Generating and Exploiting Probabilistic Monocular Depth Estimates: Supplementary Material

Zhihao Xia, Patrick Sullivan, Ayan Chakrabarti

A. Additional Results

We include additional example results of depth reconstruction for various applications in Figure 5.

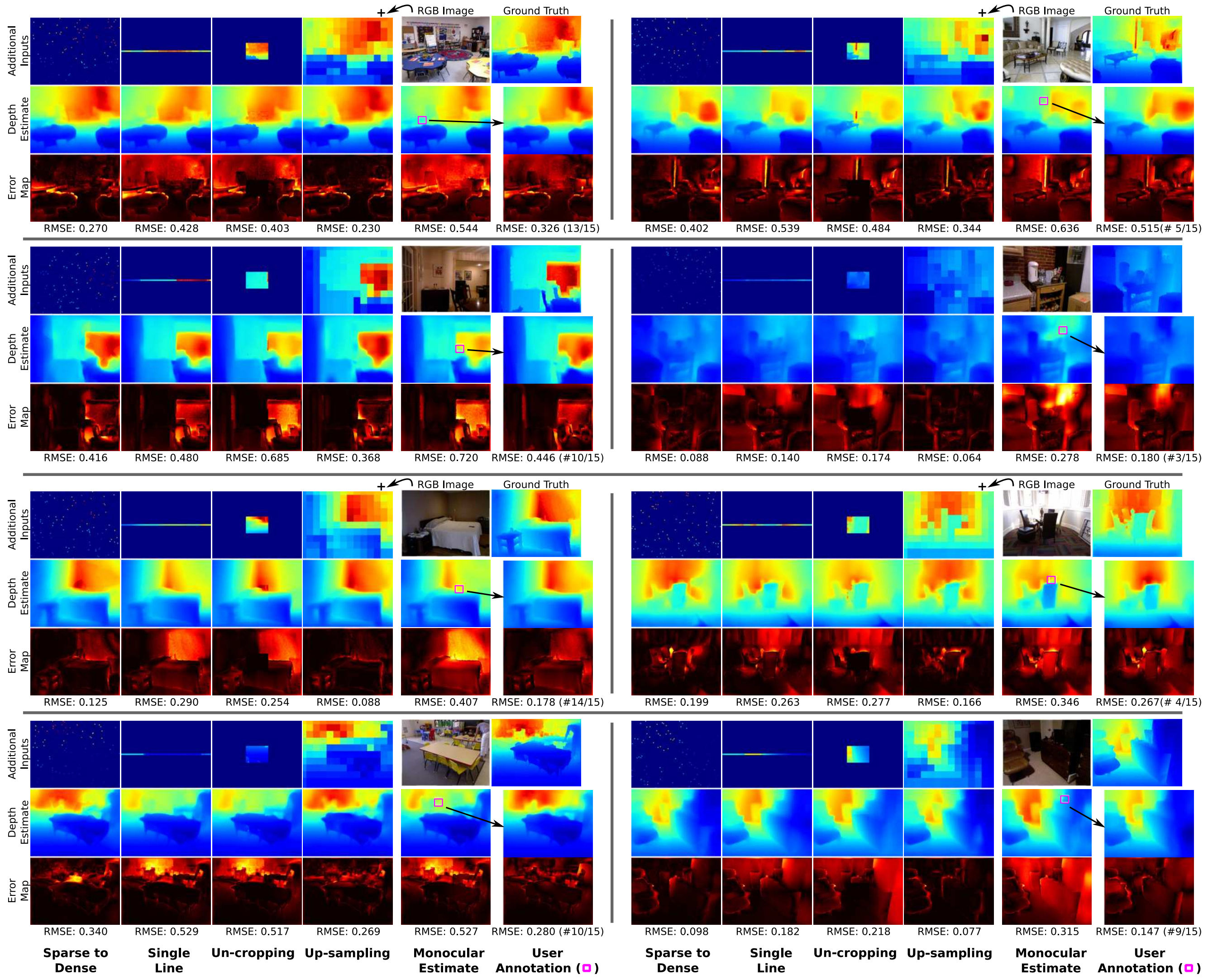

Figure 5. Additional examples of depth reconstructions using our common model for various applications.

B. Discussion

MRF Model. Here we discuss and provide additional context for our formulation for $p(\mathbf{Z}|\mathbf{I})$ in (1). Our overall strategy is to define potential functions in (2) that model the (conditional) joint distributions of depth values within a patch, and then apply these distributions on overlapping patches to form the distribution in $p(\mathbf{Z}|\mathbf{I})$. This corresponds to an MRF where the patches are the maximal-cliques. Note that this is not a pairwise MRF, since each potential is a joint function of all depths within a patch. The distribution $p(\mathbf{Z}|\mathbf{I})$ is defined in (1) up to a proportionality constant: as is common in MRF models, this constant is intractable to compute, but is typically not needed during inference (since it does not depend on $\mathbf{Z}$). Note that the potentials $\psi_i(\cdot)$ are *not* the marginals of the distribution $p(\mathbf{Z}|\mathbf{I})$, since they are defined on overlapping patches. It is also due to this overlap that different patches (and thus all depth values) become dependent in $p(\mathbf{Z}|\mathbf{I})$ through their shared pixels.

	$h^2 = h_0^2$	$h^2 = h_0^2/2$	$h^2 = h_0^2/5$	$h^2 = h_0^2/10$
Avg-ratio r	1.2×10^2	2.4×10^2	6.2×10^2	1.3×10^3

Table 5. Average ratio of dominant to non-dominant terms in the summationin patch-potentials, for different values of bandwidth h^2.

This technique of factoring a larger distribution into smaller overlapping cliques is a classic technique. For example, Zoran & Weiss [54] use a similar strategy for modeling image intensities for restoration tasks: using potentials defined as log-likelihoods of a Gaussian mixture model (GMM) for individual patches, and applying such potentials on overlapping patches. It is worth noting that in our approach (like also in [54]), the potential functions are defined to match the marginal distributions of individual patches, even though they are then applied on overlapping patches (which, as mentioned above, means they are not marginals of the global distribution). Other works attempt to learn potentials in a manner based on the likelihood of the full distribution (e.g., [38]). However, this is significantly more expensive computationally, especially in our setting where the potentials are derived using a C-VAE, motivating our use of this more approximate approach (as in [54]).

Dominant Term Approximation. To make our estimation method in Sec. 3.2 tractable, we approximate the summation of terms from each sample in (3) with a $\max$ in (4), relying on the fact that $\mathcal{P}_i\mathbf{Z}$ is high-dimensional, and thus the maximum term in the summation will dominate the other terms. This is a common approximation for sums of exponentials of this form in high-dimensional spaces (e.g., [54] also uses this for a GMM).

We evaluate this in the context of samples generated by our C-VAE, by measuring the ratio between the dominant term to other terms in the summation, computed with respect to ground-truth depth, and averaged over all patches and non-dominant samples on multiple scenes in a validation set:

$$r = \text{Avg}\left[\max_{\mathbf{x}_i \in \mathcal{S}_i}\left(\exp\left(-\frac{\|\mathcal{P}_i\mathbf{Z}_{GT} - \mathbf{x}_i\|^2}{2h^2}\right)\right) \Big/ \exp\left(-\frac{\|\mathcal{P}_i\mathbf{Z}_{GT} - \tilde{\mathbf{x}}_i\|^2}{2h^2}\right)\right],\ \ \tilde{x_i} \neq \arg\max_{\mathbf{x}_i} \exp\left(-\frac{\|\mathcal{P}_i\mathbf{Z}_{GT} - \mathbf{x}_i\|^2}{2h^2}\right). \tag{14}$$

Note that values of these ratios will depend on the choice of the bandwidth hyper-parameter h^2 (even though we did not need to explicit set this value during inference, as it was absorbed in other parameters, such as the external cost $C(\cdot)$). As reference, we first compute the average per-pixel variance h_0^2 in depth values across sample sets $\mathcal{S}_i$ for different patches. Since, h^2 is set to be lower than h_0 to allow the summation to express multi-modality, we consider different possible values of h^2 as different fractions of h_0^2, and report corresponding average ratios (14) in Table 5. We see that the dominant term is typically two to three orders of magnitude larger than other terms included in the summation.

C. Additional Details

C.1. DORN Usage and Resolution

Our conditional VAE leverages a pre-trained feature extractor derived from a state-of-the-art monocular depth network. Specifically, we take the pre-trained DORN model [9], remove its last two convolutional layers, and use it as our feature extractor. The DORN network works at a lower resolution of 257×353, compared to the original NYUv2 resolution of 640×480, for both its RGB input and depth output. Therefore, our feature extractor takes an RGB image as input after resizing to the lower DORN resolution of 257×353. The output of the feature extractor layers is a 2560-dimensional feature map with a spatial size of 33×45. Our VAE takes this feature map as input, and reasons about an output depth map at the 257×353 DORN resolution. We consider overlapping 33×33 patches at stride 4 also at the lower DORN resolution of 257×353, giving us a total of 57×81 patches, each of size 33×33.

Thus, our distributional output corresponds to the lower DORN [9] resolution of 257×353 for the depth map. However, all error metrics in the paper are computed (inside the valid crop) at full resolution. To do so, we resize our method's outputs to 640×480 (by simple bilinear interpolation). Moreover, in all applications with additional inputs, these are also provided at the original higher resolution. For user annotations, erroneous regions are marked as 50×50 windows at the full resolution, and we map the locations of these windows to the lower resolution to construct our masks $\mathbf{W}^M$. Similarly, for depth from sparse measurements, $\mathbf{F}$ corresponds to sparse measurements of depth at the full-resolution, and our global cost $C^G(\cdot)$ is defined in terms of a full-resolution depth map (we scale our depth map to the full resolution, apply gradients, and scale the updated depth map back). For depth un-cropping, we again provide depth measurements at the full resolution, and scale these to the DORN resolution to construct our measurement and mask vectors $\mathbf{F}$ and $\mathbf{W}$. Thus, all inputs and all evaluation metrics are based on the standard benchmark resolution.

No.	*Layer*	*Output Shape*
0	features from feature extractor	1×33×45×2560
1	bilinear upsample	1×65×89×2560
2	conv 1×1	1×65×89×1024
3	conv 1×1	1×65×89×512
4	conv 3×3 dilation=2	1×61×85×512
5	conv 3×3 dilation=2	1×57×81×256
6	conv 1×1	1×57×81×256
7	conv 1×1	1×57×81×256
8	conv 1×1 (no ReLU)	1×57×81×256
9	reshape and split	(57*81)×1×1×128 Mean (57*81)×1×1×128 log-Sigma

Prior-net

No.	*Layer*	*Output Shape*
0a	features from feature extractor	1×33×45×2560
1a	bilinear upsample	1×65×89×2560
2a	conv 1×1	1×65×89×1024
3a	conv 1×1	1×65×89×512
4a	conv 3×3 dilation=2	1×61×85×512
5a	conv 3×3 dilation=2	1×57×81×256
6a	reshape	(57*81)×1×1×256
0b	sample from latent distribution	(57*81)×1×1×128
0	concat: 6a and 0b	(57*81)×1×1×384
1	conv_transpose 3×3	(57*81)×3×3×256
2	conv_transpose 3×3	(57*81)×5×5×128
3	conv_transpose 3×3	(57*81)×7×7×64
4	bilinear upsample	(57*81)×13×13×64
5	conv_transpose 3×3	(57*81)×15×15×32
6	conv_transpose 3×3	(57*81)×17×17×16
7	bilinear upsample	(57*81)×33×33×16
8	conv 1×1 + tanh	(57*81)×33×33×1
9	reshape	57 × 81 × [33 × 33]

Encoder-decoder

No.	*Layer*	*Output Shape*
0.a	features from feature extractor	1×33×45×2560
1.a	bilinear upsample	1×65×89×2560
2.a	conv 3×3 dilation=2	1×61×85×1024
3.a	conv 3×3 dilation=2	1×57×81×256
0.b	GT depth patches	57 × 81 × [33 × 33]
1.b	reshape	(57*81)×33×33×1
2.b	conv 3×3 stride=2	(57*81)×16×16×8
3.b	conv 2×2 stride=2	(57*81)×8×8×16
4.b	conv 2×2 stride=2	(57*81)×4×4×32
5.b	conv 2×2 stride=2	(57*81)×2×2×64
6.b	reshape	(57*81)×1×1×256
7.b	reshape	1×57×81×256
0	concat: 3.a and 7.b	1×57×81×512
1	conv 1×1	1×57×81×1024
2	conv 1×1	1×57×81×512
3	conv 1×1	1×57×81×256
4	conv 1×1 (no ReLU)	1×57×81×256
5	reshape and split	(57*81)×1×1×128 Mean (57*81)×1×1×128 log-Sigma

Posterior-net

Table 6. Conditional VAE architecture. We show architecture details of the three different sub-networks in our VAE, with the posterior-net used only during training. Valid padding is used everywhere. Every convolutional layer is followed by a ReLU, unless otherwise specified. The output of the encoder-decoder network has a tanh activation, followed by scaling to map to the depth range of the NYUv2 dataset.

C.2. Conditional VAE Architecture

Our conditional VAE treats the output of the DORN feature extractor—with a spatial resolution of 33×45 and 2560 feature channels—as an encoding of the input image. Closely following the formulation of [18], this VAE has the following three sub-networks:

1. *Prior-net*: Given the input image feature encoding, this network produces the mean and variance vectors for each of the 57×81 overlapping patches. These vectors represent the parameters of diagonal Gaussian distributions over the latent space of the corresponding patches. The latent space, and the per-patch mean and variance vectors, are 128-dimensional.

2. *Encoder-decoder*: This network takes as input both the image feature encoding, and per-patch latent vectors sampled as-per the distributions produced by the prior-net. The encoder produces a 256-dimensional feature vector for each patch (i.e., at a spatial resolution of 57×81), which is then concatenated with the patch's corresponding sampled latent vector. This concatenated vector is then decoded to output 33×33 depth value estimates for each patch—i.e., the output of the decoder is $57\times81\times[33\times33]$. Note that the decoder path is independent for each patch to ensure independent sampling.

3. *Posterior-net*: This network is used only during training, and takes the image feature encoding and ground-truth patch depths as input. It uses two streams to first encode each of these to 256-dimensional per-patch feature vectors, concatenates them, and uses a series of 1×1 convolution layers to predict mean and variance vectors—the "posterior" equivalents of the prior-net's outputs.

The detailed architectures of these three networks are included in Table 6—with convolution and reshape operations allowing us to run the network efficiently in a fully-convolutional way, while still producing independent samples for overlapping patches. We train these three networks in a similar way as [18], using a weighted combination of two losses: (1) an L_1 loss between ground-truth patch depth and the output of the encoder-decoder network; and (2) a KL-divergence loss between the

Application	Un-cropping	Up-sampling	Sparse Meaus.	User Sel.	User Sel. w/ Annot.
Time	1.0 s	0.4 s	0.7 s	0.8 s	2.2 s

Table 7. Optimization running time for different applications (does not include sample generation time). Note that for user-guidance, the reported time is for each generated mode $\mathbf{Z}^m$.

No.	*Layer*	*Output Shape*
0	features from feature extractor	1×33×45×2560
1	resize	1×65×89×2560
2	conv 1×1	1×65×89×1024
3	conv 1×1	1×65×89×512
4	conv 3×3 dilation=2	1×61×85×512
5	conv 3×3 dilation=2	1×57×81×256
6	reshape	(57*81)×1×1×256
	dropout as noise	
7	conv 1×1	(57*81)×1×1×256
	dropout as noise	
8	conv 1×1	(57*81)×1×1×256
	dropout as noise	
9	conv 1×1	(57*81)×1×1×256
	dropout as noise	
10	conv_transpose 3×3	(57*81)×3×3×256
11	conv_transpose 3×3	(57*81)×5×5×128
12	conv_transpose 3×3	(57*81)×7×7×64
13	resize	(57*81)×13×13×64
14	conv_transpose 3×3	(57*81)×15×15×32
15	conv_transpose 3×3	(57*81)×17×17×16
16	resize	(57*81)×33×33×16
17	conv 1×1 + tanh	(57*81)×33×33×1
18	reshape	$57 \times 81 \times [33 \times 33]$

Generator

No.	*Layer*	*Output Shape*
0.a	features from feature extractor	1×33×45×2560
1.a	resize	1×65×89×2560
2.a	conv 3×3 dilation=2	1×61×85×1024
3.a	conv 3×3 dilation=2	1×57×81×256
4.a	reshape	(57*81)×1×1×256
0.b	true/fake depth patches	$57 \times 81 \times [33 \times 33]$
1.b	reshape	(57*81)×33×33×1
2.b	conv 3×3 stride=2	(57*81)×16×16×8
3.b	conv 2×2 stride=2	(57*81)×8×8×16
4.b	conv 2×2 stride=2	(57*81)×4×4×32
5.b	conv 2×2 stride=2	(57*81)×2×2×64
6.b	reshape	(57*81)×1×1×256
0	concat: 4.a and 6.b	(57*81)×1×1×512
1	conv 1×1	(57*81)×1×1×1024
2	conv 1×1	(57*81)×1×1×512
3	conv 1×1	(57*81)×1×1×256
4	conv 1×1 + sigmoid	(57*81)×1×1×1

Discriminator

Table 8. Conditional GAN architecture. We show architectures for the generator and discriminator for the GAN used in our ablation study, which follows a similar overall design as our VAE. For all dropout layers, we use probability 0.5. Every convolutional layer is followed by a ReLU, unless otherwise specified, and valid padding is used everywhere.

distributions produced by the prior-net and posterior-net; with a weight of $1e-4$ for the latter. After training, we discard the posterior-net. Given an image, we run the prior-net and the encoder-half of the encoder-decoder network, and then run the decoder-half multiple (100) times with different samples from the latent distributions to produce multiple samples of depth estimates for each patch.

C.3. Inference Hyperparameter Selection

For applications with a per-patch cost $C_i(\cdot)$—i.e., user-guidance and depth un-cropping—the value of λ is chosen based on a small validation set, as $\lambda = 10$ for user-guidance, and 150 for un-cropping. Moreover, for user guidance, we find that slowly increasing the value of λ from 5 to its final value of 10 during optimization leads to convergence to better solutions. For depth completion from sparse (both random and regularly spaced) measurements, we set the value of the parameters for gradient-based updates for the global cost—step-size γ (in range $[0.1, 1.0]$) and number of steps (in range $[1, 10]$)—based on a validation set as well.

C.4. Running Time

Our method works by first generating multiple (100) samples for each overlapping patch, and then carrying out inference using these samples for different applications. In particular, for "MAP" estimation to compute depth estimates with additional information, this involves running our iterative optimization method. We report these running time (on a 1080Ti GPU) for this optimization for different applications in Table 7—these times vary both because of variance in time taken per-iteration, and number of iterations needed for convergence.

C.5. Ablation: Conditional GAN architecture

Our conditional GAN architecture features generator and discriminator networks, with similar architecture design choices to the VAE—the generator has a similar architecture the encoder-decoder network in the VAE, and the discriminator to the

Setting	Method	lower is better			higher is better		
		rms	m-rms	rel	δ_1	δ_2	δ_3
	20 Ma [34]	-	0.351	0.078	92.8	98.4	99.6
	Ours	**0.363**	**0.303**	**0.070**	**94.0**	**98.7**	**99.7**
	50 Ma [34]	-	0.281	0.059	95.5	99.0	99.7
	Ours	**0.309**	**0.257**	**0.053**	**95.8**	**99.2**	**99.8**
	200 Ma [34]	-	0.230	0.044	97.1	99.4	99.8
	Ours	**0.237**	**0.196**	**0.037**	**97.6**	**99.6**	**99.9**

Table 9. Performance on depth estimation from arbitrary sparse measurements, using the same evaluation setting as [34] (half-resolution, evaluated on a center-crop).

posterior-net. The generator uses dropout as the noise-source to enable sampling in different runs of the generator—and ensure that per-patch estimates are independent by ensuring that different patches are based on different instantiations of dropout noise values. The architecture is detailed in Table 8.

C.6. Half-resolution Comparison to Ma *et al*. [34].

Note that [34] evaluate their methods by reporting errors on a centered crop of half-resolution depth-maps, and also derive their input sparse measurements at this half-resolution. In contrast, our results in Table 1 in the paper represent the official benchmark metrics (in the valid crop at full resolution) for consistency to other evaluations—in our paper and elsewhere. For a more direct comparison to [34], we also evaluated our method by replicating their setting. Specifically, to provide input sparse measurements, we first down-sample the ground-truth depth map and randomly sample depth values from this down-sampled map. We then provide these as inputs to our method (which resizes these back to the full resolution to compute the global cost $C^G(\cdot)$). Then, we take the full-resolution depth map estimates produced by our method, down-sample them to half-resolution, and compute error metrics on the same centered crop as [34]. We report these results in Table 9, and find they are similar to standard evaluation in Table 1 in the paper.