


SparseCraft: Few-Shot Neural Reconstruction through Stereopsis Guided Geometric Linearization

—Supplementary Material—

Mae Younes^{†*} , Amine Ouasfi^{*}, and Adnane Boukhayma

Inria, [†]Univ. Rennes, CNRS, IRISA, M2S, France

1 Additional Experiments and Results

Additional qualitative reconstruction comparisons We provide additional qualitative comparison for the reconstruction task for both datasets DTU and Tanks and Temples in Figures 1. For DTU (3 input views), we show meshes generated by NeuS [14], per-scene fine-tuned SparseNeuS [8], VolRecon [11] and ours. In most cases, NeuS struggles to generate complete and meaningful reconstructions. SparseNeuS-ft lacks details and is overly smooth. VolRecon reconstructions can be incomplete and noisy, although having more details than SparseNeuS. Our method demonstrates the best performance in terms of surface details and completeness.

Novel view synthesis comparison on full-images We report in Table 1 quantitative results for the full images on the DTU dataset. Our method outperform per-scene optimization methods in rendering when evaluating on Full-Images in almost all settings. Our method only ranks second in 3 input-view setting in terms of PSNR compared to the generalizable NeRF method PixelNeRF [21] most likely because it benefits from its data prior when dealing with the background of DTU scenes.

Additional qualitative novel view comparisons We provide additional qualitative comparisons between our method and SOTA methods FreeNeRF [18] and RegNeRF [10] in novel view synthesis in the 3 input view setting (Figure 2) and the 6 input view setting (Figure 3).

Additional qualitative novel view results We provide rendering examples of all 15 scans of the DTU dataset for the novel view synthesis task under the 3 input-view setting in Figure 4.

2 Additional Ablation Studies

SfM vs. MVS point cloud We remind that one main motivation to exploit MVS for sparse views neural reconstruction is that it takes **only 41 seconds for**

* These authors contributed equally to this work

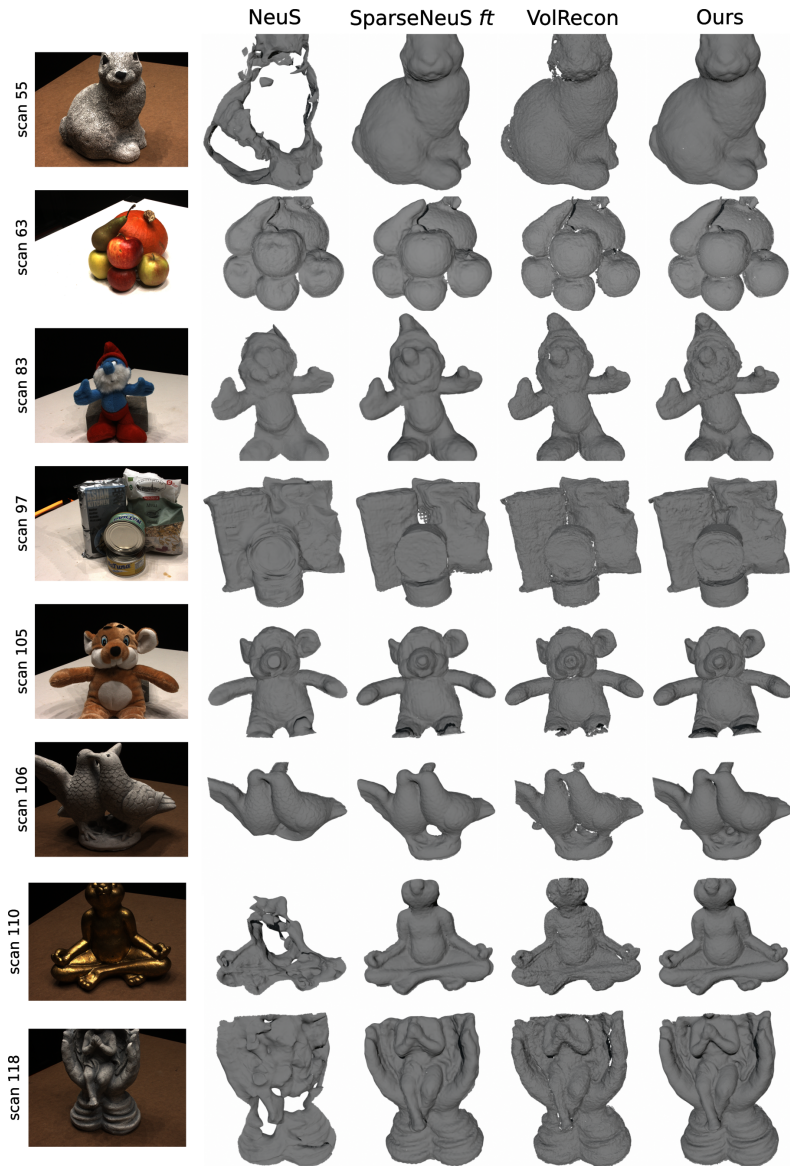


Fig. 1: Additional qualitative comparison of surface reconstructions on DTU from 3 views. **SparseNeus** and **VolRecon** use data priors, whereas we do not.

3 views, and 180 seconds for 9 views in DTU. In terms of chamfer distance, reconstruction performance drops by 45% in DTU if we use SfM instead of MVS (Ours SfM 1.47, Ours MVS 1.01 ↓). MVS carries richer supervision (pts, colors & normals), and SfM point clouds are 21 times sparser than MVS in average for

Full-Image DTU Evaluation	Full-Image PSNR \uparrow			Full-Image SSIM \uparrow			Full-Image LPIPS \downarrow			Full-Image Average \downarrow		
	3 views	6 views	9 views	3 views	6 views	9 views	3 views	6 views	9 views	3 views	6 views	9 views
SRF [2]	15.84	17.77	18.56	0.532	0.616	0.652	0.482	0.401	0.359	0.207	0.162	0.145
PixelNeRF [21]	18.74	21.02	22.23	0.618	0.684	0.714	0.401	0.340	0.323	0.154	0.119	0.105
MVSNeRF [1]	16.33	18.26	20.32	0.602	0.695	0.735	0.385	0.321	0.280	0.184	0.146	0.114
SRF ft [2]	16.06	18.69	19.97	0.550	0.657	0.678	0.431	0.353	0.325	0.196	0.143	0.125
PixelNeRF ft [21]	17.38	<i>21.52</i>	21.67	0.548	0.670	0.680	0.456	0.351	0.338	0.185	0.121	0.117
MVSNeRF ft [1]	16.26	18.22	20.32	0.601	0.694	0.736	0.384	0.319	0.278	<i>0.185</i>	0.146	0.113
DietNeRF [3]	10.01	18.70	22.16	0.354	0.668	0.740	0.574	0.336	0.277	0.383	0.149	0.098
RegNeRF [10]	15.33	19.10	<i>22.30</i>	<i>0.621</i>	<i>0.757</i>	<i>0.823</i>	<i>0.341</i>	<i>0.233</i>	<u>0.184</u>	0.189	<i>0.118</i>	<i>0.079</i>
FreeNeRF [18]	<i>18.02</i>	22.39	<u>24.2</u>	<u>0.68</u>	<u>0.779</u>	<u>0.833</u>	<u>0.318</u>	<u>0.24</u>	<i>0.187</i>	<u>0.146</u>	<u>0.094</u>	<u>0.068</u>
Ours (SculptoSparse)	<u>18.37</u>	<u>22.37</u>	25.00	0.740	0.833	0.880	0.258	0.179	0.148	0.136	0.052	0.058

Table 1: Quantitative comparison on DTU. We present the PSNR, SSIM, VGG LPIPS and Average scores of full images. Best scores are in **bold**, second best are underlined and third best are in *italic*.

Downsampling ratio	Chamfer distance \downarrow
No downsampling	1.01
by x2	1.08
by x4	1.10
by x8	1.14

Table 2: Effect of uniformly downsampling the MVS point cloud leveraged by our method. We ablate on one set of 3 views for all scenes in DTU for reconstruction. Our method is only affected slightly by sparser MVS point clouds.

the DTU reconstruction benchmark.

MVS point cloud density In this study, we want to see to what extent our method relies on the density of the leveraged MVS point cloud. To this end, we uniformly downsample the MVS point cloud in all scenes by factors ranging from 2 to 8. We report results in Table 2. Downsampling the point cloud only seems to affect our method slightly, as the quality of reconstructions remains relatively good. This shows that our method is not overly relying on the density of the MVS point cloud, thanks in part to our progressive hash and Taylor based regularizations. We note that, as witnessed by numerical results in Table 1 in the main paper, in the DTU reconstruction benchmark, we achieved large improvements over SOTA on shiny scenes (110, 37), and outperformed SOTA even in scene (83) with least dense MVS (80% less than the DTU average point cloud size).

Additional qualitative ablation results We show more qualitative ablations on additional DTU scenes for the different components of our method in Figure 5 as well as a comparison of our Taylor based losses against baseline losses in Figure 6.

Full view ablation Figure 7 shows an ablation of our losses while densifying views, using DTU scene 114. Performance improves with more views. The contribution of our regularizations reduces the denser the views, as the baseline



Fig. 3: Qualitative comparison of novel views on DTU from 6 views.

Varying the number of input views We perform an experiment to see the performance of our method in surface reconstruction when varying the number of views from 2 to 6. Please note that for a meaningful comparison, we evaluate meshes after cleaning with the masks of 6 views. We evaluate on one set of 3 views for all scenes of DTU, and we report their average in Table 3. Our method shows steady improvement when increasing the number of views. We also found that our method can provide encouraging results even under the very extreme case of using only 2 input views.

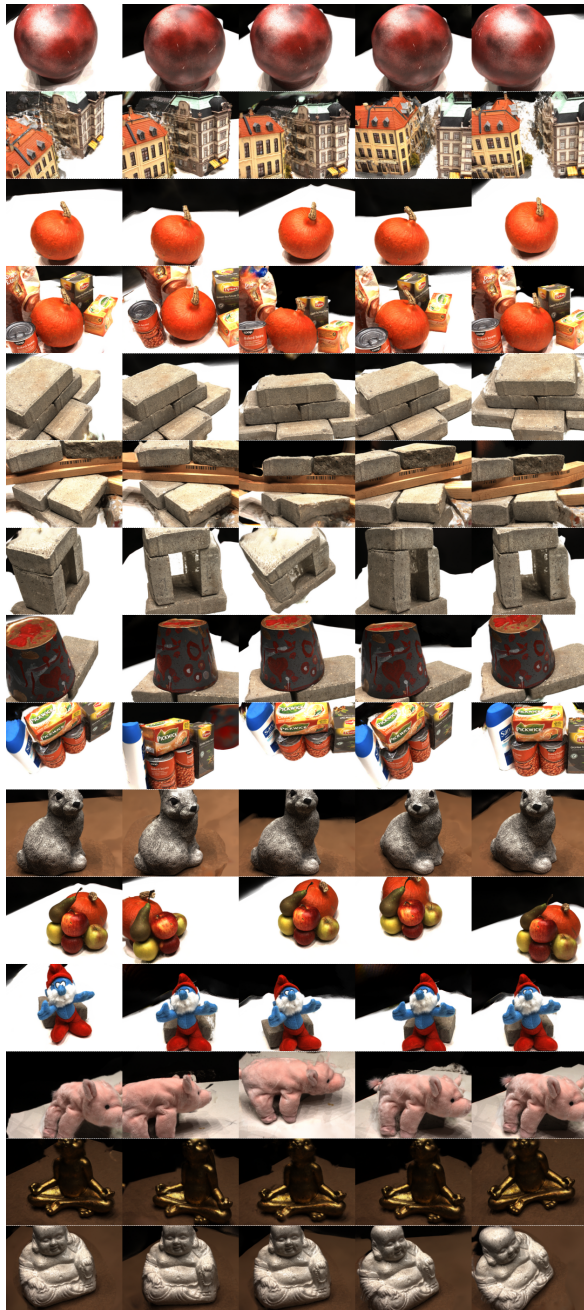


Fig. 4: Examples of novel views generated with our method from 3 input views in DTU.




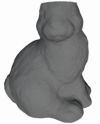
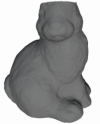
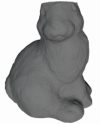







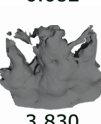





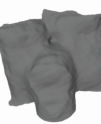
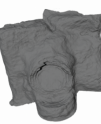
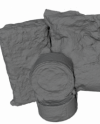
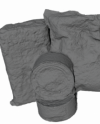
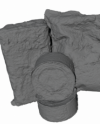


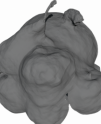
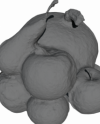
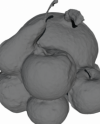
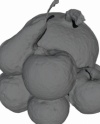
	No reg.	+ Prog. encoding	+ Input Taylor	+ MVS color	+ Query Taylor
scan 55 	 1.543	 0.834	 0.747	 0.739	 0.715
scan 122 	 6.682	 1.405	 0.902	 0.896	 0.872
scan 69 	 3.830	 1.747	 0.953	 0.896	 0.850
scan 97 	 4.436	 3.949	 1.407	 1.382	 1.273
scan 63 	 4.239	 3.967	 1.290	 1.277	 1.122

Fig. 5: Qualitative ablation of different components of our method. We report **Chamfer** scores of reconstructions.

# of views	Chamfer distance ↓
2 views	1.69
3 views	1.17
4 views	1.06
5 views	0.98
6 views	0.89

Table 3: Effect of increasing the number of views on reconstruction. For a meaningful comparison, we evaluate meshes after cleaning with masks of 6 views. We evaluate on one set of 3 views for all scenes of DTU, and we report their average. Our method shows steady improvement when increasing the number of views, and we can achieve a good Chamfer score even at 2 views.

Training Time when increasing input views The training of per-scene optimization based implicit reconstruction baselines (*e.g.* SOTA method in the dense setting Neuralangelo) under fully dense input views in DTU can take up to/ more than 16 hours depending on the scene. Differently, MVS supervision,

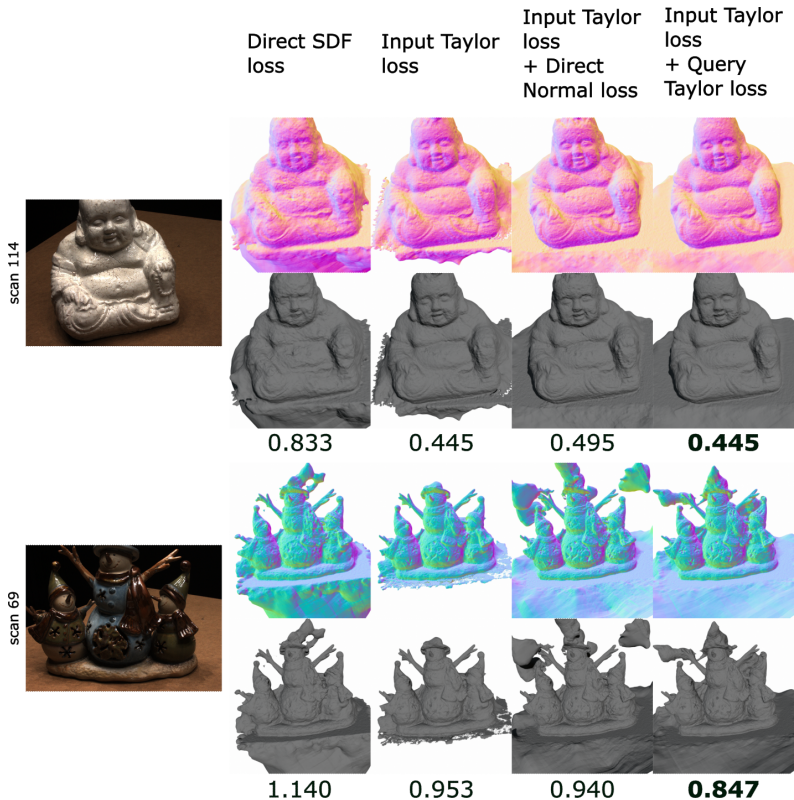


Fig. 6: Qualitative ablation of our Taylor expansion based losses, compared to applying direct standard losses. We disable the MVS color loss in this comparison. We show both the generated meshes and their surface normals. We also report **Chamfer** scores of reconstructions.

σ_ϵ values	Chamfer distance ↓
$\epsilon/2$	1.01
ϵ	1.03
$\epsilon/4$	1.08
$2 \cdot \epsilon$	1.05
Fixed 0.01	1.10

Table 4: Effect of σ used for sampling the query points around MVS point cloud. We evaluate on one set of 3 views for all scenes in DTU for reconstruction. As we use a progressive strategy that involves using progressive hash encoding as well as progressive numerical gradient step ϵ , we find that it is better to use a progressive sampling standard deviation that is proportionate to the numerical gradient step rather than having it fixed during training.

which is incidentally highly informative and less noisy under such dense settings,

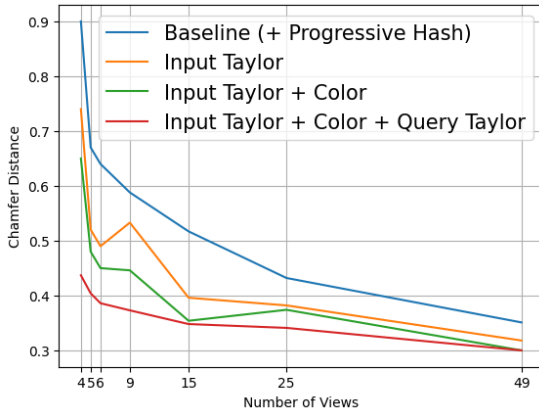


Fig. 7: Ablation of our losses under increasing number of views for DTU scan 114.

allows us to maintain very practical training times. In our experiments under full DTU views (49 images), our model converges within 40 to 50 minutes. We note that MVS running time is about 19 minutes in this setting. Hence, our accumulated overall compute time is drastically lower than NeuralAngelo’s. We remind that the focus of our work is addressing the few-shot setting. Under 3 views, MVS running time is roughly 41 seconds and training time is about 10 minutes in DTU.

Taylor query points sampling To make use of our proposed dual Taylor expansion based losses, we need to sample query points around the input MVS point cloud following a normal distribution centered at the MVS points, using a standard deviation σ . We sample 10 query points for each MVS point every 500 iterations using a KD tree. Query points should be at the vicinity of MVS points to enforce local linearity of the SDF function close to the surface. However, as we use a progressive strategy where gradients are estimated with numerically using a decreasing step size ϵ , sampling too close to the surface at the beginning of the training may not give the best results. We find in our experiments, as reported in Table 4, that it is better to use a sampling standard deviation that is proportionate to the numerical gradient step rather than having it fixed during training. We find a value of $\epsilon/2$ yields the best performance.

Gradients computation method We study the effect of using numerical gradients against analytical gradients, as our proposed Taylor-based losses also rely on these computations. Differently from NeuralAngelo [7] that computes numerical gradients to estimate surface normals, we additionally use gradients to enforce linear behavior of the SDF function near the surface through our proposed Taylor-based regularization. We find that using numerical gradients yields better results, as presented in Table 5.

Gradient Computation	Chamfer distance ↓
Numerical	1.01
Analytical	1.13

Table 5: Effect of using numerical gradients *vs.* analytical gradients. We evaluate on one set of 3 views for all scenes in DTU for reconstruction. We find that using numerical gradients for our learning, including our near the surface SDF linearization strategy, yields better results.

Prog. enc. scheduling	Chamfer distance ↓
80%	1.01
60%	1.04
20%	1.05

Table 6: Effect of scheduling the progressive encoding throughout the training, *i.e.* how many iterations until the finest hash level is activated. We evaluate on one set of 3 views for all scenes in DTU for reconstruction. We find that in the sparse setting of 3 input views, it is better to keep the progressive encoding for 80% of training iterations.

Progressive hash encoding scheduling Table 6 shows the impact of scheduling the hash encoding on the quality of reconstructions. We find that while our loss based regularization can lead to good surfaces, using progressive encoding acts as an additional regularization and helps to avoid artifacts in the reconstruction, by ensuring that the geometry model does not prematurely overfit to fine information. In addition, as our proposed losses are geometric in nature, they can sacrifice rendering quality at the expense of good reconstruction. Combining them with the progressive encoding leads to superior rendering quality than using only the progressive hash encoding.

3 Additional Experimental Details

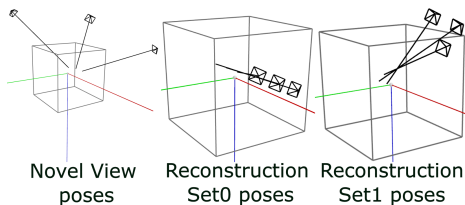


Fig. 8: Cameras in DTU benchmarks.

Camera pose distributions Following standard protocols in previous methods, we experimented with the two camera distributions of the DTU reconstruction benchmark (SparseNeus) (Set-0 & Set-1), and the DTU novel view benchmark

(RegNeRF). The baseline in the latter is wider, as illustrated in fig.8. We outperformed SOTA in both settings (see Tables 1 and 2 in the main paper). We also experimented with the 360 configurations of BMVS. The latter are very challenging for a learning prior-free method like ours. While prior-free methods shows major failures, we perform mostly on par with methods using external deep 3D priors (*e.g.* Figure 3 in the main paper).

While we use three datasets in this work, namely DTU [4], BlendedMVS [19] and Tanks and Temples [5], we adhere to two different protocols for DTU depending on the nature of the task: novel view synthesis or surface reconstruction.

DTU Dataset The DTU dataset [4] is a large-scale multi-view dataset that consists of 124 scans of different objects. Each scene has 49–64 views of resolution 1600×1200 . Generalizable methods such as PixelNeRF [21] for neural rendering and SparseNeuS [8] for surface reconstruction use a split of training scenes and test scenes to study the “pre-training & per-scene fine-tuning” setting for few-shot neural rendering and surface reconstruction respectively. Differently from these, our method does not require pre-training. We follow the protocol of [10] to optimize directly on the 15 test scenes for the novel view synthesis task, and we use another set of 15 test scenes chosen by [8] for the reconstruction task. The test scan IDs for novel view synthesis are: 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, and 114. The test scan IDs for surface reconstruction are: 24, 37, 40, 55, 63, 65, 69, 83, 97, 105, 106, 110, 114, 118 and 122.

For novel view synthesis, in each scan, the images with the following IDs (counting from “0”) are used as the input views: 25, 22, 28, 40, 44, 48, 0, 8, 13. The first 3 and 6 image IDs correspond to the input views in 3- and 6-view settings, respectively. The images with IDs 1, 2, 9, 10, 11, 12, 14, 15, 23, 24, 26, 27, 29, 30, 31, 32, 33, 34, 35, 41, 42, 43, 45, 46, 47 serve as the novel views for evaluation. The remaining images are excluded due to wrong exposure. We follow previous methods [10, 12, 13, 17, 18, 21] in using a $4 \times$ downsampled resolution, resulting in 300×400 pixels for each image.

For surface reconstruction, for each scan, there are two sets of 3 views with the following IDs used as the input views. Set-0: 23, 24 and 33, then Set-1: 42, 43 and 44 for 14 scans, except scan 37 (1, 8 and 9). We use the training views in their original resolution without downsampling.

BlendedMVS Dataset BlendedMVS [19] is a large-scale dataset for generalized multi-view stereo that consists of a variety of 113 scenes including architectures, sculptures and small objects with complex backgrounds. We use 5 challenging scenes from the low-res set, where each scene has 31–143 images captured at 768×576 . The chosen IDs for the selected scenes are : Dog : 5, 6, 8; Bear: 25, 43, 102; Man: 2, 3, 7; Sculpture: 7, 16, 17; Stone: 6, 9, 10. We use the training views in their original resolution without downsampling.

Tanks and Temples Dataset Tanks and Temples Dataset [5] includes large-scale indoor/outdoor scenes. Each scene contains 156 to 1107 high resolution images captured using a hand-held monocular RGB camera. To show that our method can handle large scale outdoor scenes, we selected 4 scenes: Family, Ignatius, Horse and Truck. These scenes are originally captured from more than 150 views. We uniformly sample 24 views and compare the result of our method to several methods used for dense views reconstruction: NeuralAngelo [7], generalizable sparse views reconstruction trained on DTU: VolRecon [11] and a sparse views per-scene optimization method leveraging deep MVS networks: S-VolSDF [16].

Metrics For the novel view synthesis task, to evaluate the PSNR scores, we assume that the maximum pixel value is 1 and use the formula $-10 \cdot \log_{10}(\text{MSE})$. Additionally, we utilize the scikit-image API¹ to compute the structural similarity index measure (SSIM) score and the pip package LPIPS to use a learned VGG model to compute the learned perceptual image patch similarity (LPIPS) score.

For the surface reconstruction task, we measure the Chamfer Distances of the predicted meshes w.r.t. ground truth point clouds of DTU scans. We use the same evaluation script used by SparseNeuS [8] that cleans the generated meshes using provided object masks, then the script that evaluates the Chamfer distance between sampled points on the generated meshes and the ground truth point cloud to output distances in both ways, then give the overall average that is usually reported. In addition, since two sets of 3 different views are used for each scan, we average the results between the two resulting meshes from each set of images and report it in the comparison as done in previous methods [8, 11].

Masked evaluation for novel view synthesis As reported by [10], due to the background evaluation bias, we follow their protocol of masked evaluation for DTU, where we use object masks and compute PSNR only within the mask. For SSIM and LPIPS, we use the masks to composite the predicted object-of-interest onto a black background before calculating the metrics.

4 Additional Implementation Details

Hash encoding Our hash resolution spans from 2^2 to 2^{11} with 32 levels, with a channel size of 2. The maximum number of hash entries of each resolution is 2^{19} . We find in our experiments that the hash encoding configuration used by NeuralAngelo [7] overfits due to the large number of parameters, especially in the few shot setting. Thus, by reducing the number of maximum hash entries and resolution range, we can avoid overfitting while vastly reducing the number of parameters as well as affording to use more hash levels.

Color network Following [7], we model the color with intrinsic decomposition into diffuse and specular components. In our case, this design allows us to lever-

¹ https://scikit-image.org/docs/stable/auto_examples/transform/plot_ssim.html

age the MVS color labels as supervision for the diffuse components. We assume here that the resulting MVS point cloud’s color is view independent, since it is aggregated from all input views. The final rendered color $C \in \mathbb{R}^3$ is the sum of the diffuse color C_d and specular color C_s : $C = \Phi(C_d + C_s)$, where Φ is the Sigmoid function used to normalize predictions into the range of 0 to 1. The albedo component predicts RGB values $C_d \in \mathbb{R}^3$ that are view independent. It gets the point encoding and geometrical features from the SDF MLP as input. The specular component predicts intensity values $C_s \in \mathbb{R}$, which are view dependent, in order to capture reflection, varying shadow, and exposure changes. The single channel design allows for flexibility in capturing specular highlights, exposure variations, and moving shadows are often intensity changes [7]. In addition to the point encoding and SDF geometrical features, the specular component is additionally conditioned on the surface normals computed with numerical gradients of the SDF network and the viewing direction encoded using spherical harmonics, differently from NeuralAngelo that uses reflection direction instead.

Training rays and samples We use a dynamic ray sampling strategy for the training of our method, where the number of training rays is initialized as 256 and increases with a momentum of 0.1 based on whether cast rays are occluded. The number of samples per ray is set to 1024. For sampling, we use a CUDA implemented occupancy grid [6] with a resolution of 256 for both the foreground and background models.

Architecture We use a learnable variance for the transparency transformation defined in [15], that is initialized with a value of 0.3. Its learning rate is set to 0.001. The SDF MLP uses 1 hidden of size 64 with softplus activation, and we use spherical initialization following prior work [7, 14, 20]. The density network for the background model is also defined as an MLP with 1 hidden layer of size 64, with ReLU activation, and its color network is an MLP with 3 hidden layers of size 64. Overall, the number of trainable parameters of our model is about 20M, as opposed to the 300M+ parameters used in NeuralAngelo.

Training protocol We start the optimization with the coarsest level and enable the next one every 100 iterations when the exponentially decreasing step size ϵ equals its grid cell size. When inactive, the hash level feature is set to zero. We use AdamW [9] and train for 4000 iterations for 3 input views. We scale this number proportionally to the number of views for 6 and 9 inputs. We use a learning rate of 1×10^{-2} , and we decay it by 10 after 2000 iterations. We set loss weight as: $\mathcal{L}_{\text{eik}} : 0.1$, $\mathcal{L}_{\text{tay}}(\mathbf{p}) : 0.5$. For losses $\mathcal{L}_{\text{tay}}(\mathbf{p})$ and $\mathcal{L}_{\text{col}}(\mathbf{q})$, we increase the weight linearly from 0.01 to 1. We found that using strong MVS supervision at the coarse stage of the model leads to unstable optimization, and that we benefit more from this loss at the finer stages. Query points (\mathcal{Q}) used for Taylor based losses are sampled every 500 iterations using a Kd-Tree of the MVS point cloud with $\sigma_\epsilon = \epsilon/2$. Similarly to [7, 14], we use the NeRF++ model [22]. The background uses a hash whose resolutions span from 2^2 to 2^6 with 16 levels.

Data preparation For the reconstruction task experiments, we use the preprocessed data provided by NeuS [14] which includes input images, object masks, calibration data and scaling matrices that transform the object of interest to be within a unit sphere. This applies to both DTU [4] and BlendedMVS [19]. For Tanks and Temples dataset [5] we use the official dataset images and run calibration using COLMAP on the selected 24 images.

For novel view synthesis, we use the data provided by PixelNeRF [21] and reprocess them to be in the same format used by NeuS [14]. To get the MVS point cloud that we leverage in our method, we use the official command line interface of COLMAP by sequentially running: `feature_extractor`, `exhaustive_matcher`, `image_undistorter`, `patch_match_stereo`, `stereo_fusion` on the input views, using the available camera matrices as part of the standard evaluation benchmarks.

References

1. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)
2. Chibane, J., Bansal, A., Lazova, V., Pons-Moll, G.: Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7911–7920 (2021)
3. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: Semantically consistent few-shot view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5885–5894 (2021)
4. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanaes, H.: Large scale multi-view stereopsis evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 406–413 (2014)
5. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)* **36**(4), 1–13 (2017)
6. Li, R., Gao, H., Tancik, M., Kanazawa, A.: Nerfacc: Efficient sampling accelerates nerfs. arXiv preprint arXiv:2305.04966 (2023)
7. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8456–8465 (2023)
8. Long, X., Lin, C., Wang, P., Komura, T., Wang, W.: Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In: Proceedings of the European conference on computer vision (ECCV). pp. 210–227 (2022)
9. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
10. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5480–5490 (2022)

11. Ren, Y., Wang, F., Zhang, T., Pollefeys, M., Süssstrunk, S.: Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction. arXiv preprint arXiv:2212.08067 (2022)
12. Seo, S., Chang, Y., Kwak, N.: Flipnerf: Flipped reflection rays for few-shot novel view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 22883–22893 (October 2023)
13. Seo, S., Han, D., Chang, Y., Kwak, N.: Mixnerf: Modeling a ray with mixture density for novel view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20659–20668 (June 2023)
14. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
15. Wang, Y., Skorokhodov, I., Wonka, P.: Hf-neus: Improved surface reconstruction using high-frequency details. *Advances in Neural Information Processing Systems* **35**, 1966–1978 (2022)
16. Wu, H., Graikos, A., Samaras, D.: S-volsdf: Sparse multi-view stereo regularization of neural implicit surfaces. arXiv preprint arXiv:2303.17712 (2023)
17. Wynn, J., Turmukhambetov, D.: Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4180–4189 (2023)
18. Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8254–8263 (2023)
19. Yao, Y., Luo, Z., Li, S., Zhang, J., Ren, Y., Zhou, L., Fang, T., Quan, L.: Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1790–1799 (2020)
20. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems* **34**, 4805–4815 (2021)
21. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)
22. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)