NeuTex: Neural Texture Mapping for Volumetric Neural Rendering Supplementary Material

Fanbo Xiang[1], Zexiang Xu[2], Miloš Hašan[2], Yannick Hold-Geoffroy[2], Kalyan Sunkavalli[2], Hao Su[1]
[1]University of California, San Diego [2]Adobe Research

1. Cube map Clarification

We briefly clarify our texture visualization (with cubemaps) used in our main paper. As discussed in Sec. 3.3 in the paper, we use spherical UVs for our texture mapping, where $\mathbf{u}$ represents a point on the surface a unit sphere. For all the figures in the main paper, we use cubemaps [1] to visualize the spherical domain. A cubemap consists of sixes faces of a unit box, recording all the color information projected from a unit sphere (as shown in Fig 1), which is widely used in graphics for spherical mapping. An alternative standard way to visualize a spherical function is to use a equirectangle map. We show the correspondence between a cube map and a equirectangle map in Fig. 2. We use cubemaps in the paper since they involve less distortion, avoiding the distorted regions in the top and bottom of equirectangle maps.

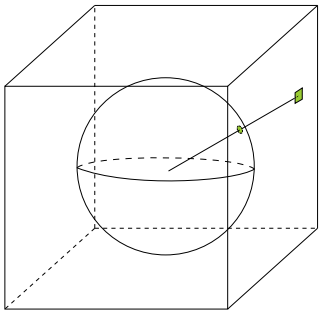

Figure 1. Cube map projection. The color at each point on the unit sphere is projected to a point on the cube centered at the origin. A cubemap is obtained by "opening up" the cube.

2. Network Implementation Details

2.1. Network structure

We show the detailed network architecture for $F_\sigma, F_{uv}, F_{uv}^{-1}$ and F_{tex} in Figure 3.

2.2. Training details in initialization

Here we describe in detail how we do the initialization stage mentioned in Sec 4.2. in the paper. Given a point cloud from Colmap, we first downsample it to one with 2,000-3,000 points. We denote this point cloud as P_{gt}. We then sample 2,500 points uniformly in the UV space (the unit sphere). We denote the set of UV coordinates as P.

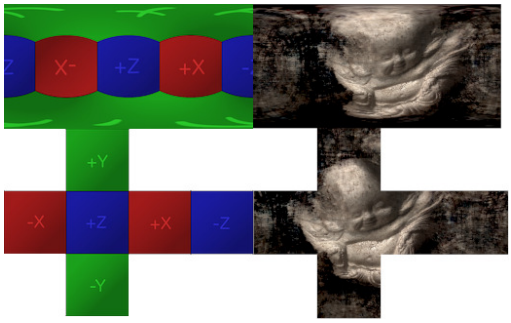

Figure 2. Cube maps on the second row corresponds to the equirectangle maps on the first row. They are different projections of the same spherical texture. A cubemap has a smaller distortion on the Y direction.

Chamfer loss. The Chamfer loss is simply the Chamfer distance between $F_{uv}^{-1}(P)$ and P_{gt}, where $F_{uv}^{-1}(P)$ corresponds to the point cloud generated by inverse-mapping very UV in P to the 3D space using the network F_{uv}^{-1}.

$$L_{\text{chamfer}} = \text{Chamfer}(F_{uv}^{-1}(P), P_{gt})$$

Inverse loss. We also leverage a loss that is similar to our cycle loss to let the initialization also influence the texture mapping network F_{uv}. In particular, instead of the 3D-to-2D-to-3D cycle mapping used the cycle loss in Eqn. 12 of the paper, we leverage a 2D-to-3D-to-2D cycle mapping in the initialization, given by:

$$L_{\text{cycle2}} = ||F_{uv}(F_{uv}^{-1}(P)) - P||_2^2$$

.

Rendering and mask loss. The same rendering and mask loss as described in section 4.1 are also applied in the initialization stage. So the loss at initialization stage is

$$L_{\text{init}} = L_{\text{chamfer}} + aL_{\text{cycle2}} + bL_{\text{render}} + cL_{\text{mask}}$$

where we set $a = 100$, $b = c = 1$.

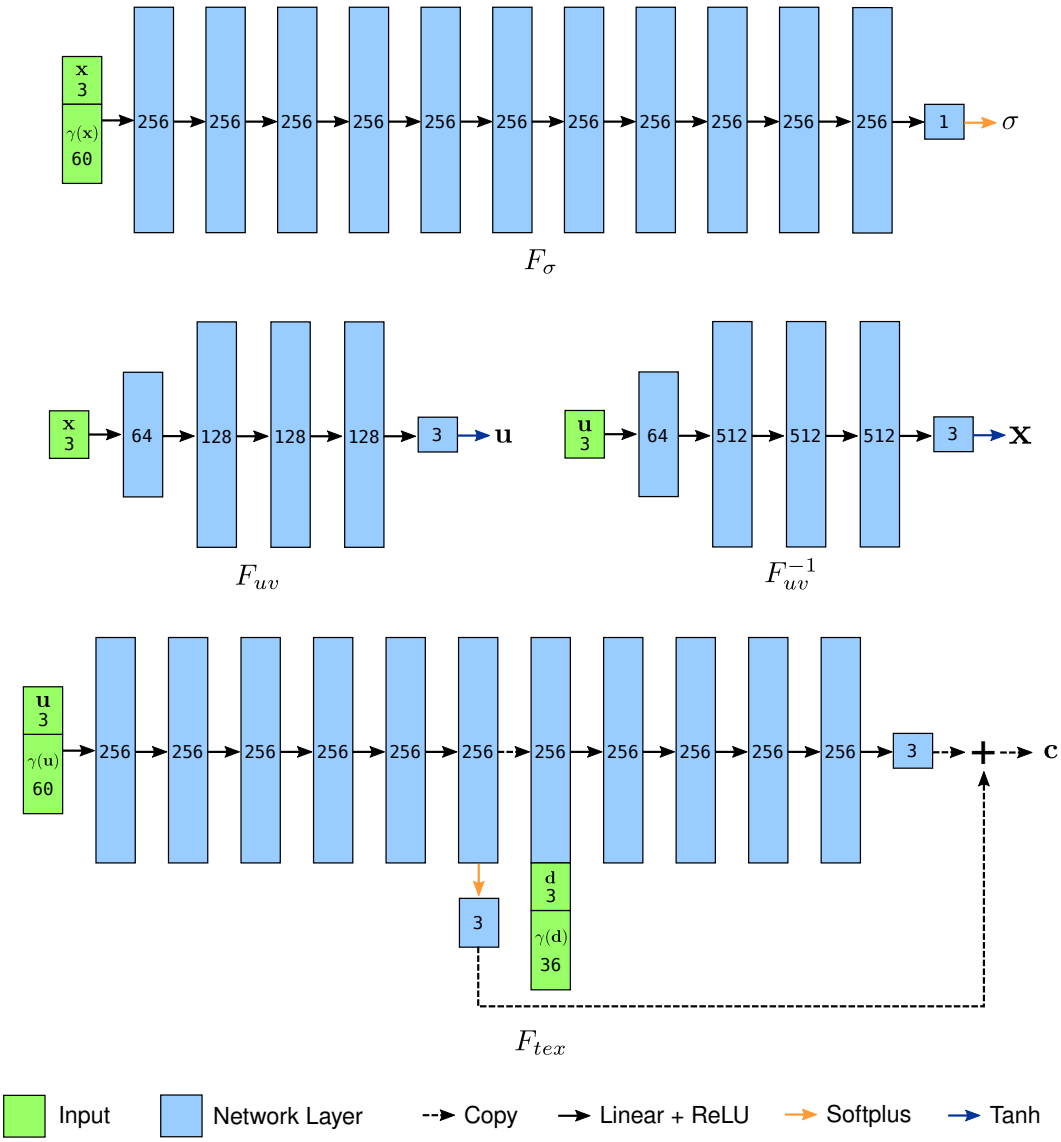

Figure 3. Network structure for the 4 networks. **x** represents 3D coordinates. γ denotes positional encoding. **u** represents texture-space points (3D points on the unit sphere). **d** represents a 3D unit vector for viewing direction. σ is predicted volumetric density. **c** is predicted radiance.

3. Additional Results

3.1. Full quantitative comparison

We have shown the averaged quantitative results across five DTU scenes in Tab. 1 of the paper. Detailed comparisons on individual scenes are provided in Table 1. Similar to the average scores, though slightly worse than NeRF, our method significantly outperforms other traditional and neural rendering methods.

3.2. Additional visual results

Figure 4 shows the visual comparison of different methods on the remaining 2 DTU scenes. Figure 5 shows additional texture editing results. Please refer to the attached video for more results on view synthesis and editing.

References

[1] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[2] 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. *arXiv preprint arXiv:2003.08934*, 2020.

[3] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstruc-

Method	55		83		114		118		122	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SRN[5]	21.35	0.673	28.68	0.929	23.75	0.808	28.74	0.900	27.75	0.877
DeepVoxels[4]	17.21	0.532	23.76	0.858	17.97	0.606	23.18	0.764	22.12	0.748
Colmap[3]	21.25	0.784	27.11	0.921	20.69	0.809	27.43	0.907	26.66	0.905
NeRF[2]	**26.78**	**0.913**	**31.77**	**0.952**	**27.38**	**0.918**	**33.98**	**0.954**	**33.72**	**0.955**
Ours	**22.67**	**0.808**	**30.61**	**0.931**	**26.45**	**0.891**	**30.67**	**0.916**	**30.75**	**0.925**

Table 1. PSNR/SSIM for novel view synthesis quality on 4 held-out views on 5 DTU scenes.

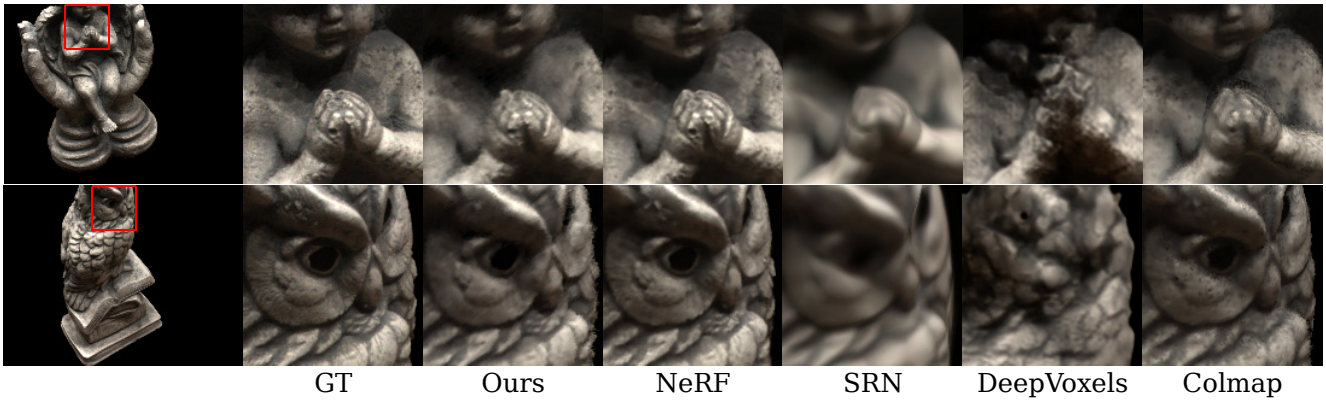

Figure 4. Comparison on the remaining DTU scenes.

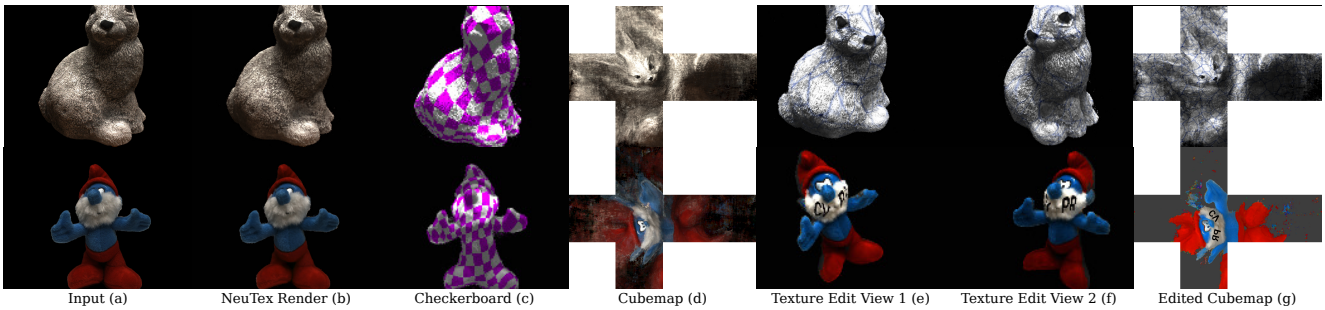

Figure 5. Additional texture editing on DTU scenes.

tured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[4] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3D feature embeddings. In *CVPR*, pages 2437–2446, 2019.

[5] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019.