

Supplementary Material: Bayesian Deep Basis Fitting for Depth Completion with Uncertainty

Chao Qu

Wenxin Liu

Camillo J. Taylor

University of Pennsylvania

{quchao, wenxinl, cjtaylor}@seas.upenn.edu

6. Network Architecture

The specific network architecture is of little importance to our approach, since all methods use exactly the same basis network, initialization and random seeds. Nevertheless, we list detailed architecture here for completeness.

We use the scaffolding approach detailed in [9] to interpolate sparse depths input. We then adopt an early fusion strategy where the interpolated depth map goes through a single convolution block (with normalization and nonlinear activation) with stride 2 to be merged with the first stage feature map of the image encoder. The fused feature map is then used as input to the subsequent stages of the encoder. The encoder is an ImageNet-pretrained MobileNet-v2 [7] which output a set of multi-scale feature maps $\{E_i\}$ at each stage with channels (16, 24, 32, 96, 320) in decreasing resolution. These feature maps except for the last one are then used as skip connection to the decoder, which eventually output another set of multi-scale features $\{D_i\}$ with channels (256, 192, 128, 64, 32) in increasing resolution. We keep the encoder essentially intact, while using ELU [4] activation throughout the decoder. This is due to the suggestion from [8], where they found a smooth activation function produces better quality uncertainty estimates. The decoder output $\{D_i\}$ are then used to generate the final multi-scale bases $\{\Phi_i\}$ with channels (2, 4, 8, 16, 32) in increase resolution. They are then upsampled (via bilinear interpolation) to the input image resolution and concatenated together [6]. This final 63-dimensional basis Φ (with a bias channel) is fed into either our proposed *bdbf* module, or a normal convolution to generate the latent prediction before the final activation function g .

7. Derivations

In this section, we provide derivation of equations given in the main paper, which roughly follows the same order of their original appearance.

7.1. Marginal Likelihood

The derivation of the full marginal likelihood function largely follows that from Chapter 3.5 in [3].

We start by writing the evidence function in the form

$$\begin{aligned} p(\mathbf{z}|\alpha, \beta) &= \int p(\mathbf{z}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)d\mathbf{w} \\ &= |\Sigma_0|^{-1/2} \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}} \left(\frac{\alpha}{2\pi}\right)^{\frac{M}{2}} \int \exp\left\{-\frac{1}{2}E(\mathbf{w})\right\} d\mathbf{w} \end{aligned} \quad (1)$$

where

$$E(\mathbf{w}) = \beta\|\mathbf{z} - \Phi\mathbf{w}\|^2 + \alpha\|\mathbf{w} - \mathbf{m}_0\|_{\Sigma_0}^2 \quad (2)$$

Completing the square over \mathbf{w} we have

$$E(\mathbf{w}) = E(\mathbf{m}) + \|\mathbf{w} - \mathbf{m}\|_{\Sigma}^2 \quad (3)$$

The integral term is evaluated by

$$\begin{aligned} &\int \exp\left\{-\frac{1}{2}E(\mathbf{w})\right\} d\mathbf{w} \\ &= \exp\left\{-\frac{1}{2}E(\mathbf{m})\right\} (2\pi)^{M/2} |\Sigma|^{-1/2} \end{aligned} \quad (4)$$

We can then write the log marginal likelihood as

$$\begin{aligned} \ln p(\mathbf{z}|\alpha, \beta) &= \frac{1}{2}(N \ln \beta + M \ln \alpha - N \ln(2\pi) \\ &\quad - E(\mathbf{m}) + \ln |\Sigma| - \ln |\Sigma_0|) \end{aligned} \quad (5)$$

7.2. Normalized Estimation Error Squared (NEES)

NEES was originally used to indicate the performance of a filter [2], *e.g.* in a tracking application. It is defined as

$$\varepsilon_k = (\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^\top \mathbf{P}_{k|k}^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_{k|k}) \quad (6)$$

where \mathbf{x} is the true state vector, \mathbf{P} is the state covariance matrix, and $(\cdot)_{k|k}$ denotes the estimated posterior at time

k . It is also used in Simultaneous Localization and Mapping algorithms (SLAM) to measure filter consistency [1]. Specifically, the average NEES over N Monte Carlo runs is used. Under the assumption that the model is correct (approximately linear Gaussian), ε_k is χ^2 distributed with $\dim(\mathbf{x}_k)$. Then the expected value of ε_k should be

$$E[\varepsilon_k] = \dim(\mathbf{x}_k) \quad (7)$$

For depth estimation, the state is the predicted depth at each pixel, which is one-dimensional. Therefore, we expect a consistent depth estimator to have an average NEES of 1.

NEES can also be extended to a Laplace distribution which is used in this work due to the choice of L1 loss. Given $z \sim \text{Laplace}(\mu, b)$ and the fact that $\frac{2}{b}|z-\mu| \sim \chi^2(2)$, the expected NEES (with one degree of freedom) is

$$E[\varepsilon] = E\left[\left(\frac{z-\mu}{b}\right)^2\right] = 1 \quad (8)$$

7.3. Re-estimation Equations

Here we provide derivation of the re-estimation equations used in the EM step during inference [3]. In the E step, we compute the posterior distribution of \mathbf{w} given the current estimation of the parameters α and β . In the M step, we maximize the expected complete-data log likelihood with respect to α and β and re-iterate until convergence. The convergence criteria is met when the change in relative magnitude of β is smaller than 1%.

The complete-data log likelihood function is given by

$$\ln p(\mathbf{z}, \mathbf{w}|\alpha, \beta) = \ln p(\mathbf{z}|\mathbf{w}, \beta) + \ln p(\mathbf{w}|\alpha) \quad (9)$$

with

$$p(\mathbf{z}|\mathbf{w}, \beta) = \mathcal{N}(\mathbf{z}|\Phi\mathbf{w}, \beta^{-1}\mathbf{I}) \quad (10)$$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \alpha^{-1}\Sigma_0) \quad (11)$$

The expectation w.r.t. the posterior distribution of \mathbf{w} is

$$\begin{aligned} & E[\ln p(\mathbf{z}, \mathbf{w}|\alpha, \beta)] \\ &= \frac{M}{2} \ln\left(\frac{\alpha}{2\pi}\right) - \frac{1}{2} \ln|\Sigma_0| - \frac{\alpha}{2} \mathbb{E}[\|\mathbf{w} - \mathbf{m}_0\|_{\Sigma_0}^2] \\ &+ \frac{N}{2} \ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2} \mathbb{E}[\|\mathbf{z} - \Phi\mathbf{w}\|^2] \end{aligned} \quad (12)$$

Setting the derivatives with respect to α and β to zero gives us the re-estimation equations.

$$\alpha = \frac{M}{\mathbb{E}[\|\mathbf{w} - \mathbf{m}_0\|_{\Sigma_0}^2]} \quad (13)$$

$$\begin{aligned} &= \frac{M}{\text{tr}(\Sigma_0^{-1}\Sigma) + \|\mathbf{m} - \mathbf{m}_0\|_{\Sigma_0}^2} \\ \beta &= \frac{N}{\mathbb{E}[\|\mathbf{z} - \Phi\mathbf{w}\|^2]} \\ &= \frac{N}{\text{tr}(\Phi^T\Phi\Sigma) + \|\mathbf{z} - \Phi\mathbf{m}\|^2} \end{aligned} \quad (14)$$

8. More Results

8.1. Using BDBF as a General Component

Here we verify our claim that *bdbf* can be used as a general, drop-in, component in a depth completion network. The best published method on the KITTI depth completion benchmark at the time of this writing is PENet [5]. It has a fully-convolutional backbone called ENet that also ranks third on the associated leaderboard. For details of their approach, we refer the reader to the original paper [5]. We only discuss the minimal changes that we made to add *bdbf*.

Figure 1 shows the network architecture for ENet. The color-dominant (CD) branch uses a convolutional layer to predict CD-depth and CD-confidence. We simply replaced this convolutional layer with our *bdbf* module. However, we needed to normalize our variance prediction to conform to their notion of confidence. This was done by inverting the standard deviation and passing it through a Tanh non-linearity to constrain the result to lie between 0 and 1. We then trained both versions for 10 epochs. Training and validation results are show in Figure 2. We see that by using *bdbf*, we achieve slightly better validation RMSE and faster convergence.

8.2. Inference Time

In table 3, we show inference time of all methods when running on an image of size 320×240 with 5% sparsity levels. All ensemble methods (*snap / snap+log*) use 5 snapshots as described in the main paper. They take roughly $5 \times$ times longer for an inference pass compared to *log*, which is the fastest among all. For *bdbf* we use a maximum iteration of 8, but we observe convergence within 2 iterations for this particular density. *bdbf* is slower than *log* due to the need to solve a small linear system and the iterations required for EM, but it is has slower latency and smaller memory footprints compared to ensemble methods. Note that during inference, we move expensive computations in our method like Cholesky and LU decomposition from GPU to CPU and then move the results back. Since this is only done in evaluation mode, no back-propagation is needed and thus no computation graph broken. The time needed to transfer small tensors ($\dim(\mathbf{w}) \times \dim(\mathbf{w})$) between host and device memory is negligible compared to carrying out those operations on GPU. Without these changes, *bdbf* runs as slow as *snap*.

8.3. Estimator Consistency

Here we report average NEES scores for all methods on various datasets. NEES is not used as a metric in our evaluation since our method explicitly uses it for uncertainty calibration and would render the comparison unfair. We present it here only to show the efficacy of our uncertainty calibration scheme. Figure 3 and 4 show NEES of all methods un-

Trained with 5%			VKITTI2						NYU-V2					
Input	Method	%	MAE	RMSE	$\mathbf{1}$	AUSE	AUCE	NLL	MAE	RMSE	$\mathbf{1}$	AUSE	AUCE	NLL
rgbd	snap	5%	1.192	3.267	95.59	0.445	0.170	-0.714	0.061	0.126	99.35	0.036	0.202	-1.390
rgbd	snap+log	5%	1.271	3.432	95.33	0.142	0.117	-1.582	0.058	0.123	99.32	0.018	0.256	-1.596
rgbd	log	5%	1.318	3.423	95.37	0.149	0.125	-1.421	0.057	0.121	99.34	0.018	0.210	-1.783
rgbd	dbf	5%	0.709	2.928	97.88	0.148	0.163	-2.489	0.026	0.083	99.64	0.007	0.054	-3.145
rgbd	bdbf	5%	0.703	2.925	97.88	0.110	0.136	-2.596	0.026	0.082	99.64	0.007	0.039	-3.151
rgbd	snap	1%	1.784	4.679	92.04	0.805	0.214	0.730	0.087	0.198	97.65	0.051	0.238	-0.387
rgbd	snap+log	1%	1.805	4.755	92.03	0.235	0.062	-1.333	0.089	0.211	97.35	0.025	0.202	-1.451
rgbd	log	1%	1.831	4.769	92.21	0.223	0.147	-1.168	0.087	0.214	97.31	0.023	0.157	-1.605
rgbd	dbf	1%	1.393	4.384	94.52	0.350	0.110	-1.670	0.055	0.155	98.45	0.016	0.072	-2.381
rgbd	bdbf	1%	1.382	4.372	94.53	0.271	0.082	-1.707	0.055	0.155	98.45	0.016	0.059	-2.374

Table 1: Quantitative results of all methods trained and tested with 5% and 1% sparsity on VKITTI2 and NYU-V2.

Trained with 500			VKITTI2						NYU-V2					
Input	Method	#	MAE	RMSE	$\mathbf{1}$	AUSE	AUCE	NLL	MAE	RMSE	$\mathbf{1}$	AUSE	AUCE	NLL
rgbd	snap	500	2.312	5.403	90.14	0.459	0.229	-0.207	0.096	0.206	97.53	0.053	0.261	0.211
rgbd	snap+log	500	2.396	5.571	89.88	0.273	0.036	-1.150	0.095	0.213	97.44	0.025	0.205	-1.393
rgbd	log	500	2.492	5.800	89.13	0.299	0.095	-0.906	0.097	0.212	97.44	0.025	0.152	-1.502
rgbd	dbf	500	2.050	5.067	92.58	0.453	0.051	-1.175	0.065	0.168	98.45	0.020	0.055	-2.186
rgbd	bdbf	500	2.015	4.994	92.71	0.392	0.014	-1.215	0.064	0.166	98.46	0.021	0.030	-2.199
rgb	bdbf	500	2.569	5.642	88.67	0.481	0.015	-0.979	0.098	0.199	98.48	0.030	0.014	-1.689
rgbd	snap	50	5.069	9.326	66.48	1.696	0.327	3.495	0.324	0.605	78.89	0.136	0.376	6.036
rgbd	snap+log	50	5.173	9.401	65.91	1.180	0.188	0.608	0.312	0.593	80.55	0.084	0.052	-0.390
rgbd	log	50	5.107	9.433	66.52	1.114	0.275	2.439	0.323	0.599	80.49	0.088	0.134	-0.166
rgbd	bdbf	50	4.098	8.440	76.91	0.977	0.021	-0.411	0.215	0.446	88.63	0.081	0.026	-0.926
rgb	bdbf	50	3.725	7.786	80.67	0.701	0.012	-0.612	0.144	0.296	94.97	0.039	0.011	-1.338

Table 2: Quantitative results of all methods trained and tested with 500 and 50 sparse depths. *rgbd* under the input column indicates the basis network uses the sparse depths scaffolding approach from [9], whereas *rgb* uses color image as basis network input only.

Inference	snap	snap+log	log	dbf	bdbf
Time [ms]	68.84	66.17	16.68	21.84	23.97

Table 3: Inference time of all methods with 5% sparsity. Image resolution is 320×240 .

der mid- and low- density with varying sparsity level. We see that our methods are the most consistent of all and remain relatively consistent even with sparsity change. Note that this consistency-based calibration can not be applied to other methods, because we did not observe the same amount of over- or under-confidence from them. While calibration of other baseline models is feasible, doing so would require extra data and thus not considered in this work.

8.4. Shared Prior

Figure 5 supports our claim in the main paper about the assumption of a shared prior across samples. Here we show the weight histograms of two training sessions on different datasets (NYU-V2 and VKITTI2). We see that within each dataset, the weight distribution exhibits a fairly sharp peak. The small bump on the right side of each plot is formed by the bias term in the weights which reflects the average log depth of the dataset.

8.5. Mid-density Depth Completion

In this section, we present extra results on mid-density depth completion. Table 1 shows quantitative results of all methods trained with 5% sparsity and test on 5% and 1% sparsity. Here we also include comparison with *dbf*, where we compute the variance by assuming an infinitely broad prior *without* the EM step. We see that *dbf* and *bdbf* have perform similar due to the large amount of sparse depths, but *bdbf* still has the best performance overall. Figure 6 and 7 show some extra in-distribution samples of all methods from VKITTI2 and NYU-V2 respectively.

8.6. Low-density Depth Completion

Table 2 shows quantitative results of all methods trained with 500 sparse depths and test on 500 and 50 sparse depths. We choose 50 because it is smaller than the number of basis (63). This would previously fail with DBF, but BDBF have no problem dealing with any amount of sparse depths. Moreover, we suggest that when designing a system working under extremely low sparsity (≤ 100 points) it is better to forgo the idea of a depth encoder altogether, because convolution is simply not designed for sparse data and interpolation schemes rarely work with only a few points.

Figure 8 and 10 show some extra in-distribution samples

Figure 1: The backbone of PENet [5], which is called ENet, and our modification. Figure taken directly from the paper.

Figure 2: Training and validation RMSE of PENet and PENet-bdbf after training for 10 epochs on KITTI Depth Completion dataset.

Figure 3: NEES scores for all methods with 5% sparsity. Closer to 1 (dashed black line) means more consistent.

of all methods from VKITTI2 and NYU-V2 respectively.

Figure 9 shows more qualitative results of (rgb) and log(rgb) on VKITTI2 with no sparse depths.

Figure 4: NEES scores for all methods with 500 sparse depths. Closer to 1 (dashed black line) means more consistent. Note the log scale on y-axis.

Figure 5: Tensorboard records of Weight histograms of two training sessions on NYU-V2 (left) and VKITTI2 (right).

8.7. Domain Shift

There are many scenarios to domain shift in depth completion, for example, different sparsity levels, sim-to-real or even dataset change.

For sim-to-real, we take the same models that are trained on VKITTI2 and directly test on KITTI without any re-tuning. The data distribution shift in this case manifests most notably in image quality as well as noise level and spatial distribution of sparse depths. Quantitative results are shown in Table 4, where we also list in-distribution test

Figure 6: Sample qualitative results of all methods trained and test with 5% sparsity on VKITTI2. Colormaps scales are different for each methods to visualize details. Axes scales are same for all methods.

Trained and tested on KITTI (original)						
Method	MAE	RMSE	ρ_1	AUSE	AUCE	NLL
snap	0.638	1.663	98.68	0.272	0.161	-1.298
snap+log	0.701	1.784	98.52	0.092	0.105	-1.605
log	0.899	2.022	98.44	0.109	0.133	-1.544
bdbf	0.364	1.682	98.74	0.060	0.100	-2.337

Trained on VKITTI2 and tested on KITTI (shifted)						
Method	MAE	RMSE	ρ_1	AUSE	AUCE	NLL
snap	1.296	2.549	92.03	0.876	0.161	-0.675
snap+log	0.853	2.245	97.69	0.134	0.117	-1.578
log	0.866	2.018	97.96	0.138	0.274	-0.755
bdbf	0.469	1.989	98.29	0.072	0.037	-2.381

Table 4: Quantitative results of all methods trained on KITTI and VKITTI with 5% sparsity and tested on KITTI.

VKITTI2 (15-deg)						
Method	MAE	RMSE	ρ_1	AUSE	AUCE	NLL
snap	1.164	3.161	95.67	0.445	0.172	-0.733
snap+log	1.206	3.260	95.43	0.131	0.103	-1.558
log	1.301	3.328	95.38	0.144	0.137	-1.347
bdbf	0.686	2.861	97.92	0.101	0.139	-2.618

VKITTI2 (30-deg)						
Method	MAE	RMSE	ρ_1	AUSE	AUCE	NLL
snap	1.084	3.009	95.39	0.428	0.175	-0.740
snap+log	1.139	3.106	95.10	0.128	0.087	-1.509
log	1.217	3.166	95.10	0.141	0.146	-1.251
bdbf	0.627	2.711	98.02	0.085	0.143	-2.676

Table 5: Quantitative results of all methods trained with 5% sparsity on VKITTI2 and test on 15-deg and 30-deg sequences from VKITTI2.

on KITTI for comparison. Most methods encounter performance drop in one or several metrics when tested under a slightly different domain. However, bdbf achieve overall best results in both scenarios, with its performance under distributional shift better than some of the baselines in the original domain.

For more tests on distributional shift, we utilize the 15-deg and 30-deg sequences from VKITTI2. These sequences are variants of one with the camera pointing at different angles. Table 5 shows quantitative results of all methods trained with 5% sparsity on VKITTI2 and test on these two sequences.

Finally, we take models trained on NYU-V2 and evalu-

ated on VKITTI2 without any modification or re-tuning. This is a complete dataset change, from indoor to outdoor

NYU-V2		Test on VKITTI2					
Name	#	MAE	RMSE	ρ_1	AUSE	AUCE	NLL
log	5%	15.16	20.89	1.143	0.256	0.479	10.112
bdbf	5%	0.914	3.044	96.88	0.121	0.162	-2.084
log	500	17.18	22.17	0.137	12.18	0.448	4.457
bdbf	500	2.364	5.900	88.51	0.785	0.086	-0.481

Table 6: Quantitative results of log vs bdbf trained with 5% and 500 sparsity on NYU-V2 and test on VKITTI2.

scenes, with different camera intrinsics and image sizes. Results are shown in Table 6. We see that completely fails on this dataset change, while hb maintains relatively high performance and uncertainty estimation.

References

- [1] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3562–3568, 2006. 2
- [2] Y. Bar-Shalom, X. Li, and T. Kirubarajan. Estimation with applications to tracking and navigation: Theory, algorithms and software. 2001. 1
- [3] C. M. Bishop. Pattern recognition and machine learning (information science and statistics). 2006. 1, 2
- [4] Djork-Arné Clevert, Thomas Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). CoRR abs/1511.07289, 2016. 1
- [5] Mu Hu, S. Wang, Bin Li, Shiyu Ning, L. Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. ArXiv, abs/2103.00783, 2021. 2, 4
- [6] Chao Qu, Ty Nguyen, and Camillo J. Taylor. Depth completion via deep basis fitting. 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 71–80, 2020. 1
- [7] Mark Sandler, A. Howard, Menglong Zhu, A. Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018. 1
- [8] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, N. Sundaram, Md. Mostofa Ali Patwary, Prabhathat, and R. Adams. Scalable bayesian optimization using deep neural networks. In ICML, 2015. 1
- [9] Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from visual inertial odometry. IEEE Robotics and Automation Letters, 5:1899–1906, 2020. 1, 3

Figure 7: Sample qualitative results of all methods trained and test with 5% sparsity on NYU-V2. Colormaps scales are different for each methods to visualize details. Axes scales are same for all methods.

Figure 8: Sample qualitative results of all methods trained and test with 500 sparse points on VKITTI2. Colormaps scales are different for each methods to visualize details. Axes scales are same for all methods.

Figure 9: More qualitative results of our method tested with 0 sparse points ($\log(\text{rgb})$) is trained as a monocular depth prediction network with NLL loss, which serves as a baseline. $\log(\text{rgb})$ is trained with 500 sparse depths.

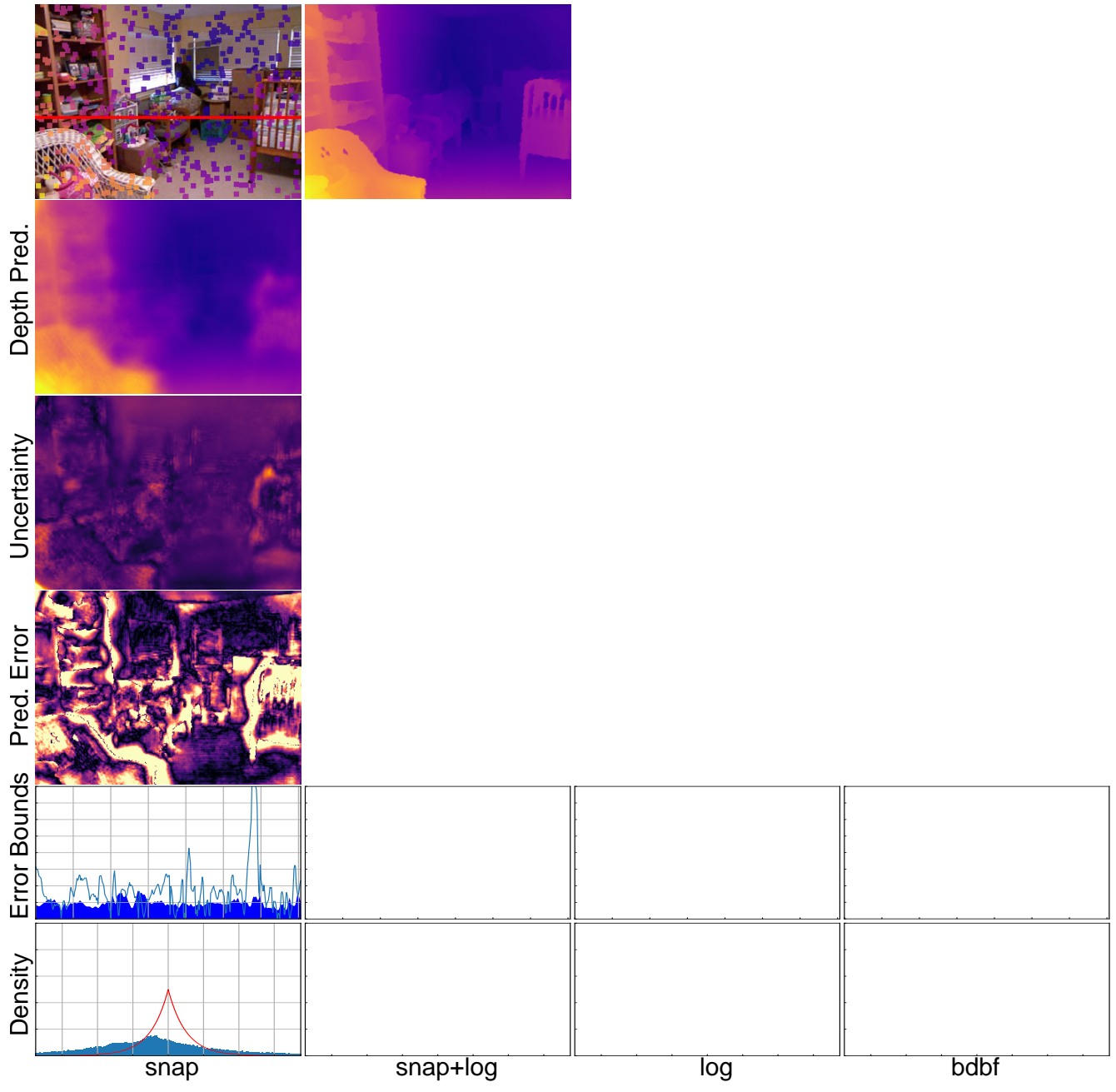


Figure 10: Sample qualitative results of all methods trained and test with 500 sparse points on NYU-V2. Colormaps scales are different for each methods to visualize details. Axes scales are same for all methods.