RetrievalFuse: Neural 3D Scene Reconstruction with a Database – Supplemental Document –

Yawar Siddiqui[1] Justus Thies[1,2] Fangchang Ma[3] Qi Shan[3] Matthias Nießner[1] Angela Dai[1]

[1]Technical University of Munich [2]Max Planck Institute for Intelligent Systems, Tübingen [3]Apple

In this appendix, we discuss additional experiments that we conducted with our neural 3D scene reconstruction method *RetrievalFuse* (Sec. 3). Specifically, we show additional ablation studies and results for both the 3D super-resolution and surface reconstruction. We also provide implementation details of our method and the used baselines (Section 1), as well as our data generation (Sec. 2). We conclude with a discussion about limitations.

1. Implementation Details

Levels of Operation for Scene Reconstruction. Fig. 1 shows the different levels of operation at which our method operates on to reconstruct a 3D scene. Larger scenes are split into fixed size windows, chunk retrievals are made on smaller sized chunks for more expressability, and attention-based blending works on yet smaller sized patches to allow the method to choose among different retrievals at a finer detail.

Network Architecture. Fig. 2 details the architecture of our networks for 3D super-resolution task. All networks are implemented in PyTorch [8].

Inference Time and Number of Parameters. We report the number of trainable parameters and the inference time for our method (both retrieval and refinement stage) along with that of the baselines in Tab. 1 for the 3D super-resolution task. All runtimes are reported on a machine with Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz processor with an NVIDIA 2080Ti GPU. We use FLANN [7] to speed up nearest neighbor lookups from the database. Our retrieval inference time is significantly higher than refinement due to multiple disk reads to retrieve chunks (= number of chunks $\times$ number of retrievals). To avoid this overhead during training, once the retrieval networks have been trained, we preprocess the entire training set to extract retrievals before starting refinement stage training.

Method	Inference Time (s)	# Parameters ($\times 10^6$)
SGNN [2]	2.297	0.64
ConvOcc [9]	1.707	1.04
IFNet [1]	0.708	2.95
Ours (Retrieval)	0.784	0.77
Ours (Refinement)	0.012	1.49

Table 1: Comparison of inference time and number of trainable parameters on the 3D super-resolution task.

1.1. IFNet-based RetrievalFuse

To demonstrate the wide applicability of our method, we also demonstrate our approach integrated into the implicit-based reconstruction of IFNet [1] to leverage our retrieved approximations. We keep the IFNet encoder and decoder unmodified, and add an additional retrieval encoder for processing the retrieved reconstruction approximations. This retrieval encoder is based on the original IFNet encoder, and

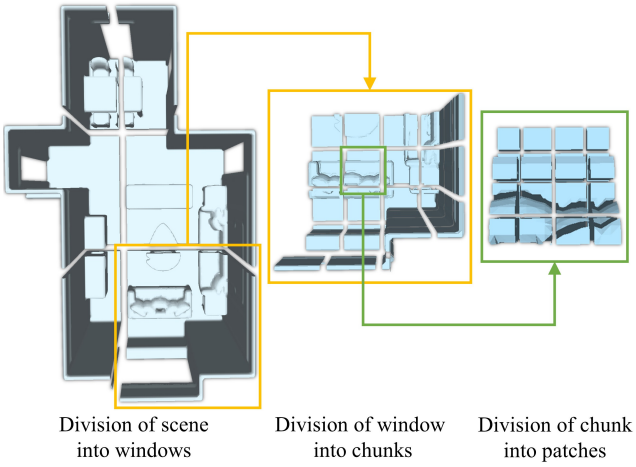


Figure 1: In our experiments, we use 64^3 target chunks for target geometry, and larger scenes work in a sliding window fashion (left). The retrieval candidates are 16^3 chunks (middle), and attention-based blending works on 4^3 patches (right).

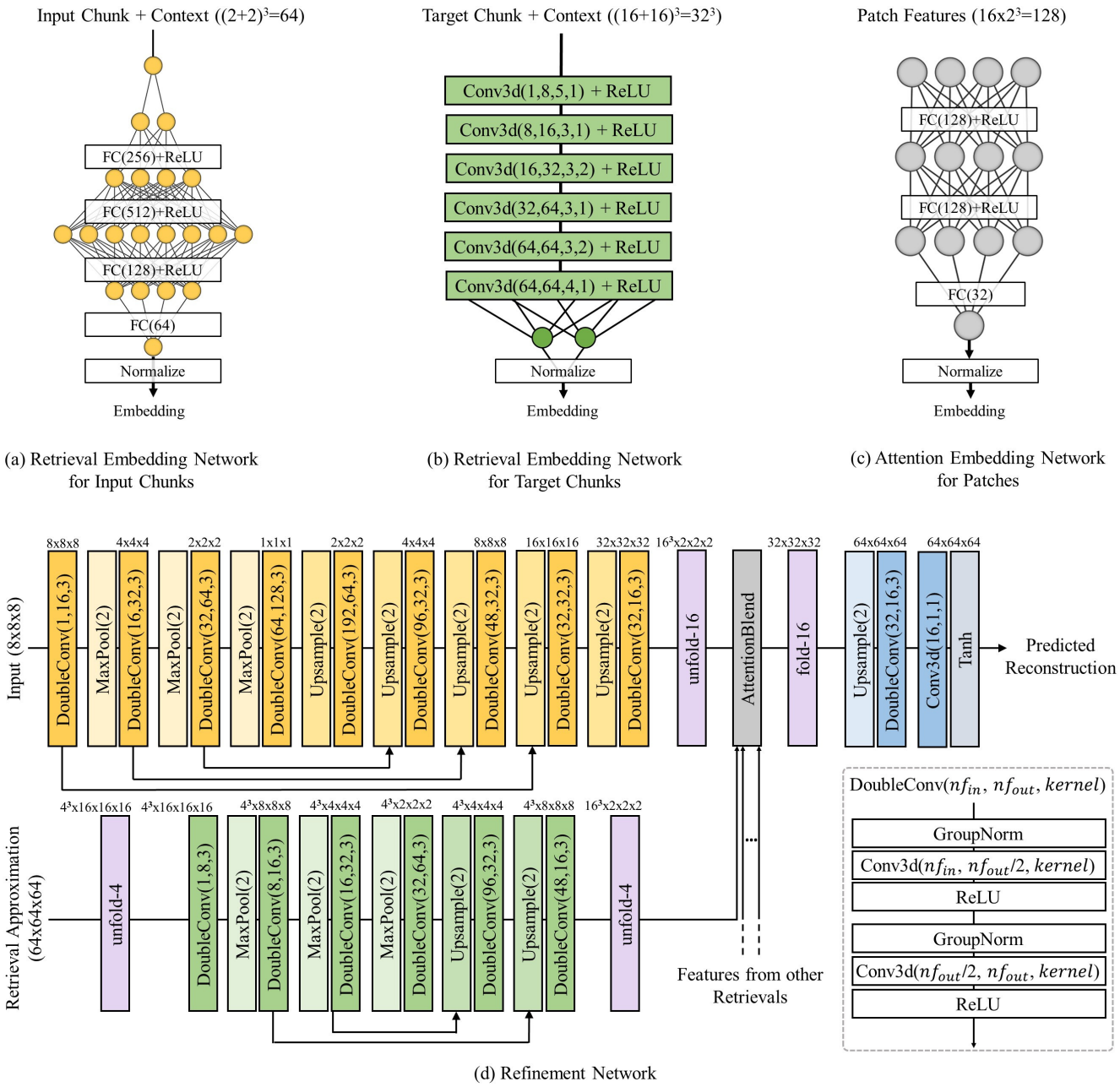


Figure 2: Network architecture used in our 3D super-resolution experiments. Convolution parameters are given as (input features, output feature, kernel size, stride), with default stride of 1 if not specified. Array of circles represent fully connected (FC) layers. For the task of point cloud to surface reconstruction, the input chunk embedding network is a convolutional layer instead of MLP with a fully connected layer at the end on account of larger input chunk size (since input is a 128^3 grid for surface reconstruction in comparison to 8^3 grid for super-resolution, we use a chunk size of 32^3 for inputs there). Additionally, the input feature extractor is deeper for point cloud to surface reconstruction on account on bigger input grid.

works with chunks from the retrievals. For a given point in space, features sampled at the point from feature volumes at different levels of the input encoder make up the input features. Features sampled from the retrieval features volumes at this point for each of the k retrievals make up the retrieval features. Next, based on the feature volume at last layer of input and retrieval encoder, a blending coefficient grid and an attention weight grid is obtained. To obtain these, the $8 \times 8 \times 8$ input feature volume and the $32 \times 32 \times 32$ retrieval feature volume are interpreted as 512 patch volumes

Chunk side (m)	Retrieval				Refinement		
	IoU↑	CD↓	NC↑	Entries	IoU↑	CD↓	NC↑
3.467	0.53	0.074	0.72	43092	0.71	0.029	0.91
1.733	0.60	0.041	0.85	344249	0.72	0.028	0.91
0.867	**0.67**	**0.033**	**0.87**	2093592	**0.75**	**0.026**	**0.92**

Table 2: Smaller sized chunk retrievals improve the performance of both retrieval and refinement, although at cost of a larger database. Evaluation performed on 3D super-resolution task on 3DFront dataset.

Variant	IoU↑	CD↓	F1↑	NC↑
Retrieval	0.364	0.781	0.525	0.708
Backbone	0.463	0.647	0.602	0.813
Naive	0.432	0.684	0.576	0.798
Ours	0.478	0.635	0.601	0.811

Table 3: In case of suboptimal retrievals, our method does not provide significant improvement over the backbone reconstruction quality. However, it is more robust to bad retrievals compared to a naive blending of retrieval features with input features. Networks trained on a ShapeNet subset with 8 classes and evaluated on a disjoined subset with 5 classes.

# Train Scenes	IoU↑	CD↓	F1↑
3750 (25%)	0.711	0.0283	0.784
7500 (50%)	0.728	0.0275	0.791
11250 (75%)	0.741	0.0269	0.796
15000 (100%)	0.751	0.0265	0.801

Table 4: Ablation study w.r.t. the number of train scenes, evaluated on the 3D super-resolution task using the 3DFront dataset.

of shape $1 \times 1 \times 1$ and $4 \times 4 \times 4$ respectively. These input and corresponding retrieval patch volumes are mapped to a shared embedding space, from which we can get the blending coefficient (Eq. 6, main paper) and attention weights (Eq. 4, main paper). Once we have the blending coefficient grid and attention weight grid, we can sample their values at the queried point. Finally we blend the sampled input features and the sampled k retrieved features (Eq. 5, main paper) to give the blended feature that is decoded by the IFNet decoder.

1.2. Baselines

We use the official implementations provided by the authors of IFNet [1], Convolutional Occupancy Networks [6], SGNN [2], Local Implicit Grids [3] and Screened Poisson Reconstruction [4] in our experiments. For 3D super-resolution experiments, the methods are provided with low-resolution distance field grids as inputs instead of voxel grid inputs. In particular, for IFNet we use the *ShapeNet32Vox*

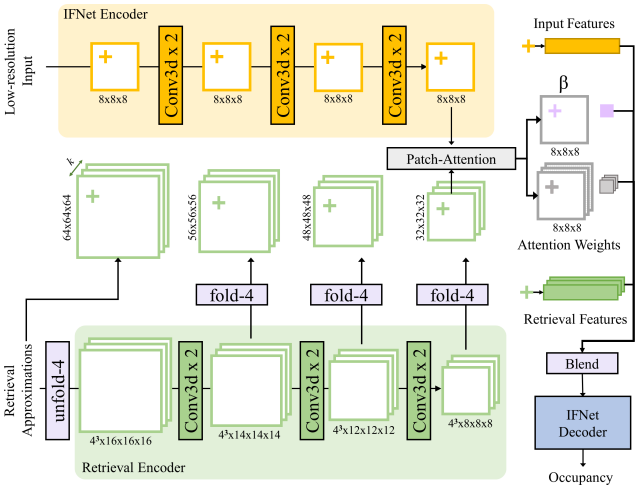


Figure 3: Integration of our RetrievalFuse approach to the implicit network of IFNet [1]. We use IFNet's encoder as the input feature encoder and their decoder as the implicit decoder. Additionally, we use a retrieval encoder similar to the IFNet encoder for obtaining features for the retrieval approximations. Further, a patch attention layer computes a blend coefficient grid and attention weight grid. For a given query point in space, features are sampled from input feature grids, retrieval feature grids. A blend coefficient value and attention weights are sampled from the blend coefficient grid and attention weight grid at the queried point. The sampled input features and retrieval features are blended based on these valued and finally decoded to an occupancy value by the IFNet decoder.

model for 3D super-resolution. For surface reconstruction from point clouds for IFNet, the 128^3 discretized point cloud is used with the *ShapeNetPoints* model. For Convolutional Occupancy Networks we use the 32^3 *voxel simple encoder* for 3D super-resolution, and a 64^3 *point net local pool* encoder for point cloud surface reconstruction. For SGNN, we use a 64^3 resolution with nearest-neighbor upsampling to a 64^3 grid for the input. For Local Implicit Grids we found that the part sizes $0.25\times$ shape size for ShapeNet and $0.35\times$ window size for 3DFront and Matterport3D worked best at the sparsity of the input point cloud.

2. Data Generation and Evaluation Metrics

Data generation. As specified in the main paper, the targets for both 3D super-resolution and surface reconstruction from point cloud tasks are 64^3 distance field grids. Training and inference on larger scenes is done in a sliding window manner with a window stride of 64. We use SDFGen[1] to generate these distance field targets. Low-resolution distance field inputs are generated in a similar manner at a

[1] https://github.com/christopherbatty/SDFGen

coarser resolution. Point cloud samples for surface reconstruction task are generated as random samples on the surface of meshes generated from target distance fields.

For IFNet [1], Convolutional Occupancy Networks [6], and our implicit variant, all of which need supervision in the form of points along with their occupancies, we first extract meshes from the target distance fields using the marching cubes algorithm [5]. These meshes are then made watertight using *implicit waterproofing* [1] from which points and their occupancies are finally sampled. SGNN is provided the same inputs and targets as ours for training, with the respective inputs upsampled to match the target 64^3 resolution grid. Local Implicit Grids [3] is trained on ShapeNet, and Screened Poisson Reconstruction [4] does not require training; however, both methods are provided high-resolution normals to obtain oriented point clouds as inputs.

Evaluation Metrics. We follow the definition and implementations of Chamfer ℓ_1 Distance, Normal Consistency, and F-Score from [9]. Specifically, Chamfer ℓ_1 Distance (CD) is defined as:

$$\mathrm{CD}(\mathcal{M}_{pred}, \mathcal{M}_{gt}) = \frac{1}{2}(\mathrm{Acc}(\mathcal{M}_{pred}, \mathcal{M}_{gt}) + \mathrm{Comp}(\mathcal{M}_{pred}, \mathcal{M}_{gt}))$$

where $\mathcal{M}_{pred}$ and $\mathcal{M}_{gt}$ are the predicted and target meshes (obtained by running marching cubes on predicted and target distance fields). $Acc(.)$ and $Comp(.)$ are accuracy and completeness given as:

$$\mathrm{Acc}(\mathcal{M}_{pred}, \mathcal{M}_{gt}) = \frac{1}{|\partial\mathcal{M}_{pred}|} \int_{\partial\mathcal{M}_{pred}} \min_{\mathbf{q}\in\partial\mathcal{M}_{gt}} \|\mathbf{p}-\mathbf{q}\| \, \mathrm{d}\mathbf{p},$$

and

$$\mathrm{Comp}(\mathcal{M}_{pred}, \mathcal{M}_{gt}) = \frac{1}{|\partial\mathcal{M}_{gt}|} \int_{\partial\mathcal{M}_{gt}} \min_{\mathbf{p}\in\partial\mathcal{M}_{pred}} \|\mathbf{p}-\mathbf{q}\| \, \mathrm{d}\mathbf{q}$$

with $\partial\mathcal{M}_{pred}$ and $\partial\mathcal{M}_{gt}$ denoting the surfaces of the meshes. Normal Consistency (NC) is defined as:

$$\mathrm{NC}(\mathcal{M}_{pred}, \mathcal{M}_{gt}) = \frac{1}{2\,|\partial\mathcal{M}_{pred}|} \int_{\partial\mathcal{M}_{pred}} |n(\mathbf{p}) \cdot n(\mathrm{proj}_2(\mathbf{p}))| \, \mathrm{d}\mathbf{p} + \frac{1}{2\,|\partial\mathcal{M}_{gt}|} \int_{\partial\mathcal{M}_{gt}} |n(\mathbf{q}) \cdot n(\mathrm{proj}_1(\mathbf{q}))| \, \mathrm{d}\mathbf{q}$$

where $(.)$ indicates inner product, $n(\mathbf{p})$ and $n(\mathbf{q})$ are the unit normal vectors on the mesh surface, and $\mathrm{proj}_2(\mathbf{p})$ and $\mathrm{proj}_1(\mathbf{q})$ are projections of $\mathbf{p}$ and $\mathbf{q}$ onto mesh surfaces $\partial\mathcal{M}_{pred}$ and $\partial\mathcal{M}_{gt}$ respectively. F-Score [10] is defined as the harmonic mean of precision and recall, where recall is fraction of points on $\mathcal{M}_{gt}$ that lie within a certain distance to $\mathcal{M}_{pred}$, and precision is the fraction of points on $\mathcal{M}_{pred}$ that lie within a certain distance to $\mathcal{M}_{gt}$. For calculating the volumetric IoU, we first voxelize the meshes $\mathcal{M}_{gt}$ and $\mathcal{M}_{pred}$ with voxel sizes of 0.054m for 3DFront, 0.0375m for Matterport3D, and resolutions 64^3 for ShapeNet. The IoU is then given as:

$$\mathrm{IoU} = \frac{\mathrm{Voxels}(\mathcal{M}_{pred}) \cap \mathrm{Voxels}(\mathcal{M}_{gt})}{\mathrm{Voxels}(\mathcal{M}_{pred}) \cup \mathrm{Voxels}(\mathcal{M}_{gt})}$$

3. Additional Evaluation

3.1. Ablation Studies

Chunk Embedding Space Visualization. Fig. 5 visualizes the embedding space used for retrieving chunks from our database. Chunks with similar geometry end up lying closer in this space.

Effect of retrieved chunk size on the performance of our method. Tab. 2 evaluates our method with retrieval approximations of different chunk sizes for retrieval. A chunk size that is too large cannot effectively capture the diversity of various scene arrangements, while smaller sizes can represent a wider variety of geometry, at the cost of an increased database size.

Effect of number of training scenes used for creating the database Tab. 4 shows the effect of number of chunks in the database on our method's performance. Availability of a wider variety of chunks helps reconstruction.

3.2. Additional Qualitative Results

We provide additional qualitative evaluation of our method on 3DFront and Matterport3D super-resolution and point cloud to surface reconstruction tasks in Fig. 8 and Fig. 9 respectively. Qualitative evaluation on ShapeNet for both of the tasks is provided in Fig. 10. Further, additional qualitative visualization for *Effect of retrieval and attention-based refinement* (main paper section 4.3) is provided in Fig. 4.

4. Additional Discussion

The result in the main paper as well is the additional experiments in this document show the broad applicability of our method, achieving state-of-the-art reconstruction and super-resolution outputs. Nevertheless, our approach still has limitations as discussed in the main paper. In particular, if the retrieval approximations are suboptimal, they will not help in the refinement process. Fig. 6 visualizes some samples where the retrieval approximations don't help the reconstruction. However, in these cases, even though the retrievals don't help the reconstruction, they also don't worsen the reconstruction. This is achieved by the blending network effectively ignoring the retrievals in such cases. The dependence on good retrievals can be observed more clearly in the following experiment. We train our retrieval

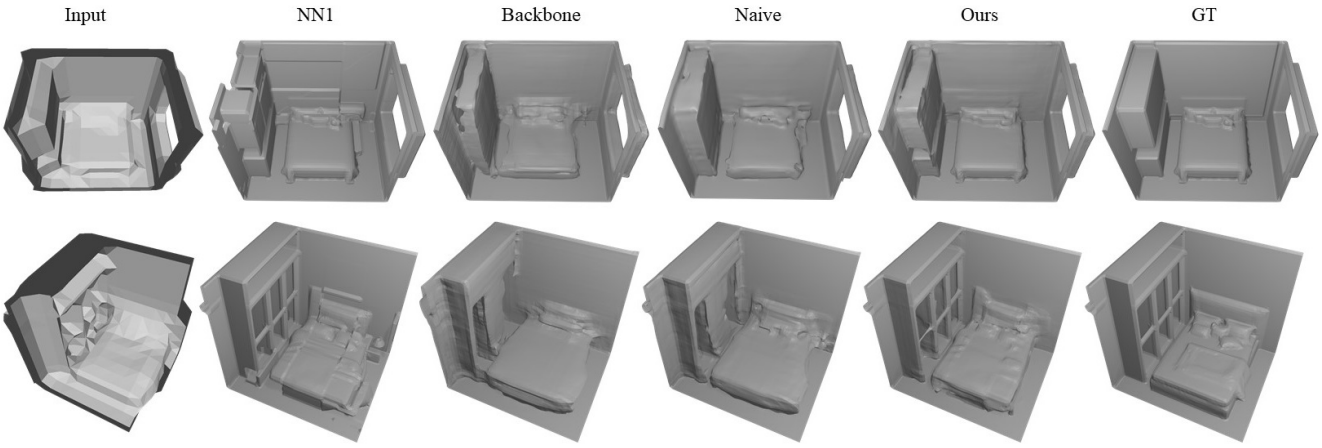
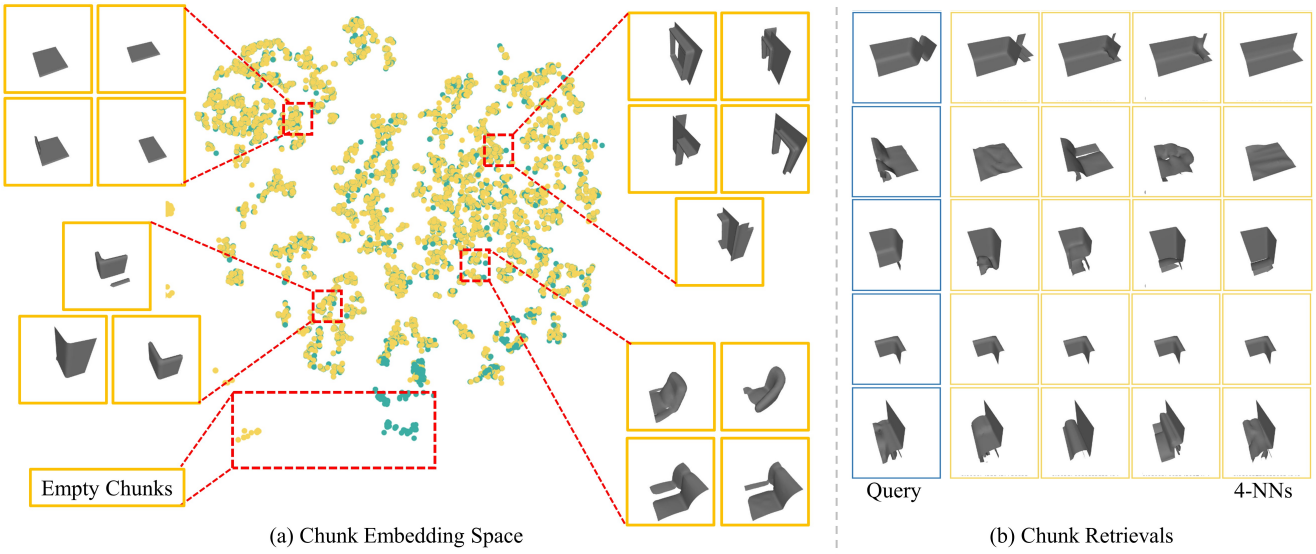


Figure 4: Additional qualitative evaluation of our method (*Ours*) in comparison to 1st nearest neighbor retrieval (*1-NN Retrieval*), our refinement network without retrievals (*Backbone*) and naive fusion of retrieved approximations during refinement (*Naive*).



(a) Chunk Embedding Space

(b) Chunk Retrievals

Figure 5: (a) Chunk embedding space visualized for 5000 chunks from 3DFront test set. This embedding space used for retrievals from the database by projecting an input chunk into this space (visualized as green dots) and retrieving k-nearest database chunks (visualized by yellow dots) from it. (b) Input queries and their corresponding 4 nearest neighbors from the embedding space. For the sake of visual clarity, input queries are visualized as their corresponding ground truth reconstruction.

and refinement networks on a ShapeNet subset of 8 classes. The dictionary is created using chunks from the same 8 classes. The trained networks are evaluated on a subset of new 5 classes. As shown in Tab. 3 and Fig. 7, our method doesn't improve significantly over the backbone network due to low quality retrievals. Compared to a naive fusion of features from retrievals however, which learns to rely on retrievals during training, our method is more robust.

A limitation of our method is cubic growth in number of chunks in the database with the decrease in patch size. As observed in Tab. 2, smaller chunk retrievals help both retrieval and refinement. This however comes at the cost of more patches in the database, making the database indexing and retrieval slower.

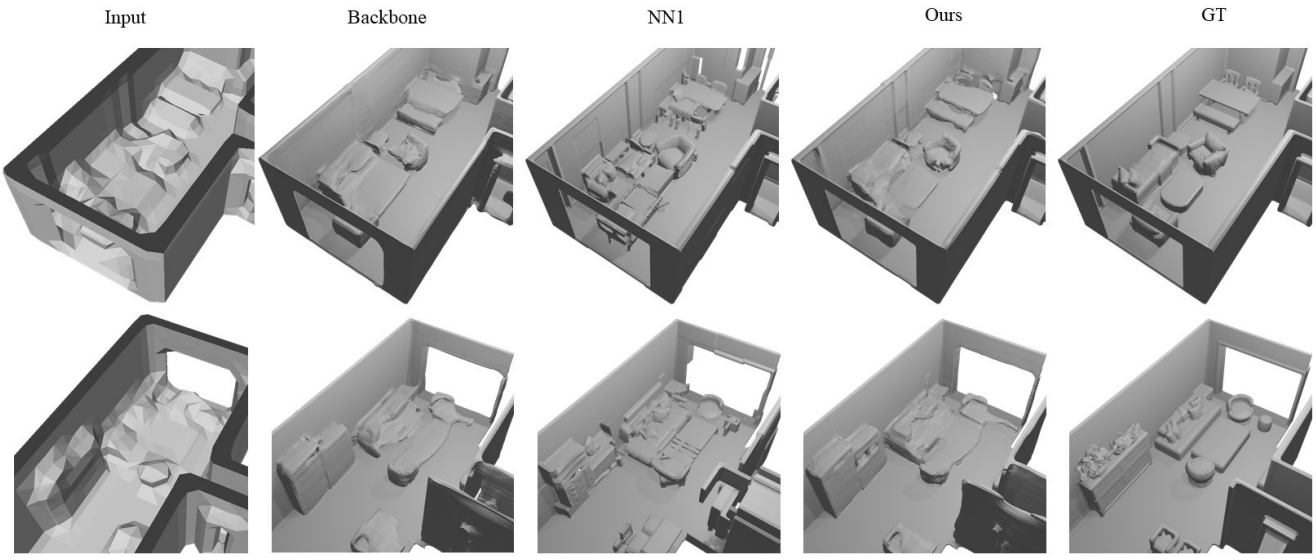


Figure 6: Suboptimal retrievals do not improve results significantly over our Backbone network. However, reconstruction produced are also not degraded due to subobtimal retrievals. Qualitative results from 3DFront super-resolution task.

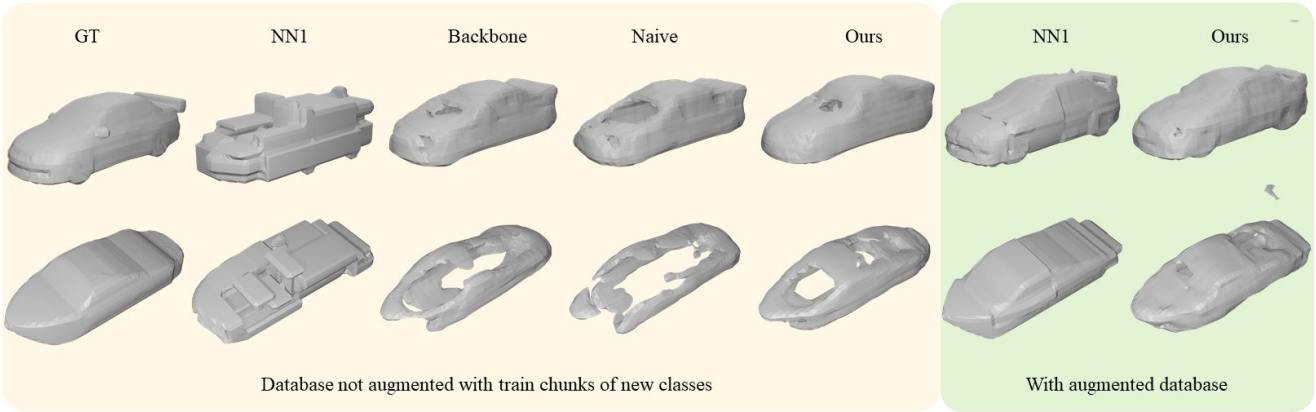


Figure 7: (Left) Suboptimal retrievals (NN1) when the our method is trained on a ShapeNet subset of 8 classes and evaluated on another 5 classes. The database contains chunks only from the original 8 classes. In this case, the suboptimal retrievals don't help the reconstruction, and the quality of reconstruction does not significantly improve over our backbone network. However, in contrast to naive fusion of retrieval features, our reconstruction quality does not degrade over the backbone. (Right) If the database if augmented with new chunks from train set of the new 5 classes, the reconstruction quality visibly improves without retraining.

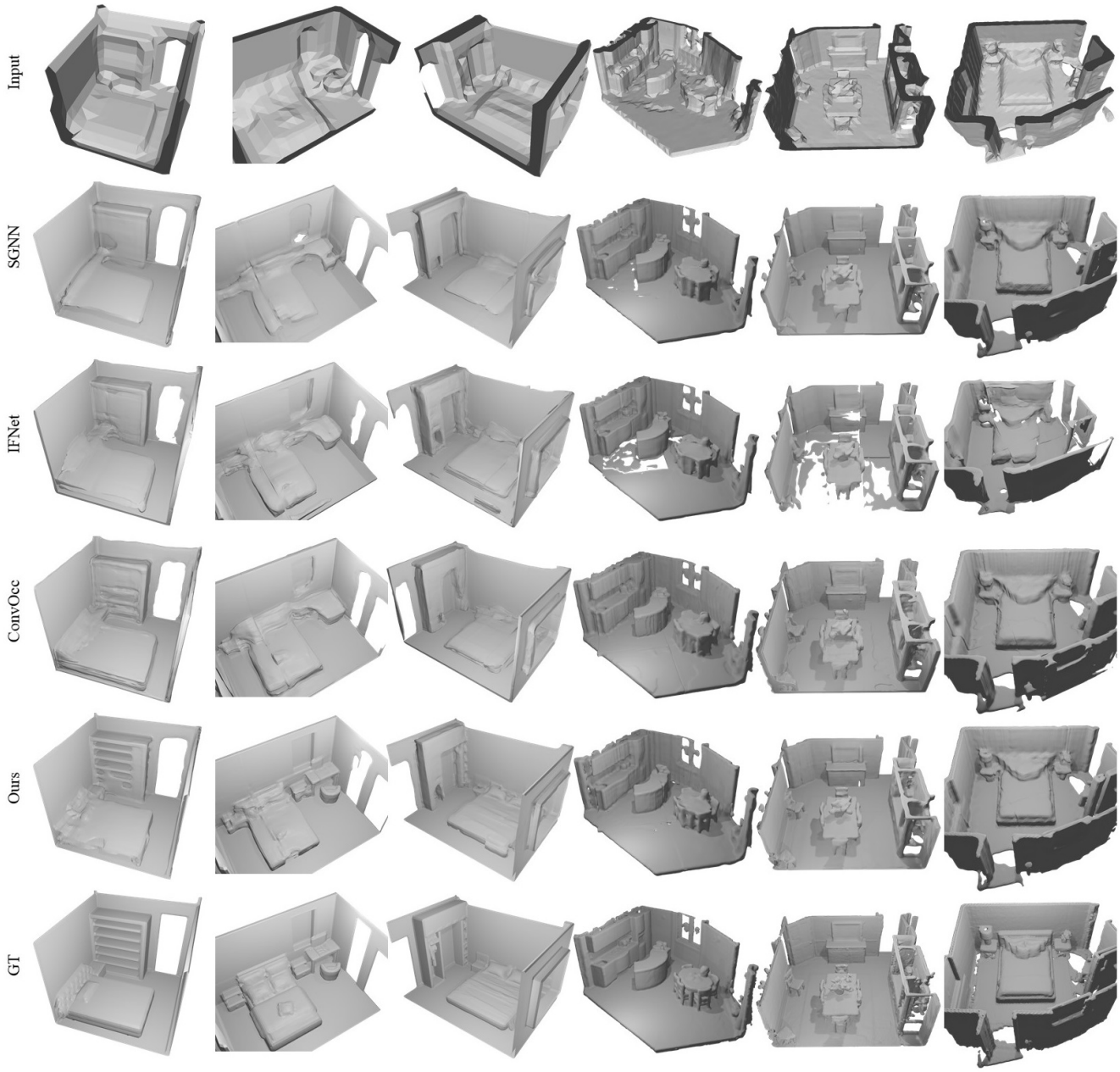


Figure 8: Additional qualitative results on 3DFront (left three) and Matterport3D (right three) on 3D super-resolution task.

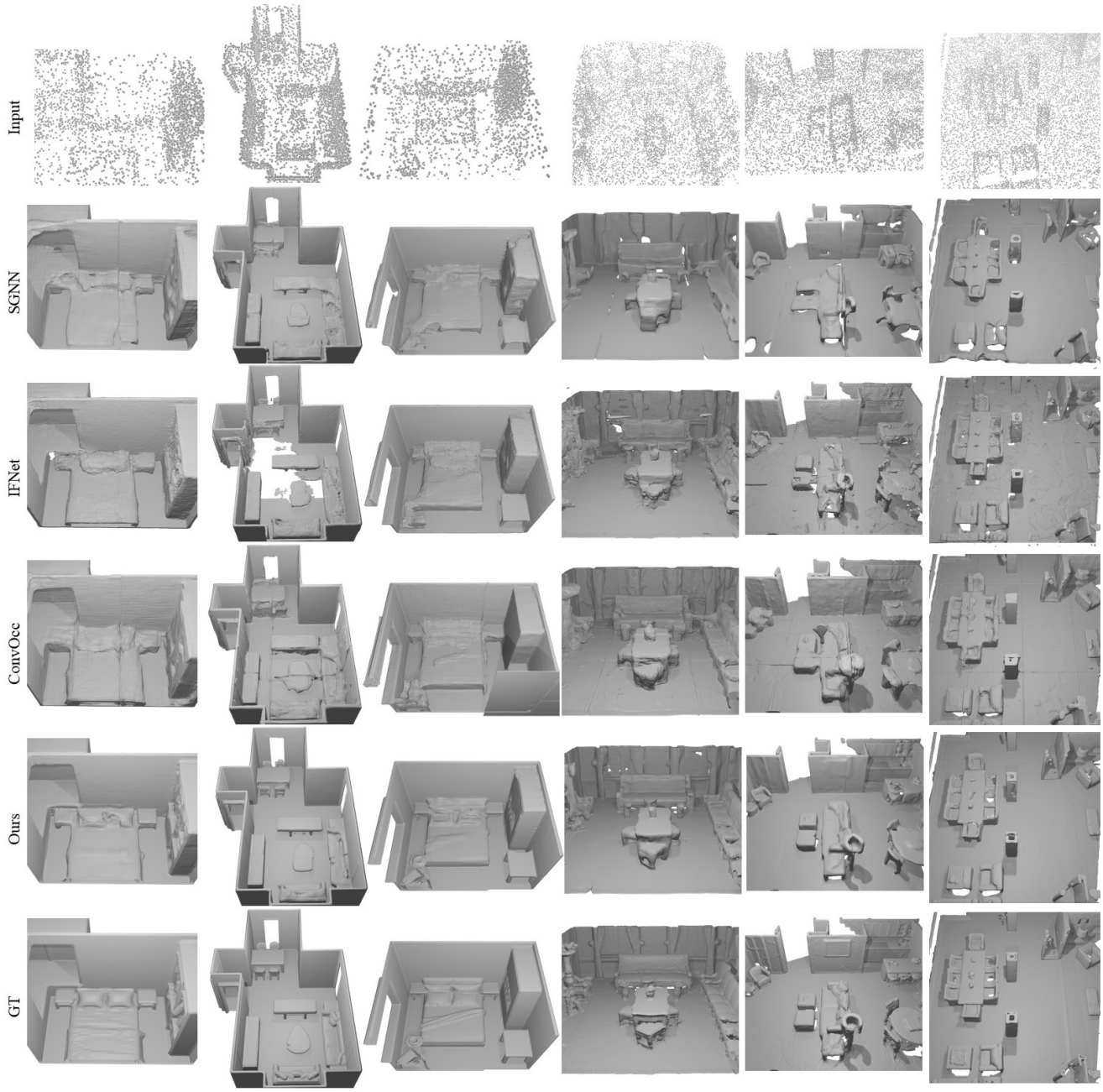


Figure 9: Additional qualitative results on 3DFront (left three) and Matterport3D (right three) on point cloud to surface reconstruction task.

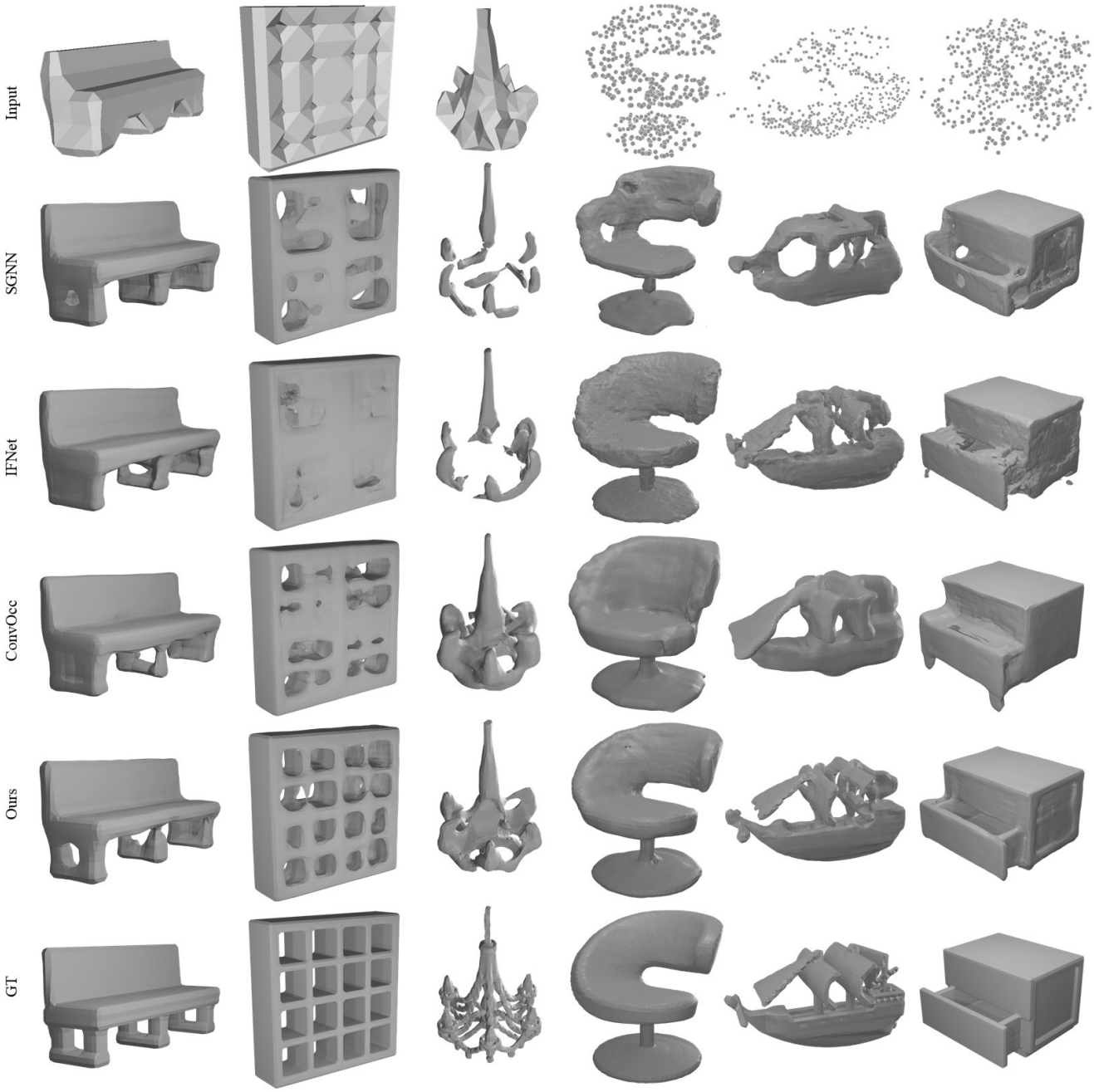


Figure 10: Qualitative results on ShapeNet dataset on 3D super-resolution (left three) and point cloud to surface reconstruction (right three) tasks.

References

[1] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 1, 3, 4

[2] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *CVPR*, pages 849–858, 2020. 1, 3

[3] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 3, 4

[4] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, 2013. 3, 4

[5] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 4

[6] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 3, 4

[7] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009. 1

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[9] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *arXiv preprint arXiv:2003.04618*, 2, 2020. 1, 4

[10] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019. 4