Crowdsampling the Plenoptic Function
Supplementary Material

Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely

Cornell Tech, Cornell University

This supplementary document includes the following:

1. Details of priors on the plenoptic function.
2. Scene statistics.
3. Losses used for optimization and learning.
4. Network architectures and implementation details.
5. Visual illustration of components in our framework.
6. Summary of user study.
7. Additional results and comparisons.

1 Priors on the Plenoptic Function

Our scene representation and approach to training are motivated by simple priors on structure in the plenoptic function. Given that our crowdsampling of a scene is unpredictable and unregistered in time, we focus primarily on periodic changes in reflectance and illumination—most notably, this includes how the appearance of a scene changes from day to night. For most scenes, this change is dominated by the motion of the sun. However, we also see scenes where, for example, visible lights turn off and on throughout the day (e.g., a cityscape or an attraction that lights up at night). Below we describe two priors that motivate the design of our representation and training.

1.1 Constant Visibility and Light Field Gradients

Even in non-Lambertian scenes with changing reflectance and illumination, we may expect the structure of visibility to remain relatively constant over time. If we consider slices of the plenoptic function at different times, we can think of this expectation in terms of gradients in the respective light fields. To see this, consider that every scene point corresponds to some 4D hyperplane in the light field. If the light transport function around our point is smooth, then we can expect that this hyperplane will be locally constant. Gradients then primarily occur at boundaries between hyperplanes, which include occlusion boundaries and edges in reflectance (e.g., surface texture) or illumination (e.g., shadows). The structure of visibility in a scene determines the adjacency of these hyperplanes, thereby limiting the set of gradients that can be introduced by changing reflectance or illumination. Our approach leverages this prior that different slices of the plenoptic function share visibility structure by fixing alpha values of our DeepMPI. Recall that each voxel of an MPI can be interpreted as a floating semi-transparent surface point.

This corresponds to a constant hyperplane in our reconstructed light field just as an analogous real surface point would. Fixing the alphas determines the visibility of such points, and therefore the adjacency of their corresponding hyperplanes in the reconstructed light field.

1.2 Common Light Sources, Material Properties, and Normals

We can think of the light transport function around every point in our scene as mapping incoming light to outgoing light. The appearance of a point in a particular image is then a sample from this transport function. Without explicitly modeling the transport function, we can reason about correlations among the samples provided by different scene points and across different viewing conditions. For example: it is reasonable to expect that many visible points in a given scene will share the same material properties, and that the relationship between surface normals at different points will remain constant (this is true so long as surface geometry does not change). Furthermore, we can expect correlation due to different points being lit by the same source—often, the sun. We learn how to leverage these many sources of correlation by training our DeepMPI with feature vectors attached to each voxel. Intuitively, this creates a latent space where surface points with highly correlated appearance end up with similar feature vectors, preserving important correlations in our generated MPIs.

2 Scene Statistics

Table 1 shows statistics for each scene, including number of valid images, field of view (FoV) of the reference DeepMPI, and depth of the near and far MPI planes. We adopt the same method as that of Zhou *et al.* [11] to estimate the scale of each scene in order to set near and far plane depth. All data, including original images, registered poses, and SfM reconstructions will be released to the research community.

3 Losses

3.1 Losses optimizing DeepMPI color and α planes

Recall that in Section 3.3 of the main manuscript, we compare the rendered base color image $\hat{B}^k$ and real photo I^k at the target viewpoint using a reconstruction loss $\mathcal{L}_{\mathsf{recon}}$. $\mathcal{L}_{\mathsf{recon}}$ consists of a pixel-wise l_1 loss and a multi-scale gradient consistency loss [5]:

$$\mathcal{L}_{\mathsf{recon}} = ||\hat{B}^k(0) - I^k(0)||_{1,1} + w_{\mathsf{grad}} \sum_{i=1}^{N_s} ||\nabla \hat{B}^k(s) - \nabla I^k(s)||_{1,1}, \qquad (1)$$

where S is the number of scales we create for calculating the gradient consistency loss, and $I(s)$ is the image at scale s (where $s = 0$ is equivalent to the

Table 1: **Scene statistics.** We include (1) total number of images, 2) field of view (FoV) of the reference DeepMPI, and (3) depth of near and far MPI planes. The first five scenes are used for evaluation in the main manuscript.

Scenes	# images	FoV (°)	(near/far) plane depth
Trevi Fountain	3453	70	1/4
Sacre Coeur	2112	65	1/20
The Pantheon	1917	65	1/25
Top of the Rock	2232	75	1/75
Piazza Navona	606	70	1/25
Mount Rushmore	3075	30	1/4
Lincoln Memorial	2582	45	1/4
Eiffel Tower	1999	65	1/20

original resolution). In our experiments, we set $S = 3$ and use nearest neighbor downsampling to create image pyramids for both rendered and ground truth images.

3.2 Training Losses

Recall that in Section 3.4 of the main manuscript, to train the rendering network G, the appearance encoder E, and the latent feature in the DeepMPI, we compute losses between output views and ground-truth exemplar views. Specifically, our training loss is composed of three terms:

$$\mathcal{L} = \mathcal{L}_{\mathsf{VGG}} + w_{\mathsf{GAN}}\mathcal{L}_{\mathsf{GAN}} + w_{\mathsf{style}}\mathcal{L}_{\mathsf{style}}, \tag{2}$$

$\mathcal{L}_{\mathsf{VGG}}$ is a normalized VGG perceptual loss similar to that used in [11, 1]:

$$\mathcal{L}_{\mathsf{VGG}} = \sum_l w_l ||\phi_l(\hat{I}^k) - \phi_l(I^k)||_1, \tag{3}$$

where $\phi_l(x)$ indicates an output from VGG layer $l \in \{$ conv1_2, conv2_2, conv3_2, conv4_2, conv5_2 $\}$ with input x, and weight w_l is proportional to the reciprocal of the number of neurons in the corresponding VGG layer.

Furthermore, we add an adversarial loss $\mathcal{L}_{\mathsf{GAN}}$ to improve the realism of the rendered images. In particular, $\mathcal{L}_{\mathsf{GAN}}$ is computed from multi-scale discriminators [10] with an objective similar to LSGAN [6]:

$$\mathcal{L}_{\mathsf{GAN}} = \mathcal{L}_{\mathrm{GAN}}(D) + \mathcal{L}_{\mathrm{GAN}}(G), \tag{4}$$

$$\mathcal{L}_{\mathrm{GAN}}(D) = \mathbb{E}_{I^k \sim p(\mathbf{I})}\left[\left(D(I^k) - 1\right)^2\right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(I^k)}\left[D\left(G(M^k(B, \alpha, F), \mathbf{z})\right)^2\right], \tag{5}$$

$$\mathcal{L}_{\mathrm{GAN}}(G) = \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(I^k)}\left[\left(D\left(G(M^k(B, \alpha, F), \mathbf{z})\right) - 1\right)^2\right], \tag{6}$$

where $\mathcal{L}_{\text{GAN}}(D)$ is the loss for the discriminator, and $\mathcal{L}_{\text{GAN}}(G)$ is the loss for our neural render.

To further enforce that the appearance of rendered images match the appearance of the exemplar images, we add a style loss $\mathcal{L}_{\mathsf{style}}$, which compares the $l_{1,1}$ norm of the difference between Gram matrices constructed from VGG features at different layers:

$$\mathcal{L}_{\mathsf{style}} = \sum_{l} ||g(\phi_l(\hat{I}^k)) - g(\phi_l(I^k))||_{1,1} \tag{7}$$

where $g(x)$ is the Gram matrix from a VGG feature x.

4 Network Architecture

Let D and C denote number of depth planes and channels in our DeepMPI, and let H^k and W^k denote the height and width of the view at target viewpoint c_k.

Appearance encoder. Our appearance encoder consists of two encoders, denoted E_1 and E_2. E_1 takes as input a reference feature buffer $\mathcal{B}^r$ with fixed resolution of 512×512, and produces a latent feature vector $\mathbf{z}_1 \in \mathbb{R}^{512}$. We adopt the encoder implemented by Park *et al.* [8] for E_1. E_2 takes as input an exemplar image I^s with varying aspect ratios, and produces a latent feature vector $\mathbf{z}_2 \in \mathbb{R}^{256}$. We adopt the encoder from Huang *et al.* [3] as E_2. The two latent vectors are then passed through a fully connected layer in order to produce a final latent appearance vector $\mathbf{z} \in \mathbb{R}^{16}$ we describe in the main manuscript.

Neural renderer. We adopt the U-Net modified from Zhu *et al.* [12] as our neural rendering network. In summary, during training and evaluation, we feed the DeepMPI at the target viewpoint with size $D \times C \times H^k \times W^k$ to the rendering network, and the network predicts RGB MPI planes with size $D \times 3 \times H^k \times W^k$. However, the rendering network operates at each depth slice of the DeepMPI at target viewpoint *independently* (with size $C \times H^k \times W^k$), and predicts the corresponding RGB color image (with size $3 \times H^k \times W^k$). Therefore, our rendering network independently processes every depth slice of DeepMPI, without considering interactions between them.

Our rendering network consists of five convolutional layers in both the encoder and decoder. Each layer of the encoder consists of a 3×3 stride-2 convolutional layer followed by Instance Normalization [9] and leaky ReLu. Each layer of the decoder consists of bilinear sampling followed by a 3×3 convolutional layer. Adaptive Instance Normalization layers (AdaIN) [2] are embedded between bilinear sampling and feature concatenation of skip connections.

Discriminator. We adopt the network architecture from Huang *et al.* [3] as the discriminator used for our GAN loss. In particular, the discriminator takes as input images at three scales, and predicts scores from each patch of the input image.

5 Training and Implementation

We implement our framework using PyTorch. In all our experiments, we empirically set hyper-parameters $w_{\mathsf{grad}} = 0.25, w_{\mathsf{GAN}} = 0.2, w_{\mathsf{style}} = 5$. We set the resolution of reference DeepMPI to 784×784.

In the first stage, we optimize base color and α planes in the reference DeepMPI for 100 epochs in total (70 epochs in phase one, and 30 epoch in phase two) using a single Tesla T40 GPU. We adopt the Adam [4] optimizer with initial learning rate 1×10^{-3} for the optimization.

In the second stage, we use 4 Tesla T40 GPUs to jointly train the rendering network G, appearance encoder E, and latent features F^r in the DeepMPI for 50 epochs. During training, we adopt the Adam optimizer [4] and set a learning rate of 3×10^{-4} for E, G and F^r, and a learning rate of 1×10^{-5} for the discriminator. In addition, since Internet photos have varying aspect ratios and orientations, we resize their weight and height to a factor of 32 depending on images' original aspect ratios. Due to GPU memory limits, during training, we randomly crop a patch of 256×256 from the resized images and we only render a view corresponding to the patch. However, our method cann render a full image with resolution up to 640×480 at inference time on a single GPU.

6 Visual Illustrations

Examples of mean RGB PSV and base color. Figure 1 shows examples of the reference mean RGB color PSV at different depth layers from different scenes. These are used for initializing base color planes, as described in Section 3.3 of the main manuscript. In addition, Figure 3 shows estimated reference base colors, which are over-composited from base color planes in the reference DeepMPI.

Examples of rectified RGB images. Figure 2 shows visual examples of rectified RGB images in the feature buffer $\mathcal{B}^r$ aligned with the reference viewpoint, described in Section 3.4 of the main manuscript.

7 User Study

Table 2 shows scores from 1104 votes (46 participants $\times$ 24 comparisons) for each of following questions, respectively (where users are shown results from multiple algorithms to choose from):

Q1: "Which one looks **most photo-realistic**? e.g. which video best reproduces the details of geometry and illumination you would expect of a real world scene?"

Q2: "Which one appears to be **most consistent** across viewpoints, with the least jitter or flicker across frames?"

Q3: "Which one is most faithful to **the appearance of the source image**? For instance, which image best resembles the illumination and shading on the building in the source image?"

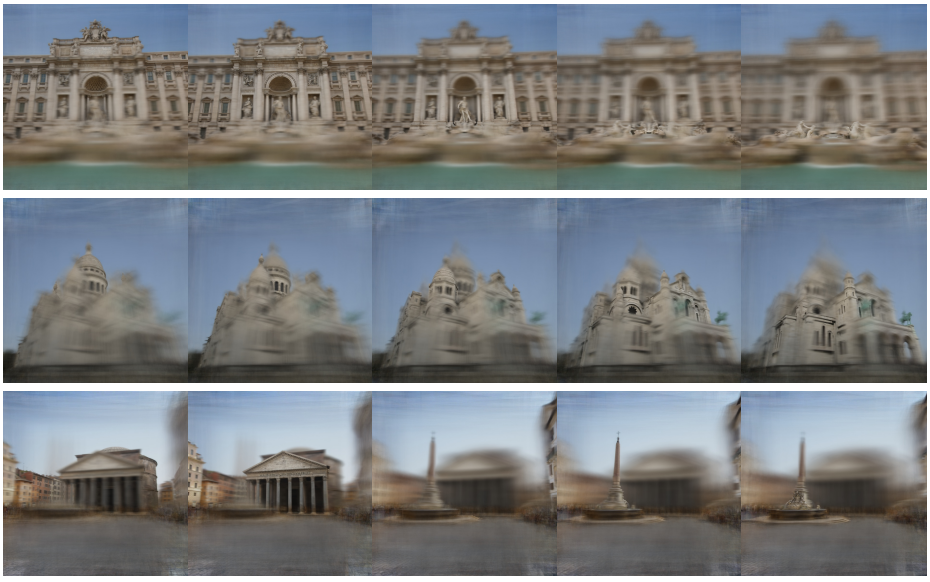

Fig. 1: **Visual illustration of reference mean RGB PSV**. Different images in each row indicate different depth planes of the plane sweep volume (PSV). The mean RGB images at different depth planes have different in-focus regions.

Table 2: **User study**

(a) **Share of votes on Q1.**

	MUNIT [3]	NRW [7]	Ours
Trevi Fountain	2%	12%	86%
Piazza Navona	6%	25%	69%
Top of the Rock	0%	8%	92%
Sacre Coeur	1%	14%	85%
The Pantheon	4%	18%	78%
Total	3%	15%	82%

(b) **Share of votes on Q2.**

	MUNIT [3]	NRW [7]	Ours
Trevi Fountain	1%	7%	93%
Piazza Navona	1%	28%	71%
Top of the Rock	0%	9%	91%
Sacre Coeur	1%	10%	89%
The Pantheon	0%	7%	93%
Total	1%	11%	88%

(c) **Share of votes on Q3.**

	MUNIT [3]	NRW [7]	Ours
Trevi Fountain	4%	13%	83%
Piazza Navona	9%	30%	61%
Top of the Rock	0%	9%	91%
Sacre Coeur	3%	16%	81%
The Pantheon	4%	27%	69%
Total	4%	19%	77%

The user study contains three sets of video comparisons and two sets of image comparisons randomly selected from each scene. Our method received the majority of votes on all questions across all five scenes.

Fig. 2: **Visual examples of rectified RGB images**. The reference rectified images are geometrically stable and globally aligned up to disocculusion.

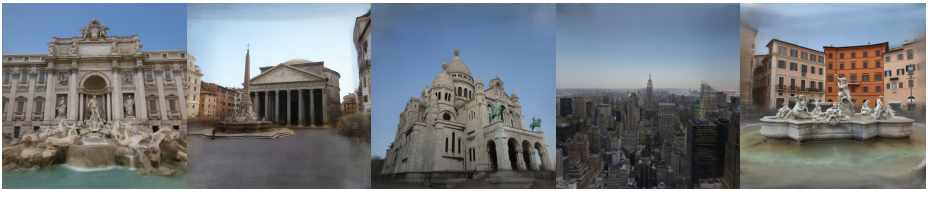

Fig. 3: **Visual examples of reference base color images**. These are over-composited from base color planes of the reference DeepMPI.

8 Additional Results

We include additional results on our project webpage, `crowdsampling.io`. In particular, we compare our approach with two baselines NRW [7] and MUNIT [3] in terms of novel view synthesis, appearance transfer and appearance interpolation. Further, we demonstrate the results of 3D hyperlapse effects (space-time interpolation) as well as novel view synthesis under diverse illumination from each of the eight landmarks.

References

1. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: Proceedings of the IEEE international conference on computer vision.

pp. 1511–1520 (2017)
2. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017)
3. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 172–189 (2018)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
5. Li, Z., Dekel, T., Cole, F., Tucker, R., Snavely, N., Liu, C., Freeman, W.T.: Learning the depths of moving people by watching frozen people. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4521–4530 (2019)
6. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2794–2802 (2017)
7. Meshry, M., Goldman, D.B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., Martin-Brualla, R.: Neural rerendering in the wild. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6871–6880 (2019)
8. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2337–2346 (2019)
9. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
10. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8798–8807 (2018)
11. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)
12. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Advances in Neural Information Processing Systems (2017)