Pushing the Boundaries of View Extrapolation with Multiplane Images: Supplementary Materials

Pratul P. Srinivasan[1] Richard Tucker[2] Jonathan T. Barron[2]
Ravi Ramamoorthi[3] Ren Ng[1] Noah Snavely[2]
[1]UC Berkeley, [2]Google Research, [3]UC San Diego

1. Supplementary Video

We have included a supplementary video that compares renderings from MPIs predicted by our model to those predicted by the original MPI method [6] and the "Disocclusion Inpainting" baseline that inpaints disoccluded pixels in each rendering using a state-of-the-art deep learning approach [5]. We invite readers to view this video for qualitative evidence of our method's ability to produce improved renderings with fewer depth discretization and repeated texture artifacts compared to the original MPI method, and with more convincing temporally-consistent disocclusions compared to the "Disocclusion Inpainting" baseline.

2. Section 3.2 Derivation Details

In this section, we provide additional details for the derivations in Section 3.2 of our main manuscript.

Figure 1 illustrates a 2D slice of the camera setup geometry where we hold the y dimension constant and view the xz-plane. An MPI in the frame of the reference camera (green dot) is viewed by a novel view camera (blue dot) at a translation (u, s) relative to the reference camera. x is the pixel coordinate of the red diffuse scene point on the blue camera's sensor plane, and x' is the pixel coordinate of the red scene point on the visualized MPI plane at disparity d. Note that the MPI plane pixel coordinate x' scales linearly with $1/d$, because each MPI plane RGBα image contains the same number of pixels sampled evenly within the camera frustum. It is straightforward to use the similar triangles of this diagram (above and to the right of the blue dot) to derive Equation 1 of our main manuscript:

$$r_{u,s}(x) = \sum_{d \in \mathcal{D}} c(x', d) = \sum_{d \in \mathcal{D}} c\left(\left(1 - sd\right) x + ud, d\right) \quad (1)$$

where $c(x, d)$ is the pre-multiplied RGBα at each pixel coordinate x and disparity plane d within the set of MPI disparity planes $\mathcal{D}$. Note that u and s are in units of pixels (such that the camera focal length $f = 1$), and we limit s to

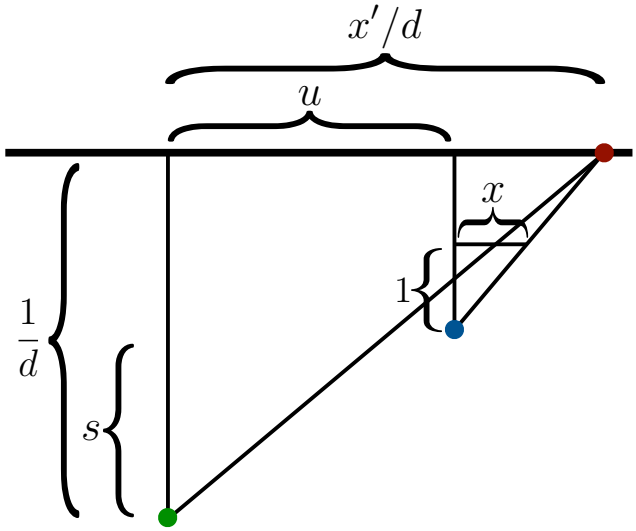

Figure 1. **Camera setup geometry.** An MPI in the frame of a reference camera (green dot) is viewed by a novel camera position (blue dot) at a translation (u, s) relative to the reference camera.

the range $-\infty < s < 1/d_{max}$ because renderings are not defined for viewpoints within the MPI volume. Additionally, note that the disparity d is in units $1/pixel$.

We wish to study the limits of views rendered from an MPI, so let us consider a worst-case MPI with content in the subset of closest planes, for which we make a locally linear approximation to the coordinate transformation $(x, d) \rightarrow (x', d)$:

$$r_{u,s}(x) = \sum_{d \in \mathcal{D}} c\left(\left(1 - sd_{max}\right) x + ud, d\right) \quad (2)$$

where d_{max} is a constant. Now, we have expressed the rendering of mutually-visible content as a sheared and dilated integral projection of the MPI. We apply the generalized Fourier slice theorem [4] to interpret this integral projection of an MPI as a 2D slice through the 3D MPI's Fourier transform. Using operator notation, the generalized Fourier slice

theorem can be expressed as:

$$\mathcal{F}^M \circ \mathcal{I}_M^N \circ \mathcal{B} \equiv \mathcal{S}_M^N \circ \frac{\mathcal{B}^{-\mathrm{T}}}{|\mathcal{B}^{-\mathrm{T}}|} \circ \mathcal{F}^N \tag{3}$$

where $\mathcal{F}^M$ is the M-dimensional Fourier transform operator, $\mathcal{I}_M^N$ is the integral projection operator of an N-dimensional function to M dimensions by integrating out the last $N - M$ dimensions, $\mathcal{B}$ is a basis transformation operator (where $|\mathcal{B}^{-\mathrm{T}}|$ is the determinant of the inverse transpose of the transformation matrix), and $\mathcal{S}_M^N$ is the slicing operator that takes an M-dimensional slice from an N-dimensional function by setting the last $N - M$ dimensions to zero. The relevant values of the resulting transformation of MPI's Fourier transform, given the sheared and dilated MPI in Equation 2, are:

$$\begin{aligned} \mathcal{B} &= \begin{bmatrix} (1 - sd_{max}) & u \\ 0 & 1 \end{bmatrix} \\ \mathcal{B}^{-\mathrm{T}} &= \frac{1}{1 - sd_{max}} \begin{bmatrix} 1 & 0 \\ -u & (1 - sd_{max}) \end{bmatrix} \end{aligned} \tag{4}$$

We use these values to express the Fourier transformation of our sheared and dilated MPI as:

$$C(k_{x'}, k_d) = C\left(\frac{k_x}{1 - sd_{max}}, \frac{-uk_x}{1 - sd_{max}} + k_d\right) \tag{5}$$

where $C(k_x, k_d)$ is the Fourier transform of $c(\mathbf{x}, d)$. We omit the $1/|\mathcal{B}^{-\mathrm{T}}|$ term since it is simply a scaling factor and can be absorbed into the definition of C. Finally, we compute the resulting rendered view as the inverse Fourier transform of the slice taken from the MPI's Fourier transformation by setting $k_d = 0$:

$$r_{u,s}(x) = \mathcal{F}^{-1}\left\{C\left(\frac{k_x}{1 - sd_{max}}, \frac{-uk_x}{1 - sd_{max}}\right)\right\} \tag{6}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform. This connects our detailed supplementary derivation back with Equation 3 in the main manuscript.

3. Network Architecture and Training Details

Table 1 contains precise specifications of the 3D convolutional neural network architecture described in Section 3.3 of the main manuscript.

We implement our system in TensorFlow [1]. We train using the Adam algorithm [2] for 300,000 iterations, with a learning rate of 2×10^{-4}, default parameters $\beta_1 = 0.9, \beta_2 = 0.999$, and a batch size of 1.

For our randomized-resolution training, we uniformly sample input PSV tensors with sizes [height, width, #planes, #channels] of any of the following:

Figure 2. **Computed disocclusion masks.** We visualize example disocclusion masks for target view pixels that are occluded in the reference view, as used for our quantitative evaluations.

[576,	1024,	2^4,	6]
[576/2,	1024/2,	2^5,	6]
[576/4,	1024/4,	2^5,	6]
[576/4,	1024/4,	2^6,	6]
[576/4,	1024/4,	2^7,	6]
[576/8,	1024/8,	2^5,	6]
[576/8,	1024/8,	2^6,	6]
[576/8,	1024/8,	2^7,	6]

At test time, we apply this network on input PSV tensors with size [576,1024,2^7,6], which has the maximum spatial and depth resolutions seen during training.

4. Disocclusion Mask Examples

Figure 2 visualizes disocclusion masks computed by our method (Equation 12 in the main manuscript) between pairs of reference and target viewpoints.

5. "r_{init} + Adversarial Disocclusions" Details

As described in our main manuscript, "r_{init} + Adversarial Disocclusions" is a two-step MPI prediction strategy we use as a baseline for comparisons. It uses an identical Φ_1 to predict the initial MPI, but Φ_2 directly predicts RGBα layers instead of flows, and we apply an adversarial loss to each final rendered target image to encourage realistic disocclusions. We adopt the SN-PatchGAN discriminator architecture with spectral normalization [3] and a hinge loss objective function, as proposed in [5]. We add the adversarial loss to our main objective (Equation 10 in the main manuscript) with a weight $\lambda = 5.0$.

Downsampling		
1-3	$(3 \times 3 \times 3$ conv, 8 features) $\times 3$	$H \times W \times D \times 8$
4	$3 \times 3 \times 3$ conv, 16 features, stride 2	$H/2 \times W/2 \times D/2 \times 16$
5-6	$(3 \times 3 \times 3$ conv, 16 features) $\times 2$	$H/2 \times W/2 \times D/2 \times 16$
7	$3 \times 3 \times 3$ conv, 32 features, stride 2	$H/4 \times W/4 \times D/4 \times 32$
8-9	$(3 \times 3 \times 3$ conv, 32 features) $\times 2$	$H/4 \times W/4 \times D/4 \times 32$
10	$3 \times 3 \times 3$ conv, 64 features, stride 2	$H/8 \times W/8 \times D/8 \times 64$
11-12	$(3 \times 3 \times 3$ conv, 64 features) $\times 2$	$H/8 \times W/8 \times D/8 \times 64$
13	$3 \times 3 \times 3$ conv, 128 features, stride 2	$H/16 \times W/16 \times D/16 \times 128$
14-15	$(3 \times 3 \times 3$ conv, 128 features) $\times 2$	$H/16 \times W/16 \times D/16 \times 128$
Bottleneck		
16	$3 \times 3 \times 3$ conv, 128 features, dilation rate 2	$H/16 \times W/16 \times D/16 \times 128$
17	$3 \times 3 \times 3$ conv, 128 features, dilation rate 4	$H/16 \times W/16 \times D/16 \times 128$
18	$3 \times 3 \times 3$ conv, 128 features, dilation rate 8	$H/16 \times W/16 \times D/16 \times 128$
19	$3 \times 3 \times 3$ conv, 128 features	$H/16 \times W/16 \times D/16 \times 128$
Upsampling		
20	$2\times$ nearest neighbor upsample	$H/8 \times W/8 \times D/8 \times 128$
21	concatenate 20 and 12	$H/8 \times W/8 \times D/8 \times (128 + 64)$
22-23	$(3 \times 3 \times 3$ conv, 64 features) $\times 2$	$H/8 \times W/8 \times D/8 \times 64$
24	$2\times$ nearest neighbor upsample	$H/4 \times W/4 \times D/4 \times 64$
25	concatenate 24 and 9	$H/4 \times W/4 \times D/4 \times (64 + 32)$
26-27	$(3 \times 3 \times 3$ conv, 32 features) $\times 2$	$H/4 \times W/4 \times D/4 \times 32$
28	$2\times$ nearest neighbor upsample	$H/2 \times W/2 \times D/2 \times 32$
29	concatenate 28 and 6	$H/2 \times W/2 \times D/2 \times (32 + 16)$
30-31	$(3 \times 3 \times 3$ conv, 16 features) $\times 2$	$H/2 \times W/2 \times D/2 \times 16$
32	$2\times$ nearest neighbor upsample	$H \times W \times D \times 16$
33	concatenate 32 and 3	$H \times W \times D \times (16 + 8)$
34-35	$(3 \times 3 \times 3$ conv, 8 features) $\times 2$	$H \times W \times D \times 8$
36	$3 \times 3 \times 3$ conv, 4 features (tanh)	$H \times W \times D \times 4$

Table 1. **3D CNN network architecture.** Our initial MPI prediction CNN Φ_1 uses the architecture described in the above table. Our final MPI prediction CNN Φ_2 uses the same architecture without the bottleneck dilated convolutional layers. All convolutional layers in Φ_1 and Φ_2 use a ReLu activation, except for the final layer. Φ_1 applies a tanh to all channels of the final layer, while Φ_2 just applies a tanh on one output channel (corresponding to α) and does not apply an activation to the predicted flows.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. 2

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 2

[3] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018. 2

[4] R. Ng. Fourier slice photography. In *ACM Transactions on Graphics (SIGGRAPH)*, 2005. 1

[5] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. *CVPR*, 2018. 1, 2

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