

Surface Normal Clustering for Implicit Representation of Manhattan Scenes

Supplementary material

Nikola Popovic¹, Danda Pani Paudel^{1,2}, Luc Van Gool^{1,2}

¹Computer Vision Laboratory, ETH Zurich, Switzerland

²INSAIT, Sofia University, Bulgaria

{nipopovic, paudel, vangool}@vision.ee.ethz.ch

A. Supplementary Overview

This supplementary material provides additional details, which complement the main paper. We first give additional details about our implementation and experimental setup in Section B. In Section C, we complement the results from the main paper by showing additional comparisons and studies. Furthermore, Section D comments about the limitations and failures of our approach. Finally, we provide additional qualitative examples in Section E.

B. Implementation Details

In this paragraph we describe the experimental setup when using the InstantNGP backbone. We use a hash table of size 2^{19} containing 16 levels with 2 features per level, maximum grid resolution of 2048, and an occupancy grid of resolution 128. We jointly train the neural network weights and the hash table entries by applying Adam with $\epsilon = 10^{-15}$ and a learning rate of 10^{-2} . We also apply \mathcal{L}_2 regularization with a factor of 10^{-6} , but only on neural network weights. These choices are based on suggestions in [5]. Also, for the sake of efficiency, we update the density grid after every 16 steps similarly to the procedure described in [5]. To further stabilize the training we also use the opacity regularization [4] with a factor of 10^{-3} , as well as gradient norm clipping with a factor of 0.05. We also use a cosine annealing learning rate scheduler and perform each training on 30k iterations with batch size 8190.

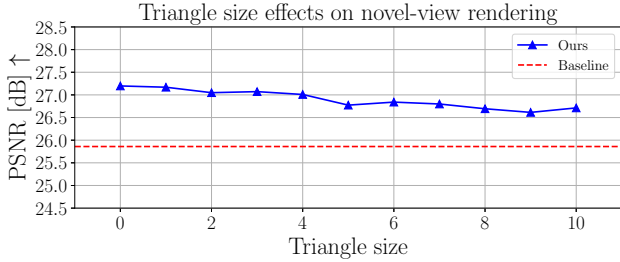
When it comes to our proposed method we set the loss weights $\lambda_{ctr} = \lambda_{ort} = 2 \cdot 10^{-3}$ in the case of Hypersim, $1 \cdot 10^{-2}$ in the case of ScanNet, and $5 \cdot 10^{-4}$ in the case of Replica. We turn on \mathcal{L}_{ort} and \mathcal{L}_{ctr} after 500 steps, and linearly increase their λ weights to the specified values over the next 2500 steps. Also, when using methods which rely on explicit surface normals, we randomly sample rays for one-third of every batch size and select their left and upper pixel neighbor to form a triplet to facilitate obtaining these explicit normals. We call this a triplet triangle. Every triangle in the batch is sampled randomly from a set of all

possible triangle triplets of all available images.

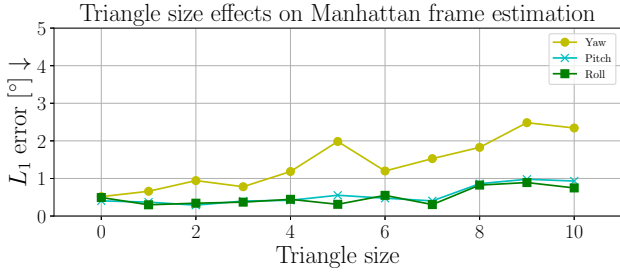
For k-means clustering, we use $k = 20$ clusters when processing training batches. However, in order to estimate the Manhattan frame for all methods after the training ends, we cluster the normals from the whole test set into $k = 30$ clusters. In addition, during both training and testing, we merge the three selected orthogonal clusters with their opposites. This is achieved by comparing the similarity of a selected cluster centroid n_i and the opposite vector of every other cluster centroid $(-c_j)$. In the case $|n_i^T(-c_j)| > 1 - t$ holds true, all cluster elements of \mathcal{C}_j are multiplied by -1 and added to \mathcal{N}_i . Also, if any cluster centroid c_j is close to one of the selected centroids n_i ($|n_i^T c_j| > 1 - t$), elements from \mathcal{C}_j are added to \mathcal{N}_i .

During the implementation of ManhattanDF [2], we initially had stability issues with the semantic segmentation cross-entropy loss. In order to alleviate these issues, we used label smoothing with a factor of 0.1, as well as a class weight of 0.3 for the background class. Also, in the Hypersim dataset, the scenes are much richer in content, and there are less wall & floor labels compared to Replica. Therefore we merged the floor class with the floor mat class, and we also merged the window class with the wall class. We also observed in Hypersim that in a small number of scenes the ceiling is labeled as wall, or that there are no wall & floor labels since they are all labeled as void.

Also, when it comes to comparing baselines and different methods on different datasets, we always perform a thorough search over additional hyperparameters before reporting results. On Hypersim the weight for the semantic loss was $1 \cdot 10^{-1}$, while the weight for surface normals loss was $5 \cdot 10^{-4}$ in the regular case. For ScanNet, the weight for the semantic loss was $1 \cdot 10^{-1}$, while the weight for surface normals loss was $5 \cdot 10^{-3}$. When using additional sparse depth (from SfM) supervision for ScanNet, the depth loss weight was $1 \cdot 10^{-1}$, while the surface normals loss weight was changed to $1 \cdot 10^{-2}$. For Replica, the weight for the semantic loss was $4 \cdot 10^{-2}$, while the weight for surface normals loss was $5 \cdot 10^{-4}$.



(a) Triangle size choice effects on novel-view rendering.



(b) Triangle size choice effects on MF estimation.

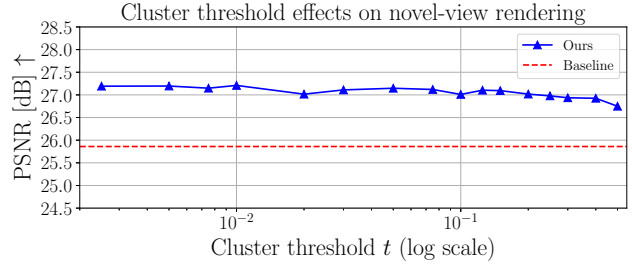
Figure 1: **Triplet triangle size choice effects on Hypersim-A.** The performance of our method slightly decreases with the triangle size increase. This is due to a bigger probability that the triplet will not lie on a planar surface segment, and thus the estimated normals will contain certain error.

C. Further Analysis

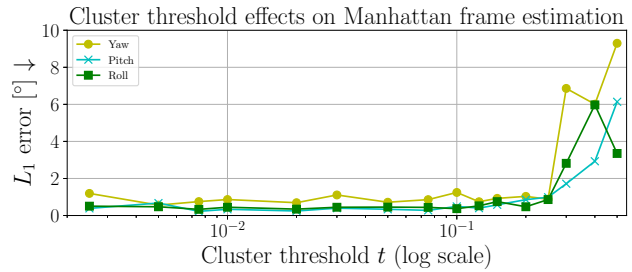
In this section, we perform further analyses of the experiments presented in Section 5 of the main paper.

Results in 435 scenes of Hypersim: In Table 1 we present results on all Hypersim scenes. Our proposed method performs better than the baseline with respect to all observed metrics. We were not able to evaluate ManhattanDF on all scenes, because experiments on a large portion of the scenes had convergence issues related to the specific loss function.

Triplet triangle size: As previously mentioned, when using methods that rely on explicit surface normals, we randomly sample rays for one-third of every batch size and select their left and upper pixel neighbor to form a triplet triangle to facilitate obtaining explicit normals. We call this a triangle of size 0, since the immediate left and upper neighbor are taken to form a triplet. Similarly, a triangle of size k is when there is a k pixel gap between the selected pixel and its left and upper triplet pair. In Figure 1 we show results of our method for different triangle sizes. As the triangle size increases, the performance of our method slightly decreases in terms of novel-view rendering as well as MF estimation. With larger triangle sizes, there is a bigger probability that the triplet will not lie on a planar surface segment and thus normals estimated with a planar assumption will contain a certain degree of error. Nevertheless, this study shows that whenever one is certain about the bigger planar regions, our



(a) Cluster threshold choice effects on novel-view rendering.



(b) Cluster threshold choice effects on estimating the MF.

Figure 2: **Cluster threshold t choice effects on Hypersim-A.** Our method is not very sensitive to threshold values below 0.1.

method of explicit normal computation can successfully be applied to potentially gain both efficiency and performance. The further study in this regard however is out of the scope of this work, which remains as our future work.

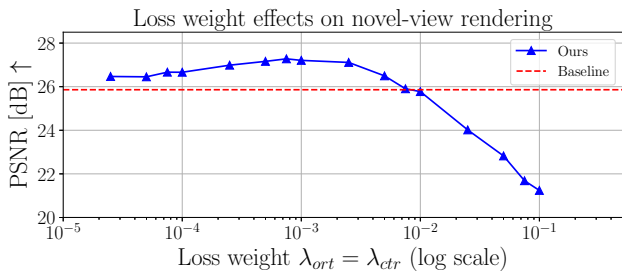
Cluster threshold: We analyze the sensitivity of our proposed method to the threshold parameter t , used to merge selected orthogonal clusters with their opposites and their nearby clusters. In Figure 2, we see that our algorithm is not very sensitive to different threshold t values below 0.1.

Ray sampling: As previously mentioned, when using methods which rely on explicit surface normals, we randomly sample pixel triplet triangles from a set of all possible triangles in all available images. However, when using the InstantNGP baseline, we randomly sample rays from the set of all available rays in all available images. The number of selected rays is always the same in total as the specified batch size, for all methods. In Table 2, we see that there is no significant difference if we sample triplet triangles during the baseline training instead of just randomly sampling rays. Therefore the improvements in our approach come from the proposed algorithm, and not the slightly different way of batch sampling.

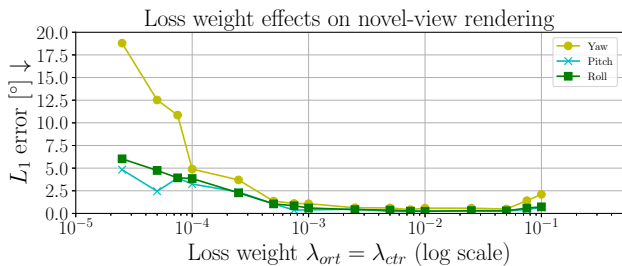
Sensitivity in hyperparameters selection: We further analyse the sensitivity of the loss weights $\lambda_{ort} = \lambda_{ctr}$ multiplying our proposed loss terms, to the overall performance. The quantitative results are presented in Figure 3. Very low λ values lead to a bad MF estimation, whereas very high λ values lead to bad novel-view rendering. A good trade-off is achieved around the chosen value

Table 1: **Experiments on all Hypersim scenes.** We observe that our method outperforms the baseline with respect to all observed metrics.

		PSNR \uparrow	SSIM \uparrow	Normals $^\circ\downarrow$	Pitch $^\circ\downarrow$	Roll $^\circ\downarrow$	Yaw $^\circ\downarrow$	D-MAE \downarrow	D-RMSE \downarrow
435 scenes	InstantNGP [5] (baseline)	19.17	0.729	61.88	7.12	7.08	21.99	0.119	0.162
	Ours	20.17	0.737	54.93	3.15	3.05	8.17	0.100	0.142



(a) Loss weight effects on novel-view rendering.



(b) Loss weight effects on estimating the MF.

Figure 3: **Loss weight $\lambda_{ort} = \lambda_{ctr}$ choice effects on Hypersim-A.** Very low weights lead to bad MF estimation, whereas very high weights lead to bad novel-view rendering.

of $2 \cdot 10^{-3}$. In addition, visual results are presented in Figure 5, where we can observe that very low λ produces noisy explicit surface normals, whereas very high λ makes all details appear “blocky” in normals. Moreover, this analysis has been performed statistically, on 20 scenes from the Hypersim-A split using the same hyperparameters. We observe that the scene-specific tuning of these hyperparameters could further improve the performance. These scene-specific hyperparameters differ only slightly from the ones selected for the whole set of scenes. We however, do not perform scene-specific tuning, demonstrating the generaliz-

Table 2: **Ray sampling strategy effects Hypersim-A.** There is no significant difference for the InstantNGP baseline between sampling random triplet triangles or random rays during training.

	PSNR \uparrow	D \downarrow -MAE	Norm. $^\circ\downarrow$	Rot. $^\circ\downarrow$
Random rays (same image)	25.72	0.072	60.11	12.36
Random rays (random images)	25.86	0.064	57.12	10.63
Random triangles (random images)	26.16	0.061	56.72	10.25

ibility of our method across the diverse scenes of Hypersim.

Runtime: We further analyze the run time of our proposed method. We use Python and the PyTorch Deep Learning library. The InstantNGP baseline implementation is adapted from [1], where some functionalities (e.g. volume rendering) are efficiently implemented directly in CUDA code. In addition to the baseline, our proposed approach computes explicit surface normals for every batch, followed by k-means clustering (implemented using the FIASS library [3]), and finally followed by computing two additional loss terms \mathcal{L}_{ctr} and \mathcal{L}_{ort} . The baseline needs 22.77 ± 2.07 minutes on average to train on Hypersim-A, whereas our method needs 29.06 ± 1.64 minutes. The inference time is the same for both methods, and it takes about half a minute on average to render the test set. The experiments were performed on a single NVIDIA GeForce RTX 2080 Ti GPU, with 11 Gb of RAM memory.

D. Limitations

Missing details in surface normals: While inspecting qualitative results of novel-view rendering, we observed a few limitations and failure cases that occurred occasionally. Sometimes our method merges one Manhattan direction with one of the other two directions, e.g. producing the same surface normals for a roof (horizontal surface) and one of the walls (vertical surface). This can be seen in the first example in Figure 4. Another phenomenon is having severe “blocky” artifacts or missing details in the computed explicit normals, e.g. as in the second example in Figure 4. Nevertheless, this usually offers better novel-view RGB rendering, compared to not imposing any structure priors. This behavior is also related to the choice of loss weights $\lambda_{ort} = \lambda_{ctr}$, discussed in Section C of this supplementary material, as well as in Figure 5.

Easy scenes: Our method is not beneficial for very easy scenes. This is shown in Figure 5 of the main paper. When it comes to easy scenes, the 3D structure of these scenes can be learned without much problem, without imposing any structure priors.

E. More Qualitative Results

We provide more qualitative results related to experiments from Section 5 of the main paper. Figure 6 depicts visual results from the Hypersim-A dataset, Figure 7 depicts visual results from the ScanNet dataset, and Figure 8 depicts visual results from the Replica dataset. We again see that our method leverages many Manhattan objects and sur-

faces in the scene, which improves the geometrical structure of 3D compared to the InstantNGP baseline. This is visible in surface normals and depth, obtained using volume rendering. Furthermore, unlike ManhattanDF, our method relies on many Manhattan cues other than the walls & floors. For example, different component and parts of furniture and stairs respect the Manhattan assumption, which is successfully exploited by our method.

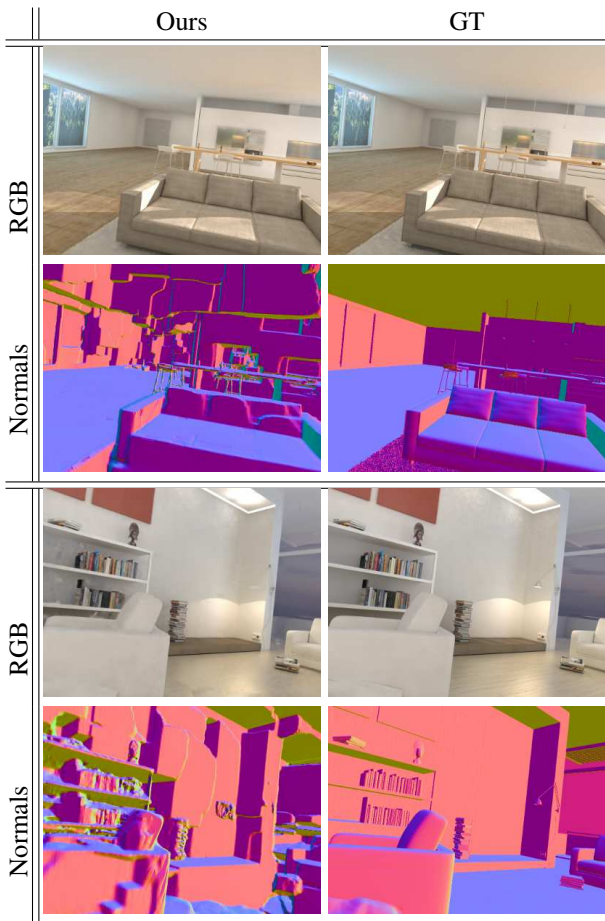


Figure 4: **Common failure cases.** In the first example, we see our method merging explicitly computed surface normals of one Manhattan direction (vertical ceiling surface) with one of the other two directions (horizontal wall surface). In the second example, we see severe “blocky” artifacts in the normals. Nevertheless, in both cases, there is not much trouble, nor big artifacts, when performing novel-view rendering.

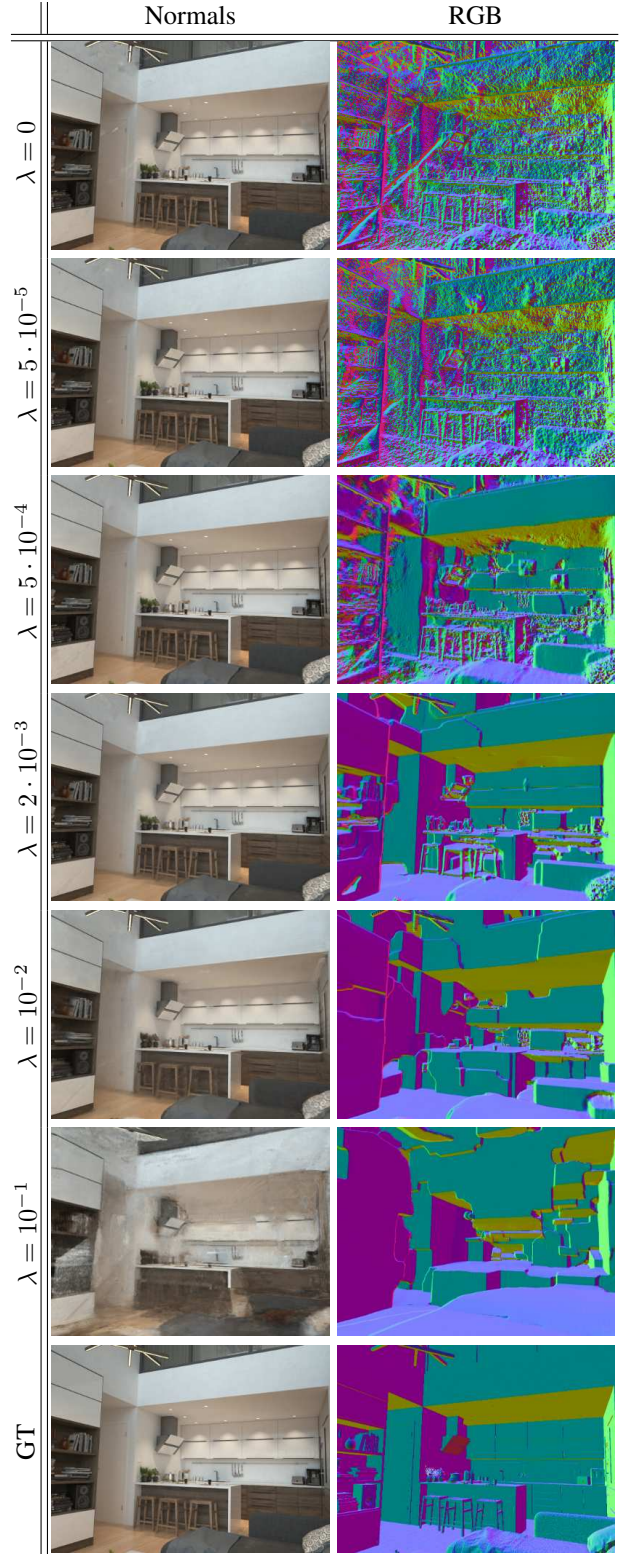


Figure 5: **Loss weight $\lambda_{ort} = \lambda_{ctr}$ choice effects on Hypersim-A.** Very low λ values lead to very noisy explicit surface normals, whereas very high λ values lead to “blocky” artifacts in normals and bad novel-view rendering.

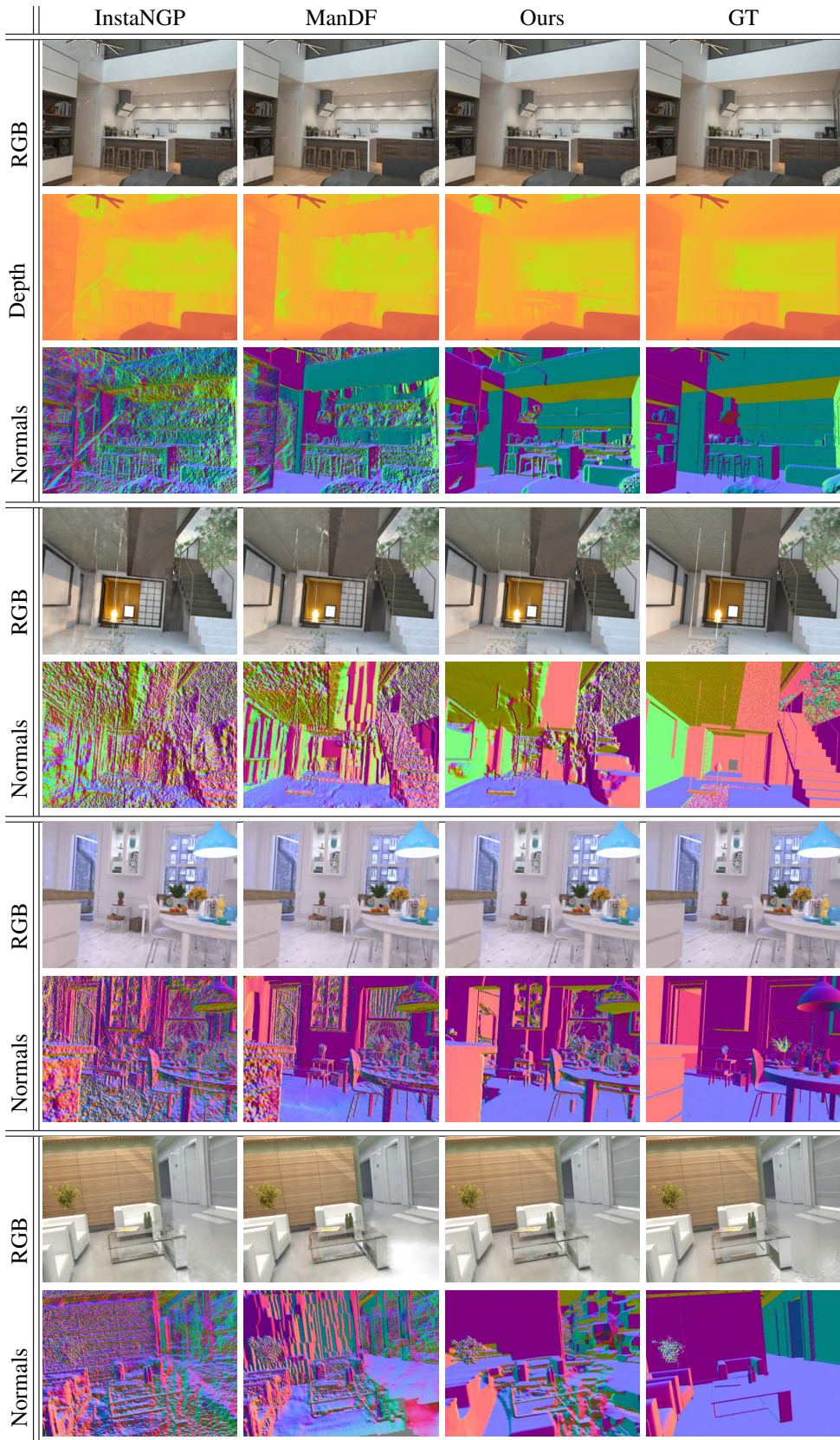


Figure 6: **Qualitative results on Hypersim-A.** Our method leverages many Manhattan objects and surfaces in the scene, which improves the implicit geometrical representation compared to the baseline. Unlike ManhattanDF, our method relies on many Manhattan cues other than the walls and floors.








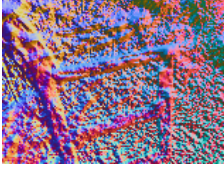

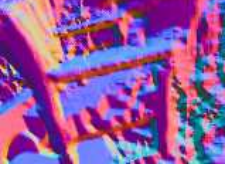




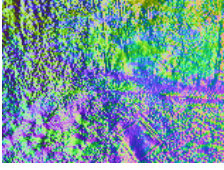
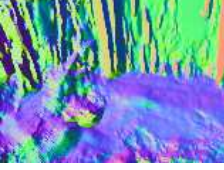
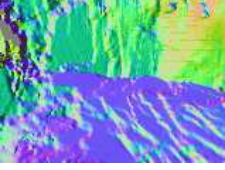




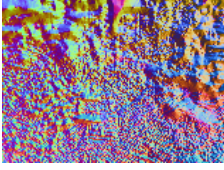
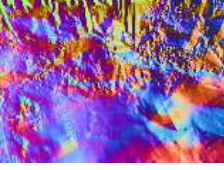
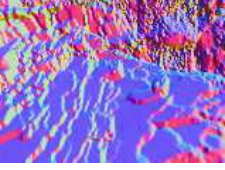




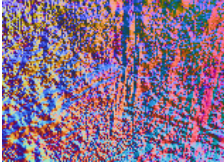

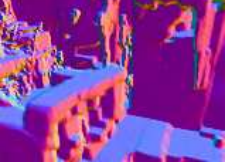
	InstaNGP	ManDF	Ours	GT
RGB				
Depth				Not available
Normals				Not available
RGB				
Normals				Not available
RGB				
Normals				Not available
RGB				
Normals				Not available

Figure 7: **Qualitative results on ScanNet.** Our method leverages many Manhattan objects and surfaces in the scene, which improves the implicit geometrical representation compared to the baseline. Unlike ManhattanDF, our method relies on many Manhattan cues other than the walls and floors.

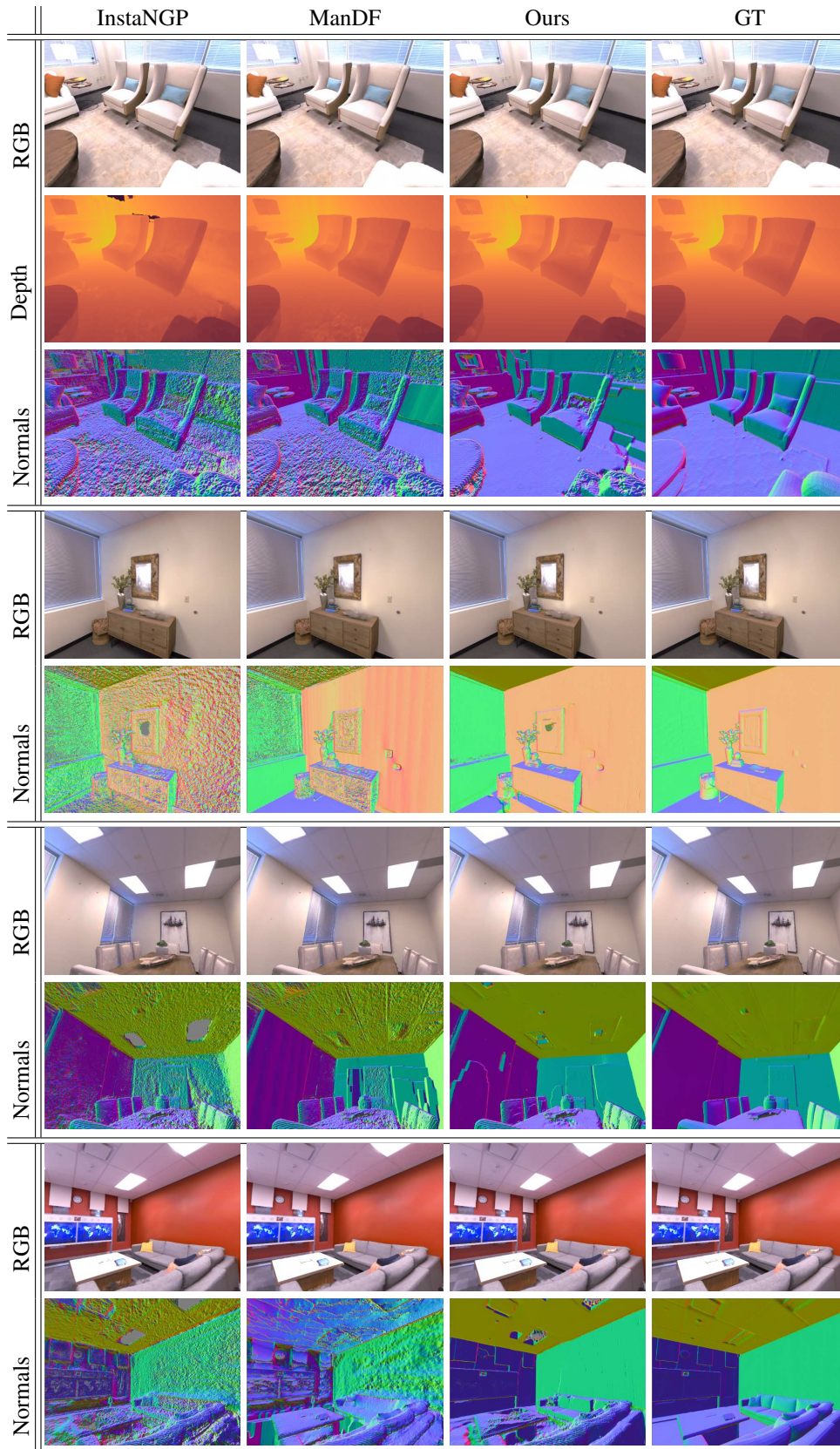


Figure 8: **Qualitative results on Replica.** Our method leverages many Manhattan objects and surfaces in the scene, which improves the implicit geometrical representation compared to the baseline. Unlike ManhattanDF, our method relies on many Manhattan cues other than the walls and floors.

References

- [1] https://github.com/kweal23/ngp_pl. Last checked : 06.09.2022.
- [2] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3d scene reconstruction with the manhattan-world assumption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5511–5520, 2022.
- [3] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- [4] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes. *ACM Transactions on Graphics*, 38(4):1–14, aug 2019.
- [5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.