FoamSurf: High-Quality Surface Reconstruction Using RadFoam

Mats Grobe
TUM
Munich
mats.grobe@tum.de

Ignacio Dassori
TUM
Munich
ignacio.dassori@tum.de

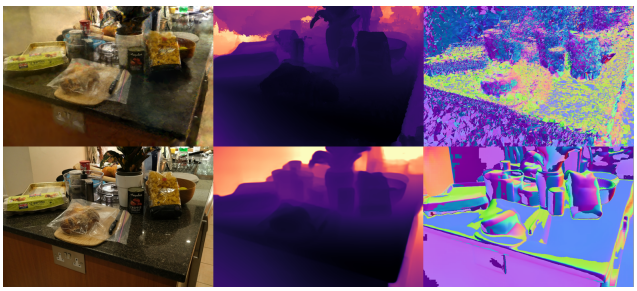

Figure 1. Full size image: Full model implementation. Top row: RadFoam novel-view render (left), predicted depth map (center), predicted normals (right). Bottom row: ground truths.

1. Introduction

Neural Radiance Fields (NeRF) [7] revolutionized the field of novel view synthesis through optimizing a continuous volumetric radiance field, parameterized by an MLP, while achieving impressive quality. However, their reliance on per-sample network evaluations leads to slow train and render times.

To address this, Gaussian Splatting, a scene representation as collections of anisotropic Gaussians was introduced. This enabled real-time rasterization on the GPU without sacrificing fidelity [6]. Follow-up works such as DN-Splatter [8] further leverage monocular depth and normal priors to improve surface reconstruction.

Most recently, the authors of Radiant Foam (RadFoam) [3] proposed a volumetric mesh representation based on a differentiable Voronoi tessellation. Despite its strong novel view synthesis results the framework has not yet been optimized for surface extraction. In this work, we introduce **FoamSurf**, a lightweight extension of the RadFoam that incorporating expected *depth* and *normal* supervision using off-the-shelf monocular priors. Our code is currently non-publicly available at https://github.com/matsgrobe/FoamSurf.

2. Related work

2.1. RadFoam

RadiantFoam (RadFoam) [3] proposes representing scenes via a foam structure of polyhedral cells. Through a volumetric mesh ray tracing algorithm it is capable to jointly take advantage of rasterization on the GPU like 3D Gaussian Splatting (3DGS) [6] whilst avoiding approximations that make the implementation of light transport phenomena such as reflection and refraction more difficult. Cells are continuously optimized by parameterizing them as a Voronoi diagram which is shown to be differentiable under volume rendering. Another major benefit of the framework is its nonreliance on NVIDIA RT cores.

2.2. Surface reconstruction

Following the introduction of *3D Gaussian Splatting*, numerous works focused on extending it for more accurate surface geometry reconstruction, addressing issues caused by the multi-view inconsistency of 3D Gaussians. By improving the alignment of Gaussians with the underlying geometry, these methods enable the extraction of high-quality meshes from the reconstruction.

SuGaR [4] introduces a regularization term that encourages Gaussians to align with the scene surface. They achieve this by deriving an SDF from the Gaussians and minimizing its distance to a desired SDF. Additionaly, they introduce an efficient mesh extracting method that exploits this alignment to produce highly detailed meshes.

2DGS [5] simultaneously reconstructs geometry and appearance by optimizing oriented 2D disks in favor of 3D Gaussians. The elliptical disks align tightly to surfaces, and training requires only a sparse point cloud obtained from calibration and RGB supervision.

DN-Splatter [8] integrates readily available geometric cues to regularize the 3D Gaussian Splatting optimization process. Both depth and normal priors are employed, coming from sensors present in modern devices or off-the-shelf monocular networks.

3. Methodology

Following the approach in [8], we incorporate depth- and normal-based regularization into our pipeline through the use of pretrained monocular priors. We then integrate over the rendered color and depth maps to extract meshes.

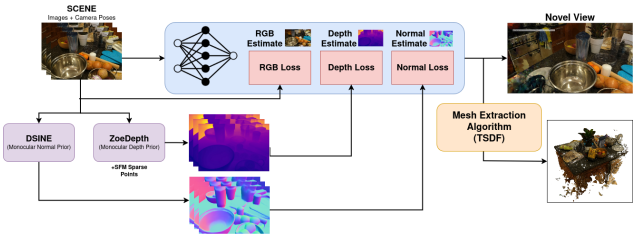

Figure 2. FoamSurf Pipeline

3.1. Depth estimation

Both depth and normal estimation are directly integrated into RadFoam's CUDA raytracing pipeline for maximal compute efficiency. We leverage the same piecewise-constant volume rendering formulation used for color. For each camera ray r traversing N Voronoi cells, let σ_n be the density of cell n, δ_n the ray's path length through that cell, and d_n the exit depth. let

$$d_n = \tfrac{t_n^{(0)}+t_n^{(1)}}{2}$$

be the midpoint depth (where $t_n^{(0)}$ and $t_n^{(1)}$ are the entry and exit distances along the ray). We define

$$\alpha_n = 1-\exp(-\sigma_n\,\delta_n), \quad T_n = \prod_{j=1}^{n-1}(1-\alpha_j), \quad w_n = \alpha_n\,T_n.$$

Forward pass

$$\hat{D}(r) \;=\; \sum_{n=1}^{N} w_n\,d_n. \tag{1}$$

Backward pass

Let

$$g_D \;=\; \frac{\partial L}{\partial \hat{D}(r)}, \qquad D_{\text{rest}} = \frac{\hat{D}(r) - \sum_{k\le n} w_k\,d_k}{T_n\,(1-\alpha_n)}.$$

Then the gradient with respect to α_n gains the additional term

$$\frac{\partial L}{\partial \alpha_n} \;+\!=\; T_n\,(d_n - D_{\text{rest}})\,g_D \tag{2}$$

This $\partial L/\partial\alpha_n$ is then back-propagated through $\alpha_n = 1 - e^{-\sigma_n\,\delta_n}$ into σ_n and δ_n. Additionally, each segment's midpoint depth $d_n = \frac{t_n^{(0)}+t_n^{(1)}}{2}$ contributes

$$\frac{\partial L}{\partial t_n^{(0)}} \;+\!=\; \tfrac{1}{2}\,w_n\,g_D, \quad \frac{\partial L}{\partial t_n^{(1)}} \;+\!=\; \tfrac{1}{2}\,w_n\,g_D,$$

which are then scattered into the point positions.

3.2. Normal Estimation

Surface normals are estimated by accumulating weighted segment directions. For each cell crossing between points $\mathbf{p}_n$ and $\mathbf{p}_{n+1}$, define

$$\mathbf{f}_n = \frac{\mathbf{p}_{n+1}-\mathbf{p}_n}{\|\mathbf{p}_{n+1}-\mathbf{p}_n\|}, \quad w_n = \alpha_n\,T_n.$$

Forward pass

$$\mathbf{n}_{\text{num}} \;=\; \sum_{n=1}^{N} w_n\,\mathbf{f}_n, \quad \hat{\mathbf{n}} \;=\; \frac{\mathbf{n}_{\text{num}}}{\|\mathbf{n}_{\text{num}}\|}. \tag{3}$$

Backward pass

Let

$$\mathbf{g}_n = \frac{\partial L}{\partial \hat{\mathbf{n}}}, \qquad \mathbf{n}_{\text{rest}} = \frac{\hat{\mathbf{n}} - \sum_{k\le n} w_k\,\mathbf{f}_k}{T_n\,(1-\alpha_n)}.$$

Then the gradient with respect to α_n gains the normal-supervision term:

$$\frac{\partial L}{\partial \alpha_n} \;+\!=\; T_n\,\big(\mathbf{f}_n - \mathbf{n}_{\text{rest}}\big)\cdot\mathbf{g}_n. \tag{4}$$

Additionally, consider points $\mathbf{p}_n$ and $\mathbf{p}_{n+1}$ and define

$$\mathbf{raw}_n = \mathbf{p}_{n+1} - \mathbf{p}_n$$

and let $\mathbf{g}_{\mathbf{f}_n} = w_n\,\frac{\partial L}{\partial \hat{\mathbf{n}}}$ be the upstream gradient w.r.t. $\mathbf{f}_n$. Then the contribution of the normal-supervision loss to the raw vector is

$$\frac{\partial L}{\partial \mathbf{raw}_n} = \frac{\mathbf{g}_{\mathbf{f}_n}}{\|\mathbf{raw}_n\|} \;-\; \mathbf{raw}_n\,\frac{\mathbf{raw}_n^{\top}\,\mathbf{g}_{\mathbf{f}_n}}{\|\mathbf{raw}_n\|^3}.$$

Finally, this splits into point-position gradients as

$$\frac{\partial L}{\partial \mathbf{p}_{n+1}} \;+\!=\; \frac{\partial L}{\partial \mathbf{raw}_n}, \qquad \frac{\partial L}{\partial \mathbf{p}_n} \;-\!=\; \frac{\partial L}{\partial \mathbf{raw}_n}.$$

3.3. Monocular priors

Depth regularization: Since the MipNerf360 dataset does not contain any depth sensor data, we employ *ZoeDepth* [2] to obtain pseudo-ground truth depth maps. Due to the scale ambiguity of monocular predictions, we align the estimated depth maps with the sparse pointcloud given by COLMAP, by solving the closed-form linear regression problem

$$\hat{a}, \hat{b} = \arg\min_{a,b} \sum_{i,j} \|(a * D_{\text{mono},i,j} + b) - D_{\text{sparse},i,j}\|_2^2, \tag{5}$$

where the parameters a, b are scale and shift respectively.

Normal regularization: We obtain pseudo-ground truth normal maps using *DSINE* [1], and convert them from world space to tangent space to match the output of our network during training. To reduce computational overhead, both depth and normal priors are precomputed offline for all samples in the dataset.

3.4. Optimization

We optimize using a composite loss function comprising three different terms. First, we use the Huber loss for color supervision

$$\mathcal{L}_{\hat{\mathbf{C}}} = \begin{cases} 0.5\,(C-\hat{C})^2 & \text{if } |C-\hat{C}| < 1 \\ |C-\hat{C}| - 0.5 & \text{else.} \end{cases} \tag{6}$$

Then, we regularize depth using the gradient-aware depth loss proposed in [8]

$$\mathcal{L}_{\hat{D}} = g_{\text{rgb}} \frac{1}{|\hat{D}|} \sum \log\left(1 + \|\hat{D} - D\|_1\right), \tag{7}$$

where $g_{rgb} = \exp(-\nabla I)$ is the gradient of the color image. The last term is the cosine similarity loss, which we use for normal regularization

$$\mathcal{L}_{\hat{N}} = \frac{1}{|\hat{N}|} \sum \left(1 - \frac{\hat{N} \cdot N}{\|\hat{N}_2\| \|N_2\|}\right). \tag{8}$$

Since batches are made of randomly sampled rays, we modify the dataloader so that it includes the depth and normal "ground truths", as well as the pre-computed rgb gradient of the corresponding pixel.

Additionaly, the original work applies a regularization term on the distribution of contribution to the volume rendering integral along the ray given by

$$\mathcal{L}_{\text{quantile}} = \mathbb{E}_{t_1,t_2 \sim \mathcal{U}[0,1]} \left[|W^{-1}(t_1) - W^{-1}(t_2)| \right], \tag{9}$$

where W^{-1} is the quantile function. This term, similar to our depth regularization, encourages volumes to accumulate near the surfaces of the scene. The final loss is given by

$$\mathcal{L} = \mathcal{L}_{\hat{C}} + \lambda \mathcal{L}_{\text{quantile}} + \lambda_d \mathcal{L}_{\hat{D}} + \lambda_n \mathcal{L}_{\hat{N}}. \tag{10}$$

For our experiments, we use $\lambda = 1e{-}4$, $\lambda_d = 0.2$, and $\lambda_n = 0.05$.

3.5. Evaluation setup

We evaluate our method on scenes from the *MipNeRF360* and *NeRF Synthetic* datasets. Specifically, all experiments are conducted on the *counter* and *lego* scenes, using an RTX 5080 GPU, training for 20,000 iterations and limiting the final number of points to 524,288, one-fourth of the amount used in the original *RadFoam* paper. For the *counter* scene, we train with a downsampling factor of 8 for the first 5,000 iterations, then switch to a factor of 4 for the remainder.

For the *lego* scene, we train on the original resolution for the complete run. The synthetic dataset provides ground truth maps, which we employ for depth supervision, as well as for evaluating rendered outputs. Additionally, a reference mesh can be obtained through *Blender*, allowing for quantitative evaluation of the extracted meshes.

3.6. Mesh extraction

After optimization, we render depth maps for all views and integrate them using the Truncated Signed Distance Function (TSDF) fusion method from Open3D [11]. Notably, TSDF expects depth values measured from the image-plane, but our network outputs depth estimates from the camera center. To address this, we convert the networks depth outputs to image-plane using the pinhole camera model

$$d_z = \frac{d_{\text{ray}}}{\sqrt{\left(\frac{u-c_x}{f_x}\right)^2 + \left(\frac{v-c_y}{f_y}\right)^2 + 1}}, \tag{11}$$

where c_x, c_y, f_x, f_y are the camera intrinsics, and u, v the pixel coordinates. Omitting this conversion leads to radial distortion and poor meshes, as visible in Figure 9.

4. Results

4.1. Ablation

As an ablation and to quantify the individual contributions of color, depth, and normal supervision, we evaluate six loss variants on the MipNeRF360 dataset. Our metrics are peak signal-to-noise ratio (PSNR), structural similarity (SSIM), depth root-mean-square-error (RMSE_d), and normal cosine-similarity (CosSim_n) all evaluated at test time. Table 2 summarizes the results. In line with expectations, both PSNR and SSIM decrease with the introduction of depth/normal supervision. Furthermore, RMSE_d and CosSim_n are lowest when trained on their corresponding losses only. Lastly, changing depth supervision from the initial $L_{\hat{D}}$ to a simple log-scaled depth loss still yields acceptable depth supervision without overly damaging PSNR/SSIM.

Table 1. Ablation (MipNerf360)

Loss variant	PSNR	SSIM	RMSE_d	CosSim_n
$L_{\hat{C}}$	**28.74**	**.8807**	1.272	52.07
$L_{\hat{C}} + L_{\hat{D}}$	21.91	.7087	**.9628**	50.47
$L_{\hat{C}} + L_{\hat{N}}$	21.30	.6096	27.391	**29.35**
$L_{\hat{C}} + L_{\hat{D}} + L_{\hat{N}}$	21.87	.6436	1.195	36.21
$L_{\hat{C}} + L_{\hat{D}} \rightarrow logL_1$	23.86	.7882	1.007	49.83

4.2. 2DGS Comparison

To evaluate our model's performance in depth estimation and mesh quality, we perform both quantitative and visual comparisons against RadFoam and 2DGS [5] for the Lego synthetic scene. For a fairer comparison, we also run 2DGS with a limit on the maximum number of Gaussians proportional to the final point limit in our pipeline. As shown in the meshes in Figure 3, there is a clear improvement in surface

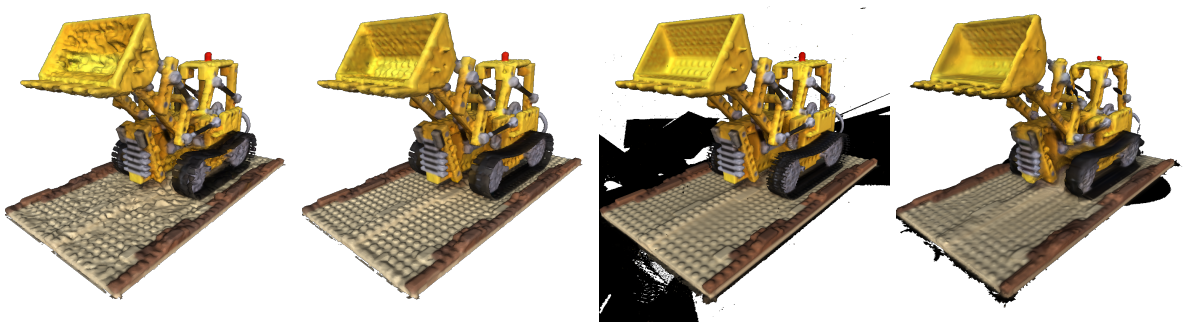

Figure 3. Meshes extracted via TSDF integration applied to rendered color and depth maps produced by four different methods. From left to right: Radiant Foam, FoamSurf, 2DGS, and downsampled 2DGS.

quality from RadFoam to FoamSurf, especially noticeable on the ground and bucket areas. Both 2DGS reconstructions have black artifacts but still manage a good overall reconstruction of the lego object. Notably, the downsampled 2DGS version loses some finer details that our model is able to preserve.

Quantitative results in Table 2 show that our regularization on the geometry negatively affects the PSNR, which is to be expected. However, our method using the edge-aware loss improves RMSE and MAE for depth estimation when compared to base RadFoam and FoamSurf with LogL1 loss for regularization. This in turn translates to better mesh accuracy, as evidenced by the lower Chamfer-L_1 distance. Compared to 2DGS, our method achieves slightly worse RMSE but considerably better MAE. This suggests that our method has more large outliers in the depth maps, but fewer consistent small errors than 2DGS, which can be seen by their meshes loosing finer details such as the lego floor tiles. Lastly, while we achieve a better Chamfer-L_1 distance, this is most likely because of the black artifacts present in the 2DGS meshes.

While a quantitative analysis of the counter scene is not possible due to the lack of ground truth data, a qualitative comparison of the meshes can be seen in Figure 10. Unlike the synthetic scene supervised with ground truth data, the pseudo-ground truth estimates seem to introduce numerous artifacts, especially in the background of each view. This can be attributed to a missmatch in the alignments of the depth estimates obtained from the monocular prior, which in turn causes floating artifacts to appear in the mesh when performing TSDF.

5. Conclusion

5.1. Limitations and future work

Due to limited compute resources more extensive scenes such as in ScanNet are yet to be trained but would give more insights into the real world usability of our method. Furthermore, we were not able to implement DN-Splatters

Table 2. Depth estimation and mesh evaluation

Method	PSNR	RMSE	MAE	Chamfer
Radfoam	28.50	.1631	.0338	0.018
FoamSurf (LogL1)	27.44	.1395	.0198	0.016
FoamSurf (Edge aware)	24.34	.1088	**.0161**	**0.014**
2DGS	**37.75**	**.0903**	.0674	0.281
2DGS (downsampled)	29.32	.0984	.0728	0.022

prior on the total variation of predicted normals, encouraging smooth normal predictions at neighboring pixels due to the nature of the pipeline which returns uncoordinated raybatches for regular training iterations. Given the highly oscillating nature of the normals loss such a regularizer would, however, prospectively improve normal training which would warrant a refactoring of the training pipeline for this specific use case.

The quality of depth priors directly affects the accuracy of meshes, as seen in the difference between models trained with monocular priors versus those trained with ground truth data. Improvements could be made by employing recent methods for pseudo-ground truth generation, such as *DepthAnythingV2* [10] and *VGGT* [9].

5.2. Conclusion

FoamSurf extends the Radiant Foam framework by incorporating off-the-shelf depth and normal monocular priors for more accurate surface reconstruction. Our method successfully integrates these priors in the optimization process, leading to improved multiview depth map estimation. For scenes with available ground truth depth data, we are able to extract high fidelity meshes, competitive with state of the art methods like 2DGS. Scenes that rely on pseudo-ground truth depths however, are affected by misalignments in the monocular priors, which can cause artifacts to appear during mesh extraction. Addressing these limitations presents a promising direction for future work in extracting quality meshes from real scenes using FoamSurf.

References

[1] Gwangbin Bae and Andrew J. Davison. Rethinking inductive biases for surface normal estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[2] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 2

[3] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. *arXiv:2502.01157*, 2025. 1

[4] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 1

[5] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 1, 3

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1

[7] 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 *ECCV*, 2020. 1

[8] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. 1, 2, 3

[9] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 4

[10] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 4

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

FoamSurf: High-Quality Surface Reconstruction Using RadFoam

Supplementary Material

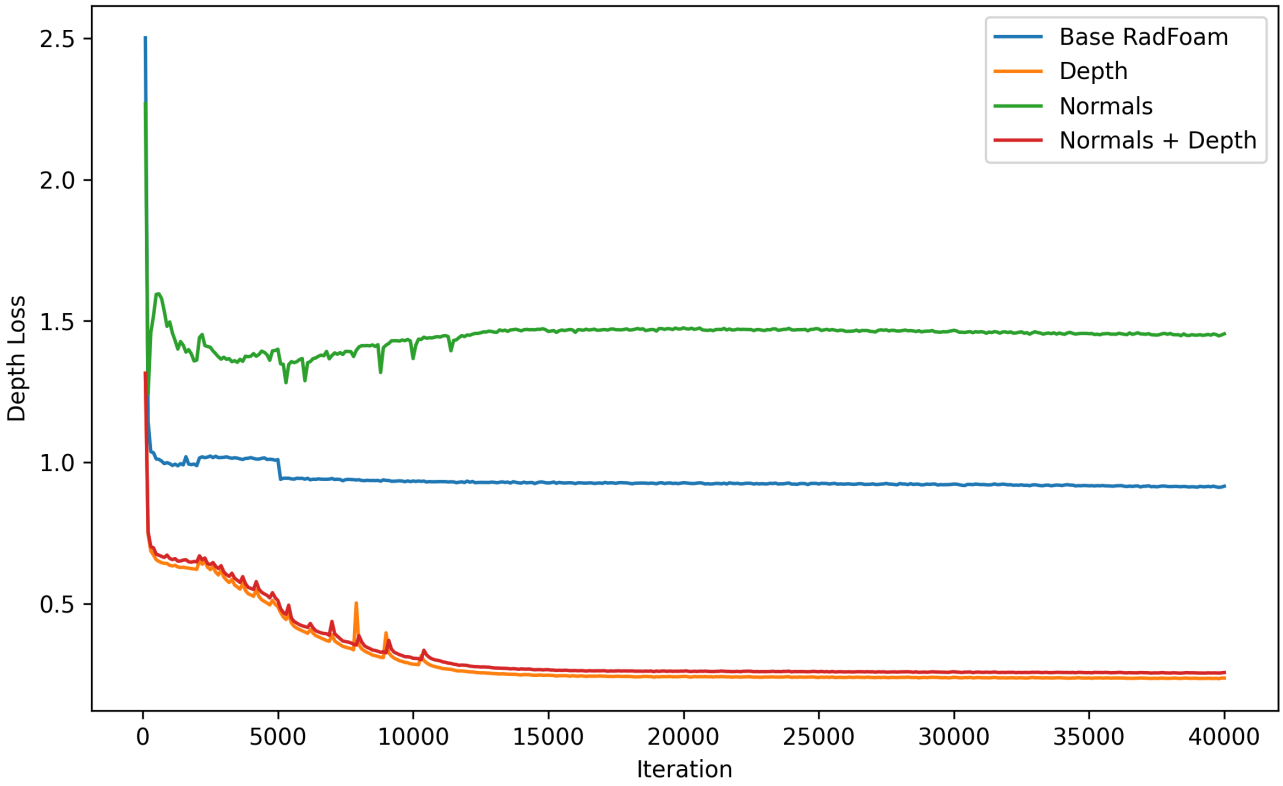

Figure 4. Ablation on gradient-aware depth loss.

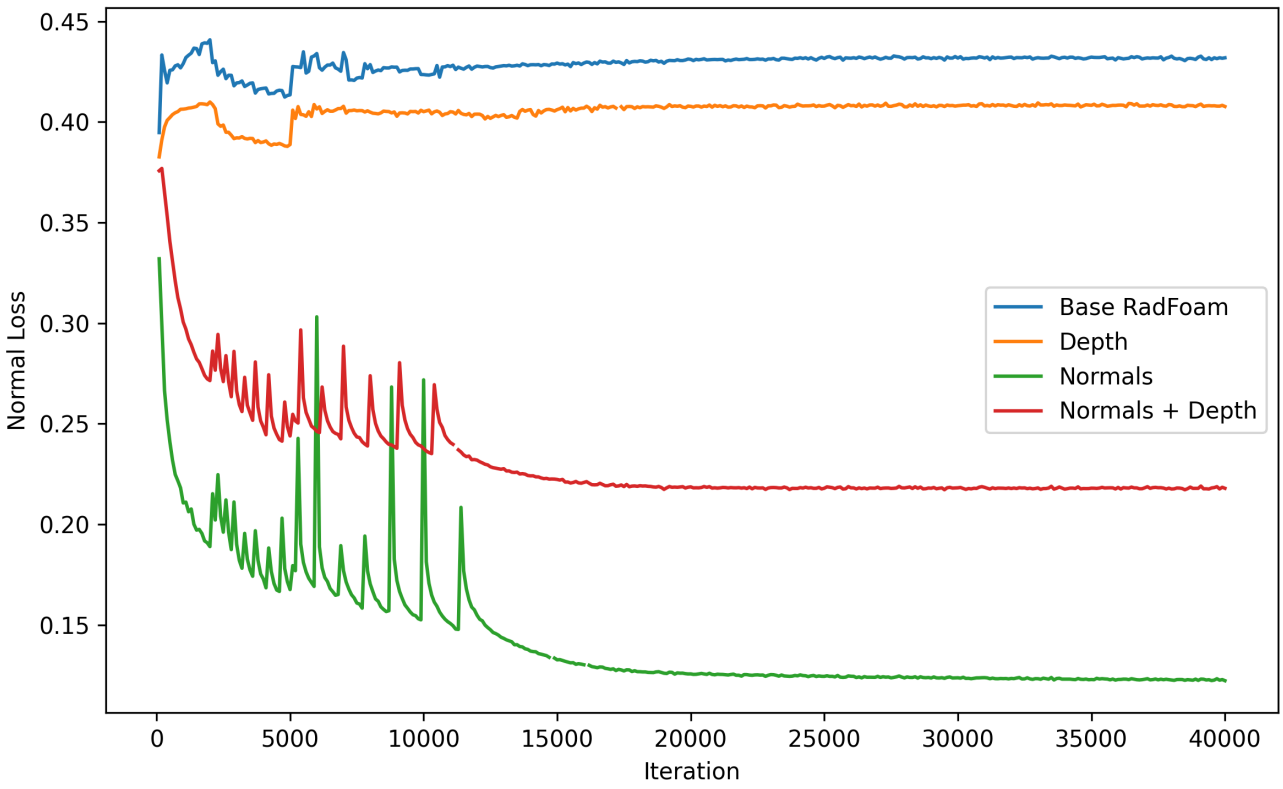

Figure 5. Ablation on normal-supervision loss.

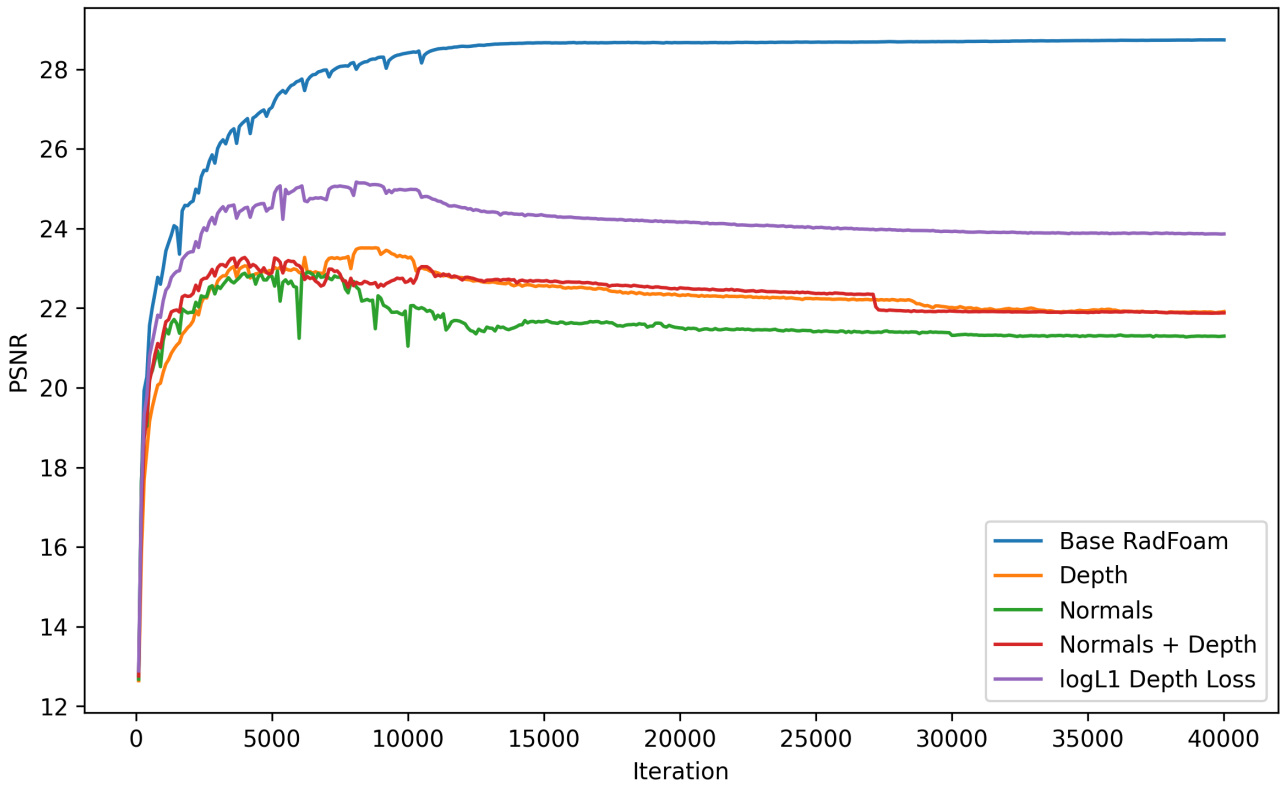

Figure 6. Ablation on PSNR.

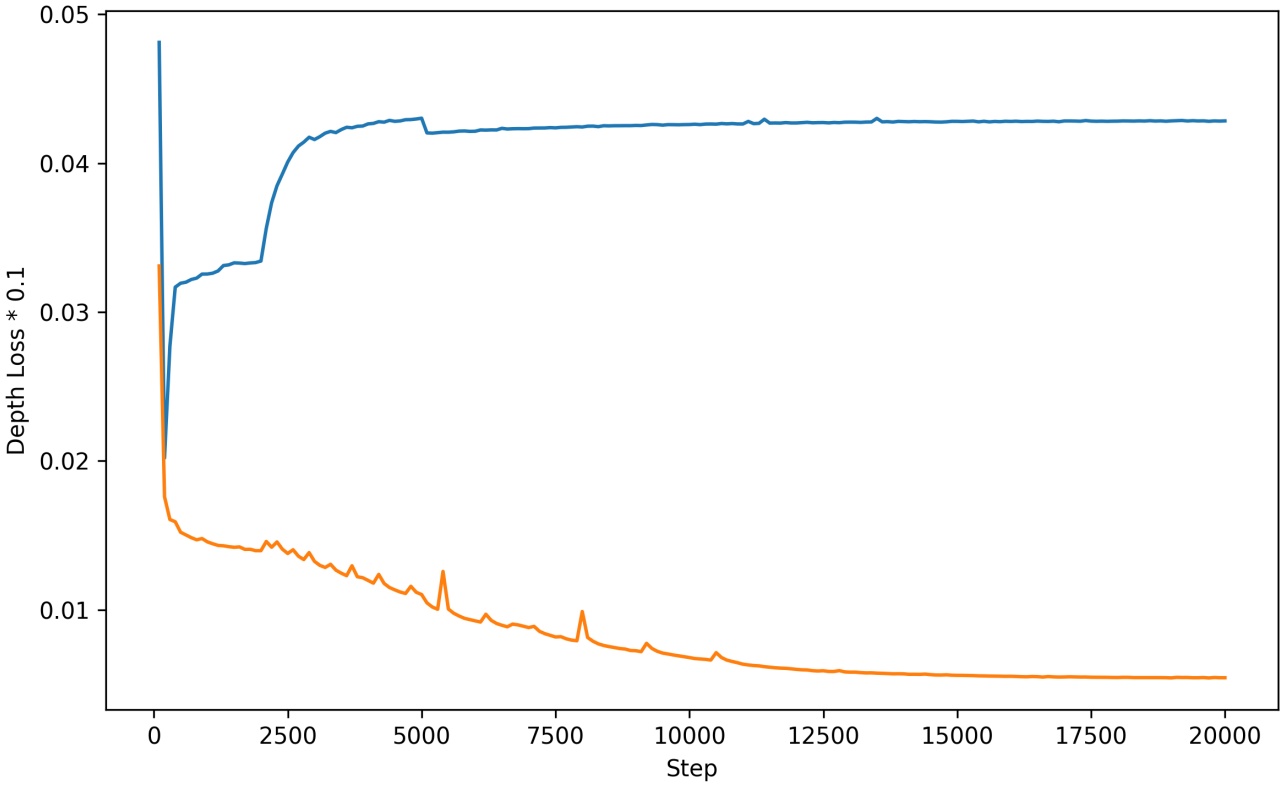

Figure 7. The initial CUDA backward pass was missing the transmissivity scaling term.

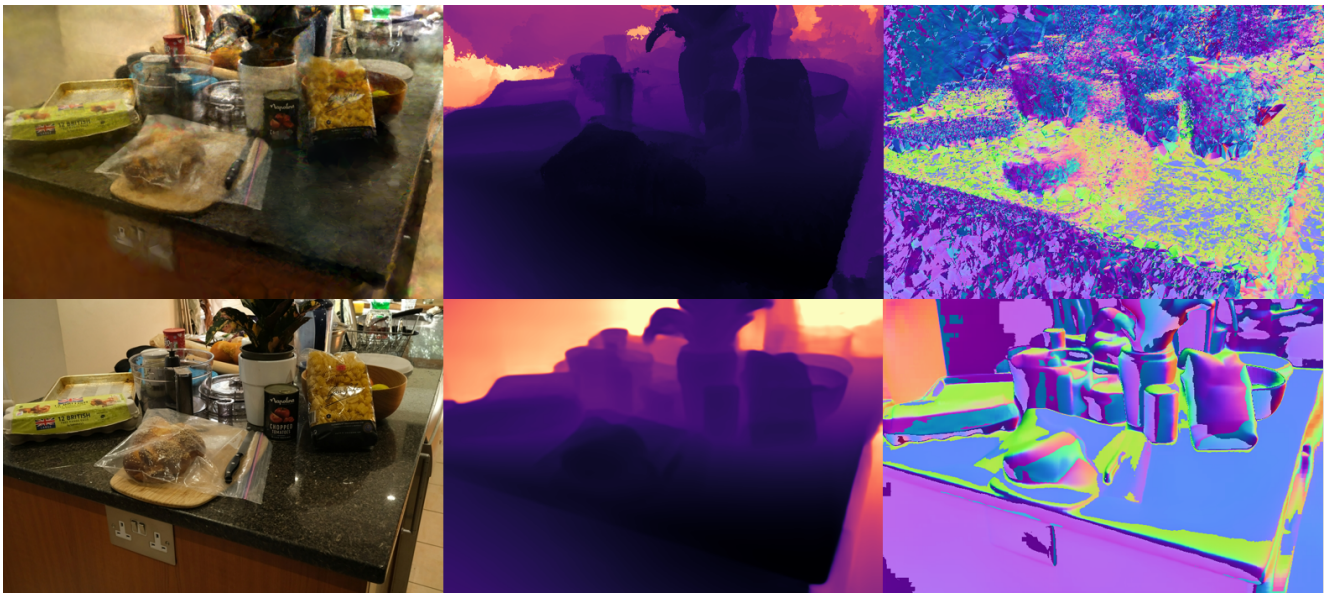

Figure 8. Full size image: Full model implementation. Top row: RadFoam novel-view render (left), predicted depth map (center), predicted normals (right). Bottom row: corresponding ground truths.

Figure 9. Effects of not converting depth estimates to image-plane coordinates on meshing.

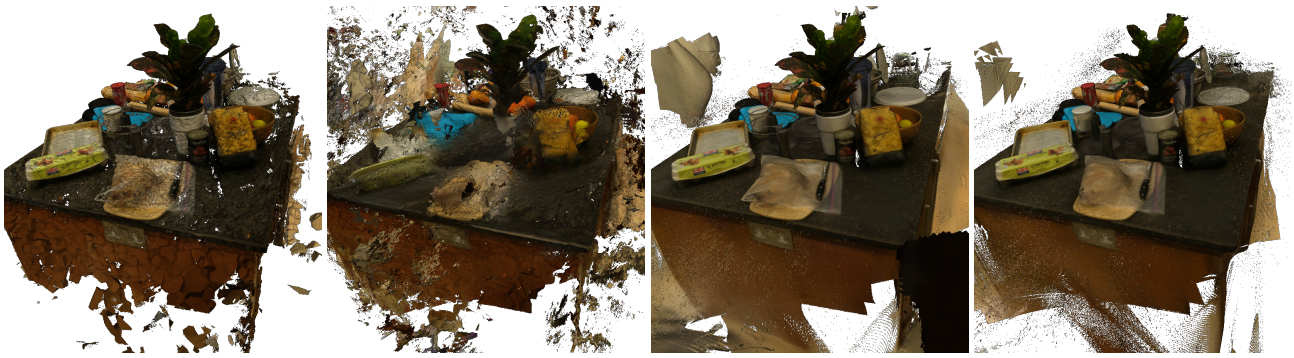

Figure 10. Meshes extracted from counter scene via TSDF integration applied to rendered color and depth maps produced by four different methods. From left to right: Radiant Foam, FoamSurf, 2DGS, and downsampled 2DGS.