Behind the Veil: Enhanced Indoor 3D Scene Reconstruction with Occluded Surfaces Completion

Su Sun* [1], Cheng Zhao* [2], Yuliang Guo[2], Ruoyu Wang[2], Xinyu Huang[2], Yingjie Victor Chen[1], Liu Ren[2]
[1]Purdue University, [2]Bosch Research North America, Bosch Center for Artificial Intelligence (BCAI)
{sun931, victorchen}@purdue.edu
{cheng.zhao, yuliang.guo2, ruoyu.wang, xinyu.huang, liu.ren}@us.bosch.com

In this appendix, we provide more details on the datasets, evaluation metrics, and baselines in Appendices 1, 2, and 3. Additionally, we elaborate on our method implementation, including data preparation, network designs and training in Appendix 4. Furthermore, the runtime analysis and video demo are provided in Appendices 5 and 6, respectively. Lastly, we include further information about our newly developed 3D Complete Room Scene (3D-CRS) dataset in Appendix 7.

1. Datasets

We evaluate the proposed method with the baselines on two datasets: 3D-CRS and iTHOR scene dataset from AI2-THOR [3]. Most of the existing public indoor RGB-D datasets don't provide the completed 3D meshes of room scenes. To validate the generalization of our approach, we created a new dataset named 3D Complete Room Scene (3D-CRS) using Unreal Engine 4.27 for the main experiment. 3D-CRS contains 20 distinct indoor room scenes, each with RGB, depth, normal, semantic/instance masks, and camera trajectories. Notably, we provide the completed 3D scene meshes for each room, consisting of both 3D furniture meshes and room layout meshes. We are capable of generating an infinite amount of data from the 3D complete scene mesh, by using a virtual camera in Unreal Engine. For the experiments conducted in this paper, we only utilized depth images and 3D room meshes. We further employ the iTHOR dataset as the complementary experiment, which provides the complete 3D scene meshes. iTHOR is a near photo-realistic interactable framework for embodied AI agents, including 120 room-scale scenes manually modelled by 3D artists. From this collection, we selected 13 living room models and exported their scene meshes by Unity for our second experiment.

*Equally contributed as co-first author. This work was done during Su's internship at Bosch.

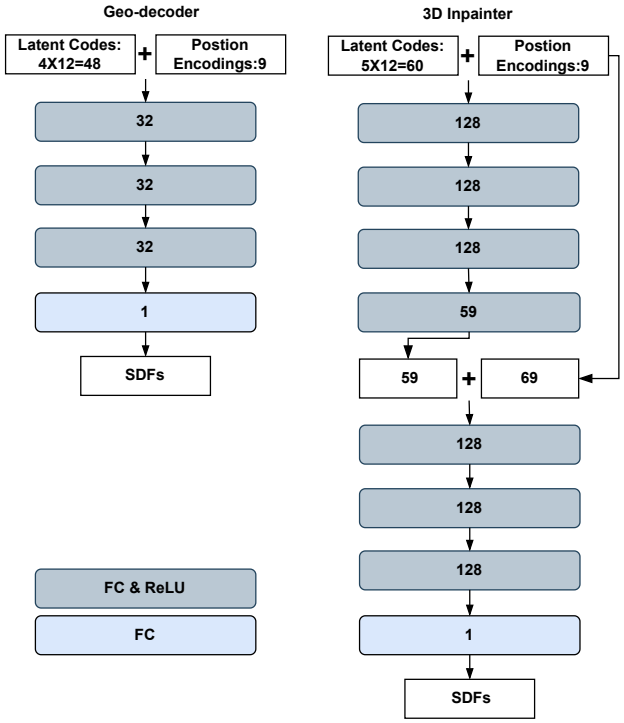

Figure 1. Network Architecture of Geo-decoder and 3D Inpainter

2. Metrics

Following the evaluation metrics in BNV-Fusion [4], the standard metrics Accuracy (Accu.), Completeness (Comp.) and F1 score (F1) are employed for the quantitative analysis. To be specific, we firstly uniformly sample 100,0000 points from the ground truth completed 3D meshes and generated 3D meshes, respectively, then compute the Accu., Comp. and F1 metrics. Accu. calculates the fraction of points from the reconstructed 3D mesh which is closer to points from the ground truth completed 3D mesh than a threshold distance of 2.5cm. Similarly, Comp. calculates the fraction of points from the ground-truth completed 3D

mesh which is closer to points from the reconstructed 3D mesh than a threshold distance of 2.5cm. The overall performance metric, F1, is defined as the harmonic mean of Accu. and Comp.

3. Baselines

In our comparison experiments, we aimed to demonstrate the accuracy of our proposed method by comparing it against three baseline methods: TSDF-Fusion [8], Go-Surf [7] and BNV-Fusion [4]. TSDF-Fusion [8], which is implemented in the Open3D library, is currently the SOTA explicit geometry-based 3D surface reconstruction method. BNV-Fusion [4], which utilizes neural implicit representations, is currently the SOTA implicit-style 3D surface reconstruction method. For a fair comparison with the baseline Go-Surf [7], we modified its architecture by omitting the RGB branch during both the training and inference phases. All the baselines, the same as our method, use only depth images as input, ensuring a fair basis for comparison. It's important to note that the encoder used in BNV-Fusion is pre-trained on the large-scale ShapeNet dataset, notably improving the accuracy of visible surface reconstruction. However, our method does not involve this particular pre-training step.

4. Implementation Details

Data Preprocessing:

1) Mesh Processing: It was noted that human-created meshes frequently have artifacts on the exteriors of room layouts. To maintain consistency and purity in our ground truth meshes, we manually culled these unwanted external structures. Once these meshes were culled, we further processed them to be watertight using [2], enabling the computation of Signed Distance Functions (SDF).

2) Depth Image Rendering: For each scene, we first generate the camera trajectories. These trajectories are defined through Catmull-Rom spline interpolation, anchored on a series of manually chosen control points. Using the Trimesh raytracer [1], we render depth images corresponding to each camera pose. The number of sampled depth frames in each scene of 3D-CRS and iTHOR dataset are given in Table 1 and Table 2 respectively. The depth images for both datasets are rendered at a resolution of 1024×768.

3) Point Sampling: To prepare the training data for our 3D Inpainter, we extract SDF samples from each training scene. Each mesh is first normalized, followed by sampling around 8×10^6 signed distances close to the surface and an equal number of signed distances uniformly distributed within the scene bounding box. For non-surface samples, we maintain a 50% ratio between positive and negative SDFs, representing visible free space and invisible interior of furniture, respectively.

Octree Feature Volume: The voxel resolution in our configuration is set to 2 cm, and the latent code assigned to each corner of the octree node is set to 12 dimension.

Network Architecture: Figure 1 illustrates the network architecture of the Geo-Decoder and 3D Inpainter. The Geo-Decoder employs a relatively shallow multilayer perceptron (MLP) consisting of 4 fully connected layers. Each of these layers is followed by a ReLU activation, except for the final layer. In contrast, the 3D Inpainter utilizes a network structure similar to DeepSDF [6], consisting of 8 fully connected layers. These layers are applied with weight normalization and interconnected via ReLU activations and a 0.3 dropout rate, except for the last layer. A skip connection is integrated at the 4th layer of the 3D Inpainter.

3D Inpainter Training: In the training phase of the 3D Inpainter, our approach starts with randomly selecting a scene, loading its octree feature volume along with training samples, and then optimizing both the features and the 3D Inpainter over several consecutive iterations. Subsequently, we save the feature volume and repeat this process for each scene until all scenes have been trained within an epoch. We experimentally found that 100 consecutive iterations per scene produces a desirable balance between generalization across different scenes and accuracy within a specific scene. The 3D Inpainter is offline trained using Adam optimizer with a learning rate of 1e-3. The training of our 3D Inpainter is conducted on an NVIDIA RTX 3090 GPU for 100 epochs, which takes about 8 hours.

Geo-decoder Optimization: The Geo-decoder is online optimized using Adam optimizer with a learning rate of 1e-2. The optimization process is conducted on an NVIDIA RTX 3090 GPU for 1000-10000 iterations, which approximately takes 3-30 mintues. The number of iterations can be chosen based on different application requirements, which is a trade-off between speed and accuracy.

Octree Feature Training and Optimization: During both offline training and online optimization, we observed that high-level features in the octree have less fluctuation and faster convergence during training, in contrast to low-level features. Therefore, we implemented a learning rate decay approach for octree features, progressively reducing the learning rate from high to low levels. This decay starts from an initial rate of 1e-3 and decreases for each level at a rate of 0.5.

3D Surface Generation: Our method is capable of inferring SDFs at arbitrary 3D locations. For the extraction of triangle meshes, we use Marching Cubes [5], applying a spatial resolution of 1cm.

5. Runtime Analysis

The network is implemented under the PyTorch framework and is tested on a machine equipped with an NVIDIA RTX 3090 GPU accelerated by CUDA and cuDNN. The infer-

Scene	*01*	*02*	*03*	*04*	*05*
Depth Frames	199	172	188	235	156
Scene	*06*	*07*	*08*	*09*	*10*
Depth Frames	235	223	240	176	121
Scene	*11*	*12*	*13*	*14*	*15*
Depth Frames	275	110	243	199	200
Scene	*16*	*17*	*18*	*19*	*20*
Depth Frames	155	210	200	187	278

Table 1. The number of depth images sampled from each scene in the 3D-CRS dataset.

FloorPlan	*202*	*205*	*206*	*207*	*210*
Depth Frames	100	100	100	100	100
FloorPlan	*213*	*217*	*219*	*220*	*225*
Depth Frames	100	100	100	100	100
FloorPlan	*226*	*228*	*229*		
Depth Frames	100	100	100		

Table 2. The number of depth images sampled from each scene in the iTHOR dataset.

ence runtime of our framework can be mainly divided into two distinct steps: 1) the octree feature volume building costs around 0.2 second per depth frame with 1024×768 resolution; 2) the Geo-decoder and feature optimization takes about 0.2 second per iteration. The number of iterations can vary between 1000 to 10000, allowing flexibility to balance speed and accuracy based on the specific application requirements.

6. Video Demo

We attach a video demo of visual comparison results in one scene. The video showcases the 3D surface reconstruction of the scene using our method, as well as the BNF-Fusion and TSDF-Fusion methods, along with the ground truth, while following a virtual camera trajectory.

7. 3D-CRS Dataset

To address the limitations of existing public indoor RGB-D datasets, which typically lack complete 3D meshes of room scenes and furniture, we build a novel dataset, named the 3D Complete Room Scene (3D-CRS) upon a commercial indoor 3D dataset provided by Coohome LLC [1]. Although the scene dataset from AI2-THOR [3] includes complete 3D scene meshes, their data does not meet the high-quality photorealistic standards, and the complexity of their room layouts is insufficient. Consequently. we created the 3D-CRS dataset to facilitate further research on 3D surface completion. The 3D-CRS dataset, illustrated in Figure 4: right, is meticulously crafted by 3D artists using Unreal Engine 4.27. It features 20 high-quality indoor scenarios that represent typical household rooms, including kitchens, bathrooms, living rooms, and bedrooms. We utilized the virtual camera in Unreal Engine to render an extensive amount of photorealistic data from these high-quality 3D scene models. The camera trajectory is determined using Catmull-Rom spline interpolation, based on a series of manually selected control points, aiming to encompass the majority space of each room scene. We render 2D images at 1024×768 resolution using a 120-degree FOV camera along the camera trajectory, under the automatic exposure and default indoor light in Unreal Engine. As shown in Figure 2, each scene in the 3D-CRS dataset includes RGB images, depth images, normal images, semantic/instance masks, and albedo. Besides the 2D rendering data, we also provide the complete 3D scene meshes (Figure 4: left) for each room, consisting of both complete 3D furniture meshes (Figure 3) and complete 3D room layout meshes (Figure 4: middle). The 3D-CRS dataset's configuration allows AI agents to interact with objects in these environments in various ways. Our 3D scene models are designed to be seamlessly compatible with robotics simulations, such as NVIDIA's Isaac Sim[2]. Our 3D-CRS dataset greatly enhances the potential for research in 3D surface completion and other robotics-related fields.

[1] https://www.coohom.com/b2b

[2] https://developer.nvidia.com/isaac-sim

References

[1] Dawson-Haggerty et al. trimesh. 2

[2] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 2

[3] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1, 3

[4] Kejie Li, Yansong Tang, Victor Adrian Prisacariu, and Philip HS Torr. Bnv-fusion: Dense 3d reconstruction using bi-level neural volume fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6166–6175, 2022. 1, 2

[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. 2

[6] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2

[7] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022. 2

[8] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 2

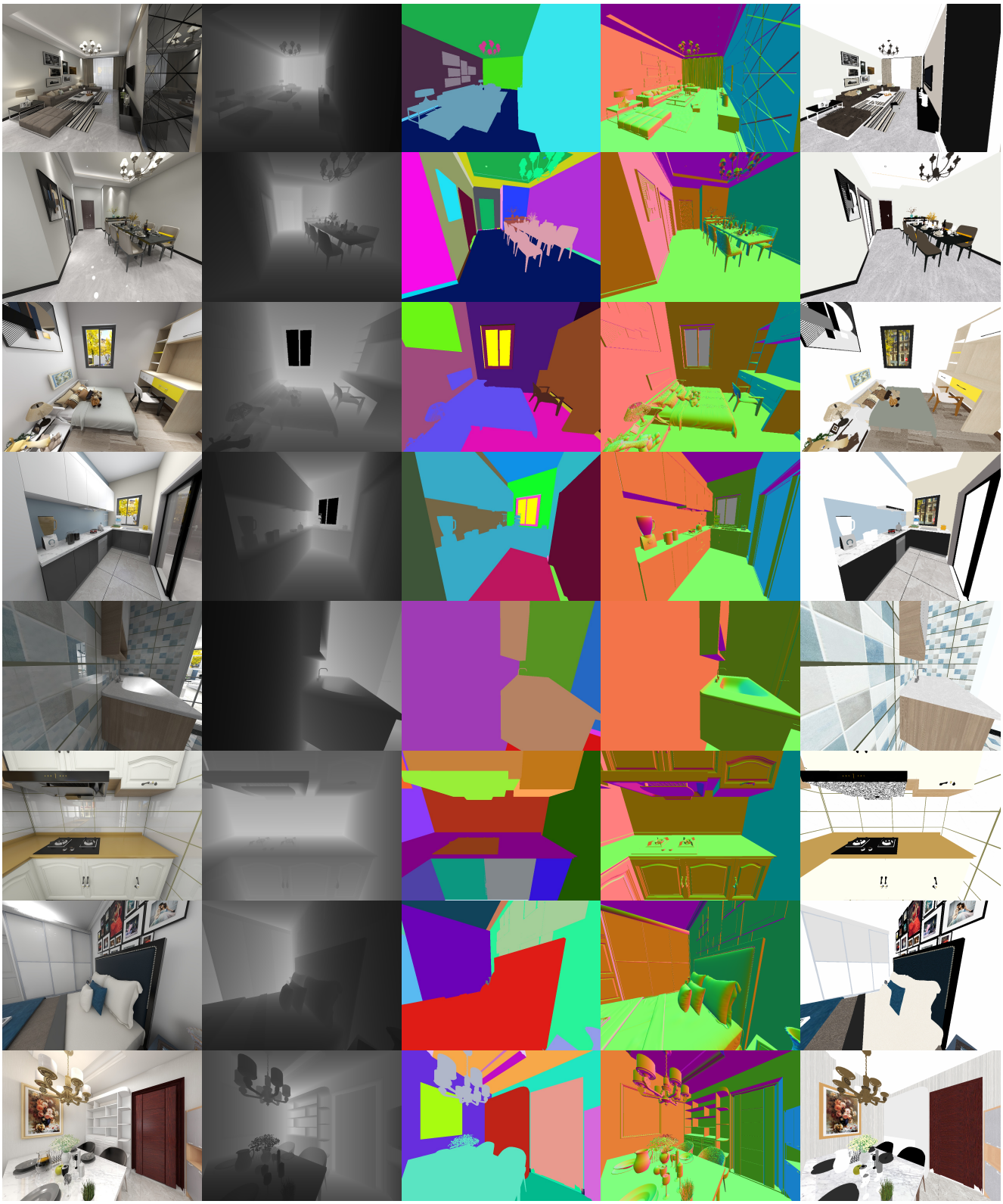

Figure 2. Some examples of 2D rendered data from the 3D-CRS dataset: showcasing from left to right, RGB, depth, semantic mask, normal and albedo. We only use depth images and 3D meshes in the experiment of this manuscript.

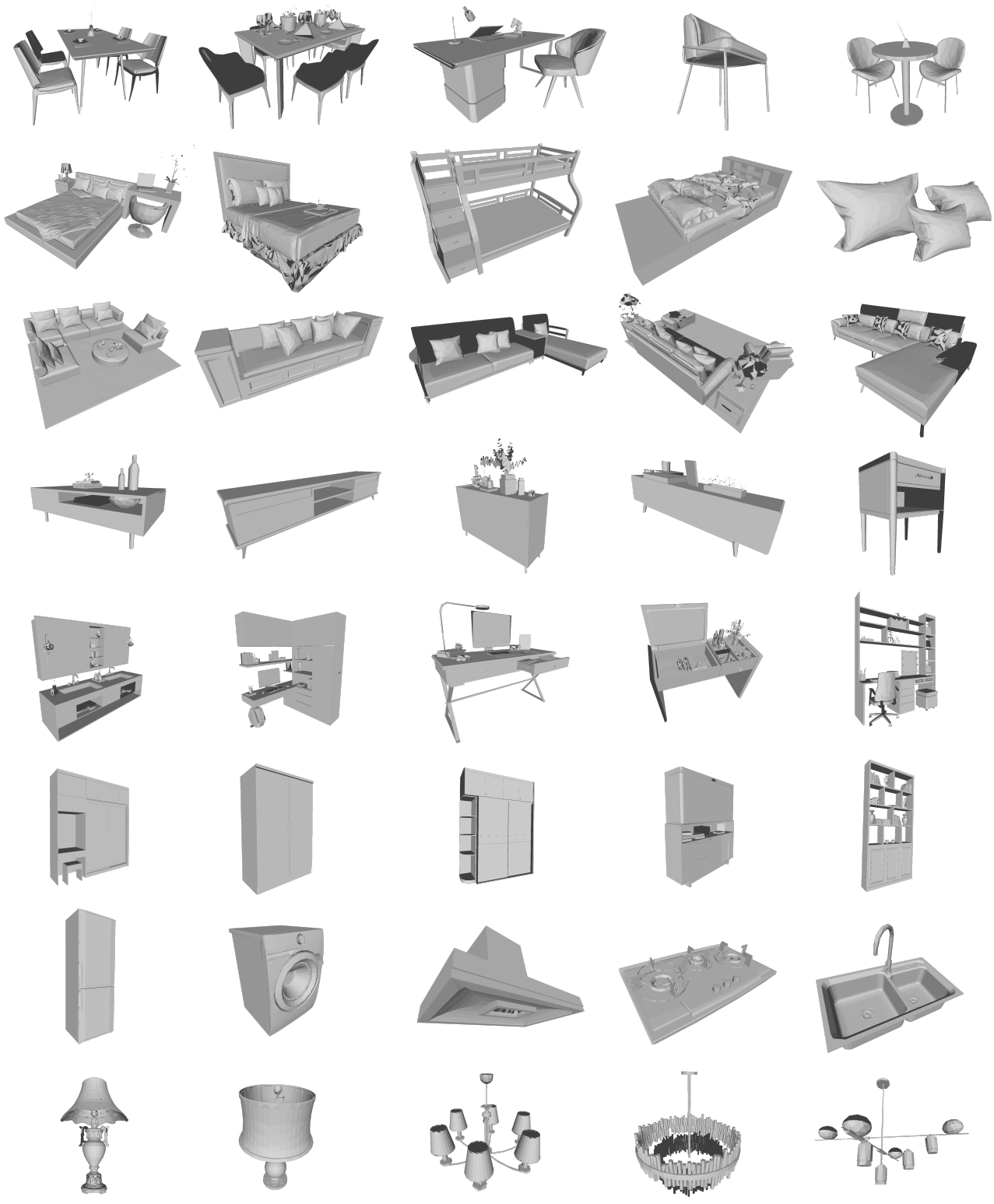

Figure 3. Some examples of complete 3D furniture mesh from the 3D-CRS dataset.

Figure 4. Some examples of complete 3D scene mesh (left), 3D room layout mesh (middle) and 3D scene model (right) from the 3D-CRS dataset.