Supplementary Material

Abhinav Narayan Harish
CVIG Lab, IIT Gandhinagar
abhinav.narayan@iitgn.ac.in

Rajendra Nagar
IIT Jodhpur
rn@iitj.ac.in

Shanmuganathan Raman
CVIG Lab, IIT Gandhinagar
shanmuga@iitgn.ac.in

A. Overview

In this document, we include additional qualitative results that we could not fit in the main paper. We further explain in detail the different part-orderings and architectural variants used in our ablation study. We will release our code and pre-trained models upon acceptance.

B. Training Details

We train our model with weight 10 assigned to $\mathcal{L}_r$ and 1 assigned to $\mathcal{L}_t$ and $\mathcal{L}_s$. We use Adam optimizer with standard parameters and a learning rate of $1e^{-3}$ with a decay of 0.9 every 5000 steps of training. During each training step, we execute our model 5 times and backpropagate using the minimum-over-N (MoN) [1] loss. We train each model till convergence. Training takes 2.5 days for the chair, 1.5 days for the lamp and 4 days for the table category on the NVIDIA RTX 2080Ti GPU.

Baseline Methods. We train the baseline methods B-Global [3, 5], B-LSTM [7] using the open-source code provided by the authors of [2]. We conduct training using the standard hyper-parameter settings provided by the authors. For the baseline B-DGL [2], we use their pre-trained models.

C. Shape Recovery from Latent Space

Our latent space contains information about the shape structure. For shape recovery, we utilize the point-cloud decoder of TreeGAN [6]. Although TreeGAN was developed for point-cloud generation, we customize it for our application by removing the discriminator.

We concatenate the forward and reverse hidden state of the last time-step $\mathbf{g}_1^{(t)}$ and $\mathbf{h}_N^{(t)}$ and process it using an MLP to a compact latent code $\mathbf{y}^{(t)} \in \mathbb{R}^{256}$. We sample 2048 points from each assembled shape to create the ground truth. The network architecture utilizes 5 layers of tree-based graph convolution with feature dimensions of $\{64, 32, 32, 16, 3\}$ and an upsampling of $\{1, 2, 4, 8, 32\}$. The network is trained with a learning rate of $1e^{-4}$ for 100 epochs. We conduct our experiments on the two largest categories of our dataset the chair and table. Our additional qualitative results are included in Figure 4. For more detail on the network architecture please refer to [6].

D. Optimal Order for Assembly

In our main paper, we presented various ways to order part-components and justified that the top to bottom order of a shape is the optimal one. In this section, we explain each of these orders in more detail. We use $\mathcal{A}$ to represent the inter-part adjacency matrix and $\mathcal{B}$ to represent a group of similar parts.

- **Part-wise connectivity order.** In this ordering we maintain a queue to decide the order in which to process parts. We start at a random part $\mathbf{P}_i$ and add to the queue all parts $\mathbf{P}_j$ directly connected to it, i.e, $\mathcal{A}(i, j) = 1$. If there are multiple neighbours for a given part, we consider them in the order they appear in the adjacency matrix. We iteratively carry out this process until the all parts are considered.
- **Group-wise connectivity order.** In this setting, we create a group adjacency matrix by combining similar parts. Two groups $\mathcal{B}_m$ and $\mathcal{B}_n$ are connected if for any $\mathbf{P}_i \in \mathcal{B}_m$ and $\mathbf{P}_j \in \mathcal{B}_n$, $\mathcal{A}(i, j) = 1$. Then, similar to part-wise connectivity, we start from a random group $\mathcal{B}_k$ and iteratively processing the neighbouring groups.
- **Volume Ordering.** We sort parts from minimum volume to maximum volume. We measure volume as the axis-aligned bounding box volume of the component point-cloud in its canonical PCA orientation. If a component point-cloud belongs to a part-group, we assign it minimum volume among the components belonging that group.
- **Central-Part Connectivity.** We define the central part as the one with maximum neighbours. We then follow a similar procedure as part-wise connectivity starting at the central part.
- **Top-Bottom Order.** We consider parts in the canonical top-bottom ordering of the PartNet [4] dataset. This order resembles the group-connectivity order, with similar groups being placed together. However, the

starting group is the group occurring at the top of the assembled object.

- **Random Order:** We randomly order the parts.

E. Structural Variants

In this section, we discuss the structural variations of our framework.

- **Bottom to Top Encoding.** We use a unidirectional GRU, which operates in the top to bottom ordering.
- **Top to Bottom Encoding.** We use a unidirectional GRU, which operates in the bottom to top ordering.
- **Initializing Hidden States.** The hidden states at subsequent time-step $(t + 1)$ are initialized with final hidden state of the current time-step (t).
- **Noise in Pose Decoder.** In this variant, we evaluate the performance of our algorithm with part-wise randomness. We do this by adding noise to the pose decoder (part-wise randomness) instead of the hidden state (global randomness).
- **Without Graph Learning.** We remove the graph learning module of our algorithm and only test the performance of our progressive framework.
- **Sequential before Graph.** The sequential module is incorporated first to obtain the updated features. The updated features are processed along with the part-message for pose-regression.

F. Additional Qualitative Results

In this section, we present additional qualitative results of our algorithm on the table (Figure 1), chair (Figure 2) and lamp (Figure 3) category. We also include additional results of shape recovery in Figure 4.

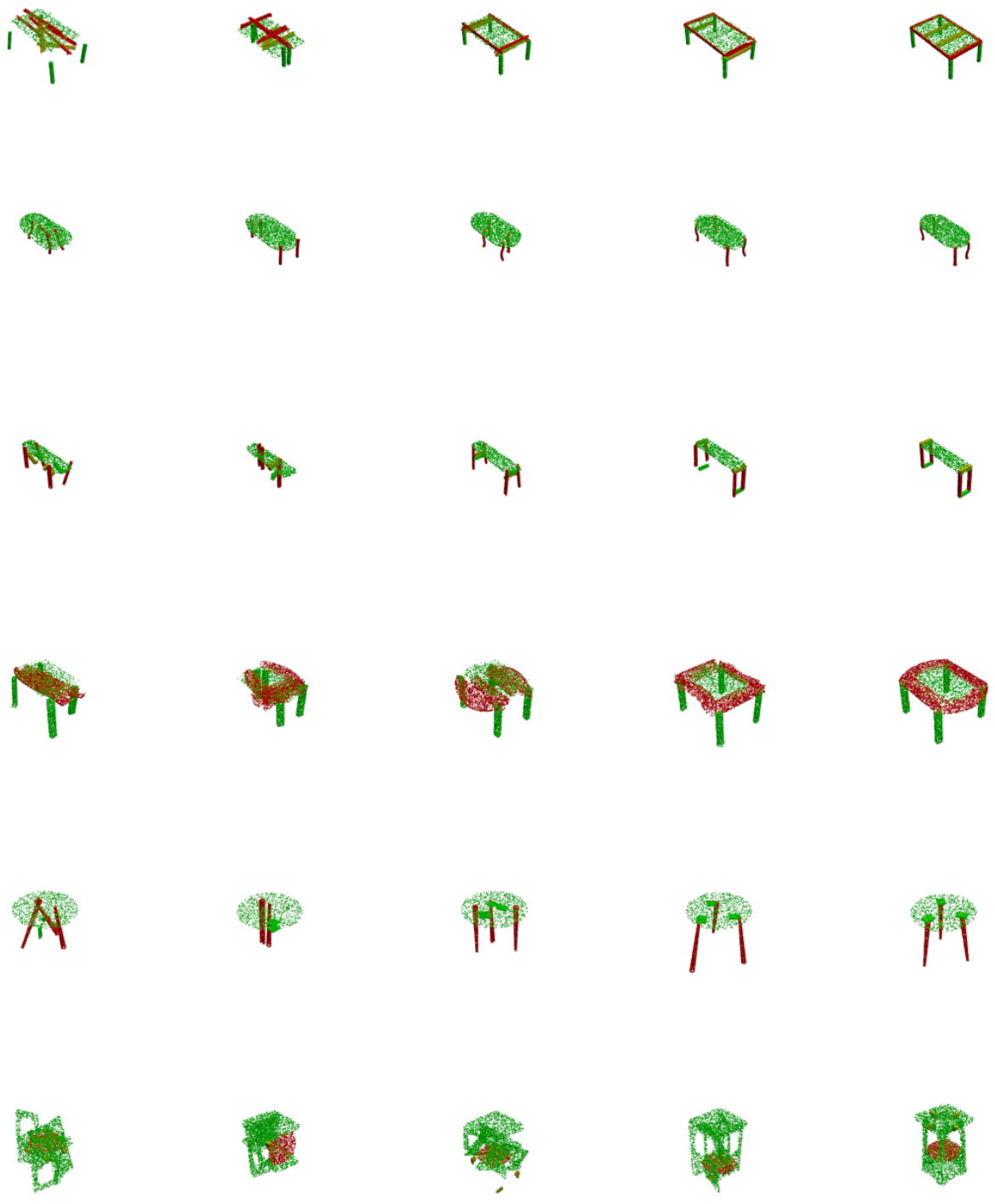

Figure 1. Additional qualitative comparison on the PartNet Table dataset

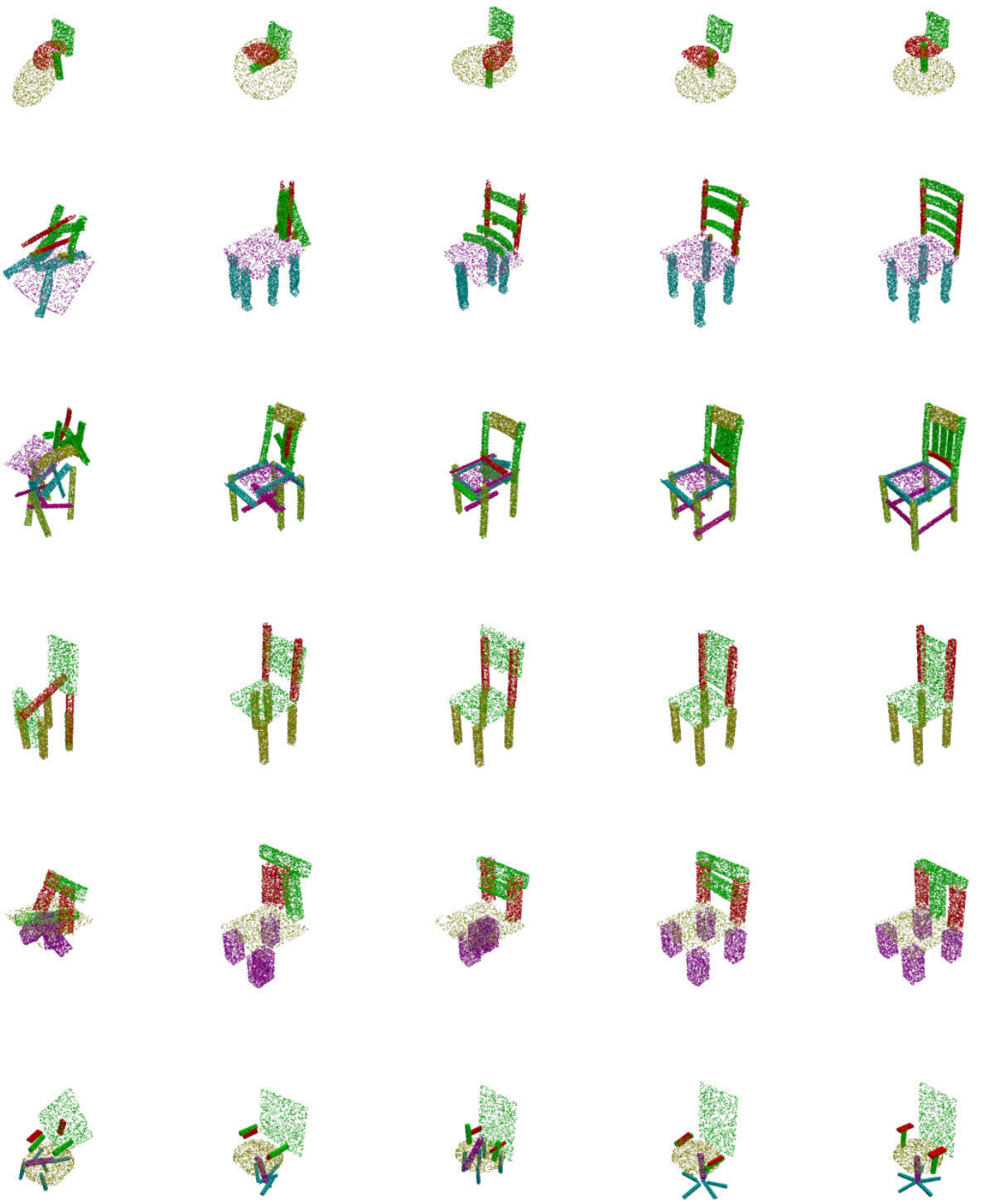

Figure 2. Additional qualitative comparison on the PartNet Chair dataset

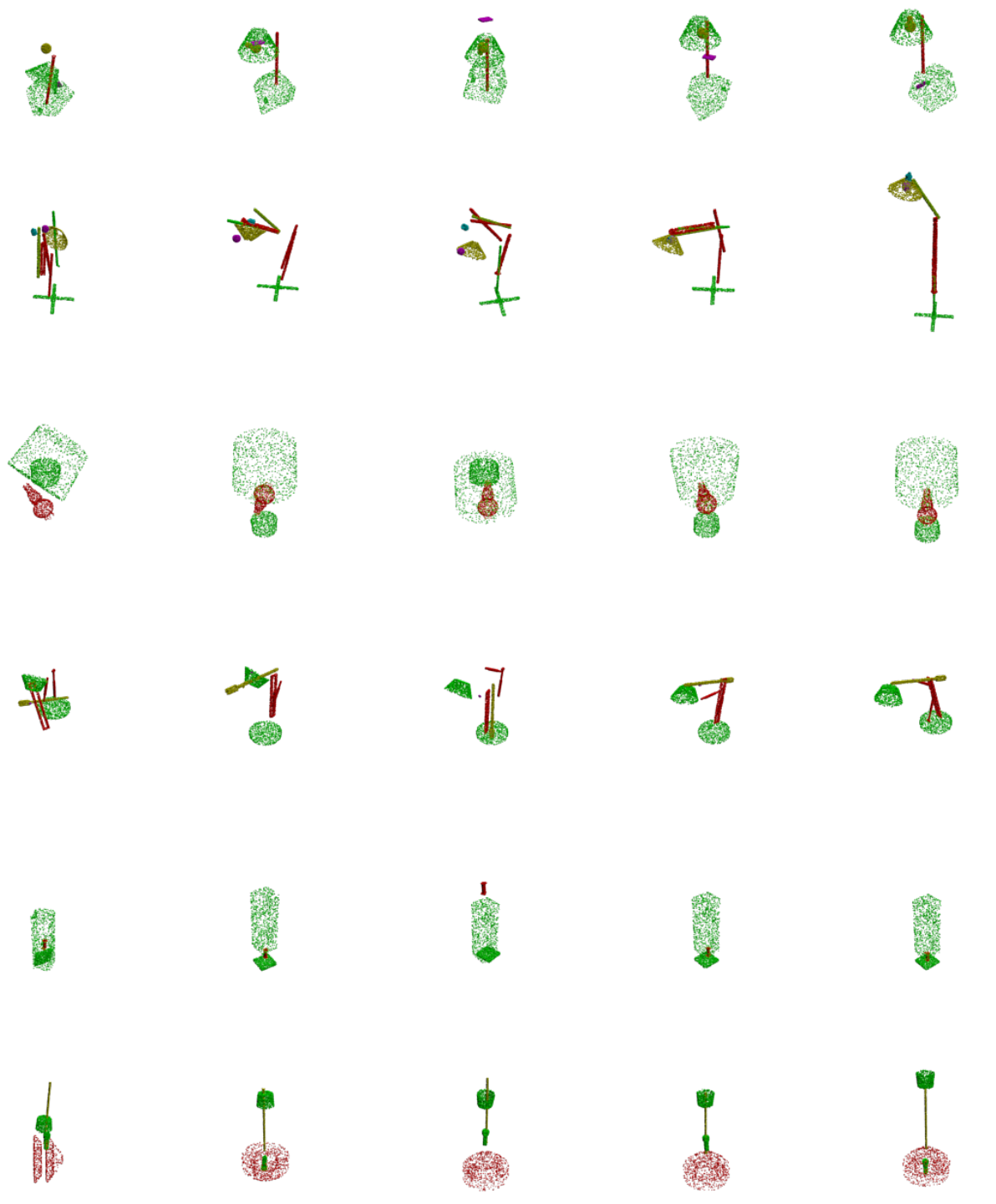

Figure 3. Additional qualitative comparison on the PartNet Lamp dataset

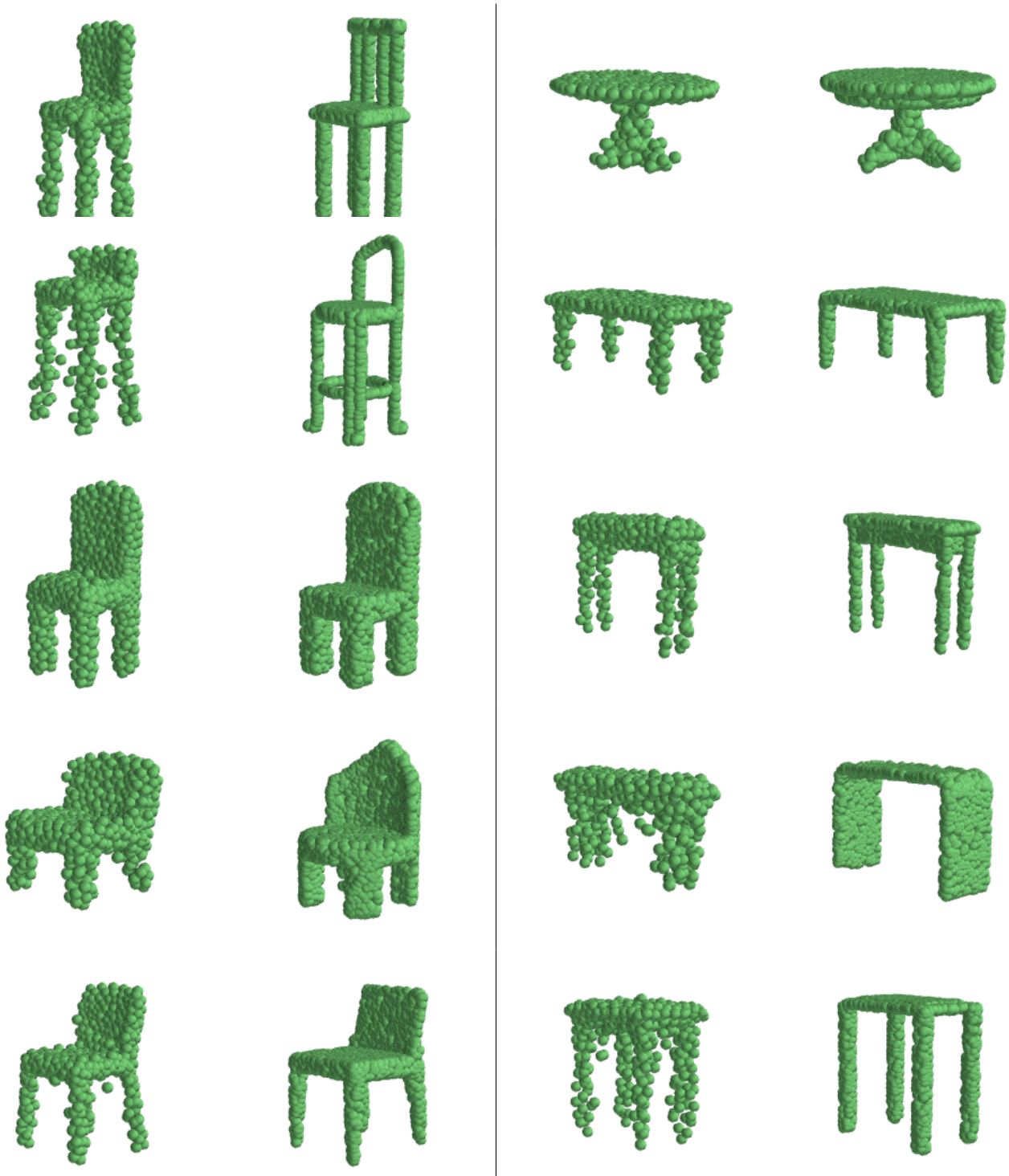

Figure 4. Additional qualitative results on shape recovery from our hidden state

References

[1] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[2] Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, and Hao Dong. Generative 3d part assembly via dynamic graph learning. *The IEEE Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11362–11369, 2020.

[4] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[5] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis and composition. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8759–8768, 2019.

[6] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3859–3868, 2019.

[7] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020.