CASAGPT: Cuboid Arrangement and Scene Assembly for Interior Design

Supplementary Material

In this supplementary material, we include the following details: implementation specifics in Appendix A, dataset-related statistics in Appendix B, computation cost in Appendix C, cuboid generation procedure evaluation in Appendix D, some failure cases of CASAGPT in Appendix E and further visual results Appendix F.

A. Implementations

A.1. Model architecture

The model architecture is based on LLaMA-3 with modifications to the model size and position encoding. Specifically, the model consists of 8 layers, with a hidden dimension of 512 and a parameter size of 27.4 million. It uses the SwiGLU activation function and a learned position encoding. Additionally, both the number of attention heads (*Head n*) and key-value heads (*KV head*) are set to 8.

A.2. Cuboid merging algorithm

Voxel space segmentation. We describe the detailed algorithm for the initial simple segmentation on the target voxel O here.

We start by studying how to cover the binary voxel representation using at most k rectangles while minimizing the total volume. Given a binary 3D array where 1 represents the target shape and 0 represents the background, we begin by identifying the minimum bounding box that encloses all target voxels.

We then iteratively refine this initial box through a dynamic programming approach. At each iteration, we identify the rectangle that, when subdivided, would eliminate the largest contiguous region of background voxels.

The subdivision process depends on the position of the removed background region. When removing a background region from a corner, the remaining volume is partitioned into 2 new rectangles. When removed from an edge (but not a corner), the remaining volume is partitioned into 3 new rectangles. When removed from the interior (not touching any edge), the remaining volume is divided into 4 new rectangles, splitting along horizontal or vertical planes. This position-dependent subdivision optimally preserves the target geometry while eliminating empty spaces.

The algorithm considers both horizontal-first and vertical-first splitting strategies, selecting the one that better minimizes total volume. The iterative refinement continues until either the maximum allowed number of rectangles k is reached or no further optimization is possible. This results in a compact representation that closely approximates the original voxel shape while maintaining a controlled number of segments. This initial segmentation serves as the foundation for subsequent refinement steps in our pipeline.

Dynamic threshold. To encourage the merging of smaller cuboids while penalizing the merging of larger ones, we introduce a dynamic threshold mechanism. The dynamic threshold τ_{dynamic} is computed as a function of the merged cuboid's volume, using the following formula:

$$\tau_{\text{dynamic}}(V_C) = \tau_{\text{min}} + (\tau_{\text{max}} - \tau_{\text{min}}) \cdot \exp\left(-\frac{V_C}{S}\right), \quad (1)$$

where τ_{min} and τ_{max} are the minimum and maximum threshold values, V_C is the volume of the merged cuboid C, and S is a scaling parameter that controls the rate of exponential decay.

The merging algorithm proceeds iteratively. We attempt to merge each pair of cuboids using the predefined threshold (either static or dynamic). If merging is successful, the two cuboids are replaced by the merged cuboid, and the process continues until no further merges are possible.

we provide an algorithm to illustrate the cuboid merging process, which plays a key role in maintaining an efficient representation of 3D scenes. The pseudocode for the algorithm is presented in Algorithm 1.

B. Dataset

B.1. Failure cases in 3D-FRONT

As described in the main text, the 3D-FRONT dataset contains numerous object intersections. As shown in Figure 1, the 3D-FRONT dataset contains numerous examples of object intersections. These intersections can occur between various objects, such as tables and chairs, beds and nightstands, as well as wardrobes.

B.2. 3D-FRONT and 3DFRONT-NC comparison

We computed the non-intersection rate (NIRate) for both the 3D-FRONT and 3D-FRONT-NC datasets. Figure 3 illustrates our findings, showing that our curated dataset, 3D-FRONT-NC, has significantly fewer object intersections.

B.3. Distributions of 3DFRONT-NC

Figure 4 illustrates the average cuboid lengths across various furniture categories. The category related to desks exhibits the longest average cuboid length, highlighting the inherent complexity of desks.

Algorithm 1 Cuboid Merging with Dynamic Threshold

Input: List of cuboids $C = \{C_1, C_2, \ldots, C_n\}$, initial threshold τ_{init}, dynamic scaling factor S
Output: Merged cuboids list $\hat{C}$
Initialize $\hat{C} \leftarrow C$
Set *changed* to `True`
while *changed* **do**
  Set *changed* to `False`
  Select pairs of adjacent cuboids from $\hat{C}$
  for each selected pair (C_i, C_j) **do**
    Calculate the bounding cuboid C_{bounding}
    Compute volumes V_{C_i}, V_{C_j}, and $V_{C_{\text{bounding}}}$
    Calculate dynamic threshold $\tau_{\text{dynamic}}(V_{\text{C}})$
    if $\frac{V_{\text{C}}}{V_{\text{A}}+V_{\text{B}}} < \tau_{\text{dynamic}}$ **then**
      Merge C_i and C_j into C_{bounding}
      Remove C_j from $\hat{C}$
      Set *changed* to `True`
      break from the loop once a merge occurs
    end if
  end for
end while
return $\hat{C}$

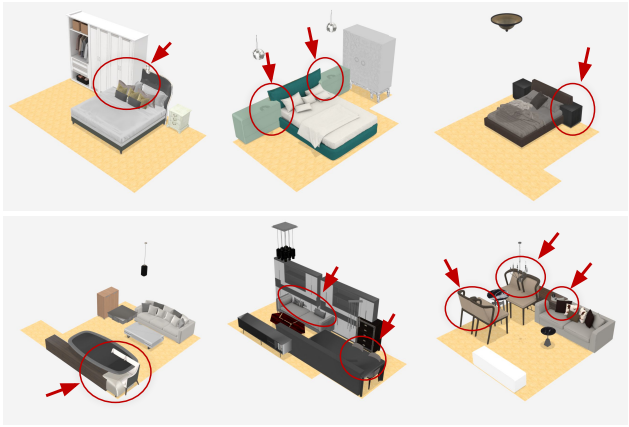

Figure 1. Examples of object intersections in the 3D-FRONT dataset, with collided objects highlighted in red circles.

We also analyzed the cuboid representation lengths in bedroom, living room, dining room, and library scenes and plotted their distributions. Figure 2 illustrates the distribution of cuboid lengths in each scene. It can be observed that the living room and dining room have longer cuboid lengths, likely due to the higher number of objects in these scenes.

C. Computation cost

We compare the model size, inference time per forward pass, as well as the GPU time and GPU memory consumption during training of our method with others in Table 1.

Additionally, We measured the average object retrieval time over 1,000 sampled scenes and compared it with ATISS and DiffuScene. The average retrieval time per object for ATISS, DiffuScene, and our method is 0.57s, 0.45s, and 0.70s, respectively, showing that our approach adds only a small overhead.

Methods	Params (M)	Forward time	GPU hours	GPU Mem.	Retr. (s/obj)
ATISS	8.4	36.68 ms	190 h	1.99 G	0.57
DiffuScene	77.6	41.12 ms	185 h	3.69 G	0.57
Ours	27.4	29.63 ms	55 h	11.8 G	0.70
Ours w/ rej.	27.4	29.63 ms	175 h	11.8 G	0.70

Table 1. Comparison of computation cost for different methods.

D. Cuboid generation procedure evaluation

During the initial experiments, we explored some shape abstraction methods, but we found that some methods [1, 2] do not meet our goal of modeling with cuboids, and some methods are trained on specific ShapeNet categories and lack sufficient generalization to 3D-FRONT. Therefore, we designed our own cuboid shape abstraction method that works well on 3D-FRONT.

We compare our method with a relatively recent work [3] on 3D-FRONT. For convenience, we refer to it as CAVS. We use the authors' checkpoint trained on ShapeNet Chair and test it on 3D-FRONT Dining Chair. Our results, shown in Tab. 2 and Fig. 5, demonstrate superior performance.

Method	ChamferDistance-L_1 ↓	ChamferDistance-*fscore* ↑	3D-IoU ↑
CAVS. [3]	0.013	0.331	0.246
Ours	0.008	0.412	0.368

Table 2. Comparison between our shape abstraction method with CAVS. Our shape abstraction method does not rely on neural networks, yet still achieves significant improvements.

E. Failure cases of CasaGPT

Cuboid generation plausibility. We perform object retrieval by computing 3D IoU on voxel grids. For a generated object's cuboid assembly, we compute its OBB box, align it to the origin, and select the candidate with the highest 3D IoU. Fig. 6 highlights extreme anomaly cases with very low max 3D IoU. In most cases, our cuboid sequences resemble reasonable objects and failure cases are rare.

Cross boundary. Figure 7 presents some failure cases of our method. Firstly, boundary constraints were not explicitly considered during training, leading to results where some objects exceed the floor boundaries (first row).

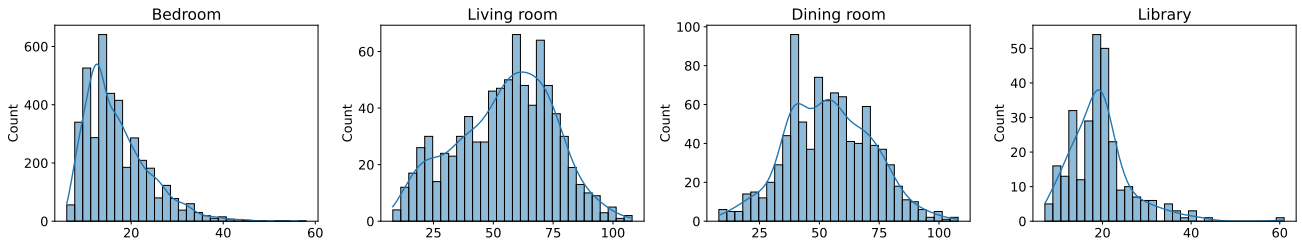

Figure 2. Distribution of cuboid representation lengths in different room scenes. The histograms depict the frequency of cuboid lengths in bedroom, living room, dining room, and library scenes, showcasing the variation in cuboid sizes across these environments.

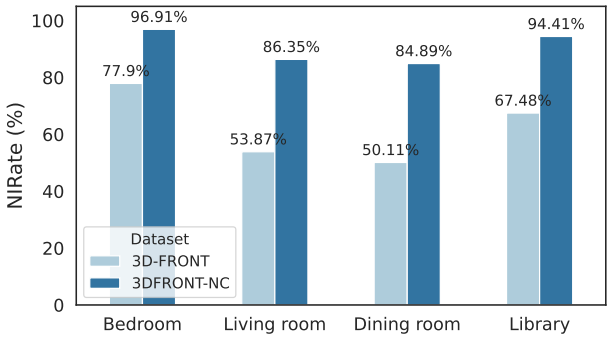

Figure 3. Comparison of non-intersection rates (NIRate) between the 3D-FRONT and 3D-FRONT-NC datasets across different room types. The curated 3D-FRONT-NC dataset exhibits significantly higher NIRate, indicating fewer object intersections, particularly in bedrooms and libraries.

Ergonomic issue. Secondly, while we aim for the generated results to comply with *ergonomic* principles, our model occasionally produces results that do not meet these standards. For instance, in the first column of the second row, there are areas on the floor that are inaccessible to people and disconnected from other floor areas. In the second column of the second row, there is no lighting. In the third column of the second row, a chair is placed in an inaccessible location. In the fourth column of the second row, the sofa's main orientation does not face the TV cabinet.

F. More visual results

We present additional results demonstrating the capabilities of our model in various tasks. As shown in Figure 8, our model excels in scene completion tasks. Figure 9 shows our model exhibits randomness and diversity. Furthermore, we tested our model on customized floor plans, achieving strong performance on these tailored designs, as illustrated in Figure 10. Finally, our model's effectiveness in synthesis tasks is showcased in Figures 11 to 13.

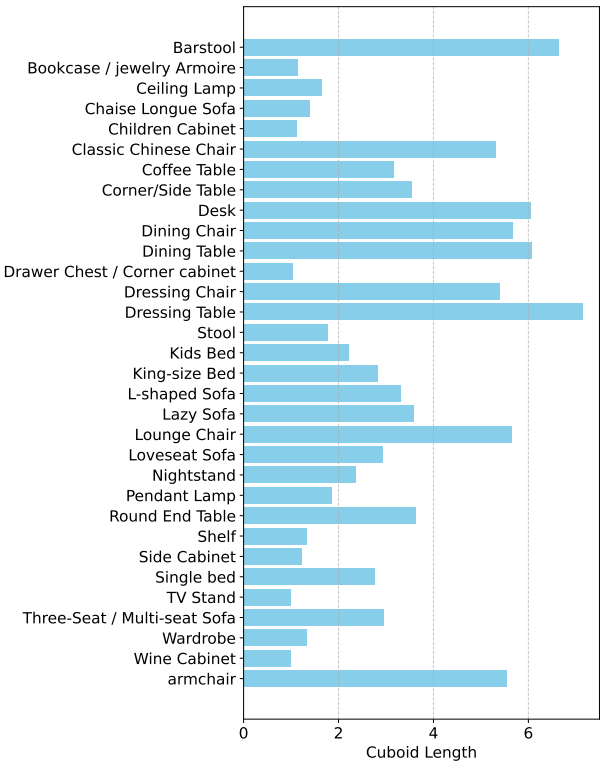

Figure 4. Average cuboid length for different furniture categories. Note that "*footstool / sofa stool / bed end stool / stool*" is simplified as "*stool*", "*lounge chair / cafe chair / office chair*" as "*lounge chair*", and "*sideboard / side cabinet / console table*" as "side *cabinet*".

References

[1] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *CVPR*, 2020. 2

[2] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics Revisited: Learning 3D shape parsing beyond cuboids. In *CVPR*, 2019. 2

[3] Kaizhi Yang and Xuejin Chen. Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. *ACM TOG*, 2021. 2

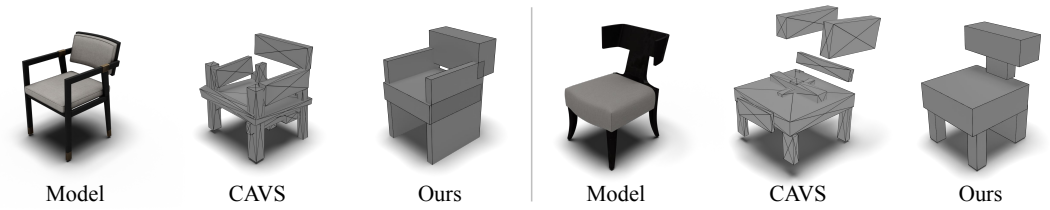

Figure 5. CAVS performs reasonably well on relatively in-distribution models (left), but 3D-FRONT contains many OOD models (right), where CAVS struggles.

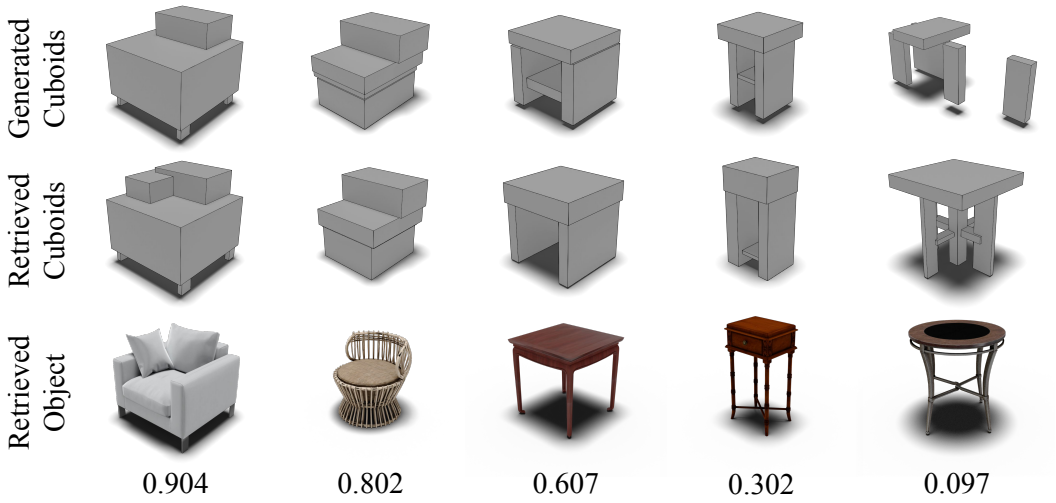

Figure 6. Cuboid assemblies and retrieved objects under different max 3D IoU conditions.

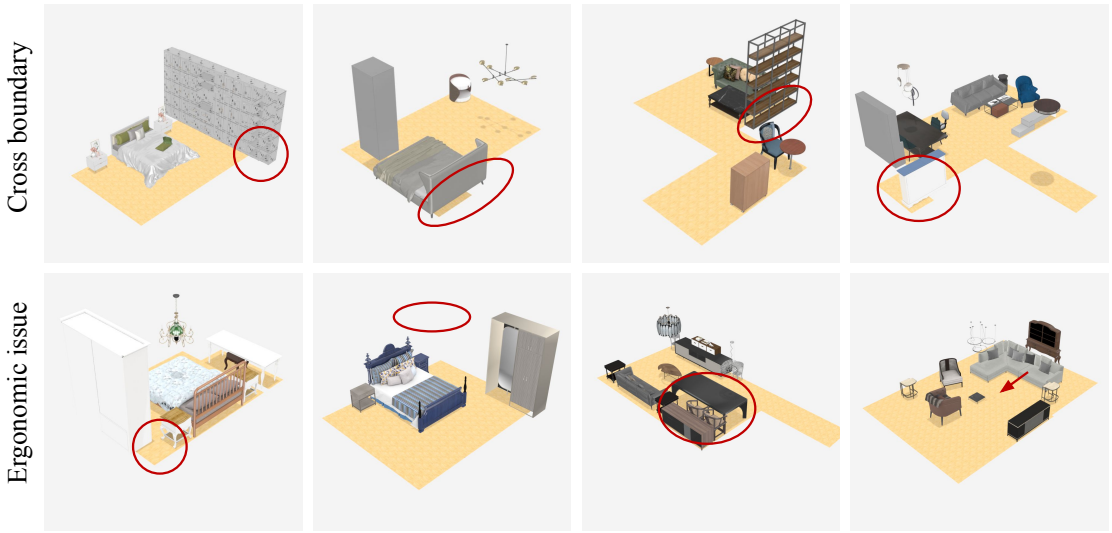

Figure 7. Failure cases of our method, including boundary crossing (first row) and ergonomic issues (second row).

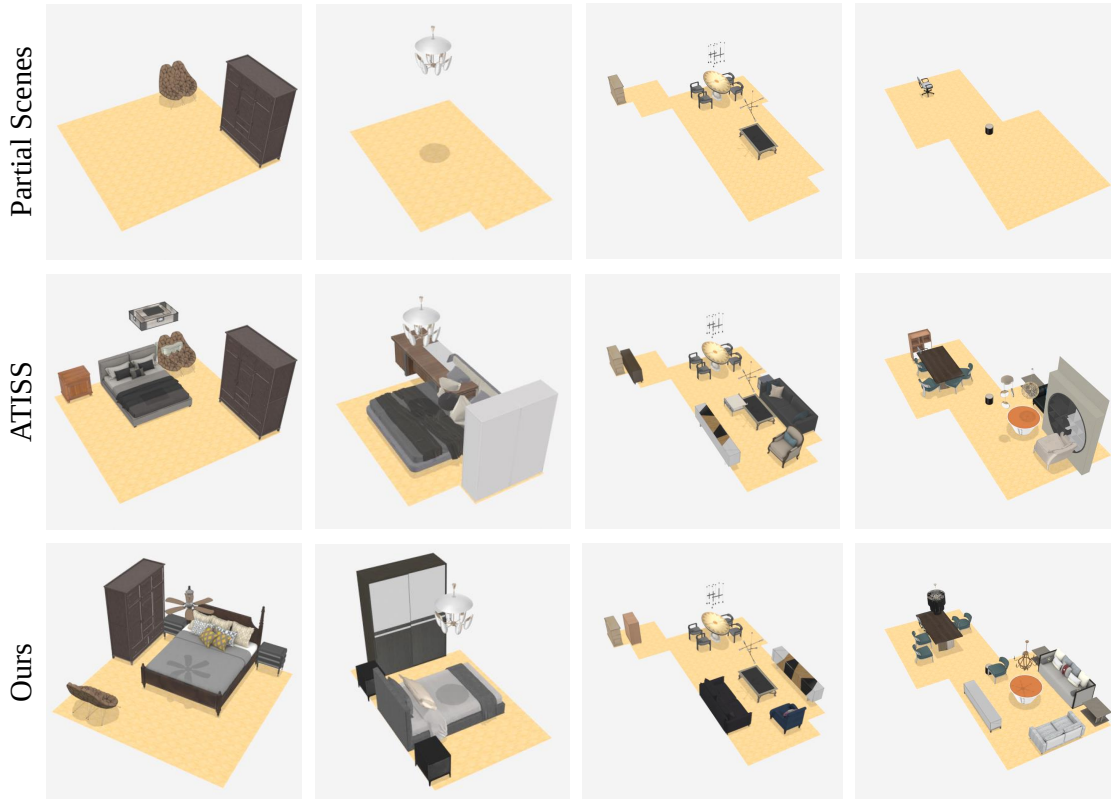

Figure 8. Comparison of scene completion results between ATISS and our method. We present results for the *bedroom* scene (left two columns) and the *living room* scene (right two columns).

Figure 9. Sampling results for 2 different floor plans in the living room scene.

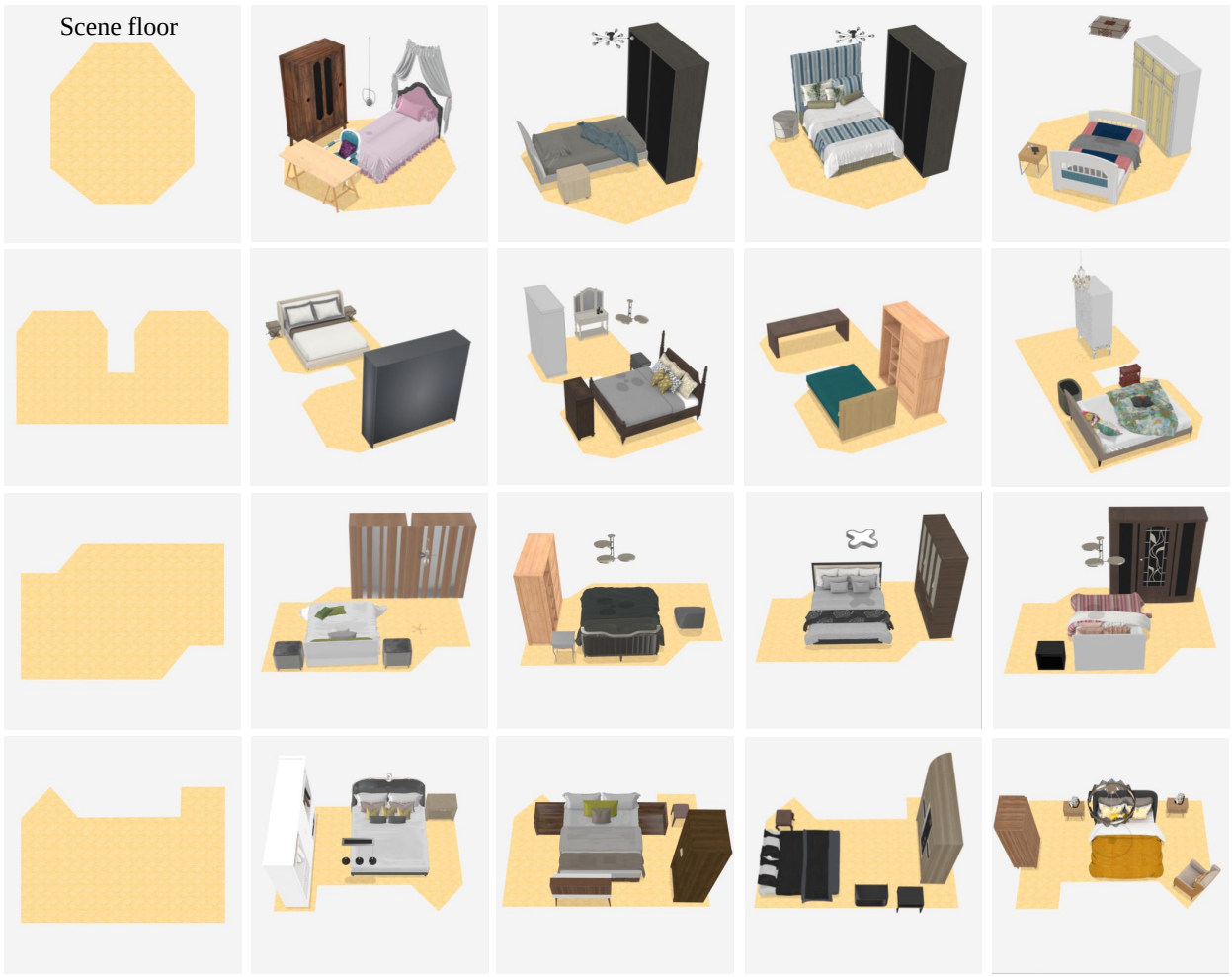

Figure 10. Generalization beyond training data. We present four synthesized bedrooms generated based on four customized room layouts using our model.

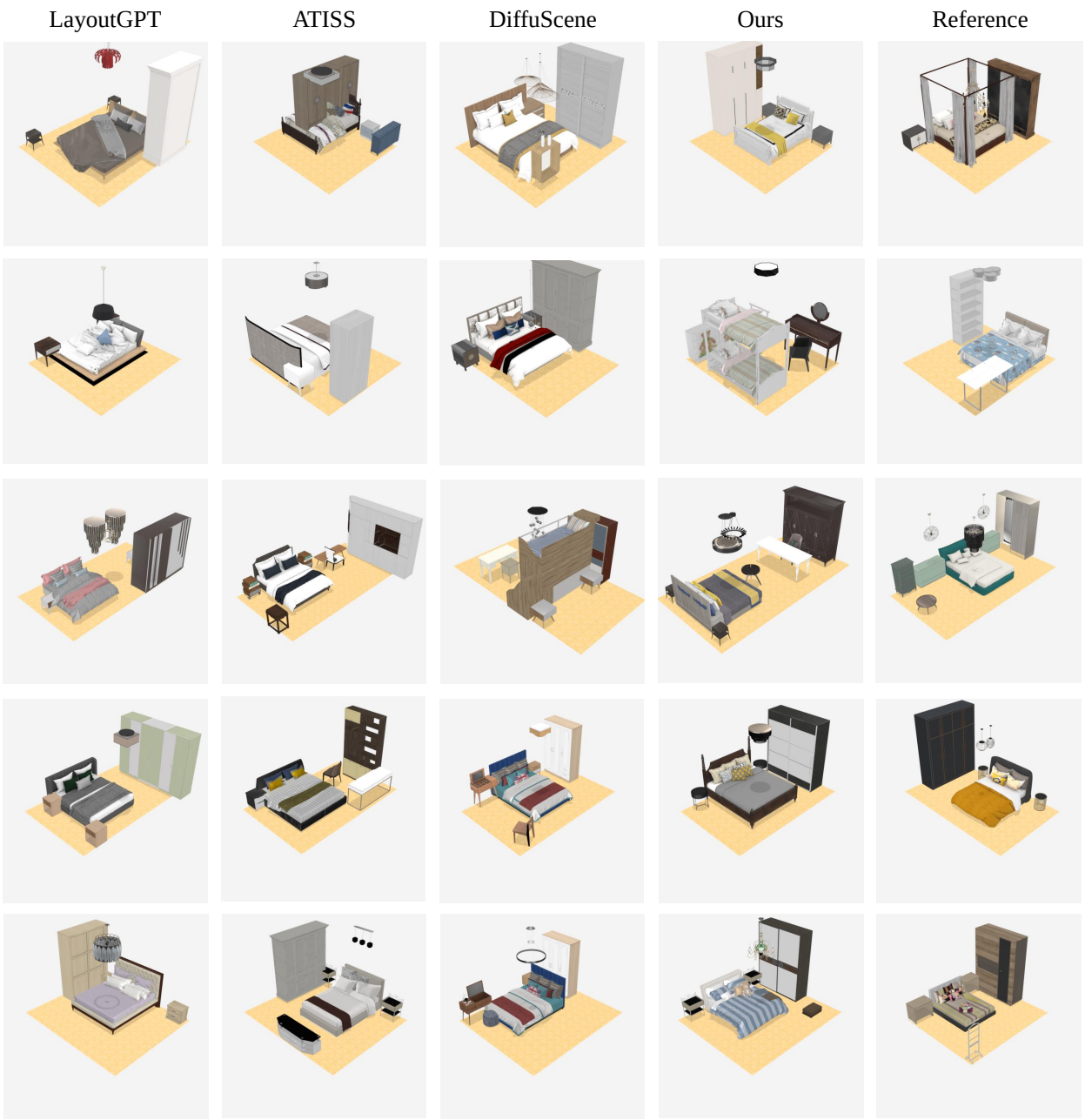

Figure 11. Additional results of *bedroom* scene synthesis. We compare our method with the state-of-the-art methods, where our results present better plausibility with fewer object collision issues.

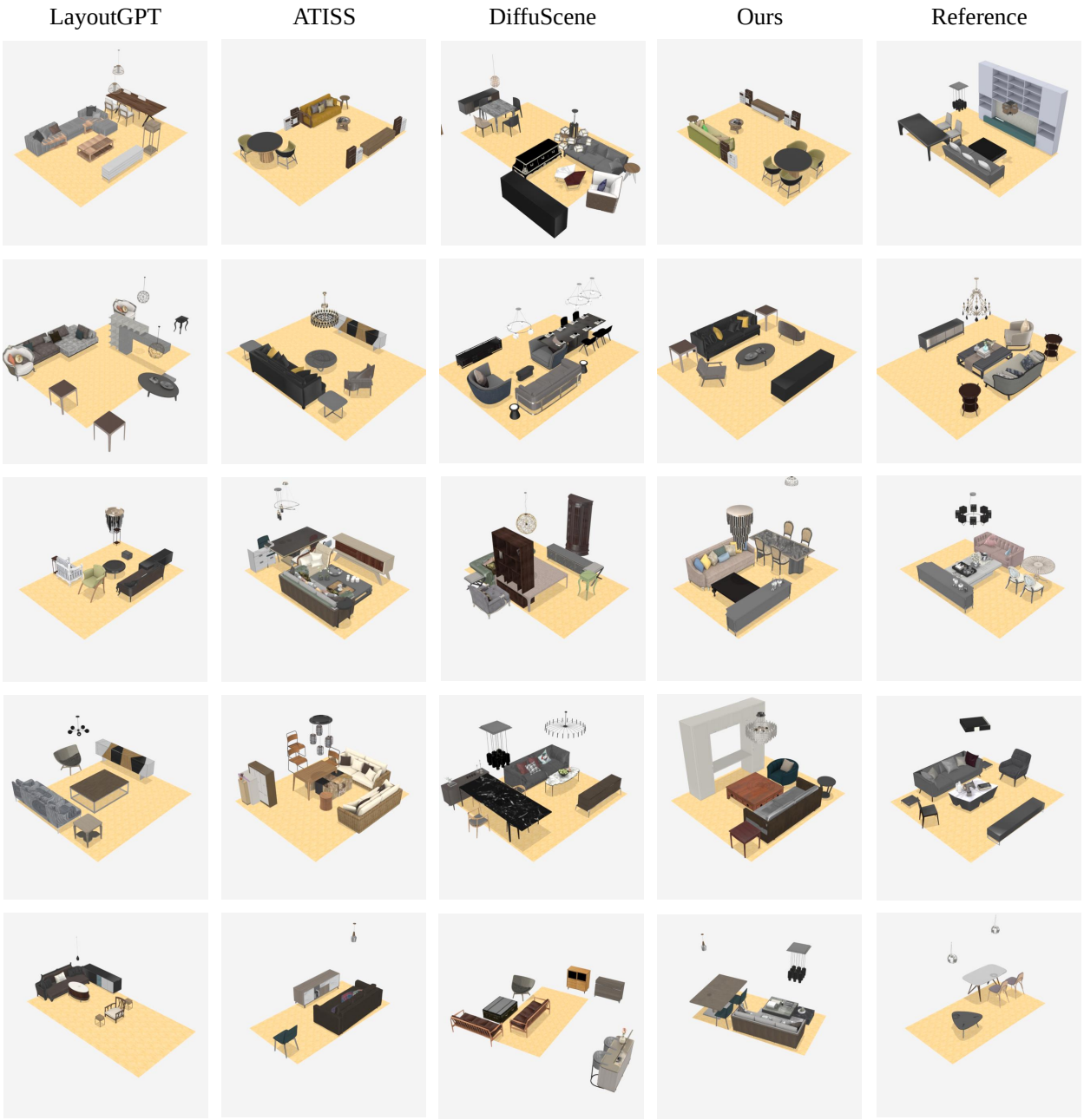

Figure 12. Additional results of *living room / dining room* scene synthesis show that compared to the state-of-the-art methods, our approach has fewer object collision issues, especially in the pairing of tables and chairs.

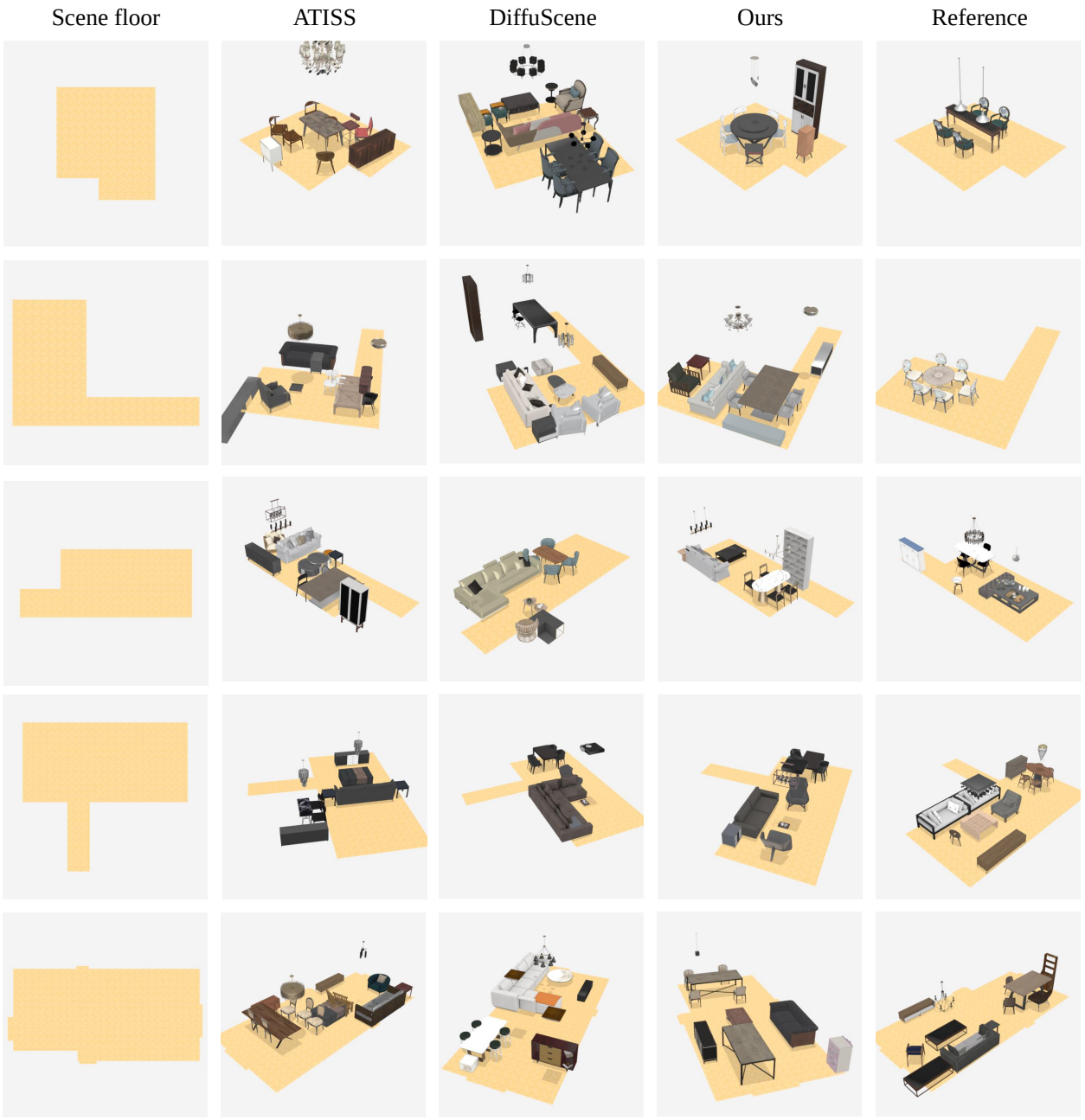

Figure 13. Additional results of *living room / dining room* scene synthesis with various floor planes demonstrate that our method can fit well to the given floor layouts.