BoxSplitGen: A Generative Model for 3D Part Bounding Boxes in Varying Granularity — Supplementary Material

Juil Koo[1*] Wei-Tung Lin[2*] Chanho Park[1] Chanhyeok Park[1] Minhyuk Sung[1]
[1]KAIST [2]NVIDIA
{63days,charlieppark,chpark1111,mhsung}@kaist.ac.kr weitungl@nvidia.com

Contents

[*]Equal contribution.

S.1. Overview

In this supplementary document, we provide additional details and results that complement the main paper. We first describe our user-interactive box and shape editing demo and show example outputs in Section S.2. We then report additional results on box-splitting over PartNet in Section S.3, followed by detailed descriptions of the data, evaluation metrics, and experimental setups for box-splitting generation in Section S.4 and for box-conditioned shape generation in Section S.5. Next, we discuss alternative approaches for box-splitting generation in Section S.6 and present implementation details of all components in Section S.7, along with more details on our user-interactive demo (Section S.9), a runtime analysis (Section S.8), and further quantitative results of box splitting (Section S.10). Finally, we visualize full splitting sequences in Section S.11 and provide additional qualitative results for box-splitting generation in Section S.12 and for box-conditioned shape generation in Section S.13.

S.2. User-Interactive Generation Demo and Examples

In Figure 1 of the main paper and Figure S1, we showcase the interface and exemplary outputs of our user-interactive box and shape editing framework. The system allows users to create 3D shapes using bounding boxes as conditions, enabling not only the manipulation of boxes but also their splitting and merging, thereby controlling the granularity. This controllability of granularity is a crucial difference from prior work [11, 12, 16, 17, 24], which operates at a fixed level of detail. It allows users to envision diverse shapes from coarser bounding boxes while generating specific shapes with detailed bounding boxes, as shown in Figure 1 of the main paper and at the top of Figure S1.

The system suggests a pivot box to split using our pivot classifier and generates the resulting child boxes with our Child-Boxes Diffusion model. Given the set of bounding boxes, the bounding box-to-shape diffusion model generates a variety of aligned 3D shapes. For further details, please refer to the **supplementary video**.

Additionally, the system allows users to edit local parts of a 3D shape by transforming the corresponding bounding boxes. Figure S1 bottom rows presents results achieved through box manipulation. When a selected box is modified, the decoded shape adjusts accordingly, highlighting our framework's user-friendly and intuitive interface for shape editing.

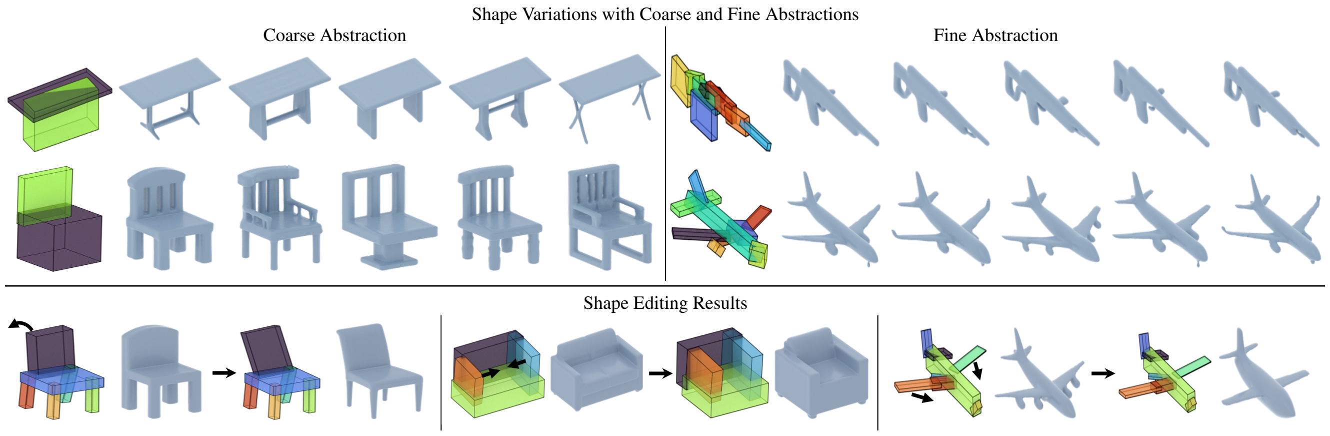

Figure S1. **Shape variations and editing guided by bounding boxes.** The top row demonstrates shape variations with different granularity of the bounding boxes, while the bottom illustrates shape editing achieved by manipulating bounding boxes.

S.3. Box-Splitting on PartNet

Table S1. Comparison of SMART [23] and PartNet [21] across different metrics for Chair and Table categories. Bold indicates the best for each column.

Dataset	Chair				Table			
	Tightness ↑	COV-EMD ↑	MMD-EMD ↓	1-NNA-EMD ↓	Tightness ↑	COV-EMD ↑	MMD-EMD ↓	1-NNA-EMD ↓
PartNet [21]	**2.25**	32.49	**16.645**	85.87	**2.20**	29.02	**14.190**	**78.21**
SMART [23]	1.61	**33.97**	16.910	**83.43**	1.78	**30.58**	15.132	79.22

Table S1 shows the Tightness of bounding boxes introduced in SMART [23] and results using same setup as Table 1, replacing SMART's unsupervised over-segments with PartNet's [21] clean annotated leaf parts. As shown below, PartNet slightly improves MMD with cleaner decompositions but slightly worsens other metrics due to looser bounding boxes from semantic-level parts.

These comparable results show our method works with hand-crafted data when available, while SMART, despite being unsupervised, provides structurally meaningful hierarchical data, demonstrating its scalability to large-scale datasets like Objaverse [6], where collecting part annotations is costly.

S.4. Details on the Experiment Setups of Box Splitting Generation

Data. We use 3D shapes from the ShapeNet [2] dataset for our experiments. Each class in ShapeNet contains between 300 and 2,700 shapes. The average number of bounding boxes in the SMART outputs for each class ranges from 5 to 12. See Table S2 for detailed statistics. We construct a training set and a validation set by splitting the shapes within each class at an 8:2 ratio. For the box-splitting training, we train separate models for each class, whereas our box-to-shape generation model is trained jointly for all classes.

Evaluation metrics. The evaluation metrics used in Section 5.1 of the main paper are computed using two shape distance metrics: Chamfer Distance [1] (CD) and Earth Mover's Distance [27] (EMD). To measure CD and EMD between bounding boxes, we first convert a set of bounding boxes for an object into a watertight mesh [8]. The conversion process begins by representing the bounding boxes as a union of implicit surfaces, which is then converted into a surface mesh, effectively removing intersections across the bounding boxes. Next, we sample 2,048 points on the watertight mesh surfaces using Poisson disk sampling [26].

S.5. Details on the Experiment Setups of Box-Conditioned Shape Generation

To evaluate the diversity and fidelity of the 3D shapes generated from the input bounding boxes, we randomly sample 1,000 bounding boxes from the validation set for each class to construct the evaluation set for shape generation. For the reference set, we use the shape set from IM-Net [3]. As in the evaluation of the box splitting stage, we sample 2,048 points over the surface of the shapes and measure geometric distances using Chamfer Distance (CD) and Earth Mover's Distance (EMD) for the COV, MMD, and 1-NNA metrics.

For assessing box alignment, we use a different evaluation bounding box set to ensure that the bounding boxes are sufficiently fine, such that the decoded shape tightly fits within the input boxes. This prevents input bounding boxes from being overly loose and simply enclosing the shapes. Specifically, we use the bounding boxes at the finest granularity level for each shape in the validation set to ensure a rigorous alignment evaluation between input bounding boxes and their decoded shapes.

Given a set of bounding boxes $\{b_i\}$ and a 3D shape S, TOV and VIoU are calculated as follows:

$$\text{TOV} = \frac{\text{vol}(\bigcup_i b_i \setminus S)}{\text{vol}(S)}, \quad \text{VIoU} = \frac{\text{vol}(S \cap \bigcup_i b_i)}{\text{vol}(S \cup \bigcup_i b_i)}. \tag{1}$$

In addition to the volume-based alignment metrics, we measure geometric alignment (Box-CD and Box-EMD) by computing CD and EMD between two point clouds: one sampled from the surface of the bounding boxes and the other from the decoded 3D shape.

S.6. Details on Alternative Approaches for Box Splitting Generation

We explore two alternative approaches for learning the conditional distribution of child boxes given a pivot box and its context. First, we investigate the use of unconditional diffusion models combined with inpainting techniques to generate child boxes while preserving the existing remaining boxes. Second, we explore a sequence generation approach inspired by their recent advances in 3D shape generation [20, 30, 34]. Below, we present more details on each alternative approach.

Unconditional Diffusion with Inpainting. Diffusion models have demonstrated great success in solving inverse problems [4, 5, 10, 15, 18, 32, 33, 38], guiding the denoising process with some conditions in various applications (*e.g.* background regions for image inpainting, blurred images for image deblurring, and low-resolution images for image super-resolution). The guided denoising process enables generating data that satisfies the given conditions with a *unconditional* diffusion model. Inspired by this, we experimented with sampling the two child boxes using a unconditional diffusion model and diffusion-based image inpainting technique.

Table S2. **Dataset statistics.** We report the number of shapes and the average number of bounding boxes for each class.

Class	Table	Chair	Couch	Airplane	Bench	Display	Rifle	Lamp
# of shapes	2,725	1,976	1,108	420	409	407	398	322
Avg. # of Boxes	6.96	9.54	5.37	11.46	8.92	3.98	8.05	7.46

Specifically, we train the noise prediction network $\boldsymbol{\epsilon}_\theta$ in Equation 3 of the main paper without the condition encoder $\mathcal{E}_\theta$ and cross-attention layers in the decoder $\mathcal{D}_\theta$. Given the unconditional diffusion model, the set of input bounding boxes $\mathcal{B}_s$, and the sampled pivot box $b_v \in \mathcal{B}_s$, we first duplicate b_v, increasing the total number of boxes to $|\mathcal{B}_s| + 1$, and then perform the DDIM inversion [7], obtaining the standard normal sample $\mathbf{x}_T$ from the input boxes. Next, we reset the portion of $\mathbf{x}_T$ corresponding to the duplicates of b_v to random standard normal samples. Then, we perform inpainting while treating $\mathcal{B}_s \setminus \{b_v\}$ as the background.

Conditional Token Prediction Model. While sequence generation models have shown remarkable capabilities in 3D shape generation [9, 22, 30, 34], their application to hierarchical box-splitting poses unique challenges. Unlike traditional sequence generation tasks that generate 3D shapes, our goal is to learn the conditional probability of splitting one pivot node into two child nodes while removing the selected pivot.

Since typical GPT-like models require a discretized representation to model categorical distributions, we quantize our continuous 15-dimensional box vector representation using VQ-VAE [36]. The VQ-VAE [36] is trained to encode input box vectors into a token space, with the encoded tokens being decoded back into the original vector. The VQ-VAE consists of simple MLPs. Following MeshGPT [30], we also incorporate a residual quantization technique [14, 19], where an input latent is discretized using a stack of D ordered codes. We set $D = 2$, and the number of codes, $|V|$, is set to 16,384.

For sequence modeling, we adapt the network of the conditional diffusion model to output the logits of two elements $[l_1, l_2] \in \mathbb{R}^{2\times|V|}$ in the token space instead of predicting their noise. Unlike traditional sequence generation tasks, our splitting process has unique characteristics: (1) it does not impose any order on the boxes—for instance, the selected pivot box can be an intermediate token in the input sequence, and (2) the splitting process generates two tokens simultaneously. To address these requirements, we introduce essential modifications to the standard sequence generation model training and inference approaches. For non-sequential autoregressive modeling, we forgo positional encoding techniques and use the Transformer [37] architecture to ensure order invariance. To predict two logits without considering their order, the training objective computes the cross-entropy loss for both possible orderings of the ground truth tokens and selects the minimum loss. The loss function is defined as: $\mathcal{L} = \min(\text{CE}([l_1, l_2], [v_1, v_2]), \text{CE}([l_1, l_2], [v_2, v_1]))$, where CE represents the cross-entropy loss, and v_1 and v_2 are the ground truth token indices corresponding to l_1 and l_2, respectively. At inference time, we first sample the first token. When sampling the second token, we mask out the index of the first predicted token to ensure that the same box is not sampled again.

S.7. Implementation Details

Child-Boxes Diffusion. As discussed in Section 3.4 of the main paper, the noise prediction network of Child-Boxes Diffusion $\boldsymbol{\epsilon}_\theta$ consists of a Transformer encoder $\mathcal{E}_\theta$ and a decoder $\mathcal{D}_\theta$. The encoder consists of 6 self-attention layers with a hidden dimension of 512. To indicate the pivot box $b_v \in \mathcal{B}_s$, we use a class embedding $\mathbf{e}_c \in \mathbb{R}^{|\mathcal{B}_s|\times 512}$ encoded from an indicator highlighting the pivot box's index. Additionally, we also encode the number of input boxes $|\mathcal{B}_s|$ into a cardinality embedding $\mathbf{e}_d \in \mathbb{R}^{512}$. These two embeddings are added to the output of each self-attention layer, yielding the final encoder output: $\mathbf{h} = \mathcal{E}_\theta(\mathcal{B}_s, b_v, |\mathcal{B}_s|) \in \mathbb{R}^{|\mathcal{B}_s|\times 512}$. The decoder $\mathcal{D}_\theta$ has a similar architecture to the encoder, with each self-attention layer followed by a cross-attention layer. The condition latent $\mathbf{h}$ is fed as the key and value in every cross-attention layer, while the noisy two child boxes $\mathbf{x}_t \in \mathbb{R}^{2\times 15}$ are fed as query. We set a learning rate and batch size to $8e^{-4}$ and 2048, respectively. For sampling, we use the DDIM [31] deterministic sampling process with 50 steps.

Inpainting with Unconditional Diffusion Models. The unconditional diffusion model for the inpainting technique adopts a similar architecture to that of the conditional diffusion model but without the decoder part. The learning rate, batch size and number of training epochs are the same as those used in BoxSplitGen. We use 50 steps for both DDIM inversion [7] and denoising process with masking.

Table S3. **Quantitative comparison of shape abstraction generation with different pivot selection strategies.** MMD-CD scores and MMD-EMD scores are scaled by 10^3 and 10^2, respectively. The best results are highlighted in **bold**. The numbers are the averages across split levels $s = 5$ and $s = 8$.

Pivot Selection	Models	COV ↑		MMD ↓		1-NNA ↓		COV ↑		MMD ↓		1-NNA ↓	
		CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
		Chair						**Airplane**					
Random	Token Prediction Model	22.05	25.76	27.960	21.335	94.13	93.02	55.25	58.80	10.645	15.680	90.27	90.08
	Uncond. Diffusion	29.70	30.97	18.784	18.128	88.68	87.93	63.82	68.21	7.207	12.953	88.50	87.90
	Cond. Diffusion (Ours)	**32.84**	**33.97**	**16.896**	**16.910**	**85.28**	**83.43**	**77.75**	**77.38**	**6.842**	**12.599**	**86.35**	**85.64**
Classifier	Token Prediction Model	27.41	30.87	24.615	20.235	91.39	90.19	66.87	70.41	8.185	13.870	87.16	86.49
	Uncond. Diffusion	33.03	35.68	18.174	17.598	88.53	85.77	71.76	77.75	6.811	12.505	86.15	85.55
	Cond. Diffusion (Ours)	**46.08**	**47.08**	**14.166**	**15.580**	**75.87**	**73.33**	**82.89**	**80.56**	**6.478**	**12.188**	**85.62**	**84.81**
		Table						**Rifle**					
Random	Token Prediction Model	18.77	17.24	32.175	22.580	90.77	92.59	57.42	58.67	3.400	9.475	87.18	86.72
	Uncond. Diffusion	25.05	25.89	21.840	18.784	83.79	84.94	**66.46**	**70.48**	3.015	8.813	**84.88**	**84.86**
	Cond. Diffusion (Ours)	**30.17**	**30.58**	**14.229**	**15.132**	**78.22**	**79.22**	64.95	68.97	**2.716**	**8.285**	87.28	85.65
Classifier	Token Prediction Model	25.45	24.47	25.895	19.680	87.47	88.72	68.72	71.86	3.495	9.230	84.28	82.67
	Uncond. Diffusion	30.36	31.68	17.726	17.104	81.96	82.26	**75.38**	**79.02**	3.110	8.717	**79.17**	**79.96**
	Cond. Diffusion (Ours)	**36.79**	**37.96**	**12.314**	**14.218**	**71.38**	**72.82**	74.75	75.25	**2.607**	**8.076**	82.86	81.96
		Couch						**Bench**					
Random	Token Prediction Model	44.80	40.73	12.810	13.750	87.41	89.30	**67.45**	60.11	11.155	14.520	**87.15**	92.40
	Uncond. Diffusion	56.56	55.42	9.127	11.296	**74.66**	**77.95**	64.69	62.33	**9.318**	**12.660**	88.45	**89.73**
	Cond. Diffusion (Ours)	**58.18**	**56.28**	**8.677**	**11.220**	77.68	79.70	64.43	**69.68**	10.630	13.167	87.23	90.43
Classifier	Token Prediction Model	48.91	42.00	12.600	13.545	85.19	88.12	**78.22**	70.08	10.535	13.450	85.53	89.84
	Uncond. Diffusion	56.51	54.71	**9.527**	**11.497**	**75.19**	**77.77**	67.85	69.82	**9.195**	**12.266**	86.12	87.29
	Cond. Diffusion (Ours)	**59.40**	**57.91**	9.724	12.743	84.40	85.00	76.12	**78.35**	9.764	13.222	**85.47**	**86.54**
		Lamp						**Display**					
Random	Token Prediction Model	65.09	64.75	25.340	22.940	87.78	87.08	50.00	58.84	14.010	15.605	92.52	92.64
	Uncond. Diffusion	71.36	71.69	**18.849**	**19.921**	**84.10**	84.93	73.46	75.43	**10.042**	**12.287**	**77.88**	**78.81**
	Cond. Diffusion (Ours)	**80.00**	**79.83**	25.810	21.988	87.01	**83.94**	**74.57**	**80.47**	11.492	13.368	84.34	84.36
Classifier	Token Prediction Model	73.73	76.95	24.640	22.500	**87.54**	**86.14**	51.23	58.60	14.860	16.175	92.06	92.09
	Uncond. Diffusion	67.79	71.36	**22.328**	**21.207**	87.80	86.60	72.48	**74.44**	**12.702**	**13.345**	**80.21**	**81.05**
	Cond. Diffusion (Ours)	**79.49**	**82.20**	27.892	23.571	89.02	86.93	**76.17**	72.97	15.325	15.688	85.71	88.41

Conditional Token Prediction Model. The conditional token prediction model adopts a similar architecture to the conditional diffusion model, with key modifications. The timestep embedding from the original network is removed. To condition the network on $\mathcal{B}_s$, its discretized representation is obtained by encoding the boxes into quantized vectors using the pre-trained VQ-VAE, which are then fed into the encoder $\mathcal{E}_\theta$. Two learnable tokens are initialized and passed through the cross-attention layers in the decoder, where these tokens attend to the quantized vectors. The final linear layer of the network is modified to output logits of dimension $|V|$. The same learning rate, batch size, and number of training epochs as those used in the conditional diffusion model are used for training.

Pivot Classifier. Similar to the architecture of the diffusion models, we utilize a Transformer encoder [37] to model the categorical distribution $p(b_v|\mathcal{B}_s)$ introduced in Section 3.1 of the main paper. The encoder contains 6 self-attention layers, each with a hidden dimension of 512 and 4 attention heads. An MLP layer follows the final self-attention layer, mapping the 512-dimensional latent of each box to a scalar logit. To process inputs with a varying number of bounding boxes, we condition the network on the current number of boxes $|\mathcal{B}_s|$ via adaLN-layers [25]. We set the learning rate and batch size to $8e^{-4}$ and 2048, respectively, and train the network for 100 epochs. To choose the pivot at the inference time, we sample the index of the pivot v from a learned categorical distribution $p(b_v|\mathcal{B}_s)$.

S.8. Runtime Analysis

Our Child-Boxes Diffusion takes 0.33 seconds per split, and BoxSplitGen (1.8s) introduces marginal increase over 3DShape2VecSet [39] (1.1s), ensuring an efficient, responsive design workflow. All measurements were taken on an RTX 3090 GPU using a 128^3 resolution for Marching Cubes.

Table S4. **Quantitative comparison of shape abstraction generation using the pivot classifier for pivot selection. Results are evaluated at split levels s=5 and s=8.** MMD-CD scores and MMD-EMD scores are scaled by 10^3 and 10^2, respectively. The best results are highlighted in **bold**.

s	Models	COV ↑		MMD ↓		1-NNA ↓		COV ↑		MMD ↓		1-NNA ↓	
		CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
		Chair						**Airplane**					
	Token Prediction Model	29.24	31.73	23.008	19.648	90.83	89.64	65.04	69.44	7.722	13.552	87.59	86.47
5	Uncond. Diff.	33.62	36.49	18.802	17.843	89.48	87.45	74.33	78.73	6.631	12.464	86.01	85.55
	Ours	**45.08**	**45.14**	**14.408**	**15.690**	**78.23**	**75.71**	**82.40**	**79.95**	**6.35**	**12.10**	**85.51**	**85.47**
	Token Prediction Model	25.57	30.00	26.224	20.817	91.95	90.75	68.70	71.39	8.647	14.186	86.72	86.51
8	Uncond. Diff.	32.43	34.86	17.547	17.353	87.58	84.10	69.19	76.77	6.990	12.546	86.30	85.55
	Ours	**47.08**	**49.03**	**13.923**	**15.471**	**73.51**	**70.96**	**83.37**	**81.17**	**6.610**	**12.277**	**85.72**	**84.14**
		Table						**Rifle**					
	Token Prediction Model	26.31	24.64	23.917	18.890	86.25	88.28	72.11	73.37	3.237	8.647	83.19	81.44
5	Uncond. Diff.	32.00	32.03	16.720	16.761	81.77	82.50	**78.64**	**82.91**	2.855	8.487	**79.15**	**79.77**
	Ours	**36.34**	**37.50**	**12.203**	**14.087**	**70.56**	**73.18**	73.62	75.13	**2.476**	**7.912**	83.90	82.49
	Token Prediction Model	24.60	24.30	27.872	20.473	88.68	89.15	65.33	70.35	3.753	9.622	85.36	83.90
8	Uncond. Diff.	28.73	31.33	18.733	17.447	82.14	82.01	72.11	75.13	3.365	8.947	**79.19**	**80.15**
	Ours	**37.24**	**38.42**	**12.426**	**14.349**	**72.20**	**72.46**	**75.88**	**75.38**	**2.738**	**8.239**	81.82	81.44
		Couch						**Bench**					
	Token Prediction Model	52.26	45.57	11.194	12.683	83.90	86.64	**78.74**	68.77	10.453	13.628	86.94	90.63
5	Uncond. Diff.	**59.95**	**58.23**	**8.979**	11.150	**73.47**	**76.05**	70.34	71.65	**8.734**	**11.957**	**84.71**	87.23
	Ours	55.15	54.25	10.344	**13.890**	93.21	92.08	78.48	**81.10**	9.296	13.276	85.89	**86.85**
	Token Prediction Model	45.57	38.43	14.007	14.405	86.48	89.60	**77.69**	71.39	10.623	13.269	**84.12**	89.04
8	Uncond. Diff.	53.07	51.18	10.075	11.845	76.92	79.49	65.35	67.98	**9.657**	**12.574**	87.53	87.36
	Ours	**63.65**	**61.57**	**9.104**	**11.597**	**75.60**	**77.91**	73.75	**75.59**	10.231	13.169	85.05	**86.22**
		Lamp						**Display**					
	Token Prediction Model	**81.02**	80.34	22.570	21.316	**85.36**	**84.71**	57.00	61.92	13.309	14.728	90.78	90.78
5	Uncond. Diff.	71.86	74.92	**21.189**	**20.627**	85.62	85.10	76.90	**77.15**	**11.585**	**12.772**	**79.56**	**78.77**
	Ours	79.32	**81.69**	26.600	23.504	90.28	88.85	**78.38**	71.74	14.300	15.668	85.71	89.78
	Token Prediction Model	66.44	73.56	26.714	23.688	89.72	87.58	45.45	55.28	16.409	17.625	93.35	93.39
8	Uncond. Diff.	63.73	67.80	**23.467**	**21.786**	89.98	88.10	68.06	71.74	**13.819**	**13.918**	**80.85**	**83.34**
	Ours	**79.66**	**82.71**	29.185	23.638	**87.76**	**85.01**	**73.96**	**74.20**	16.349	15.707	85.71	87.04

S.9. Details on User-Interactive Generation Demo

Our demonstration consists of two main components: a web-based viewer and an inference server. The viewer, implemented based on Three.js [35] and Potree [28], provides an intuitive interface to manipulate bounding boxes and generate 3D shapes directly in the web browser. The inference server processes API requests and serves results from three pre-trained models: the Pivot Classifier, Child-Boxes Diffusion, and BOX2SHAPE. Guided by the Pivot Classifier, the viewer identifies the most suitable bounding box to split, and the Child-Boxes Diffusion subsequently performs the actual splitting. Finally, the resulting bounding boxes are used to generate complete 3D shapes via BOX2SHAPE, whose predicted occupancy fields are converted into 3D meshes using Occupancy-Based Dual Contouring [13]. This integrated workflow enables an interactive and efficient approach to bounding box manipulation and high-fidelity 3D shape generation.

S.10. More Quantitative Results of Box Splitting

In this section, we present more quantitative results of box splitting generation, as discussed in Section 5 of the main paper. Table S3 shows the average results across two split levels, $s = 5$ and $s = 8$, for all eight categories. Meanwhile, Tables S4 and S5 provide the results for $s = 5$ and $s = 8$ separately, using the pivot classifier or random pivot selection, respectively.

Table S5. **Quantitative comparison of shape abstraction generation with randomly selected pivots. Results are evaluated at s=5 and s=8.** MMD-CD scores and MMD-EMD scores are scaled by 10^3 and 10^2, respectively. The best results are highlighted in **bold**.

s	**Models**	**COV ↑**		**MMD ↓**		**1-NNA ↓**		**COV ↑**		**MMD ↓**		**1-NNA ↓**	
		CD	**EMD**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**
				Chair						**Airplane**			
5	Token Prediction Model	23.84	27.89	25.074	20.215	93.74	92.26	55.50	57.21	9.049	14.620	90.62	90.20
	Uncond. Diff.	29.30	30.81	19.471	18.489	88.75	87.97	64.30	67.73	7.236	13.011	87.84	88.17
	Ours	**31.30**	**32.65**	**17.09**	**17.10**	**85.51**	**84.60**	**79.22**	**78.00**	**6.69**	**12.51**	**85.60**	**85.72**
8	Token Prediction Model	20.27	23.62	30.850	22.453	94.52	93.77	55.01	60.39	12.238	16.745	89.91	89.95
	Uncond. Diff.	30.11	31.14	18.097	17.767	88.62	87.90	63.33	68.70	7.179	12.895	89.17	87.63
	Ours	**34.38**	**35.30**	**16.70**	**16.72**	**85.04**	**82.26**	**76.28**	**76.77**	**7.00**	**12.68**	**87.09**	**85.55**
				Table						**Rifle**			
5	Token Prediction Model	20.44	18.02	29.948	21.385	90.00	91.94	61.56	61.81	3.140	9.087	86.95	86.28
	Uncond. Diff.	26.42	26.57	20.893	18.473	82.24	83.95	**68.09**	**71.61**	2.901	8.661	**84.95**	**84.82**
	Ours	**29.43**	**29.51**	**14.22**	**15.14**	**78.43**	**79.64**	65.33	68.59	**2.65**	**8.20**	87.57	86.24
8	Token Prediction Model	17.09	16.46	34.404	23.779	91.54	93.24	53.27	55.53	3.658	9.855	87.41	87.16
	Uncond. Diff.	23.67	25.20	22.786	19.095	85.35	85.93	**64.82**	**69.35**	3.129	8.966	**84.82**	**84.90**
	Ours	**30.92**	**31.66**	**14.24**	**15.12**	**78.02**	**78.81**	64.57	**69.35**	**2.79**	**8.37**	86.99	85.07
				Couch						**Bench**			
5	Token Prediction Model	46.75	43.85	11.463	12.907	85.90	87.86	**68.50**	61.94	10.223	13.839	**87.19**	92.48
	Uncond. Diff.	57.69	**57.23**	8.617	**10.975**	**74.69**	**77.66**	66.40	61.94	**8.958**	**12.478**	87.40	**89.54**
	Ours	**57.87**	56.69	**8.42**	11.06	76.79	79.65	65.88	**71.92**	9.91	12.69	87.61	90.26
8	Token Prediction Model	42.86	37.61	14.165	14.588	88.92	90.73	**66.40**	58.27	12.086	15.199	87.11	92.31
	Uncond. Diff.	55.42	53.62	9.638	11.617	**74.63**	**78.24**	62.99	62.73	**9.677**	**12.841**	89.50	**89.92**
	Ours	**58.50**	**55.88**	**8.94**	**11.38**	78.56	79.75	62.99	**67.45**	11.35	13.64	**86.85**	90.59
				Lamp						**Display**			
5	Token Prediction Model	70.51	70.51	21.714	21.384	86.19	85.66	54.79	64.62	12.061	14.487	90.36	90.57
	Uncond. Diff.	74.92	74.92	**17.812**	**19.154**	**82.75**	**83.31**	76.17	79.85	**9.535**	**11.862**	**75.99**	**77.52**
	Ours	**83.73**	**82.03**	24.88	21.33	86.45	83.79	**78.13**	**83.54**	10.90	12.88	82.51	82.55
8	Token Prediction Model	59.66	58.98	28.974	24.502	89.37	88.50	45.21	53.07	15.964	16.718	94.68	94.72
	Uncond. Diff.	67.80	68.47	**19.887**	**20.689**	**85.45**	86.54	70.76	71.01	**10.550**	**12.712**	**79.77**	**80.10**
	Ours	**76.27**	**77.63**	26.74	22.65	87.58	**84.10**	**71.01**	**77.40**	12.08	13.86	86.17	86.17

S.11. Qualitative Results of Split Sequence

Figure S2 shows the evolution of bounding box splitting by our method over multiple splits. Starting from an initial unit cube, the boxes are gradually split to capture finer details, transitioning from a coarse and simple structure to a more refined and complex 3D form. See the first and last rows of Figure S2 that gradually produce the detail of the lamp shade and the airplane tail, respectively.

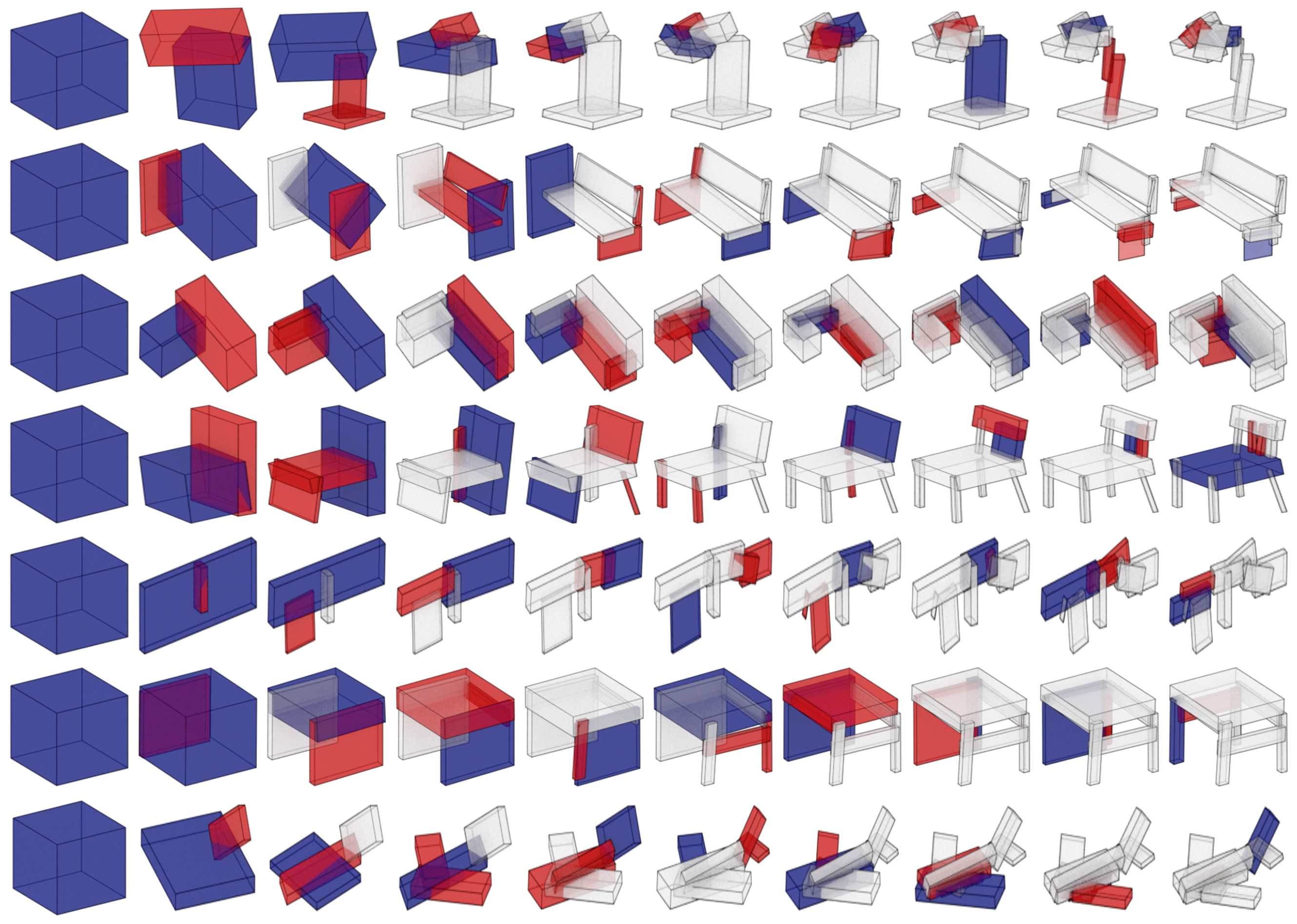

Figure S2. **The evolution of box splits by ours.** In each abstraction, new generated boxes are marked in red, and the pivot to be split in the next split is in blue.

Sections for more qualitative results are in the following pages.

S.12. More Qualitative Results of Box-Splitting Generation

In Figure S3, we present more qualitative results of box splitting generation and box-to-shape generation.

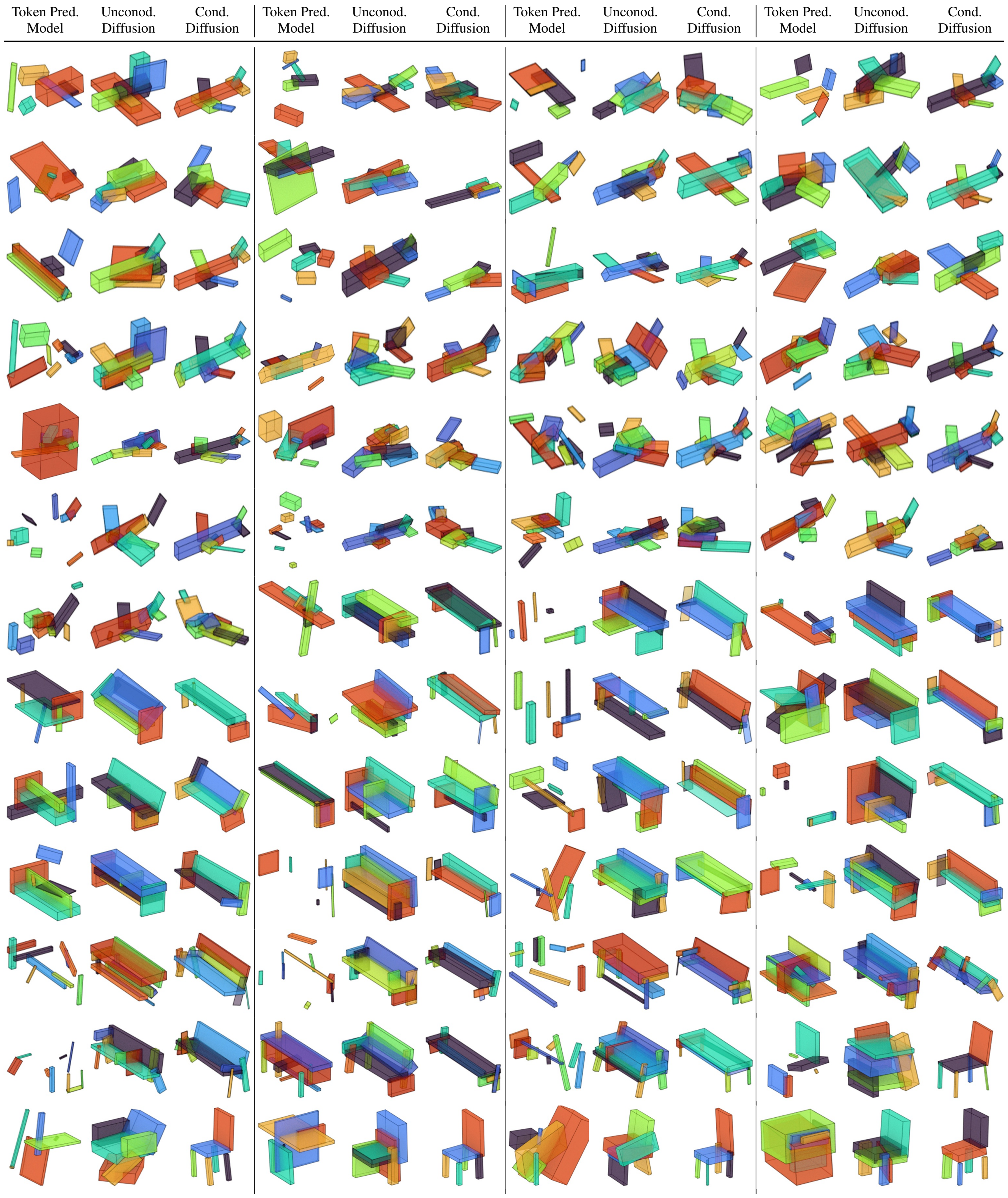

Figure S3. **Qualitative comparison of shape abstraction generation.** For each pair of columns, we query the ground truth shape and retrieve the closest generated boxes measured with chamfer distance. Our method demonstrates higher-fidelity boxes.

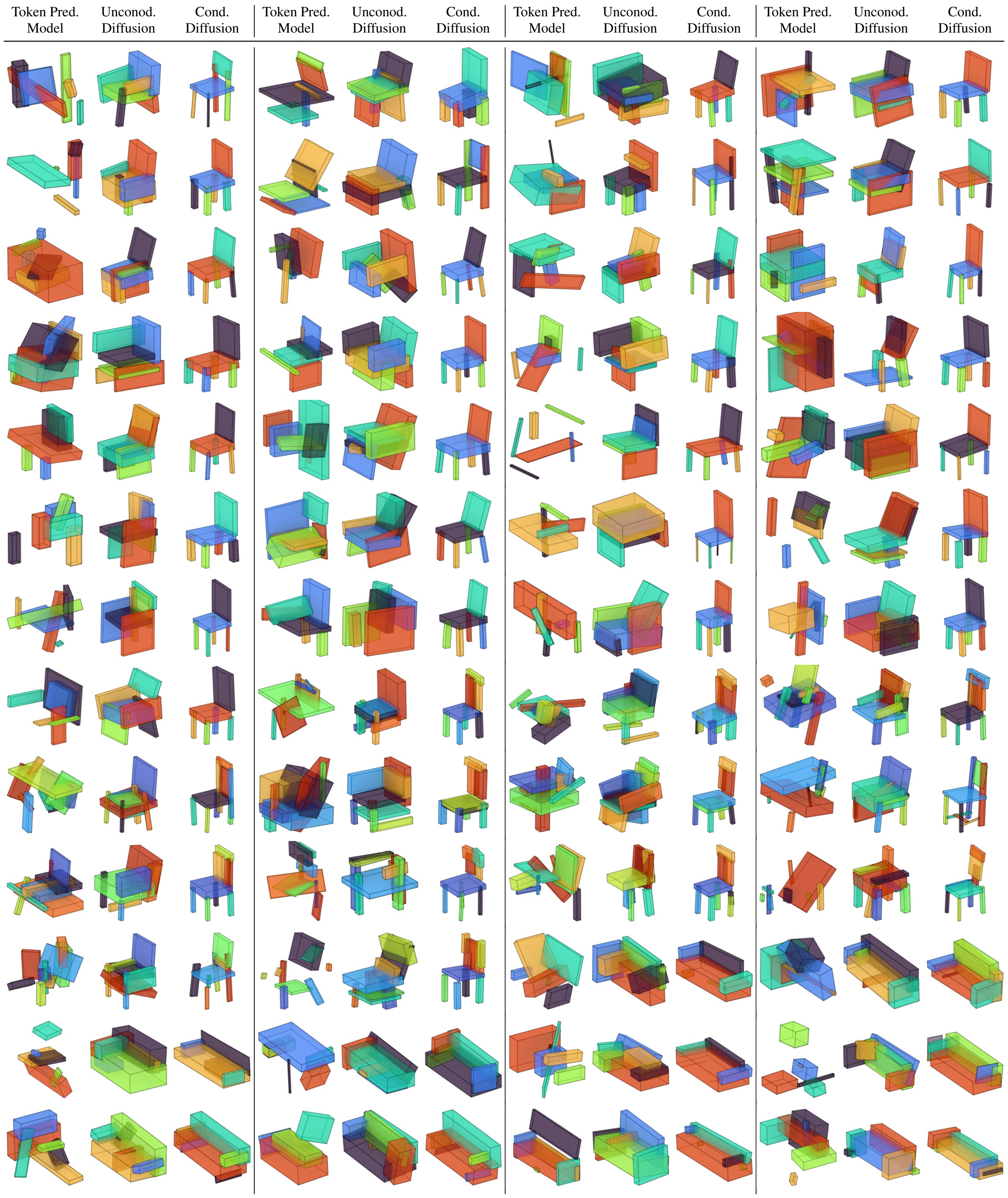

Figure S3. **Qualitative comparison of shape abstraction generation.** For each pair of columns, we query the ground truth shape and retrieve the closest generated boxes measured with chamfer distance. Our method demonstrates higher-fidelity boxes.

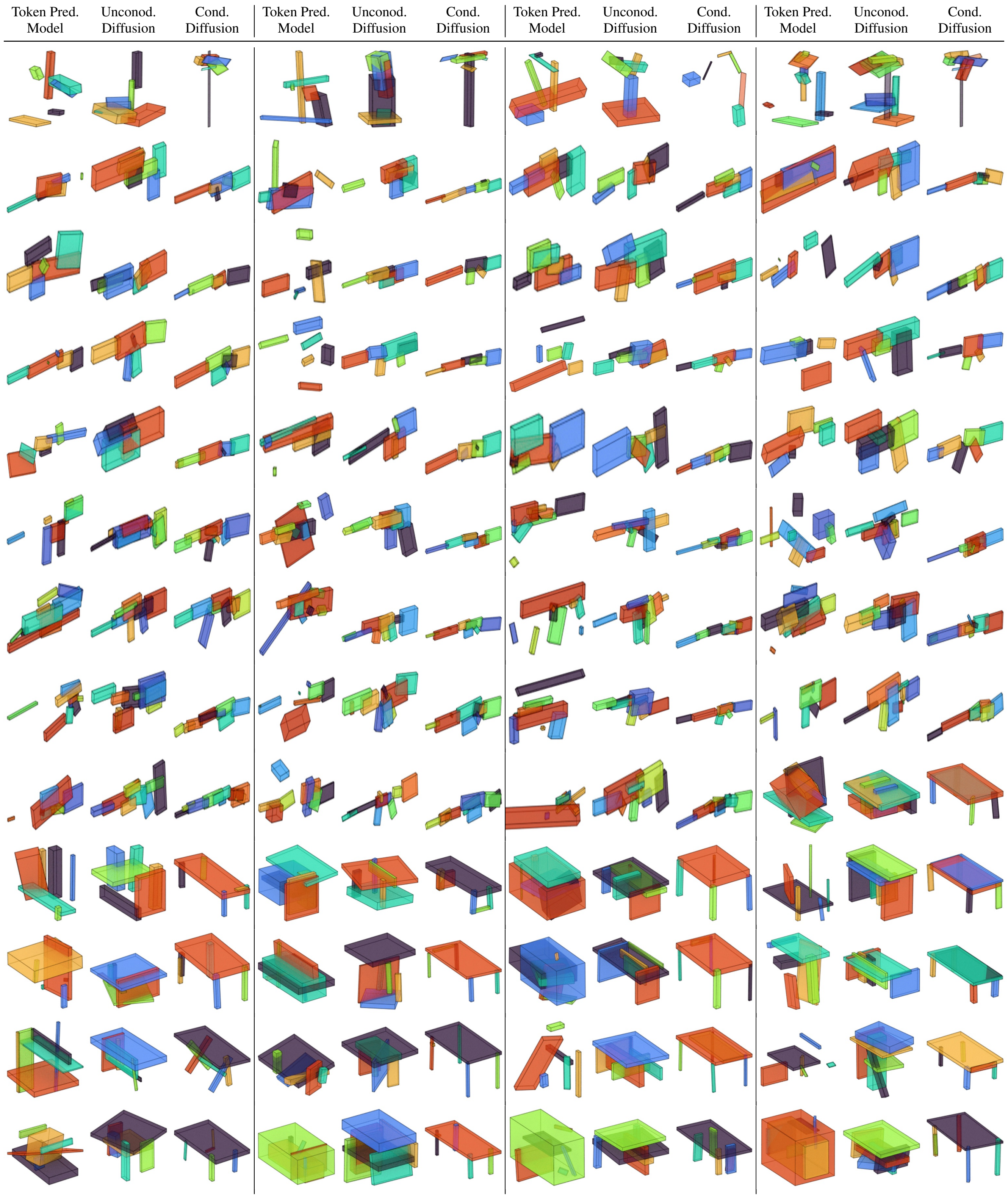

Figure S3. **Qualitative comparison of shape abstraction generation.** For each pair of columns, we query the ground truth shape and retrieve the closest generated boxes measured with chamfer distance. Our method demonstrates higher-fidelity boxes.

S.13. More Qualitative Results of Box-Conditioned Shape Generation

Figure S4 presents more qualitative results of box-conditioned shape generation.

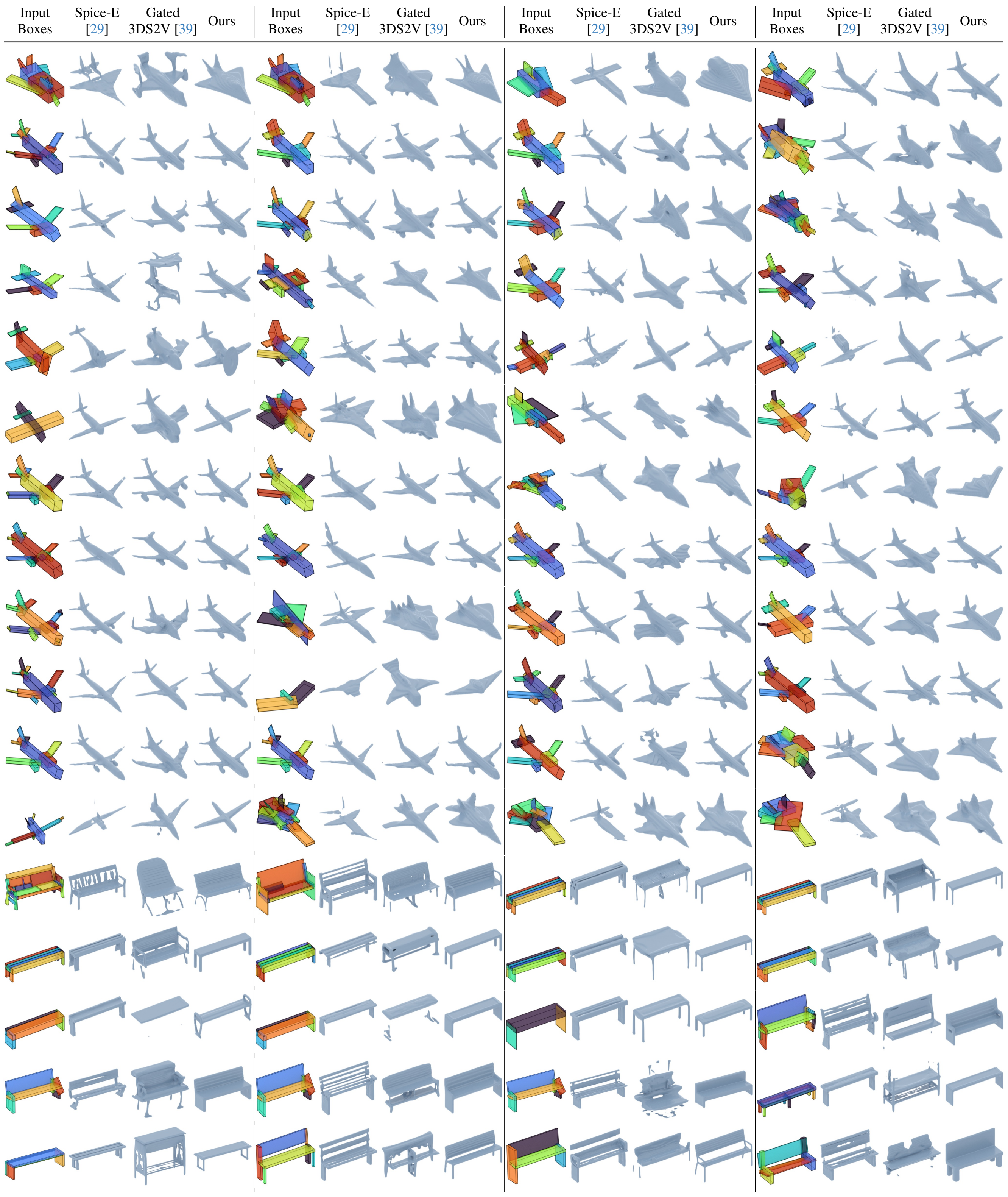

Figure S4. **Gallery of our generated bounding boxes and their final decoded 3D shapes by box-conditioned shape generation network.** Each pair of columns shows the input condition bounding box (left) and its corresponding decoded 3D shape (right).

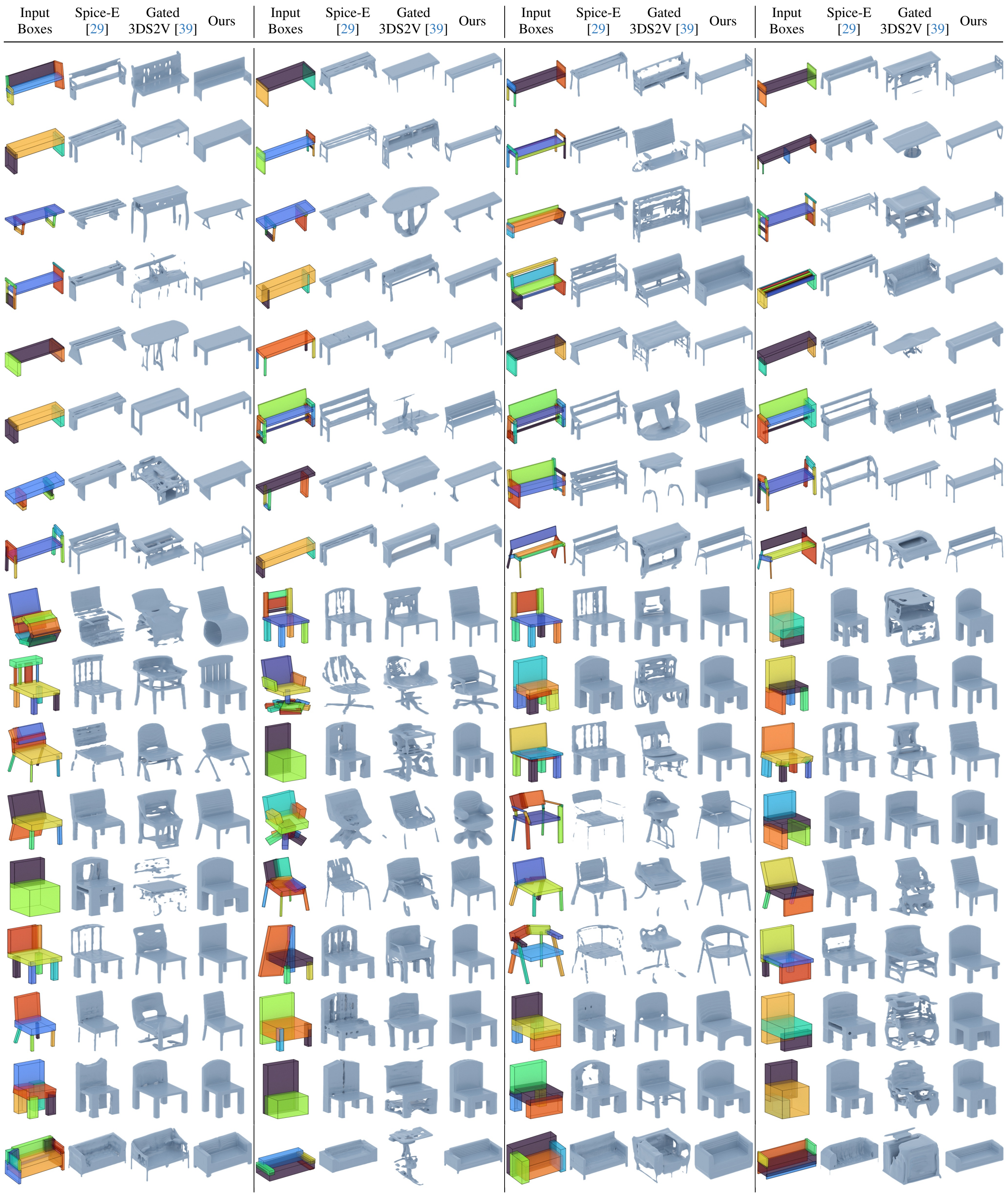

Figure S4. **Gallery of our generated bounding boxes and their final decoded 3D shapes by box-conditioned shape generation network.** Each pair of columns shows the input condition bounding box (left) and its corresponding decoded 3D shape (right).

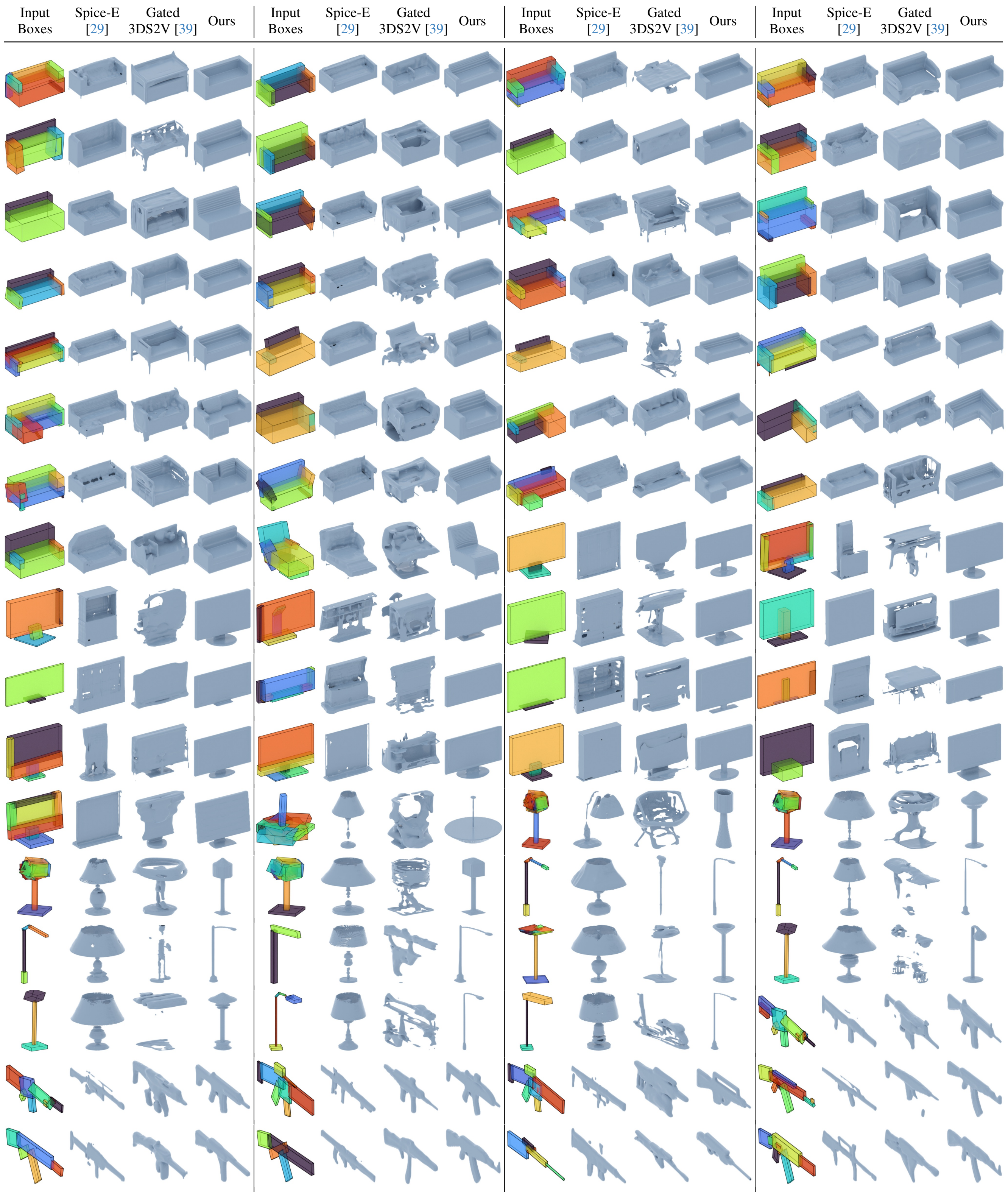

Figure S4. **Gallery of our generated bounding boxes and their final decoded 3D shapes by box-conditioned shape generation network.** Each pair of columns shows the input condition bounding box (left) and its corresponding decoded 3D shape (right).

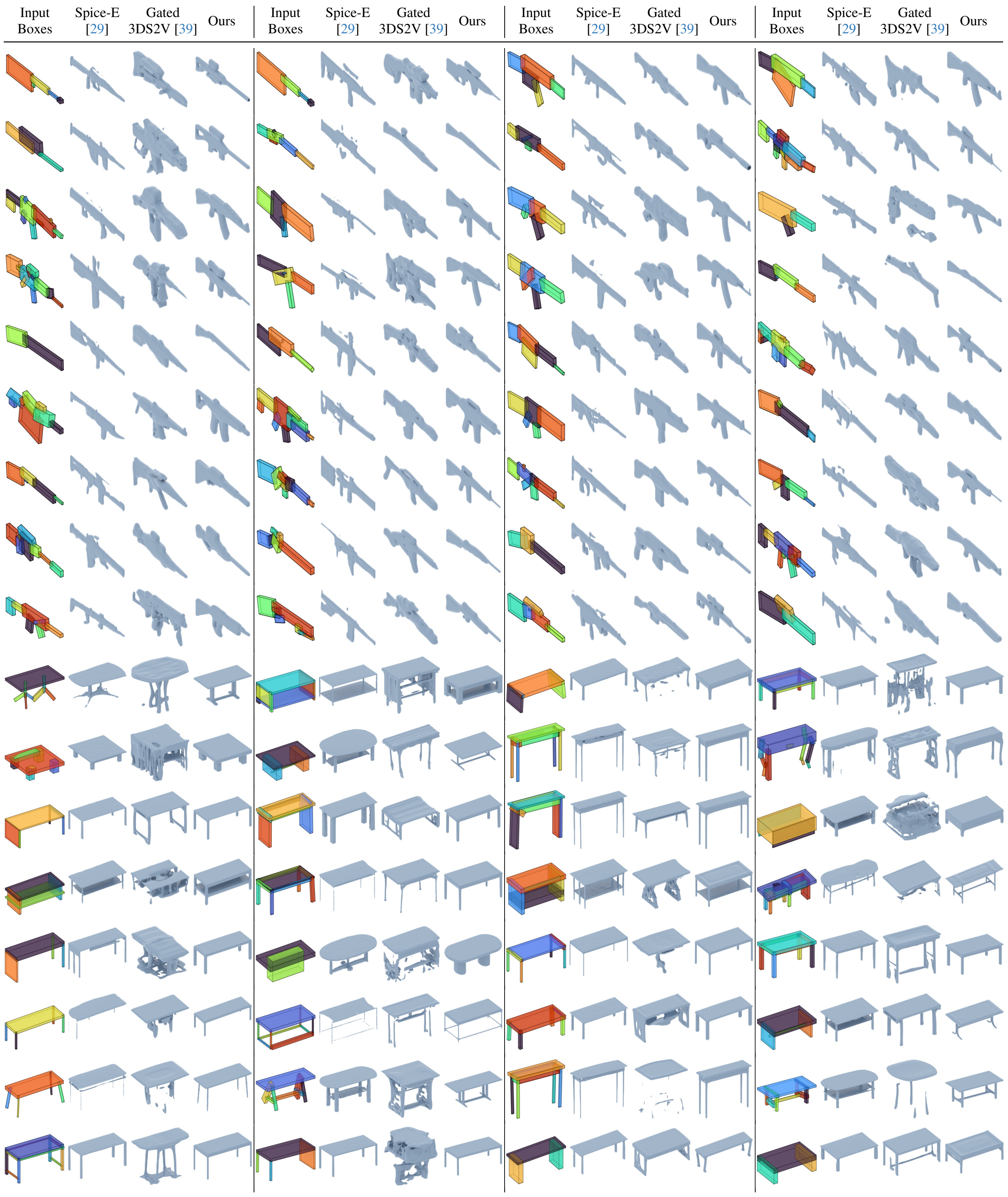

Figure S4. **Gallery of our generated bounding boxes and their final decoded 3D shapes by box-conditioned shape generation network.** Each pair of columns shows the input condition bounding box (left) and its corresponding decoded 3D shape (right).

References

[1] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen Cf Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI*, 1977. 3

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 3

[4] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022. 3

[5] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. In *NeurIPS*, 2022. 3

[6] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. 3

[7] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021. 4

[8] Emmett Lalish et al. Manifold. 3

[9] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. GET3D: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 4

[10] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023. 3

[11] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing implicit shapes through part aware generation. *ACM TOG*, 2022. 2

[12] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural template: Topology-aware reconstruction and disentangled generation of 3d meshes. In *CVPR*, 2022. 2

[13] Jisung Hwang and Minhyuk Sung. Occupancy-based dual contouring. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. 6

[14] Biing-Hwang Juang and A. Gray. Multiple stage vector quantization for speech coding. In *ICASSP*, 1982. 4

[15] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *NeurIPS*, 2022. 3

[16] Nakayama Kiyohiro, Angelina Uy Mikaela, Huang Jiahui, Hu Shi-Min, Li Ke, and Guibas Leonidas. DiffFacto: Controllable part-based 3d point cloud generation with cross diffusion. In *ICCV*, 2023. 2

[17] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. SALAD: Part-level latent diffusion for 3d shape generation and manipulation. In *ICCV*, 2023. 2

[18] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 3

[19] Julieta Martinez, Holger H Hoos, and James J Little. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*, 2014. 4

[20] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, 2022. 3

[21] 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 *CVPR*, 2019. 2, 3

[22] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020. 4

[23] Chanhyeok Park and Minhyuk Sung. Split, merge, and refine: Fitting tight bounding boxes via over-segmentation and iterative search. In *3DV*, 2024. 2, 3

[24] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *CVPR*, 2021. 2

[25] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *CVPR*, 2023. 5

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

[27] Yossi Rubner and Carlo Tomasi. *The Earth Mover's Distance*. 2001. 3

[28] Markus Schütz. *Potree: Rendering large point clouds in web browsers*. PhD thesis, Technische Universität Wien, 2015. 6

[29] Etai Sella, Gal Fiebelman, Noam Atia, and Hadar Averbuch-Elor. Spic-e: Structural priors in 3d diffusion models using cross-entity attention. 2024. 14, 15, 16, 17

[30] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *CVPR*, 2024. 3, 4

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 4

[32] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *ICLR*, 2023. 3

[33] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *ICML*, 2023. 3

[34] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024. 3, 4

[35] Threejs. Threejs. `https://threejs.org/`. 6

[36] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 4

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4, 5

[38] Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. TFG: Unified training-free guidance for diffusion models. *arXiv preprint arXiv:2409.15761*, 2024. 3

[39] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 2023. 5, 14, 15, 16, 17