Neural Lumigraph Rendering – Supplemental Material

Petr Kellnhofer[1,2], Lars C. Jebe[1], Andrew Jones[1], Ryan Spicer[1], Kari Pulli[1], Gordon Wetzstein[2,1]
[1]Raxium [2]Stanford University
{pkellnhofer,ljebe,ajones,rspicer,kpulli,gwetzstein}@raxium.com

1. Real-time rendering analysis

In Section 4.3 of the paper, we compared performance of our real-time renderer to the neural renderer. Here, we complement this comparison by demonstrating that the rendering quality is a joint product of both the shape representation in S and the emissivity function E. To that goal, we use our new head image dataset and evaluate how well a novel view can be interpolated and/or extrapolated with different choices of geometry and textures.

Setup We compare our real-time rasterized renderer (RAS) using geometries reconstructed by the surface based methods of Colmap [10], IDR [12] and our Neural Lumigraph Rendering (NLR). Additionally, we consider both the neural textures generated by the respective method (not available for Colmap) as well as alternative usage of the original training camera images.

We provide the renderer with textures corresponding to 5 of the 6 high-resolution central Back-Bone H7PRO cameras in our dataset (see Section 5), and we measure the PSNR of the held-out interpolated/extrapolated 6^{th} view. The same ground-truth masks are applied for all measurements. We average results from all 6 possible test view choices.

Results Table 1 presents PSNR scores for each scene as well as the overall average. A qualitative comparison is presented in Fig 3. For all choices of textures, we observe a favorable performance of the shape exported from our method compared to shapes exported by both the IDR and Colmap. Colmap generates very accurate but incomplete geometries resulting in holes in the rendered images (note the hair in the scene "L1"). The geometry extracted from IDR provides similar degree of view consistency as our own geometry when combined with the original captured textures, even though our method still maintains a small margin (see the column "Captured textures" in Tab. 1). However, the neural textures generated by IDR lack detail present in our neural textures which results in comparatively lower PSNR scores (see the column "Neural 1×").

Scene	Method	Captured textures	Neural textures		
			1×	2×	3×
A1	Colmap	**23.96**	N/A	N/A	N/A
	IDR	22.23	22.49	23.31	23.61
	LR	23.65	**23.83**	**26.93**	**30.33**
A8	Colmap	11.48	N/A	N/A	N/A
	IDR	19.80	19.65	20.80	21.15
	LR	**22.90**	**23.06**	**26.00**	**29.20**
L1	Colmap	5.45	N/A	N/A	N/A
	IDR	18.55	18.49	18.89	18.99
	LR	**21.24**	**21.61**	**24.13**	**26.56**
L3	Colmap	20.03	N/A	N/A	N/A
	IDR	23.56	23.66	25.24	25.95
	LR	**24.32**	**24.85**	**28.76**	**32.10**
L4	Colmap	13.36	N/A	N/A	N/A
	IDR	17.98	17.73	18.36	18.59
	LR	**19.57**	**19.80**	**22.20**	**24.25**
M2	Colmap	18.40	N/A	N/A	N/A
	IDR	22.07	21.84	22.60	22.96
	LR	**23.94**	**24.34**	**26.96**	**29.76**
P4	Colmap	**24.22**	N/A	N/A	N/A
	IDR	16.06	13.85	14.78	15.59
	LR	24.16	**24.27**	**27.41**	**30.04**
Average	Colmap	16.70	N/A	N/A	N/A
	IDR	20.04	19.67	20.57	20.98
	LR	**22.83**	**23.11**	**26.06**	**28.89**

Table 1: PSNR scores for the interpolated/extrapolated views achieved by different variants of our real-time renderer. Neural textures for the original camera poses (1×), and their supersampled variants (2×, 3×) are compared to a direct use of the original images captured by our camera array.

Texture space upsampling An important argument for using the neural textures rather than the original captured images is the possibility to use the neural renderer to produce additional virtual views. This allows to generate much denser virtual camera spacing, i.e. through view synthesis,

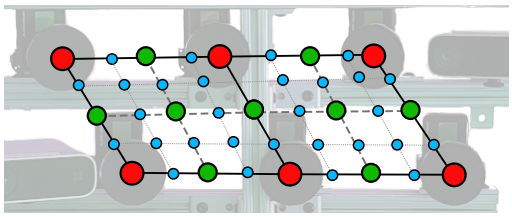


Figure 1: The positions of the original texturing cameras (red) and the interpolation pattern to generate novel textures labeled as "2×" (green) and "3×" (blue).

than would be practical with physical camera setups. To demonstrate this unique capability we resampling the texture space by subdividing the original 3×2 camera layout in our dataset as shown in Figure 1. This yields first the original 6 textures (labeled "1×"), then additional 9 textures (total of 15; labeled "2×") and finally additional 30 textures (total of 45; labeled "3×"). We follow the same procedure and remove the texture corresponding to the original ground-truth camera pose for the test.

The results are again presented in Table 1 (two rightmost columns) and in Fig 3. Both IDR and SineSDF experience significant quality boost as the texturing angular space gets denser and the interpolation distances between blended textures get smaller. However, only our technique can leverage the high spatial detail featured in our neural textures and, as a result, shows significantly larger PSNR overall. Please explore the video supplement to evaluate dynamic properties of this behavior.

2. Additional Results

In this section, we provide additional results that did not fit into the main paper. Table 2 complements Table 2 in the main paper and shows results of the image reconstruction metrics computed for the three test views held-out from the training on the DTU dataset [3]. Table 3 extends Table 3 in the main paper and shows additional image metric scores for results using variety of mutli-view datasets.

3. Supplemental Video

We provide a detailed video supplement to fully evaluate view consistency of our novel rendered views and compare to the baseline methods. A summary video with side by side comparison and comments is available in the root folder as `4945_supplemental_video.mp4` (x264 encoding in mp4 container). Additionally, individual results in form of short videos are provided separately in three folders, each showing a different type of comparison.

Neural renderings Videos in the folder `neural_videos` present neural renderings of scenes from the datasets used in our paper produced by our method as well as well Colmap, IDR, Neural Volumes and NeRF. Note, that the renderings are provided without post-processing and show the background reconstruction present in Neural Volumes and NeRF. This background was not evaluated in metrics in our paper; all tested methods use the same foreground mask when evaluating the PSNR.

Shape renderings In the folder `shape_videos`, we use a simple OpenGL renderer using the PyRender Python package to visualize meshes extracted from Colmap, IDR and our Neural Lumigraph that are later used for the real-time rendering. Note that IDR contains lot of boundary geometry noise for views that were not supervised during training.

Real-time renderings Finally, in the folder `cg_videos`, we present outputs of our real-time renderer as well as different variants presented in the study in Sup. Sec. 1.

4. Input sensitivity

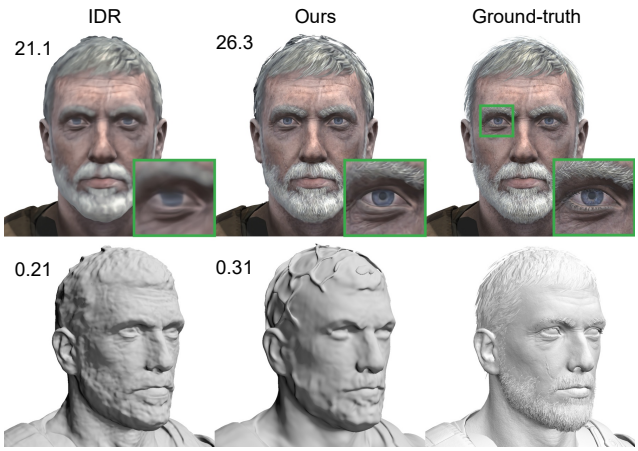


Figure 2: A comparison of image and shape reconstruction by IDR and our method with the ground-truth Unity rendering and mesh shown on the right. The upper numbers denote the image PSNR averaged over all views and the lower numbers correspond to the Chamfer distance.

Some of the key input properties that affects quality of multi-view reconstruction is the accuracy of camera calibration. We investigate how our method performs if this factor is taken out of the equation by measuring its performance on a computer-generated 3D scene. We have used Unity to render a 5×5 grid of views of a 3D head model with surface specular properties at a resolution of 2160×2160 pixel and we extracted the ground-truth camera poses as well as the object masks.

Scan	Colmap [10]			IDR [12]			NeRF [7]			NLR-ST			NLR-RAS		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
65	15.12	0.845	0.156	21.89	0.939	0.105	**27.08**	**0.948**	**0.069**	26.58	0.941	0.083	25.60	0.940	0.087
97	11.37	0.835	0.166	22.77	0.914	0.128	24.17	0.918	0.109	**26.43**	**0.922**	**0.108**	26.40	0.922	0.108
106	16.69	0.875	0.153	20.49	0.891	0.205	**30.17**	0.934	0.107	29.60	**0.939**	**0.098**	28.52	0.938	0.104
118	21.72	0.912	0.100	23.24	0.937	0.150	**31.03**	**0.955**	**0.083**	30.77	0.950	0.091	30.39	0.950	0.093

Table 2: Image error metrics PNSR, SSIM [11] and LPIPS [13] computed for the 3 held-out test views of the DTU dataset [3] complementing the respective training errors in Table 2 of the main paper. Best scores in bold, second best underlined.

Dataset	Colmap			IDR			NV			NeRF			NLR-ST			NLR-RAS		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Volucap (1)	19.57	0.975	0.037	22.27	0.978	0.043	29.32	0.975	0.034	**32.71**	**0.982**	0.026	28.59	0.981	**0.022**	28.57	0.981	**0.022**
Digit. Ira (1)		Fail.		23.89	0.835	0.286	26.54	0.829	0.287	31.18	0.845	0.267	**31.63**	**0.851**	**0.255**	31.13	**0.851**	0.260
Ours (7)	17.58	0.841	0.187	22.34	0.878	0.202	25.05	0.857	0.186	28.45	0.898	0.171	**30.45**	**0.921**	**0.147**	30.25	**0.921**	0.151

Table 3: Average image reconstruction error metrics PNSR, SSIM [11] and LPIPS [13] computed across datasets (number of scenes in parentheses). (*) Only 5 scenes tested.

As Fig. 2 shows, both methods achieve a clean shape reconstructions without artifacts. The hair region is represented by a cloud of semi-transparent billboards in the original 3D model, so this would be challenging to be accurately reconstructed by any approach. While the Chamfer distance metric favors the results of IDR, we see that our image quality is much higher as it features notably higher spatial fidelity. We conclude that our method can avoid geometry artifacts if consistent camera and image labeling can be obtained.

5. Datasets

Our Captured Dataset Each capture consists of 22 images and corresponding foreground masks. Six images are captured with high-resolution narrow field-of-view (FOV) cameras, while the remaining 16 images are captured with low-resolution wide FOV cameras. While these additional 16 images provide useful geometrical information, they do not carry high quality visual detail. This is why we use all views for training each of the evaluated methods but we only evaluate reconstruction quality for the 6 central high-resolution cameras.

Figure 4 shows scenes in our dataset. These data are available on the project website.[1]

DTU We use the multiview images and camera calibrations provided by [3] along with object masks by Niemeyer *et al*. [9] and Yariv *et al*. [12]. Each scan consists of 49 or 64 images with 1600×1200 pixels and covering approximately 90×90 degree view zone of the object. Ground truth 3D shapes in form of point captured using structured light stereo are used to compute the Chamfer distance.

Volucap We evaluate a full body reconstruction using a sample frame from a video sequence provided by the volumetric production studio Volucap GmbH [2]. The cameras are distributed in pairs around an upper hemisphere of the capture dome with a subject standing in the center. The images are cropped and rescaled to 2028×1196 px. The masks are manually annotated on top of automatic background subtraction.

Digital Ira The Digital Ira dataset [1] contains 7 very high resolution (3456×5184 px) closeups of a man's face in a studio setting. We used the provided camera calibration data and we manually annotated objects masks for each view.

6. Baselines

Colmap We follow the same procedure and settings for Colmap [10] as Yariv *et al*. [12] to reconstruct a 3D point cloud from the multi-view images, build a surface mesh using Poisson reconstruction (with trim = 7) and render the images using PyRender. For the DTU dataset, we also apply object masks to remove background points before the surface reconstruction. Note that this method failed to produce a surface mesh of the Digital Ira [1] with all variations of parameters that we tried.

[1] http://www.computationalimaging.org/publications/nlr/

NeRF We use the original code shared by the authors [8]. We modify the code to support general camera models. We execute the code using the provided high-quality configuration used for their paper results but we reduce the number of random samples to 1024 to fit to our GPU memory (11 GB on Nvidia Geforce RTX 2080Ti). Further, to support variable sensor sizes in our dataset, we upsample all images to the shape of the high-resolution cameras which retains the detailed information.

For our dataset, we are forced by the memory requirements to reduce the training resolution by factor of 2 from 3000×4000 pixels to 1500×2000 pixels. However, it is unlikely that this reduction affected the results since our experiment with even more aggressive four-fold downsizing (to 750×1000 pixels) yielded a PSNR of 28.62 dB which is comparable to 28.58 dB of the two-fold downsizing we ended up using.

IDR We use the original code and configuration shared by the authors [12]. Note, that while IDR supports training without accurate camera calibrations, we train with the same calibrations used for all other methods and keep the poses fixed during the process.

Neural Volumes We use the original code shared by the authors [6]. We train our models for at least 150K iterations. As recommended by the authors, we use three views from different angles to condition the autoencoder. The authors provide the ability to either pass the background explicitly or to let the network estimate the background. Since we don't have a background image for all datasets, we choose to let the network estimate the background. Due to the long training times, we evaluate the method on a limited subset of our dataset. We crop the lower resolution landscape images to the same aspect ratio as the six high-resolution central cameras in portrait mode. Due to GPU memory limitations, we are only able to train with the image resolution reduced to $1/4$ of the original size of the high-resolution images. Note that Neural Volumes uses an explicit voxel representation and is therefore, unlike implicit models, not resolution independent.

7. Metrics

PSNR For all methods we compute the PSNR between the reconstructed images and the ground truth only for pixels in the object masks. The same procedure is used by Yariv *et al*. [12] and we verify it by reproducing their metric scores with their pre-trained models. Therefore, even methods that do not use masks to recover the shape (NeRF, Neural Volumes) are only rated based on prediction of the foreground object.

Chamfer distance The Chamfer distance is computed in a similar manner as in the original Matlab scripts provided with the DTU dataset [3]. The shortest distance between each ground-truth 3D point and the reconstructed shape is computed in one direction by the Open3D [14] library for Python. This prevents penalizing reconstruction of background (NeRF) or reconstruction of occluded object parts. We leave out Neural Volumes from evaluating the metric since surface extraction was not demonstrated in the original manuscript [6] and the explicit nature of the representation makes recovery of fine details difficult.

8. Sphere tracing implementation details

Convergence of the main paper Eq. 5 towards the zero-level set S_0 is not guaranteed for general shape configuration. As the SDF function S indicated distance towards the nearest surface, it underestimates the optimal step length if the nearest surface is not orthogonal to the current ray. This can partially be improved by dividing the step length by the dot product $-\mathbf{r_d} \cdot \nabla_{\mathbf{x}} S(\mathbf{x}_i)$. However, computing the gradients in every step is costly and quickly diminishes the returns. Furthermore, if a ray closely misses a surface edge with a surface normal orthogonal to the ray direction, such adjustment does not improve the convergence as the step length will stay very small.

To mitigate these issues, we use bidirectional sphere tracing as described by Yariv *et al*. [12]. This approach does not rely on the sphere tracer's convergence as it uses it to only narrow a possible range of surface positions which is then further refined by additional solvers.

To this goal, we solve Eq. 5 to get the near zero-level set $\mathbf{x}_n^n$ in a forward direction for $n = 16$. We mask rays with $|S(\mathbf{x}_n)| < 5e^{-5}$ as converged. These rays are not further optimized and $\hat{\mathbf{x}}_n = \mathbf{x}_n^n$ is the final output of the sphere tracer. Next, for the remaining rays, we solve a modified equation in an opposite direction to find a far zero-level set $\mathbf{x}_n^f$ as

$$\mathbf{x}_0^f = \mathbf{r_o} + t_f \mathbf{r_d}, \quad \mathbf{x}_{i+1}^f = \mathbf{x}_i' - S(\mathbf{x}_i^f)\mathbf{r_d}. \tag{1}$$

where t_f is the rear intersection of $\mathbf{r_d}$ with a unit sphere. $\mathbf{x}_n^n$ and $\mathbf{x}_n^f$ define the nearest and the furthers possible location of the first surface point $\mathbf{x}_n$. In cases where $\mathbf{x}_n^f$ is closer than $\mathbf{x}_n^n$, we conclude that no surface exists. In the remaining cases we evaluate 100 samples with the candidate range and look for the first zero crossing of the S. Finally, assuming a locally monotonic S we further refine the zero-crossing location by 8 steps of sectioning [12] to get the zero-level set $\hat{\mathbf{x}}_n$.

Note that for performance, stability and memory efficiency, this procedure is not differentiated [12, 4, 5]. Therefore, one extra step of the forward sphere tracing is executed with gradient tracking enabled to achieve a differentiable

sphere tracing. We apply the gradient direction adjustment [12] for this last step to get the final zero-level set as

$$\mathbf{x}_n = \hat{\mathbf{x}}_n - \frac{S(\hat{\mathbf{x}}_n)}{\nabla_{\mathbf{x}} S(\hat{\mathbf{x}}_n) \cdot \mathbf{r_d}} \mathbf{r_d}, \tag{2}$$

for points where $|S(\mathbf{x}_n)| < 0.005$. A division by zero is avoided by clamping the denominator to $|\nabla_{\mathbf{x}} S(\hat{\mathbf{x}}_n) \cdot \mathbf{r_d}| > 0.01$.

References

[1] Graham Fyffe, Andrew Jones, Oleg Alexander, Ryosuke Ichikari, and Paul Debevec. Driving high-resolution facial scans with video performance capture. *ACM Trans. Graph.*, 34(1), Dec. 2015. 3

[2] Volucap GmbH. A multiview capture data sample, 2020. 3

[3] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 2, 3, 4

[4] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proc. CVPR*, 2020. 4

[5] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. CVPR*, 2020. 4

[6] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph. (SIGGRAPH)*, 38(4), 2019. 4

[7] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph. (SIGGRAPH)*, 38(4), 2019. 3

[8] 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 *Proc. ECCV*, 2020. 4

[9] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. CVPR*, 2020. 3

[10] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 3

[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 3

[12] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Proc. NeurIPS*, 2020. 1, 3, 4, 5

[13] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3

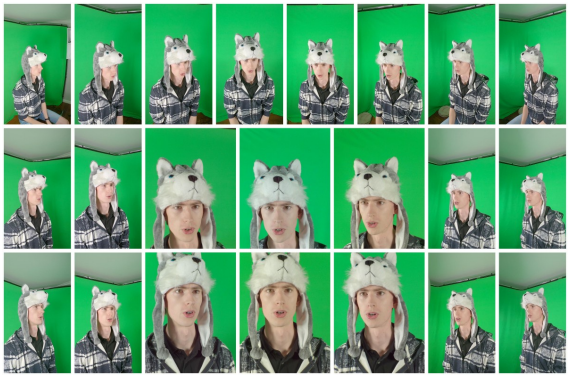
[14] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 4

Figure 3: Comparison of our real-time rendering variant evaluated on an view not present in the texture set. The IDR results for the presented view in P4 are black due to an occlusion by a false geometry generated in front of the person (see the shape rendering). While fixable and only present in this scene, we keep the geometry as is and provide detailed metrics for each single scene in Table 1. A rejection of this result would not change the overall trend.

(a) A1

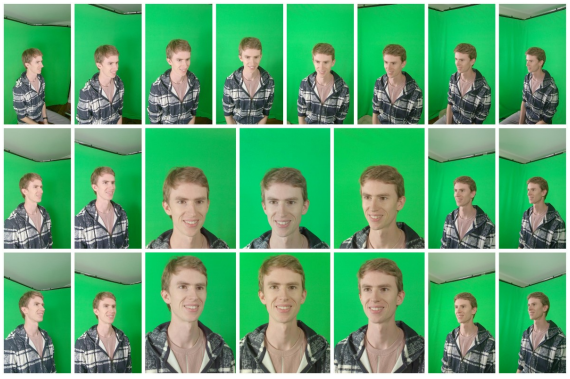
(b) A8

(c) L1



(d) L3

(e) L4



(f) M2

(g) P4

Figure 4: The seven scenes forming our dataset. Image positions approximate the camera layout. The central 6 images are high-resolution head close-ups.