel	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2 ↑	1.25^3 ↑	Mean↓	Median↓	11.25↑	22.5↑	30↑
Target	Color	*Raw*	Both	0.094	0.123	39.84	50.40	64.68	76.38	82.80	32.87	18.70	34.2	55.7	64.3
	Color	*Render*	Both	0.089	0.116	40.63	51.21	65.35	76.64	82.98	31.13	17.28	37.7	58.3	67.1
Pixel	Color	Render	*Observed*	0.091	0.121	40.31	50.88	64.92	76.50	82.91	32.16	18.44	34.7	56.4	65.5
	Color	Render	*Unobserved*	0.090	0.119	40.71	51.22	65.21	76.59	83.04	31.52	17.70	35.4	57.7	66.6
	Color	Render	*Both*	0.089	0.116	40.63	51.21	65.35	76.64	82.98	31.13	17.28	37.7	58.3	67.1
Input	*Depth*	Render	Both	0.107	0.165	38.89	48.54	61.12	73.57	80.98	35.08	23.07	27.6	49.1	58.6
	Both	Render	Both	0.090	0.124	40.13	51.26	64.84	76.46	83.05	35.30	23.59	26.7	48.5	58.1
	Color	Render	Both	0.089	0.116	40.63	51.21	65.35	76.64	82.98	31.13	17.28	37.7	58.3	67.1

Table 6. Ablation studies. Evaluations of estimated surface normals and solved depths using different training inputs and losses. For the sake of comparison, Table 1 from main paper is copied in the last three rows as comparison across different inputs.

quantitative results are shown in Table 7. As expected, the models work best on the test dataset matching the source of the training data. Actually, the model trained from Matterport3D has a better generalization capability compared to the model trained from ScanNet, which is presumably because the Matterport3D dataset has a more diverse range of camera viewpoints. However, interestingly, both models work still reasonably well when run on the other dataset, even though they were not fine-tuned at all. We conjecture this is because our surface normal prediction model is trained only on color inputs, which are relatively similar between the two datasets. Alternative methods using depth as input would probably not generalize as well due to the significant differences between the depth images of the two datasets.

Intel RealSense Depth Sensor The depth map from Intel RealSense has better quality in short range but contains more missing area compared to Structure Sensor [2] and Kinect [1]. The depth signal can be totally lost or extremely sparse for distant area and surface with special materials, e.g. shinny, dark. We train a surface normal estimation model from ScanNet dataset [12] and directly evaluate on the RGBD images captured by Intel RealSense from SUN-RGBD dataset [65] without any finetuning. The results are shown in Figure 12. From left to right, we show the input color image, input depth image, completed depth image using our method, the point cloud visualization of the input and completed depth map, and the surface normal converted from the completed depth. As can be seen, the depth from RealSense contains more missing area than Matterport3D and ScanNet, yet our model still generates decent results.

Train	Test	Rel	RMSE	1.05	1.10	1.25	1.25^2	1.25^3
Matterport3D	Matterport3D	0.089	0.116	40.63	51.21	65.35	76.74	82.98
ScanNet	Matterport3D	0.098	0.128	37.96	49.79	64.01	76.04	82.64
Matterport3D	Scannet	0.042	0.065	52.91	65.83	81.20	90.99	94.94
ScanNet	ScanNet	0.041	0.064	53.33	66.02	81.14	90.92	94.92

Table 7. **Cross-dataset performance.** We trained surface normal estimation models on each dataset, Matterport3D and ScanNet, respectively and test on both. Models work the best on the dataset where it is trained from. Model trained from Matterport3D shows better generalization capability than the one from ScanNet.

This again shows that our method can effectively run on RGBD images captured from various of depth sensors with significantly different depth patterns.

B.3. Comparisons to Depth Inpainting Methods

Section 4.2 of the paper provides comparisons to alternative methods for depth inpainting. This section provides further results of that type in Table 8. In this additional study, we compare with the following methods:

- **DCT [23]:** fill in missing values by solving the penalized least squares of a linear system using discrete cosine transform using the code from Matlab Central [4].
- **FCN [50]:** train an FCN with symmetric shortcut connection to take raw depth as input and generate completed depth as the output using the code from Zhang et.al [80].
- **CE [58]:** train the context encoder of Pathak et.al to inpaint depth images using the code from Github [5].

The results of DCT [23] are similar to other inpainting comparisons provided in the paper. They mostly interpolate holes.

The results of FCN and CE show that methods designed for inpainting color are not very effective at inpainting depth. As already described in the paper, methods that learn depth from depth using an FCN can be lazy and only learn to reproduce and interpolate provided depth. However, the problems are more subtle than that, as depth data has many characteristics different from color. For starters, the context encoder has a more shallow generator and lower resolution than our network, and thus generates blurrier depth images than ours. More significantly, the fact that ground-truth depth data can have missing values complicates the training of the discriminator network in the context encoder (CE) – in a naive implementation, the generator would be trained to predict missing values in order to fool the discriminator. We tried multiple approaches to circumvent this problem, including propagating gradients on only unobserved pixels, filling a mean depth value in the missing area. We find that none of them work as well as our method.

More results of our method and comparison to other inpainting methods can be found in Figure 14,15,16 in the end of this paper. Each two rows shows an example, where the 2nd row shows the completed depth of different methods, and 1st row shows their corresponding surface normal for purpose of highlighting details and 3D geometry. For each example, we show the input, ground truth, our result, followed by the results of FCN [50], joint bilateral filter [64], discrete cosine transform [23], optimization with only smoothness, and PDE [14]. As can be seen, our method generates better large scale planar geometry and sharper object boundary.

Method	Rel↓	RMSE↓	1.05↑	1.10↑	1.25↑	1.25^2↑	1.25^3↑
Smooth	0.151	0.187	32.80	42.71	57.61	72.29	80.15
Bilateral [64]	0.118	0.152	34.39	46.50	61.92	75.26	81.84
Fast [5]	0.127	0.154	33.65	45.08	60.36	74.52	81.79
TGV [20]	0.103	0.146	37.40	48.75	62.97	75.00	81.71
Garcia et.al [23]	0.115	0.144	36.78	47.13	61.48	74.89	81.67
FCN [80]	0.167	0.241	16.43	31.13	57.62	75.63	84.01
Ours	**0.089**	**0.116**	**40.63**	**51.21**	**65.35**	**76.74**	**82.98**

Table 8. **Comparison to baseline inpainting methods.** For the sake of comparison, we copy the methods compared in the main paper in the same table. Our method significantly outperforms baseline inpainting methods.

We also convert the completed depth maps into 3D point clouds for visualization and comparison, which are shown in Figure 13. The camera intrinsics provided in Matterport3D dataset is used to project each pixel on the depth map into a 3D point, and the color intensity are copied from the color image. Each row shows one example, with the color image and point clouds converted from ground truth, input depth (i.e. the raw depth from sensor that contains a lot of missing area), and results of our method, FCN [50], joint bilateral filter [64], and smooth inpainting. Compared to other methods, our method maintains better 3D geometry and less bleeding on the boundary.

[4] https://www.mathworks.com/matlabcentral/fileexchange/27994-inpaint-over-missing-data-in-1-d–2-d–3-d–nd-arrays

[5] https://github.com/pathak22/context-encoder

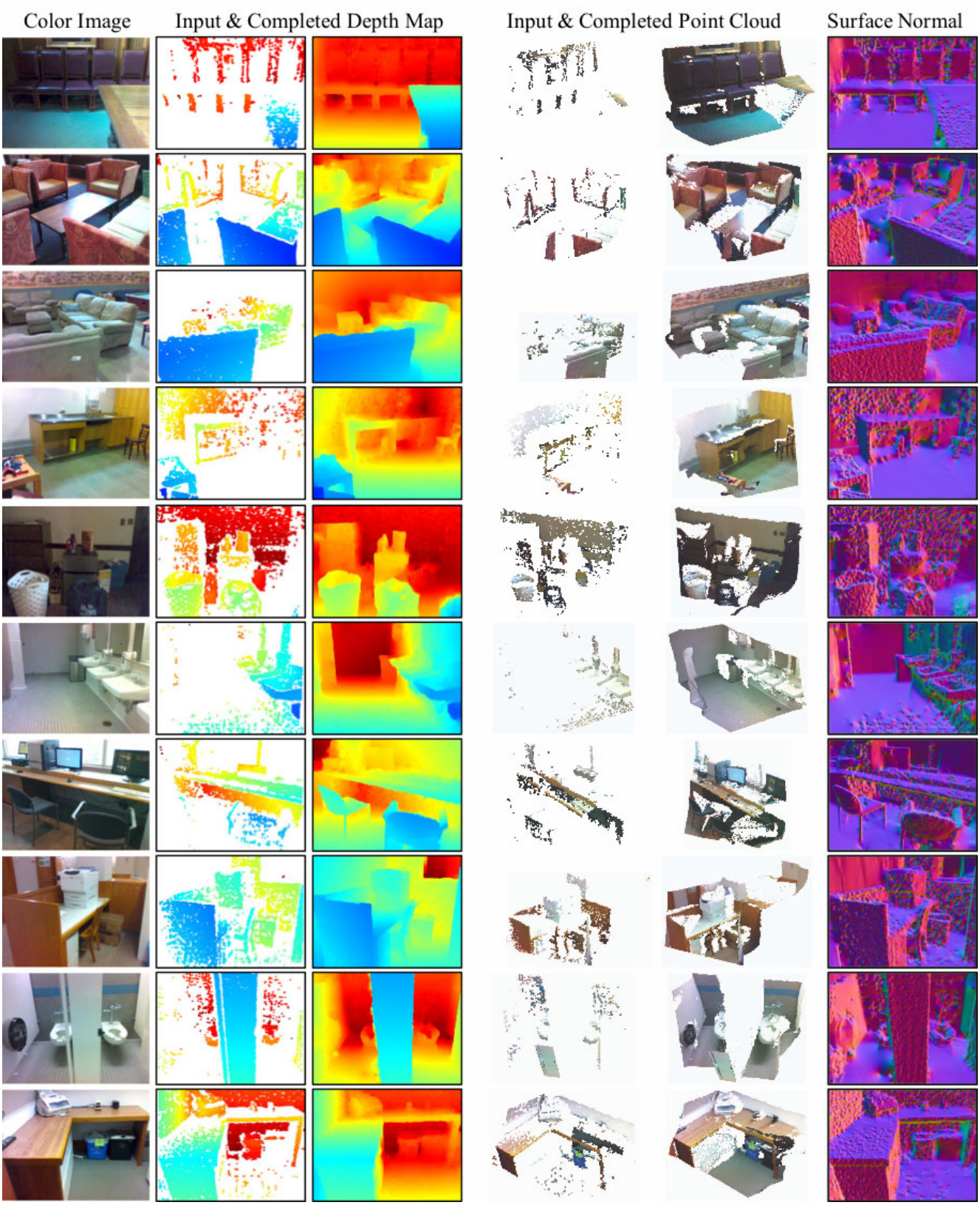


Figure 12. **Our results on RealSense data.** We run a model trained from ScanNet dataset and test on RGBD images captured by Intel RealSense without finetune. From left to right, we show the input color image, input depth image, completed depth image using our method, the point cloud visualization of the input and completed depth map, and the surface normal converted from the completed depth. Our method generates good results for depth completion.

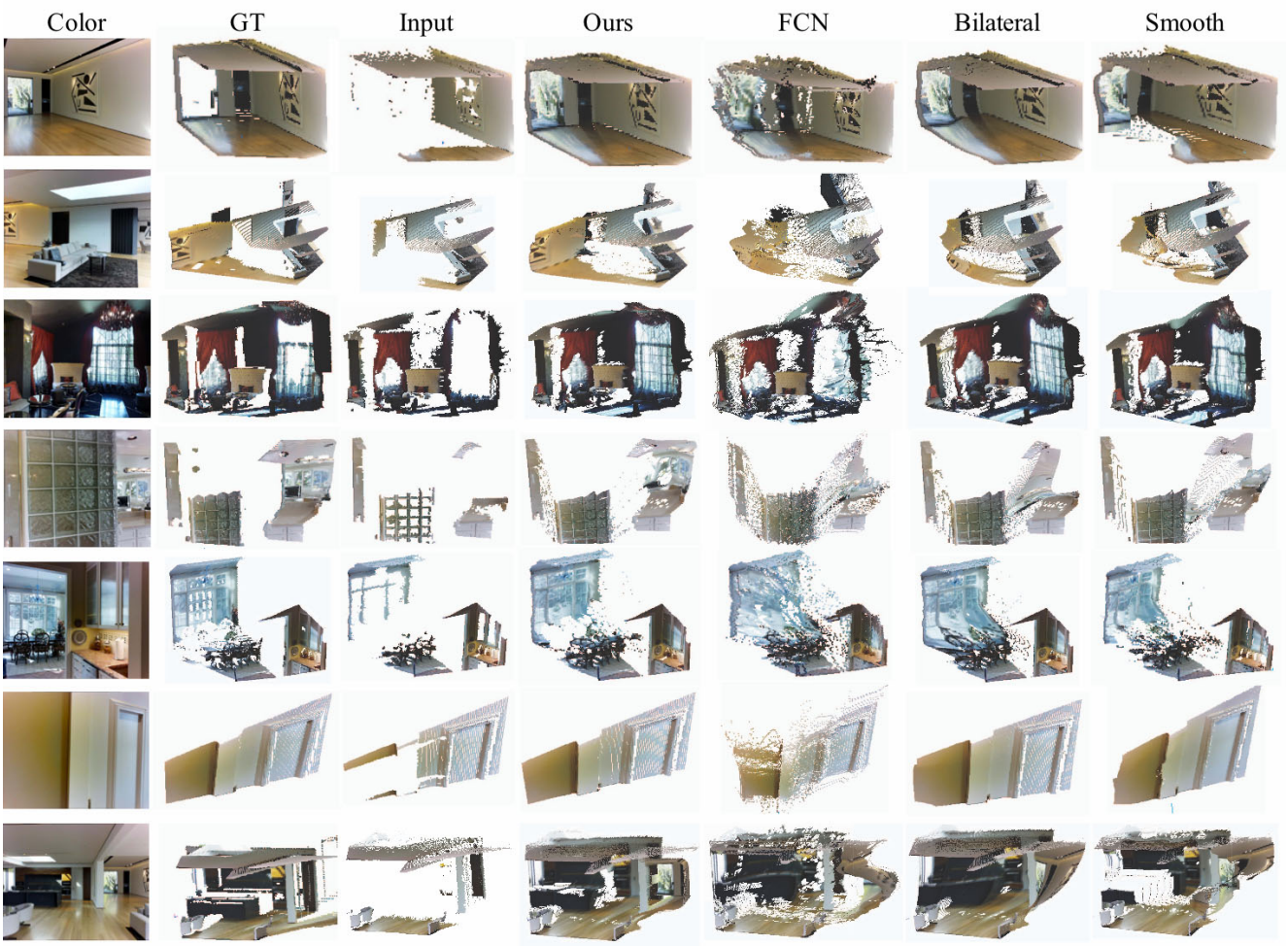


Figure 13. **Point cloud visualization of our method and other comparisons.** We convert the completed depth into point cloud. Our model produces better 3D geometry with fewer bleeding issue at the boundary.

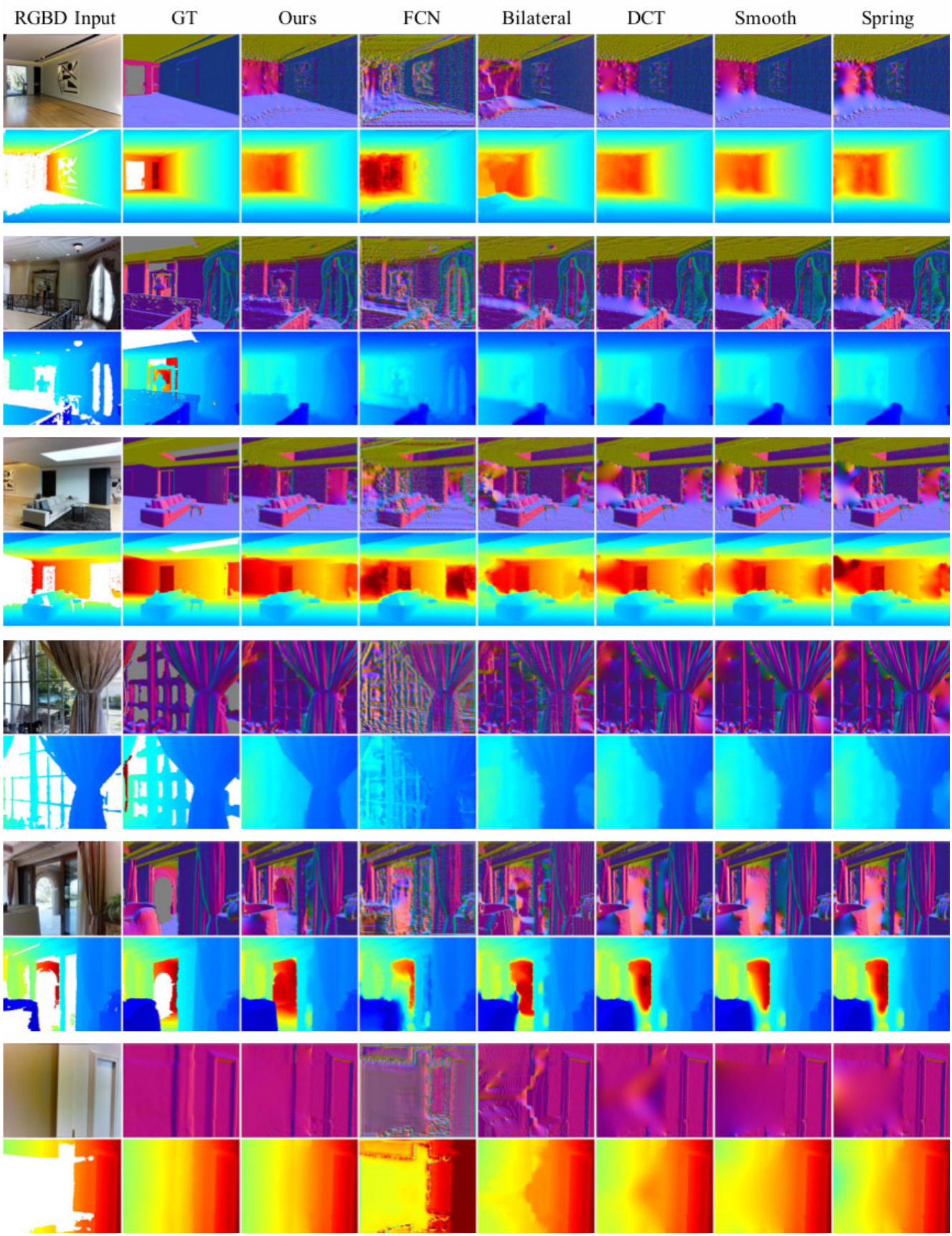


Figure 14. **More results and comparison to inpainting methods.** Each example is shown in two rows, where the second row shows the input, ground truth, and completed depth, whereas the first row shows the surface normal of each corresponding depth map on the second row to highlight details. Our method in general works better than other inpainting methods.

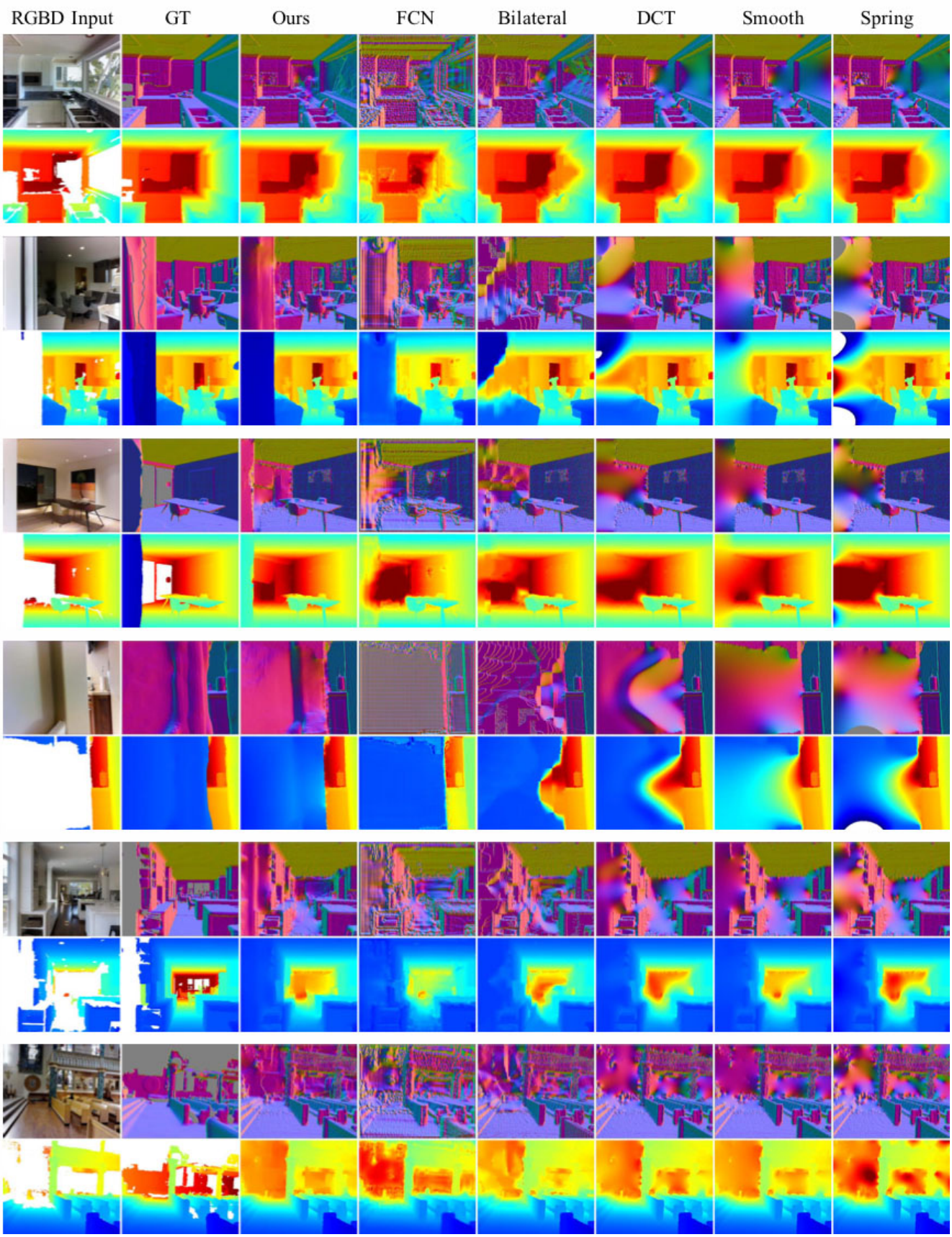


Figure 15. **More results and comparison to inpainting methods.** Each example is shown in two rows, where the second row shows the input, ground truth, and completed depth, whereas the first row shows the surface normal of each corresponding depth map on the second row to highlight details. Our method in general works better than other inpainting methods.

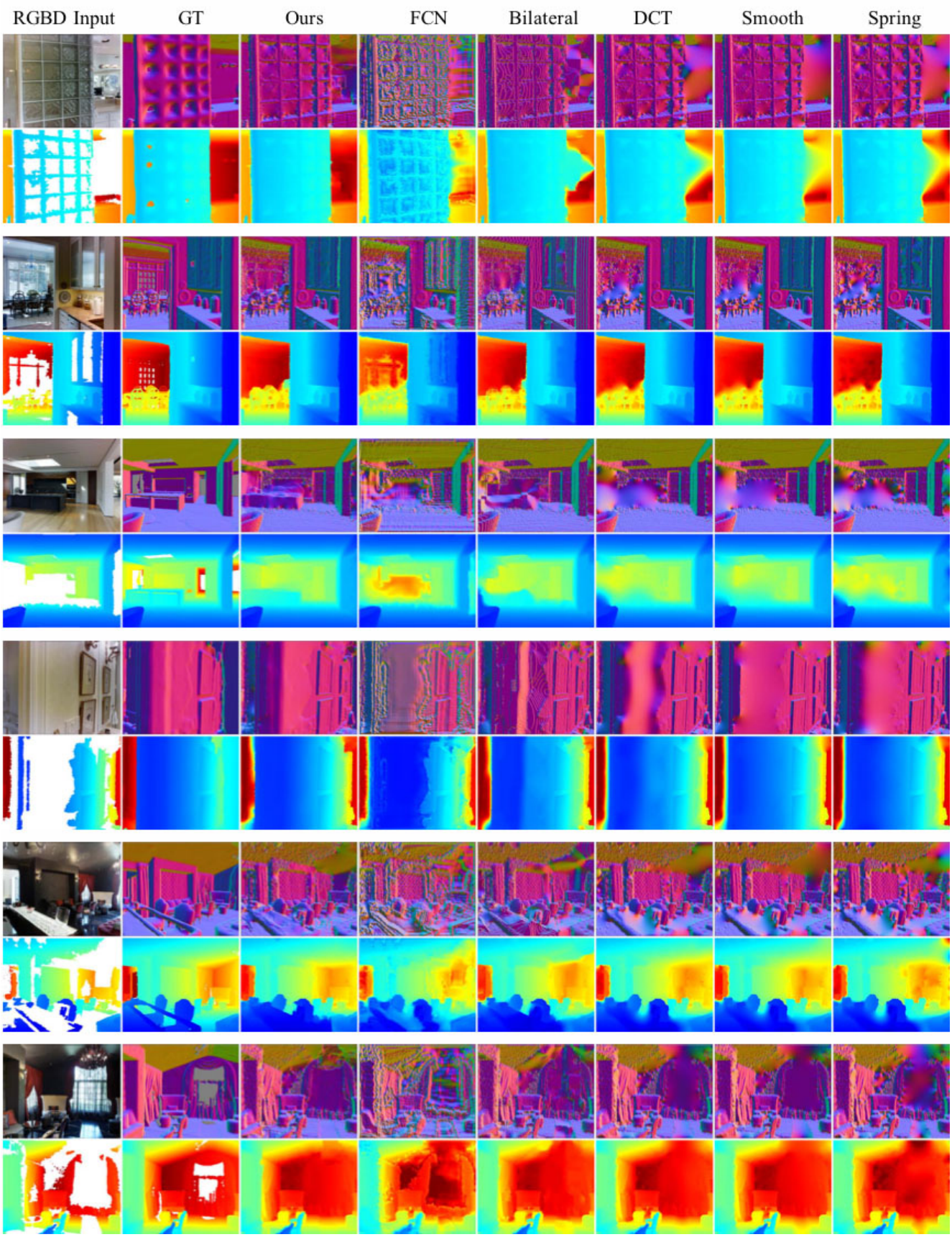


Figure 16. **More results and comparison to inpainting methods.** Each example is shown in two rows, where the second row shows the input, ground truth, and completed depth, whereas the first row shows the surface normal of each corresponding depth map on the second row to highlight details. Our method in general works better than other inpainting methods.