Single Image Depth Prediction Made Better: A Multivariate Gaussian Take —Supplementary Material—

Ce Liu[1] Suryansh Kumar[1*] Shuhang Gu[2] Radu Timofte[1,3] Luc Van Gool[1,4]
[1]CVL ETH Zürich [2]UESTC China [3]University of Würzburg [4]KU Leuven
{ce.liu, sukumar, vangool}@vision.ee.ethz.ch
shuhanggu@uestc.edu.cn, radu.timofte@uni-wuerzburg.de

Abstract

Our supplementary material accompanies the main paper and is organized as follows. Firstly, it contains detailed mathematical derivations of the proposed loss function and the covariance of the mixture of Gaussian. Secondly, details related to our neural network design and likelihood computation are presented to understand our implementation better. Next, more ablation studies are presented. Finally, visualization of our learned covariance and qualitative SIDP results on several benchmark datasets are presented for completeness.

1. Derivations

We present the detailed derivations for the negative log likelihood, and the covariance matrix for the mixture of Gaussian distributions.

1.1. Low-Rank Negative Log Likelihood

We start with the standard multivariate Gaussian distribution with mean $\boldsymbol{\mu}_\theta(I) \in \mathbb{R}^{N\times 1}$ and covariance $\boldsymbol{\Sigma}_\theta(I,I) \in \mathbb{R}^{N\times N}$:

$$\Phi(Z|\theta, I) = \mathcal{N}\big(\boldsymbol{\mu}_\theta(I), \boldsymbol{\Sigma}_\theta(I,I)\big). \tag{1}$$

The log probability density function is:

$$\log \Phi(Z|\theta, I) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det(\boldsymbol{\Sigma}_\theta(I,I)) - \frac{1}{2}(Z-\boldsymbol{\mu}_\theta)^T(\boldsymbol{\Sigma}_\theta(I,I))^{-1}(Z-\boldsymbol{\mu}_\theta). \tag{2}$$

We make the low-rank assumption:

$$\boldsymbol{\Sigma}_\theta(I,I) = \Psi_\theta(I)\Psi_\theta(I)^T + \sigma^2\mathbf{eye}(N) \tag{3}$$

where $\Psi_\theta(I) \in \mathbb{R}^{N\times M}$ and $M \lll N$, to ease the computing of the determinant and the inversion term.

*Corresponding Author (k.sur46@gmail.com)

Determinant: We follow the matrix determinant lemma [6] to simplify the computation of the determinant term in Eq.(2). The determinant of the matrix $\Psi_\theta(I)\Psi_\theta(I)^T + \sigma^2\mathbf{eye}(N) \in \mathbb{R}^{N\times N}$ is computed by

$$\begin{aligned}&\det(\Psi_\theta(I)\Psi_\theta(I)^T + \sigma^2\mathbf{eye}(N))\\ =&\det(\sigma^2\mathbf{eye}(N))\det(\mathbf{eye}(M) + \Psi_\theta^T(\sigma^{-2}\mathbf{eye}(N))\Psi_\theta)\\ =&\sigma^{2N}\det(\mathbf{eye}(M) + \sigma^{-2}\Psi_\theta^T(I)\Psi_\theta(I))\end{aligned} \tag{4}$$

The complexity of time and space for computing the determinant of the matrix $\mathbf{eye}(M) + \sigma^{-2}\Psi_\theta^T(I)\Psi_\theta(I) \in \mathbb{R}^{M\times M}$ is $O(M^3)$ [8].

Inversion: Then we use the matrix inversion lemma [6] to ease the computation of the inversion of the matrix $\Psi_\theta(I)\Psi_\theta(I)^T + \sigma^2\mathbf{eye}(N) \in \mathbb{R}^{N\times N}$:

$$\begin{aligned}&(\Psi_\theta(I)\Psi_\theta(I)^T + \sigma^2\mathbf{eye}(N))^{-1}\\ =&-(\sigma^2\mathbf{eye}(N))^{-1}\Psi_\theta(\mathbf{eye}(M) + \sigma^{-2}\Psi_\theta^T\Psi_\theta)^{-1}\\ &\Psi_\theta^T(\sigma^2\mathbf{eye}(N))^{-1} + (\sigma^2\mathbf{eye}(N))^{-1}\\ =&\sigma^{-2}\mathbf{eye}(N) - \sigma^{-4}\Psi_\theta(\mathbf{eye}(M) + \sigma^{-2}\Psi_\theta^T\Psi_\theta)^{-1}\Psi_\theta^T\end{aligned} \tag{5}$$

Again, computing the inversion of the term $\mathbf{eye}(M) + \sigma^{-2}\Psi_\theta^T\Psi_\theta \in \mathbb{R}^{M\times M}$ requires time and space complexity $O(M^3)$.

Total: We put the Eq.(4) and Eq.(5) into Eq.(2), then we can easily obtain:

$$\log \Phi(Z|\theta, I) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2}\log\det(\mathbf{A}) - \frac{\sigma^{-2}}{2}\mathbf{r}^T\mathbf{r} + \frac{\sigma^{-4}}{2}\mathbf{r}^T\Psi_\theta(I)\mathbf{A}^{-1}\Psi_\theta(I)^T\mathbf{r} \tag{6}$$

where, $\mathbf{r} = Z - \boldsymbol{\mu}_\theta(I)$, and $\mathbf{A} = \sigma^{-2}\Psi_\theta(I)^T\Psi_\theta(I) + \mathbf{eye}(M)$.

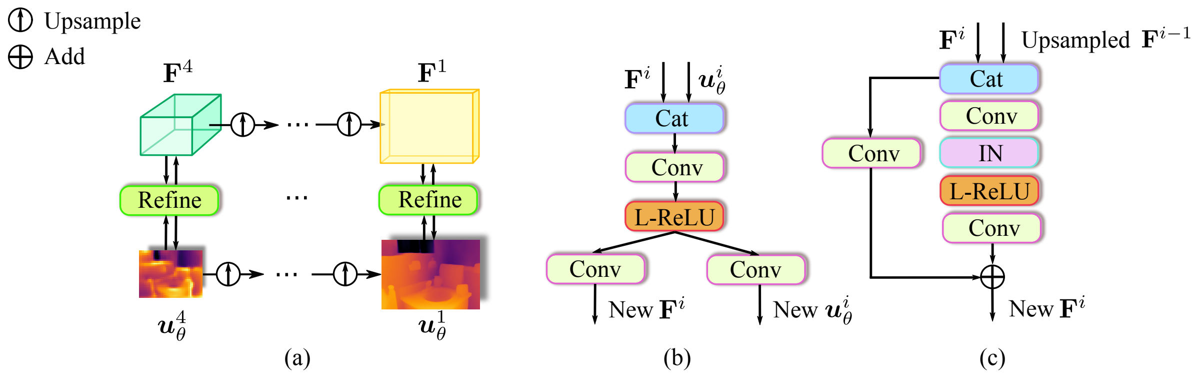


Figure 1. (a) The architecture of the U-decoder. (b) The detailed structure of the refine module. (c) The detailed structure of the fusion module. We use L-ReLU to represent the LeakyReLU operation, and IN to represent the instance normalization layer.

1.2. Covariance of Mixture of Gaussian

We present the covariance matrix of the mixture of Gaussian distributions:

$$\Phi(Z|I,\mathcal{D}) = \frac{1}{S}\sum_s \Phi(Z|\theta^s, I) \tag{7}$$

where $\Phi(Z|\theta^s, I)$ is the probability density function for a single Gaussian distribution. By the law of total variance [11], we obtain:

$$\begin{aligned}\mathrm{Var}(Z) =& \mathrm{E}[\mathrm{Var}(Z|s)] + \mathrm{Var}(\mathrm{E}[Z|s]) \\ =&\frac{1}{S}\sum_s(\Psi^{(s)}\Psi^{(s)T} + \sigma^2\mathbf{eye}(M)) \\ &+ \frac{1}{S}\sum_s(\boldsymbol{u}^{(s)} - \bar{\boldsymbol{\mu}})(\boldsymbol{u}^{(s)} - \bar{\boldsymbol{\mu}})^T \\ =&\sigma^2\mathbf{eye}(N) + \sum_s \frac{1}{\sqrt{S}}\Psi^{(s)}\frac{1}{\sqrt{S}}\Psi^{(s)T} \\ &+ \sum_s \frac{1}{\sqrt{S}}(\boldsymbol{u}^{(s)} - \bar{\boldsymbol{\mu}})\frac{1}{\sqrt{S}}(\boldsymbol{u}^{(s)} - \bar{\boldsymbol{\mu}})^T\end{aligned} \tag{8}$$

By constructing the following matrix:

$$\bar{\Psi} = \frac{1}{\sqrt{S}}\mathbf{concat}(\Psi^{(1)},\ldots,\Psi^{(S)},\boldsymbol{\mu}^{(1)}-\bar{\boldsymbol{\mu}},\ldots,\boldsymbol{\mu}^{(S)}-\bar{\boldsymbol{\mu}}), \tag{9}$$

the covariance matrix can be written as $\bar{\Psi}\bar{\Psi}^T + \sigma^2\mathbf{eye}(N)$, which shares the same form as Eq.(3).

2. Implementation Details

In this section, we present the details for the network architecture design and the likelihood computation.

2.1. Network Architecture

We introduce the details about the network architecture. The network is comprised of (a) encoder, (b) U-decoder, and (c) K-decoder. In general, we set the kernel size of the convolution layers to be 3 unless otherwise stated.

(a) Encoder. We adopt the standard Swin-Large [5] as our encoder. More specifically, the patch size is 4, the window size is 12, and the embedding dim is 192. The numbers of feature channels in four stages are 192, 384, 768, 1536, respectively. And there are 2, 2, 18, 2 blocks in the four stages, respectively. We collect the output feature map from the last block in each stage into $\mathbf{F} = \{\mathbf{F}^1, \mathbf{F}^2, \mathbf{F}^3, \mathbf{F}^4\}$, where $\mathbf{F}^1$ has 192 channels and stride 4, $\mathbf{F}^2$ has 384 channels and stride 8, $\mathbf{F}^3$ has 768 channels and stride 16, $\mathbf{F}^4$ has 1536 channels and stride 32.

(b) U-Decoder. The input to the U-decoder is $\mathbf{F} = \{\mathbf{F}^i\}_{i=1}^4$. From the input, the U-decoder will predict a set of depth maps $\{\boldsymbol{\mu}_\theta^i\}_{i=1}^4$. The network architecture of U-decoder is shown in Fig.1 (a). We start with $\mathbf{F}^4$, which has 1536 channels and stride 32. We first predict the $\boldsymbol{\mu}_\theta^4$ though a convolution layer, which has 1536 input channels and 128 output channels. We utilize a refine module to refine the $\mathbf{F}^4$ and $\boldsymbol{\mu}_\theta^4$. The refine module is shown in Fig. 1 (b). Then we upsample the $\mathbf{F}^4$ via bi-linear interpolation. The upsampled $\mathbf{F}^4$ will be concatenated with the $\mathbf{F}^3$ from the encoder. Then we adopt a fusion module to fuse the information from the $\mathbf{F}^3$ and the upsampled $\mathbf{F}^4$. The fusion module is shown in Fig. 1 (c). The fused $\mathbf{F}^3$ has 512 channels and stride 16. We upsample the $\boldsymbol{\mu}_\theta^4$ to $\boldsymbol{\mu}_\theta^3$ via bi-linear interpolation. Similar to the above procedures, the $\mathbf{F}^3$ will be refined with $\boldsymbol{\mu}_\theta^3$, and then upsampled and fused with $\mathbf{F}^2$ from the encoder. The fused $\mathbf{F}^2$ has 256 channels and stride 8. The $\boldsymbol{\mu}_\theta^3$ is also upsampled to $\boldsymbol{\mu}_\theta^2$ via bi-linear interpolation. With the same operations, we can further obtain the $\mathbf{F}^1$, which has 64 channels and stride 4. And we can also obtain $\boldsymbol{\mu}_\theta^1$. Now $\boldsymbol{\mu}_\theta^1$, $\boldsymbol{\mu}_\theta^2$, $\boldsymbol{\mu}_\theta^3$, $\boldsymbol{\mu}_\theta^4$ all have 128 channels. We upsample them to stride 1 via bi-linear operation, and compress the number of channels to 1 via a convolution layer.

(c) K-Decoder. The K-decoder aims to predict the Ψ_θ. The input to the K-decoder is $\mathbf{F} = \{\mathbf{F}^i\}_{i=1}^4$. The architecture

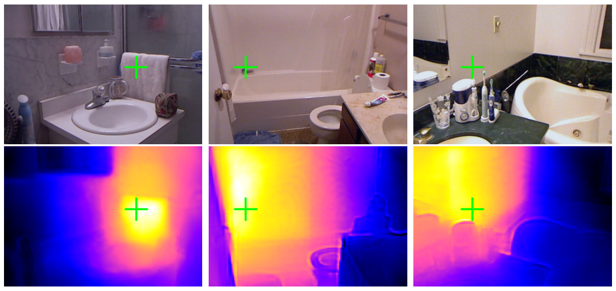


Figure 2. **Visualization of Covariance**. Top: test image. Bottom: covariance with respect to the pixel which is marked as a green cross. The yellow and light regions have higher covariance than the blue and dark ones.

of K-decoder is similar to U-decoder, except for there is no depth map predictions and refine modules. More specifically, we first upsample $\mathbf{F}^4$ via bi-linear interpolation, then fuse with the $\mathbf{F}^3$ though the fusion module. The fusion module is the same as the one in the U-decoder. The fused $\mathbf{F}^3$ has 512 channels and stride 16. Similar to the above procedures, we can further obtain the fused $\mathbf{F}^2$ and the fused $\mathbf{F}^1$. The fused $\mathbf{F}^2$ has 256 channels, and the fused $\mathbf{F}^1$ has 128 channels. We predict Ψ_θ from $\mathbf{F}^1$ by a convolution layer that has 128 input channels and 128 output channels.

2.2. Likelihood Computation

We provide the pseudo code to compute the log likelihood in Algorithm 1.

Algorithm 1 Log Likelihood Computation

```
    Input: $\boldsymbol{\mu}_\theta(I) \in \mathbb{R}^{N\times 1}$, $\Psi_\theta(I) \in \mathbb{R}^{N\times M}$, $\sigma \in \mathbb{R}^+$, and $Z^{gt} \in \mathbb{R}^{N\times 1}$
    Output: $\log \Phi(Z^{gt}|\theta, I)$
1:  $\mathbf{r} = Z^{gt} - \boldsymbol{\mu}_\theta(I)$
2:  $\mathbf{p} = \Psi_\theta(I)^T \mathbf{r}$
3:  $\mathbf{A} = \sigma^{-2}\Psi_\theta(I)^T\Psi_\theta(I) + \mathbf{eye}(M)$
4:  $\mathbf{L}\mathbf{L}^T = \text{cholesky}(\mathbf{A})$
5:  $\mathbf{q} = \mathbf{L}\backslash\mathbf{p}$                ▷ Or: $\mathbf{q} = \text{inv}(\mathbf{L}) * \mathbf{p}$
6:  Return $-\frac{N}{2}\log 2\pi\sigma^2 - \sum_i \log \mathbf{L}_{ii} - \frac{\sigma^{-2}}{2}\mathbf{r}^T\mathbf{r} + \frac{\sigma^{-4}}{2}\mathbf{q}^T\mathbf{q}$
```

3. More Ablations

In this section, we provide more ablation studies.

3.1. Comparison with Deep Evidential Regression

We compare with the Deep Evidential Regression [1] on NYU Depth V2 test set [9] and KITTI Eigen split [3]. We present the experimental results in Tab. 1. Our approach achieves better depth prediction accuracy and uncertainty estimation.

Dataset	Loss	SILog ↓	NLL ↓	RMS ↓	δ_1 ↑
NYU	DER	9.253	0.118	0.330	0.927
	Ours	**8.323**	**-1.342**	**0.311**	**0.933**
KITTI	DER	7.500	1.072	0.225	0.971
	Ours	**6.757**	**-0.222**	**0.202**	**0.976**

Table 1. Comparison with Deep Evidential Regression (DER).

3.2. FPS with K-Decoder

In general K-Decoder is used only at train time. The K-Decoder can be abandoned at test time for SIDP if uncertainty information is not required. For completeness, we present the FPS information at test time in Tab. 2.

K-Decoder	SI Log ↓	FPS
w/o	**8.323**	**9.909**
w/	**8.323**	8.445

Table 2. SI Log error and corresponding FPS on NYU Dataset.

4. Visualization of Learned Covariance

To understand the covariance learned by the proposed negative log likelihood loss function, we visualize the covariance for selected pixels. More specifically, for each image we select a pixel (marked as a green cross), and visualize the covariance between the pixel and all other pixels. The results are shown in Fig. 2. We observe that the pixels from nearby regions or the same objects usually have higher covariance.

5. Qualitative Results

We provide more qualitative results on NYU Depth V2 [9], KITTI Eigen split [3,4] and SUN RGB-D [10] in Fig. 3, Fig. 4, Fig. 5, respectively. The depth prediction from our method contains more details about the scenes, especially in NYU Depth V2 and SUN RGB-D.

References

[1] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in Neural Information Processing Systems*, 33:14927–14937, 2020.

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

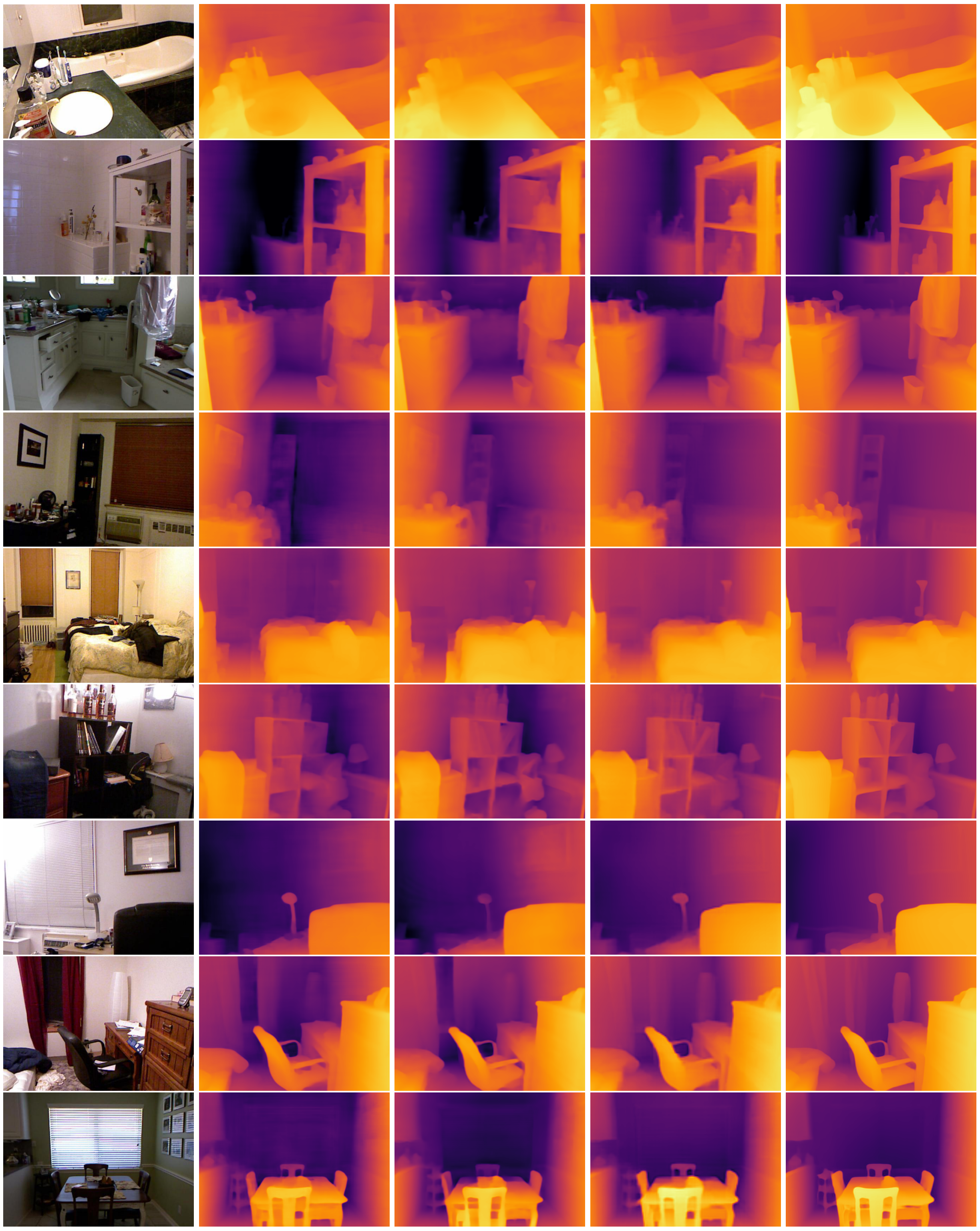
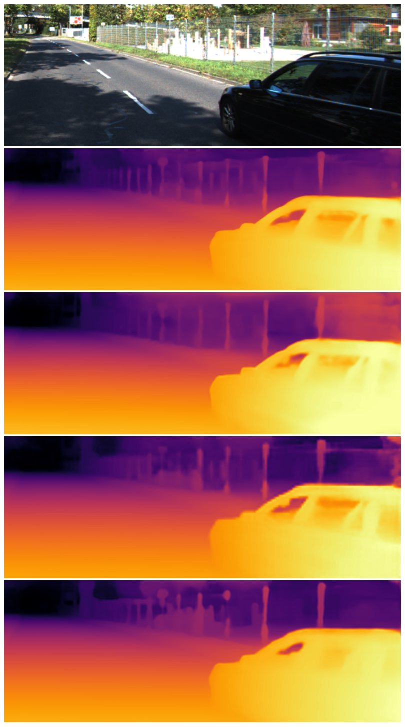
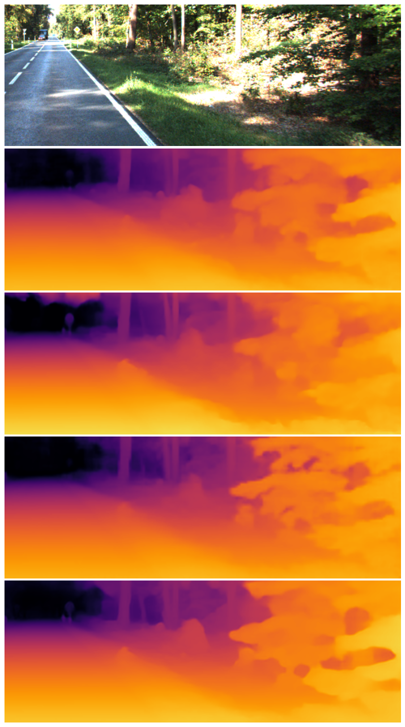
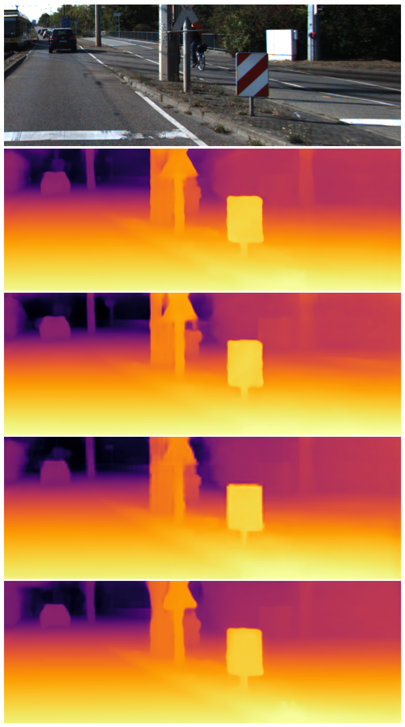
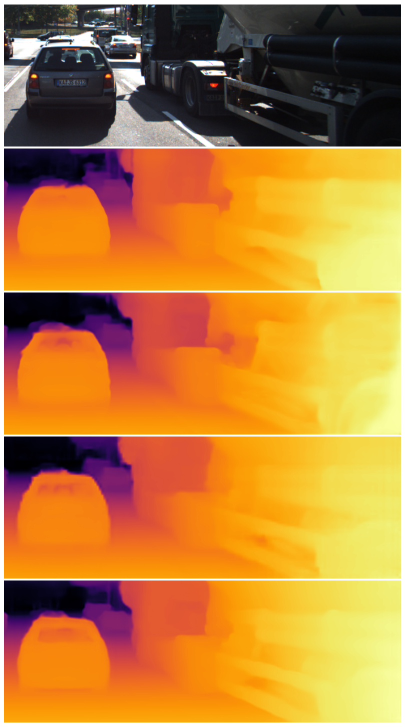


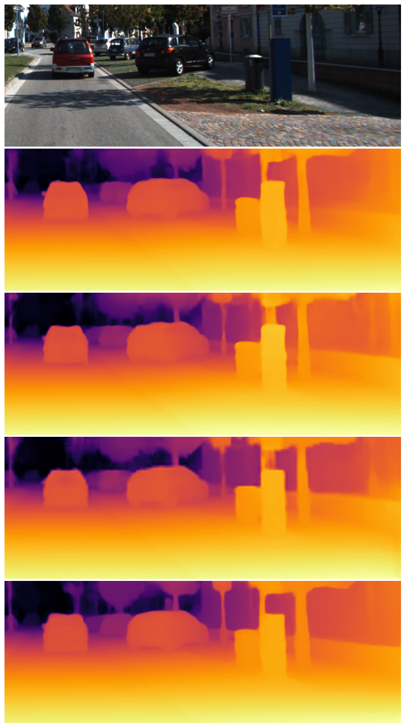
Figure 3. **Qualitative Comparison on NYU Depth V2 test set** [9]. Our method recovers better depth even for complex scenes than the prior art such as (b) DPT [7], (c) AdaBins [2], (d) NeWCRFs [12].


(a) Image 1


(b) Image 2


(c) Image 3


(d) Image 4


(e) Image 5

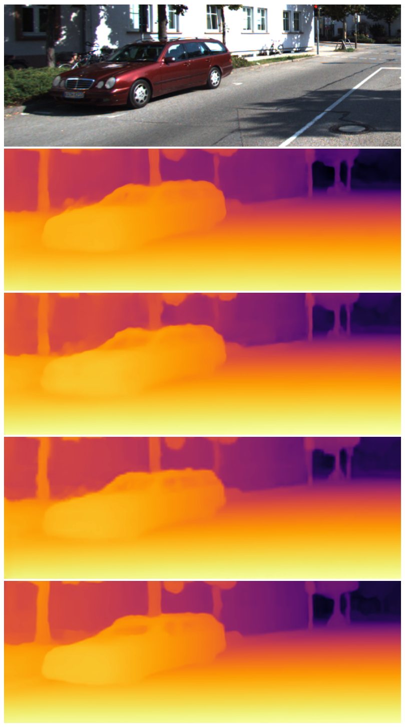

(f) Image 6

Figure 4. **Qualitative comparison on KITTI Eigen split** [3]. For each column, from top to bottom we present the input image, the prediction from DPT [7], AdaBins [2], NeWCRFs [12], and our framework respectively.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

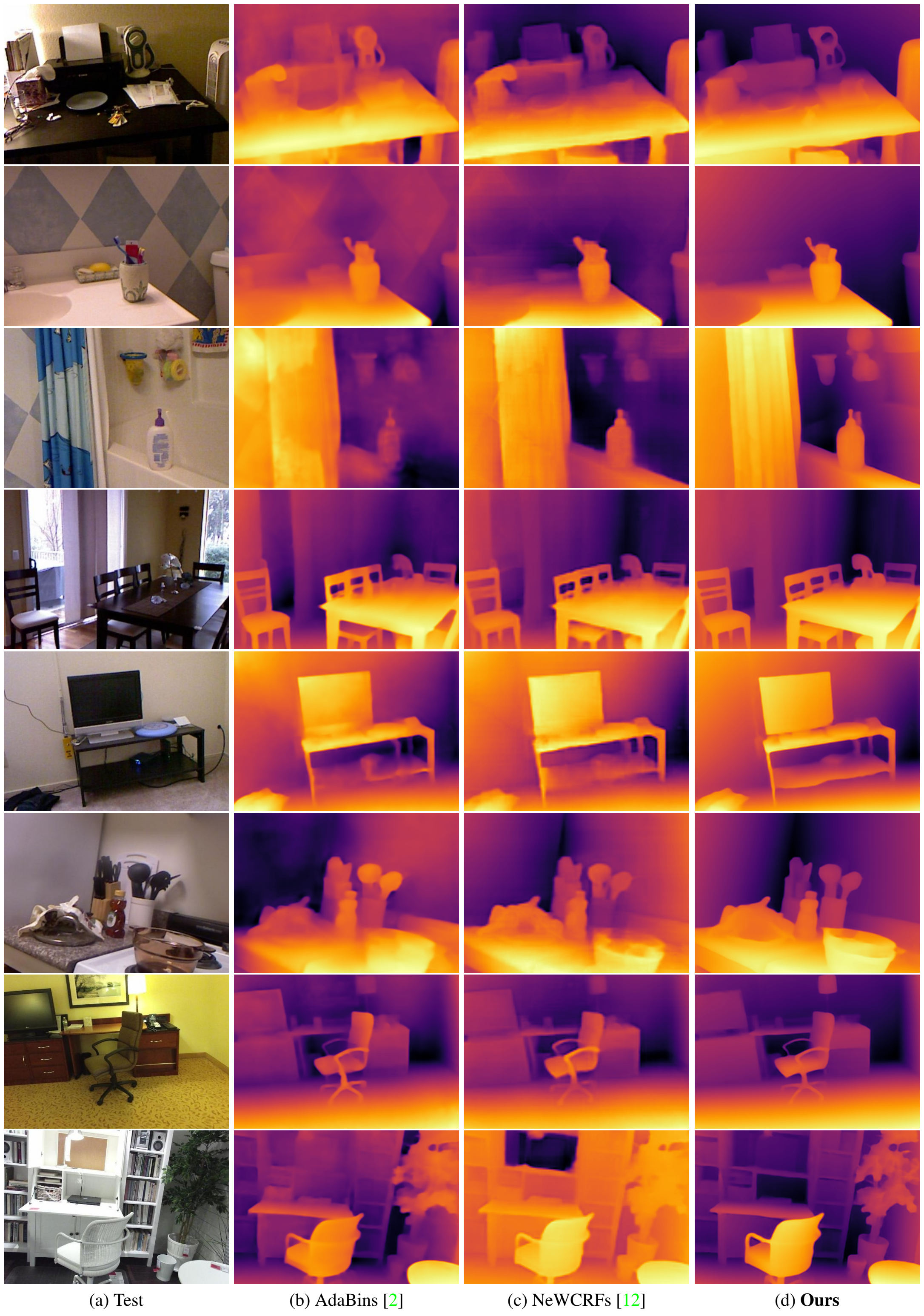

(a) Test (b) AdaBins [2] (c) NeWCRFs [12] (d) **Ours**

Figure 5. **Qualitative Comparison on SUN RGB-D** [10]. All the methods are trained on NYU Depth V2 [9] without fine-tuning on SUN RGB-D. Our method generalizes better on unseen scenes than (b) AdaBins [2] and (c) NeWCRFs [12].

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[6] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, USA, second edition, 1992.

[7] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[8] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[9] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[10] Shuran Song, Samuel P Lichtenberg, and Jianxiong 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.

[11] N.A. Weiss, P.T. Holmes, and M. Hardy. *A Course in Probability*. Pearson Addison Wesley, 2006.

[12] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3916–3925, June 2022.