3D Photography using Context-aware Layered Depth Inpainting Supplementary Material

Table 1. **Quantitative comparison** on the RealEstate10K dataset.

Methods	SSIM ↑	PSNR ↑	LPIPS ↓
Stereo-Mag [12]	**0.8906**	26.71	0.0826
PB-MPI [10] (32 Layers)	0.8717	25.38	0.0925
PB-MPI [10] (64 Layers)	0.8773	25.51	0.0902
PB-MPI [10] (128 Layers)	0.8700	24.95	0.1030
LLFF [5]	0.8062	23.17	0.1323
Xview [1]	0.8628	24.75	0.0822
Ours	0.8887	**27.29**	**0.0724**

1. Additional Quantitative Results

We further evaluate the PB-MPI method [10] with various number of depth layers. We report the results in Table 1.

2. Visual Results

Comparisons with the state-of-the-arts. We provide a collection of rendered 3D photos with comparisons with the state-of-the-art novel view synthesis algorithms. In addition, we show that our method can synthesize novel view for legacy photos. Please refer to the website[1] for viewing the results.

Ablation studies. To showcase how each of our proposed component contribute to the quality of the synthesized view, we include a set of rendered 3D photos using the same ablation settings in Section 4.4 of the main paper. Please refer to the website[1] for viewing the photos.

3. Implementation Details

In this section, we provide additional implementation details of our model, including model architectures, training objectives, and training dataset collection. We will release the source code to facilitate future research in this area.

Model architectures. We adopt the same U-Net [8] architecture as in [4] for our depth inpainting and color inpainting models (see Table 2), and change the input channels for each model accordingly. For the edge inpainting model, we use a design similar to [7] (see Table 3). We set the input depth and RGB values in the synthesis region to zeros for all three models. The input edge values in the synthesis region are similarly set to zeros for depth and color inpainting models, but remain intact for the edge inpainting network. We show the input details of each model in Table 4

Training objective. To train our color inpainting model, we adopt similar objective functions as in [4]. First, we define the reconstruction loss for context and synthesis regions:

$$L_{\text{synthesis}} = \frac{1}{N}\left\|S \odot (I - I_{gt})\right\|, \qquad L_{\text{context}} = \frac{1}{N}\left\|C \odot (I - I_{gt})\right\|, \tag{1}$$

[1] https://shihmengli.github.io/3D-Photo-Inpainting/

Table 2. **Model architecture of our color and depth inpainting models.** W denote partial convolution layer as PConv, and denote BatchNorm as BN. We add the context and synthesis region together as the partial masks for the PConv layers.

Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
PConv1	7×7	64	1	2	-	ReLU
PConv2	5×5	128	1	2	BN	ReLU
PConv3	5×5	256	1	2	BN	ReLU
PConv4	3×3	512	1	2	BN	ReLU
PConv5	3×3	512	1	2	BN	ReLU
PConv6	3×3	512	1	2	BN	ReLU
PConv7	3×3	512	1	2	BN	ReLU
PConv8	3×3	512	1	2	BN	ReLU
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv7)	-	512+512	-	-	-	-
PConv9	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	- 512	-	2	-	-	
Concatenate (w/ PConv6)	-	512+512	-	-	-	-
PConv10	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv5)	-	512+512	1	-	-	-
PConv11	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv4)	-	512+512	-	-	-	-
PConv12	3×3	512	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	512	-	2	-	-
Concatenate (w/ PConv3)	-	512+256	-	-	-	-
PConv13	3×3	256	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	256	-	2	-	-
Concatenate (w/ PConv2)	-	256+128	-	-	-	-
PConv14	3×3	128	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	128	-	2	-	-
Concatenate (w/ PConv1)	-	128+64	-	-	-	-
PConv15	3×3	64	1	1	BN	LeakyReLU(0.2)
NearestUpsample	-	64	-	2	-	-
Concatenate (w/ Input)	-	64 + 4 or 64 + 6 (Depth / Color Inpainting)	-	-	-	-
PConv16	3×3	1 or 3 (Depth / Color Inpainting)	1	1	-	-

where S and C are the binary mask indicating *synthesis* and *context* regions, respectively, $\odot$ denotes the Hadamard product, N is the total number of pixels, I is the inpainted result, and I_{gt} is the ground truth image.

Next, we define the perceptual loss [2]:

$$L_{perceptual} = \sum_{p}^{P-1} \frac{\left\| \psi_p(I) - \psi_p(I_{gt}) \right\|}{N_{\psi_p}}, \tag{2}$$

Here, $\psi_p(\cdot)$ is the output of the pth layer from VGG-16 [9], and N_{ψ_p} is the total number of elements in $\psi_p(\cdot)$.

We define the style loss as:

$$L_{style} = \sum_{p}^{P-1} \frac{1}{C_p C_p} \left\| \frac{1}{C_p H_p W_p} \left[(\psi_p^I)^\top \psi_p^I - (\psi_p^{I_{gt}})^\top \psi_p^{I_{gt}} \right] \right\|, \tag{3}$$

where C_p, H_p, W_p is the number of channels, height, and width of the output $\psi_p(\cdot)$.

Table 3. **Model architecture of our edge inpainting models.** As in [7], the edge inpainting model consists of 1 edge generator, and 1 discriminator network. SN→IN indicates that we first perform spectral normalization (SN) [6], and then apply instance normalization (IN) [11]. ResnetBlock comprises 2 conv layers with the specified hyper-parameters and a skip connection between the input and the output of the block.

Edge Generator						
Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
Conv1	7×7	64	1	1	SN→IN	ReLU
Conv2	4×4	128	1	2	SN→IN	ReLU
Conv3	4×4	256	1	2	SN→IN	ReLU
ResnetBlock4	3×3	256	2	1	SN→IN	ReLU
ResnetBlock5	3×3	256	2	1	SN→IN	ReLU
ResnetBlock6	3×3	256	2	1	SN→IN	ReLU
ResnetBlock7	3×3	256	2	1	SN→IN	ReLU
ResnetBlock8	3×3	256	2	1	SN→IN	ReLU
ResnetBlock9	3×3	256	2	1	SN→IN	ReLU
ResnetBlock10	3×3	256	2	1	SN→IN	ReLU
ResnetBlock11	3×3	256	2	1	SN→IN	ReLU
ConvTranspose12	4×4	128	1	2	SN→IN	ReLU
ConvTranspose13	4×4	64	1	2	SN→IN	ReLU
Conv14	7×7	1	1	1	SN→IN	Sigmoid
Discriminator						
Module	Filter Size	#Channels	Dilation	Stride	Norm	Nonlinearity
Conv1	4×4	64	1	2	SN	LeakyReLU(0.2)
Conv2	4×4	128	1	2	SN	LeakyReLU(0.2)
Conv3	4×4	256	1	2	SN	LeakyReLU(0.2)
Conv4	4×4	512	1	1	SN	LeakyReLU(0.2)
Conv5	4×4	1	1	1	SN	Sigmoid

Table 4. **Input of each model in our proposed method.** The check mark ✓ indicates that it is used as input for the model.

	RGB	Depth	Edge	Context& Synthesis
Color Inpainting	✓	-	✓	✓
Depth Inpainting	-	✓	✓	✓
Edge Inpainting	✓	✓	✓	✓

Finally, we adopt the Total Variation (TV) loss:

$$L_{tv} = \sum_{(i,j)\in S,(i,j+1)\in S} \frac{\|I(i,j+1)-I(i,j)\|}{N} + \sum_{(i,j)\in S,(i+1,j)\in S} \frac{\|I(i+1,j)-I(i,j)\|}{N}. \tag{4}$$

Here, We overload the notation S to denote the synthesis region. This term can be interpreted as a smoothing penalty on the synthesis area. Combine all these loss terms, we obtain the training objective for our color inpainting model:

$$L = L_{context} + 6L_{synthesis} + 0.05L_{perceptual} + 120L_{style} + 0.01L_{tv}$$

For our depth inpainting model, we use only $L_{context} + L_{synthesis}$ as the objective functions. For edge inpainting model, we follow the identical training protocol as in [7].

Training details. We illustrate the data generation process in Figure 1. We use the depth map predicted by MegaDepth [3] as our pseudo ground truth. We train our method using 1 Nvidia V100 GPU with batch size of 8, and the total training time take about 5 days.

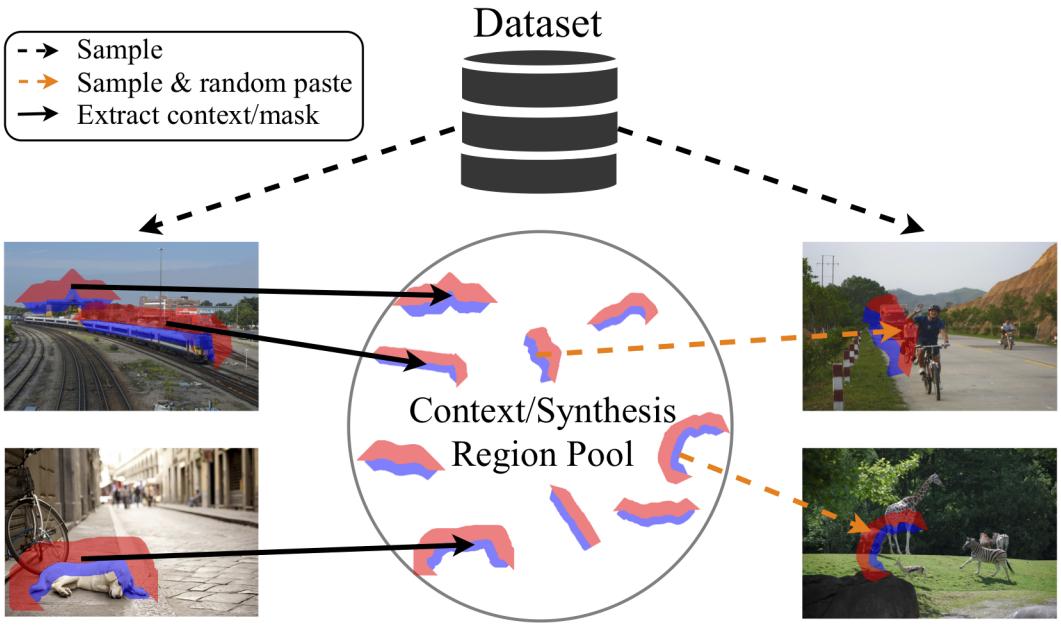

Figure 1. **Dataset generation process.** We first form a collection of context/synthesis regions by extracting them from the linked depth edges in images on the COCO dataset. We then randomly sample and paste these regions onto *different* images, forming our training dataset for context-aware color and depth inpainting.

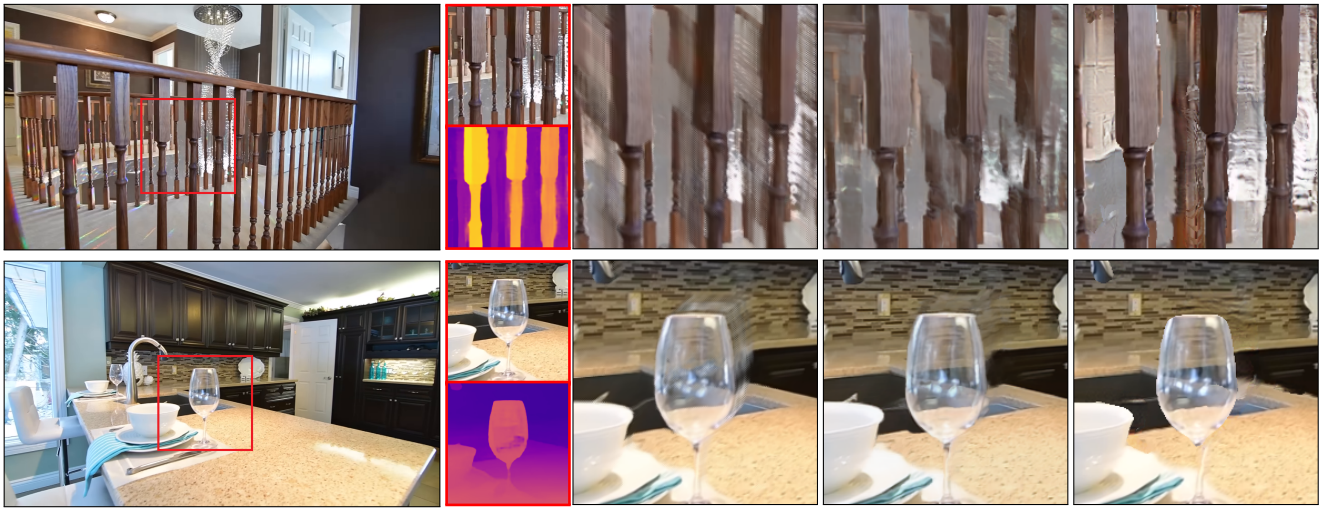

Figure 2. **Failure cases.** Single-image depth estimation algorithms (e.g., MegaDepth) often have difficulty in handling thin and complex structures and may produce overly smooth depth maps.

4. Failure cases

As estimating depth/disparity map from a single image remain a challenging problem (particularly for scenes with complex, thin structures), our method fails to produce satisfactory results with plausible motion parallax for scenes with complex structures. Due to the use of explicit depth map, our method is unable to handle reflective/transparent surfaces well. We show in Figure 2 two examples of such cases. Here, we show the input RGB image as well as the estimated depth map from the pre-trained MegaDepth model. The rendered 3D photos can be found in the supplementary webpage.

References

[1] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. In *ICCV*, 2019. 1

[2] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2

[3] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 3

[4] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018. 1

[5] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4), July 2019. 1

[6] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. 2018. 3

[7] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint*, 2019. 1, 3

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[10] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 1

[11] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[12] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 2018. 1