Supplementary Material for SOMSI: Spherical Novel View Synthesis with Soft Occlusion Multi-Sphere Images

Tewodros Habtegebrial[1,2] Christiano Gava[1,2] Marcel Rogge[1,2]
Didier Stricker[1,2] Varun Jampani[3]
[1]TU Kaiserslautern [2]DFKI [3]Google Research

1. Introduction

In this document we present more details on our proposed Soft Occlusion Multi-Sphere Images technique for Spherical Novel View Synthesis. The architecture of our network as well as training details are given in Section 2. In Section 3, we discuss results with a baseline method that directly learns an MSI representation of the scene without MLPs. Finally, in Section 4 we explain camera pose estimation for manually captured spherical light fields.

2. Network Architecture and Training Details

2.1. Network Architecture

We use a multi-scale MLP network that is implemented as 1×1 convolutional network. The network takes the reference views' 2D positional encodings of each pixel along with RGB color values. These inputs are then processed through multiple stages to produce the desired SOMSI scene representation. Note that each pixel (u, v) in the reference Equirectangular Projection (ERP) has a unique viewing direction (θ, ϕ) on the unit sphere associated with it; see preliminary section in the main paper.

	Input channels	Output channels	Input scale	Output scale
Block 1	$\mathcal{E} + 3$	$\mathcal{F}[1]$	$H/2^7$	$H/2^6$
Block 2	$\mathcal{E} + 3 + \mathcal{F}[1]$	$\mathcal{F}[2]$	$H/2^6$	$H/2^4$
Block 3	$\mathcal{E} + 3 + \mathcal{F}[2]$	$\mathcal{F}[3]$	$H/2^5$	$H/2^3$
Block 4	$\mathcal{E} + 3 + \mathcal{F}[3]$	$\mathcal{F}[4]$	$H/2^4$	$H/2^2$
Block 5	$\mathcal{E} + 3 + \mathcal{F}[4]$	$\mathcal{F}[5]$	$H/2^3$	$H/2^1$
Block 6	$\mathcal{E} + 3 + \mathcal{F}[5]$	$\mathcal{F}[6]$	$H/2^2$	$H/2^1$
Block 7	$\mathcal{E} + 3 + \mathcal{F}[6]$	$\mathcal{F}[7]$	$H/2^1$	H
Output Block	$\mathcal{F}[7]$	$k \times f + d \times (k+1)$	H	H

Table 1. SOMSI Network architecture. The input to SOMSI is the concatenation of reference image RGB values and positional encodings of its 2D pixel positions.

Table 1 summarizes the architecture of our network: It is composed of convolutional blocks, each block consisting of four 1×1 convolutions, each followed by ReLU. The positional encodings and color values of the reference ERP are fed to each convolution block at different resolutions. The output of $Block\ i$ is bilinearly upsampled by a factor of 2 and is given as input to $Block\ i+1$. We refer to Fig. 3 in the main paper for an overview of SOMSI's architecture.

In Table 1, $\mathcal{E} = 4 \times l + 2$ is the positional encoding size, for a given number of octaves l. The number of channels in the last block corresponds to the total number of outputs per pixel, i.e. d opacity values, $d \times k$ visibility/occlusion and $k \times f$ multi-occlusion appearance features. The number of features per channel is decreased linearly from $\mathcal{F}[1] = 256$ to $\mathcal{F}[7] = 128$.

Unless specified otherwise, we use the following settings for our experiments: number of spheres $d = 64$, number of occlusion layers $k = 3$ and number of octaves $l = 4$.

3. Vanilla RGBA MSI baseline without MLP.

We have experimented with a baseline that learns an MSI scene representation, directly. It is possible to randomly initialize and directly optimize an $m \times d \times 4$ dimensional MSI via gradient descent. Although cheaper to predict, it fails to produce realistic MSIs, as shown in Fig 1. PSNR score for this baseline is much lower than our method (by atleast 8 PSNR drop on Sea-Port and Residential datasets). This shows the importance of the inductive bias of using MLPs. The inductive bias of the MLP is crucial in learning realistic MSIs. Due to its poor performance, we disregarded this baseline, in favor of more robust baselines such as MatryODShka [1] and NeRF [3].

3.1. Training

We trained our model using a batch size $b = 2$ for 30000 epochs. We use the Adam [2] optimizer with the following settings: $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The network is implemented in the PyTorch [4] Deep Learning framework.The network can be trained in about 18 to 24 hours on an NVIDIA GPU that has $15 > GB$ memory.

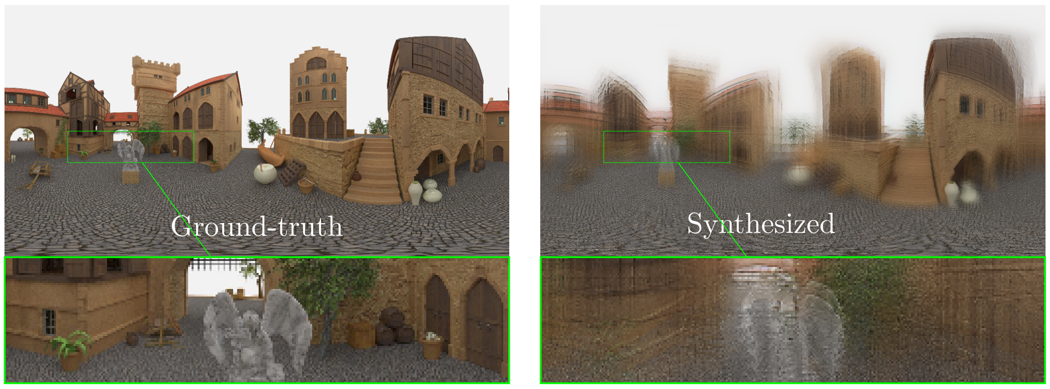

Figure 1. **Sample Vanilla-MSI Result**. Novel view (right) generated using Vanilla-MSI is quite blurry compared to GT (left). Producing realistic MSIs without the inductive bias of MLPs is challenging. Please zoom in for details.

4. Calibrating Spherical Light Fields with COLMAP [5]

The structure from motion toolbox from Shoenberger *et al*. [5] (known as COLMAP) has become the standard tool for calibrating multi-view datasets. COLMAP works for pinhole and fisheye cameras. We calibrate our spherical datasets (*Coffee Area 1-4*) by first extracting virtual perspective views from each spherical image and applying COLMAP on the collection of perspective images. After calibration each virtual perspective camera will have known rotation and translation from a reference world coordinate.

The next step is to determine the pose of all spherical cameras based on their virtual perspective counterparts. This is possible because the output camera poses of COLMAP and their corresponding spherical cameras are related by a rotation matrix R_s^p. For a perspective image whose principal axis intercepts the unit sphere at coordinates (θ, ϕ), R_s^p can computed as: $R_s^p = R_x \times R_z$, where R_x is a rotation around the x axis by angle ϕ and R_z is a rotation around the z axis by $\frac{\pi}{2} - \theta$.

Perspective to sphere rotation matrices $R_p^s = (R_s^p)^T$ are used to calculate the rotation component of the spherical camera poses: $R_w^s = R_p^s \times R_w^p$. Translation vectors are adapted accordingly: $\mathbf{t}_w^s = R_p^s \times \mathbf{t}_w^p$, where $\mathbf{t}_w^s$ is expressed in the spherical camera coordinates system.

References

[1] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision (ECCV)*, pages 441–459, 2020. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv Preprint*, 2014. 1

[3] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 1

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 1

[5] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 2