

APPENDIX

A IMPLEMENTATION DETAILS

A.1 NETWORK ARCHITECTURE

We fine-tune all three-stage models starting from pre-trained holistic 3D generators. Specifically, the layout generation model is initialized from TripoSG [Li et al. \(2025\)](#), where we replace the original tokens with our proposed box tokens and inject box ID embeddings to adapt the model for layout generation. Similarly, the coarse structure generation model and the mesh refinement model are initialized using Stage 1 and Stage 2 of TRELLIS, respectively.

In all three stages, we convert half of the DiT blocks to inter-part attention blocks, while the remaining blocks retain intra-part attention. To preserve global semantic context and stabilize fine-tuning, we maintain a holistic generation branch (assigned part ID=0), which helps prevent significant deviation from the pre-trained weights. Furthermore, conditional inputs—such as image or text—are incorporated following the original pre-trained architecture via additional cross-attention blocks. Notably, in each block, all part tokens attend to all condition embeddings.

A.2 TRAINING AND INFERENCE DETAILS

Owing to GPU memory constraints, we set the maximum number of parts per object to 30 during training. For each object sample, we sort its parts by the bounding box sizes and choose the top 30 largest parts during training. Despite this limitation, our framework supports the generation of objects with more than 30 parts through a sequential sampling strategy during inference. The process is as follows: first, we sample an initial set of 30 parts alongside a global part that captures the overall shape. Then, in a subsequent sampling round, we replace the global tokens with the noisy version of the previous denoised global token at each timestep, maintaining the global structure unchanged, and sample new 30 parts. We follow TRELLIS [Xiang et al. \(2025\)](#) to render 24 conditional images for each object and randomly choose one during training. To improve the quality of conditional generation, we adopt common practices from prior work: condition tokens are randomly dropped with a probability of 0.1 during training, and classifier-free guidance with a scale of 3.5 is applied at inference. We also apply a bounding box augmentation strategy in the training of the coarse structure generation model to make the model robust to imperfect bounding box input.

B PARTVERSE-XL DATASET

To support large-scale, high-fidelity part-aware 3D generation, we present *PartVerse-XL*, a significantly expanded and refined version of the earlier *PartVerse* [Dong et al. \(2025\)](#) dataset. Figure. 8 shows some examples in our dataset. *PartVerse-XL* comprises **40K high-quality 3D objects** sourced from *Objaverse-XL* [Deitke et al. \(2023\)](#), yielding a total of **320K semantically consistent and texture-preserving parts**, each accompanied by a detailed part-level textual description. This scale and diversity—spanning over 200 object categories—substantially surpasses existing part-level benchmarks such as PartNet [Mo et al. \(2018\)](#) and enables robust training of generative models like FullPart that demand both geometric precision and semantic grounding at the part level.

The construction of *PartVerse-XL* follows a two-stage pipeline that combines automated pre-segmentation with rigorous human refinement, ensuring both scalability and annotation quality.

Automated Pre-segmentation with Semantic Priors. We begin by leveraging intrinsic modeling cues present in artist-created 3D assets, such as mesh connectivity, UV layout boundaries, and material assignments. To align these low-level cues with high-level semantics, we integrate a 3D-aware segmentation framework built upon SAM-2 [Kirillov et al. \(2023\)](#) and Samesh [Tang et al. \(2024\)](#), enhanced with geometric priors specific to procedural 3D modeling workflows. This hybrid approach produces an initial over-segmented partition of each object, deliberately erring on the side of finer granularity. Over-segmentation is preferred because it provides annotators with atomic building blocks that can be reliably merged, whereas under-segmentation often leads to irreversible loss of part boundaries.



Figure 8: Data examples from PartVerse-XL.

Human-in-the-Loop Refinement. All pre-segmented results undergo careful manual curation using a custom Blender-based annotation interface. Annotators first filter out unsuitable assets—such as those with excessive topological complexity, non-manifold geometry, or ambiguous part semantics. They then refine the segmentation by: (1) merging fragments that belong to the same functional or visual unit (e.g., the back and seat of a chair), and (2) splitting regions that conflate distinct components (e.g., separating armrests from chair legs). The refinement protocol emphasizes semantic clarity, structural symmetry, and compatibility with downstream generation tasks. Critically, textures and material properties are preserved throughout this process, ensuring that each extracted part remains visually coherent and renderable.

Part-Aware Textual Captions. To enable vision-language conditioning in part-level generation, we generate descriptive captions for every part. For each object-part pair, we render multi-view RGB images of both the full object and the isolated part. We then identify the view that maximizes the visible overlap between the part and its context within the whole object. A composite image is formed by overlaying a bounding box around the part in the full-object render. This image is fed to a state-of-the-art vision-language model (Qwen2.5-VL [Wang et al.] (2024) in our setting) to produce a natural language description that captures the part’s shape, appearance, material, and functional or spatial relationship to the parent object.

PartVerse-XL not only provides an order-of-magnitude increase in scale over prior part datasets but also establishes a new standard for semantic consistency, texture fidelity, and multimodal alignment—making it uniquely suited for training and evaluating next-generation part-aware 3D generative models.

C MORE RESULTS AND ANALYSIS

C.1 MEMORY AND EFFICIENCY ANALYSIS

We benchmark the training and inference memory consumption of our model across varying part numbers and present a comparison with prior part generators in Table. 2. All tests are conducted on the same NVIDIA A100 GPU. Common part numbers are typically below 10. PartCrafter and CoPart are incapable of handling objects with more than 16 and 8 parts, due to the fact that their training data was explicitly limited to part counts below these thresholds. The results demonstrate that the memory overhead introduced by our method is acceptable. This efficiency can be attributed to the integration of modern attention mechanisms, specifically FlashAttention, which are highly optimized for processing long sequences. We also adopt gradient checkpointing to reduce memory during training.

Table 2: Memory usage w.r.t. different part counts.

	8	16	30
PartCrafter [Lin et al.] (2025) (infer)	8.1G	10.3G	-
OmniPart [Yang et al.] (2025b) (infer)	12.7G	13.4G	14.3G
CoPart [Dong et al.] (2025) (infer)	42.2G	-	-
Ours (infer stage-I)	7.8G	10.1G	14.3G
Ours (infer stage-II)	10.4G	15.8G	17.9G
Ours (infer stage-III)	14.1G	20.3G	36.3G
Ours (train stage-I)	21.7G	24.5G	30.4G
Ours (train stage-II)	23.7G	28.0G	35.9G
Ours (train stage-III)	28.2G	36.2G	45.0G

We benchmark the inference latency against existing part generators, as shown in the Table. 3. All tests are conducted on the same NVIDIA A100 GPU. PartCrafter achieves the fastest inference speed while suffering from more artifacts due to its one-stage framework. OmniPart exhibits minimal sensitivity to part quantity due to its shared global coordinate space design, where token count does not increase significantly with the addition of parts. In contrast, our method allocates a full-resolution grid to each part, resulting in a linear growth in token count as the number of parts increases. However, this design choice represents a deliberate trade-off between generation quality and computational efficiency. The dedicated grid space for each part is fundamental to our framework’s ability to generate high-fidelity details for small and intricate components—a capability that

shared-space approaches inherently compromise. Despite the linear growth in token count, modern optimization techniques like FlashAttention and efficient memory management prevent latency from scaling proportionally; for instance, with 8 parts (a common configuration for everyday objects), FullPart maintains a practical inference time of **55 seconds**, despite processing several times more tokens than OmniPart. This latency profile remains acceptable for applications prioritizing quality over real-time interaction, such as content creation pipelines and offline asset generation. Future work will explore hierarchical token compression strategies to further improve inference efficiency while preserving our quality advantages.

Table 3: Inference latency w.r.t. different part counts.

	8	16	30
PartCrafter Lin et al. (2025)	10s	26s	-
OmniPart Yang et al. (2025b)	17s	21s	25s
CoPart Dong et al. (2025)	46s	-	-
Ours (stage-I)	10s	25s	42s
Ours (stage-II)	22s	83s	181s
Ours (stage-III)	17s	69s	131s
Ours (stage-I + II + III)	55s	187s	370s

C.2 EVALUATION ON PARTNET AND PARTNEXT

To further validate the generalizability of our method, we constructed an additional test set comprising 100 random objects from PartNet Mo et al. (2018) and 100 random objects from PartNeXt Wang et al. (2025). As shown in Table. 4, FullPart consistently outperforms previous methods.

Table 4: Quantitative comparison on PartNet/PartNeXt test set.

	F-Score $\uparrow$	CD $\downarrow$	Part-CD $\downarrow$	ULIP-Score $\uparrow$
TRELLIS Xiang et al. (2025)	0.75	0.15	-	0.24
PartCrafter Lin et al. (2025)	0.55	0.51	-	0.10
OmniPart Yang et al. (2025b)	0.76	0.13	0.46	0.20
Ours	0.77	0.12	0.42	0.25

C.3 MORE QUALITATIVE RESULTS AND COMPARISONS

We show extensive qualitative comparisons with PartCrafter Lin et al. (2025), OmniPart Yang et al. (2025b), and CoPart Dong et al. (2025) in Figure. 9. The results reveal that PartCrafter tends to generate parts that are either coalesced or disjointed, with the latter often appearing as floating artifacts during spatial decoding (e.g., the generated chair and hat have holes and floaters). This issue stems from its use of an implicit vecset representation for direct part generation, where individual part tokens are easily influenced by others, leading to increased artifacts. OmniPart, on the other hand, struggles with fine-grained part generation due to its use of a shared global coordinate space and insufficient spatial resolution. For example, OmniPart omits the surface details of small parts in the first and second results in Figure. 9.

For CoPart, we caption the input image with a VLM Wang et al. (2024) and use the predicted caption to generate parts. The results show that the quality of the textured part mesh generated by CoPart is limited, as it directly employs a holistic 3D generator for refinement without part-specific tuning. In contrast, FullPart achieves superior results through dedicated part-data fine-tuning and a carefully designed part alignment strategy.

We also provide additional in-the-wild qualitative tests in Figure. 10. The qualitative results demonstrate the strong generalization capability of FullPart.

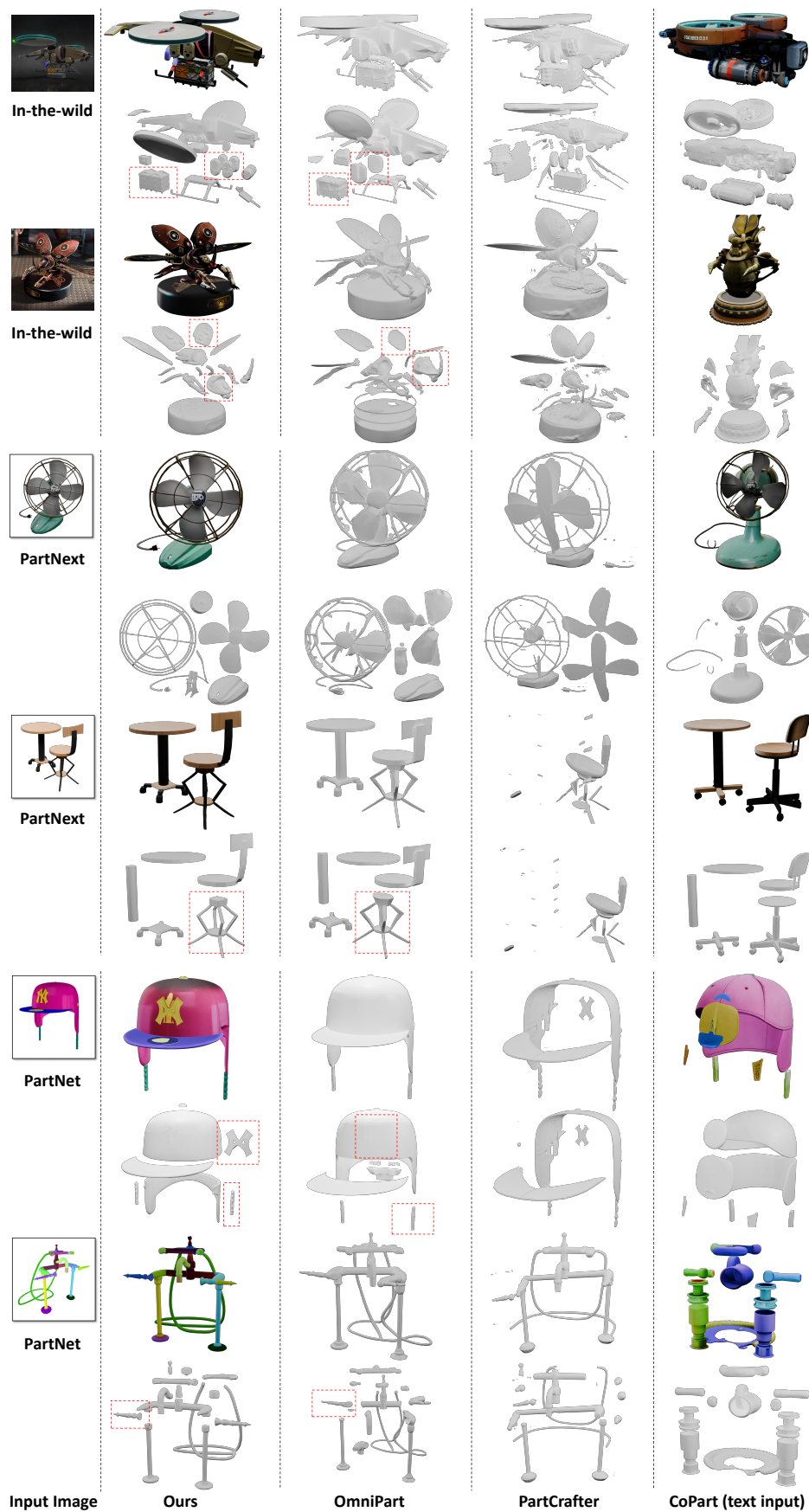


Figure 9: More comparisons with state-of-the-art 3D Part generators.

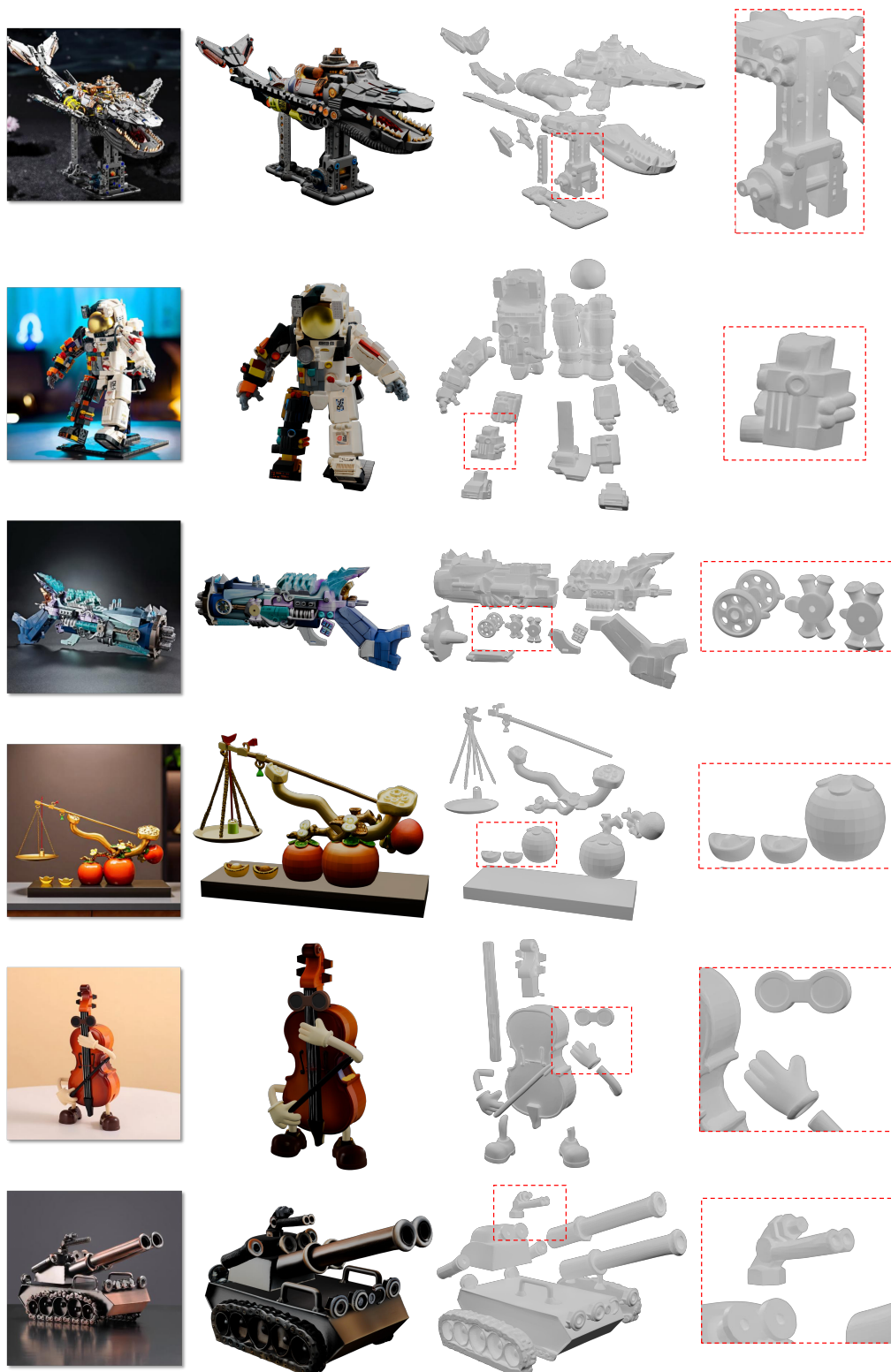


Figure 10: More in-the-wild results of FullPart.

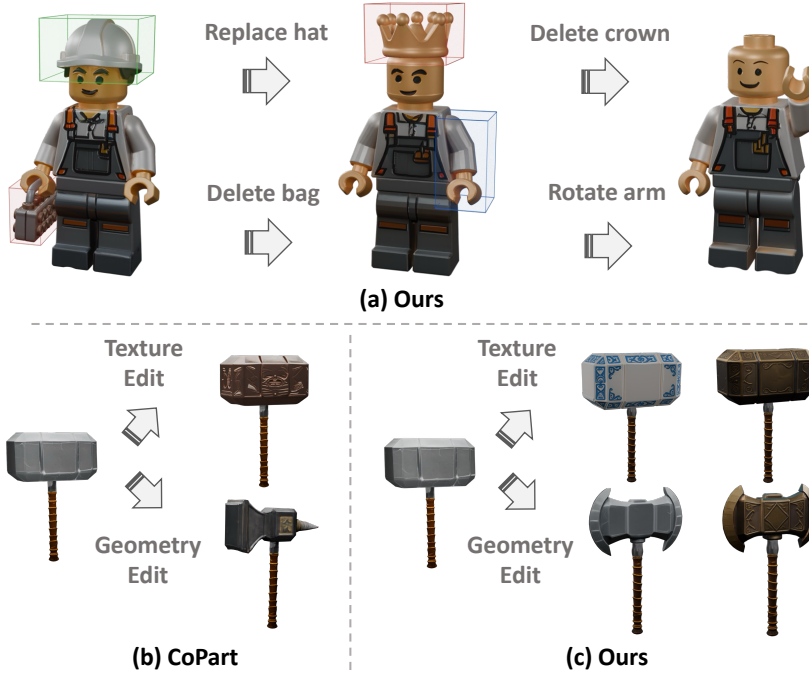


Figure 11: Editing results and comparison.

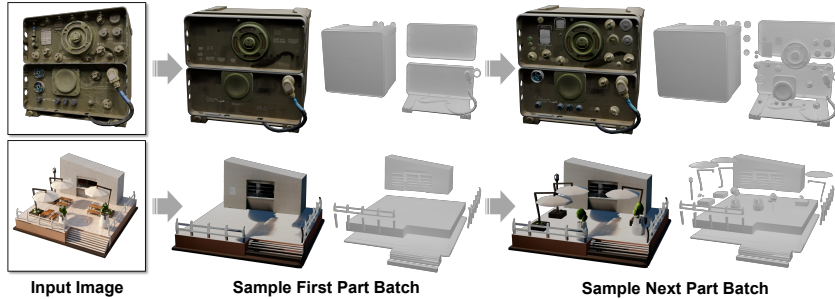


Figure 12: Sequential sampling for super long part sequence generation.

C.4 LONG SEQUENCE SAMPLING RESULTS

In Figure. 12, we show generation results for objects with over 30 parts using our sequential sampling strategy mentioned in appendix A.2. By maintaining a global branch throughout the process, our approach ensures the coherence and quality of the newly generated parts.

C.5 MORE DETAILS AND RESULTS OF THE EDITING APPLICATION.

Thanks to our hybrid pipeline, users can edit specific parts by modifying their bounding boxes and re-running the coarse generation and refinement stages. Given an assembled object, we first encode each part into its latent tokens. To regenerate a selected part while preserving others, we leverage the inherent inpainting capability of diffusion models. Specifically, we initialize the target part tokens from Gaussian noise. At each denoising step t , we replace the noisy tokens of unchanged parts by adding noise at timestep t to their clean tokens. This creates an inpainting setup where contextual information from the unchanged parts gradually and naturally guides the regeneration of the target part. The regenerated tokens for the selected parts are then inserted or replaced according to the updated box layout. This approach offers significant flexibility. If we want other unselected parts to also change coherently, we can start from a specific step in the sampling process (e.g., $t=0.5$) and stop replacing the sampled latent with the noised “clean latent”. This enables the subsequent denoising process to make subtle adjustments to the other parts. Optionally, we can opt to keep the unselected parts entirely unchanged, achieving perfect preservation of the unmodified regions.

We also demonstrate additional editing operations—including part replacement, rotation, and deletion in Figure. II (a). We provide comparisons with prior part-based generative methods Dong et al. (2025) in Figure. II (b) and (c). The results confirm that FullPart achieves competitive editing performance.

C.6 QUANTITATIVE ABLATION RESULTS

We further validate the effectiveness of our designs through quantitative ablation studies. The results in Table. 5 clearly demonstrate the critical importance of the carefully designed center-corner encoding mechanism in enabling the model to accurately perceive the actual size of each part. Moreover, we observe a significant performance drop when directly using part labels from raw meshes without human annotation. Finally, allocating a full-resolution grid to each part is shown to enhance the quality of the generated parts.

Table 5: Quantitative ablation results on PartVerse-XL test set.

	F-Score $\uparrow$	CD $\downarrow$	Part-CD $\downarrow$	ULIP-Score $\uparrow$
w/o Corner Enc	0.65	0.36	0.55	0.13
w/o Human Anno	0.72	0.20	0.48	0.17
w/o Per-part Grid	0.76	0.17	0.40	0.21
Full model	0.81	0.11	0.36	0.24

C.7 DISCUSSION OF THE LAYOUT GENERATION

Previous methods directly predict the bounding boxes (layout) parameters (x, y, z, w, h, d). For example, LayoutGPT uses LLM to predict box parameters and then retrieve objects based on boxes for 3D scene generation. OmniPart also trains an auto-regressive model to predict part box parameters. In contrast, we regard each bounding box as a box mesh, and fine-tune an implicit vecset diffusion model to generate box meshes and then compute the boxes from the extracted box meshes. In this way, FullPart can inherit the priors from the pre-trained vecset diffusion model and turn the accurate box parameter regression task into a more robust diffusion-based generation process.

C.8 DIFFERENCE WITH CoPART

The first major difference lies in the part representation. CoPart Dong et al. (2025) represents 3D parts using a combination of implicit vecset geometric latents and image latents. In contrast, FullPart represents each part with explicit voxel-based latents, which allow texture information to be directly encoded into the voxel tokens—similar to TRELLIS. Due to the constraints of its implicit representation, the part geometric tokens in CoPart share a global coordinate space, leading to insufficient resolution for fine-grained parts. In contrast, FullPart assigns each part an independent full-resolution grid, thereby enabling detailed geometric modeling.

Second, the generation stages of CoPart and FullPart differ significantly in their objectives and designs: i) In the first stage, CoPart requires users to provide part bounding boxes as input. These boxes are injected into a dual-branch DiT to generate part-centric multi-view images (via ControlNet) and coarse part shapes (via cross-attention). In contrast, FullPart eliminates the need for manual box input by automatically generating boxes through a box diffusion process. It then employs a compact single-branch DiT to generate coarse voxel-based parts. Notably, thanks to our explicit representation, each voxel latent is intrinsically defined within its corresponding box, eliminating the need for the additional architectural mechanisms used in CoPart to inject box information. ii) In the second stage, CoPart feeds the coarse shapes and multi-view images into a pre-trained holistic 3D object generator to obtain textured part meshes. Notably, CoPart treats each part as an independent whole object and directly employs this holistic generator without any fine-tuning on part-specific data. CoPart claim that the generator can produce textured part meshes when initialized with both coarse shapes and multi-view images. In contrast, FullPart’s refinement stage is fine-tuned on part data and incorporates a center-corner encoding mechanism to carefully align different parts, thereby achieving more coherent and higher-quality part generation.

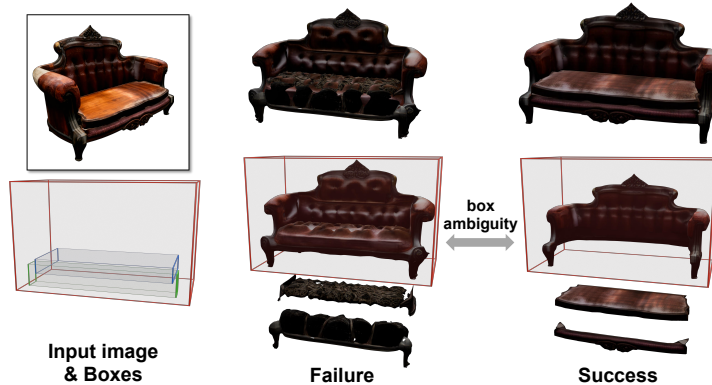


Figure 13: Typical failure case due to box ambiguity.

C.9 USER STUDY

We conducted a user study involving 73 participants from 10 professional backgrounds. Participants were asked to select the best result based on global coherence or assembly structural plausibility. The results demonstrate that FullPart outperforms all previous methods in both global quality and part-level coherence.

Table 6: User Study (% preference).

Method	Global coherence	Part structural plausibility
PartCrafter Lin et al. (2025)	7.3	2.7
OmniPart Yang et al. (2025b)	38.8	21.0
CoPart Dong et al. (2025)	5.0	16.0
Ours	48.9	60.3

D LIMITATIONS AND FAILURE CASE ANALYSIS

We observe two primary limitations. First, the model occasionally generates darker textures, which we attribute to the dark render in the data processing stage: our optimization of render settings for speed came at the cost of quality. Specifically, we lowered critical ray tracing parameters, including the number of light path bounces and the overall sampling rate. These simplifications limited how effectively light could bounce and accumulate in the scene, leading to an underexposed and less accurate image. Second, a typical failure case is shown in Figure. 13. Because the box’s structure is ambiguous, the model gets confused about the final design, which results in failures like generating an unnecessary padding layer. In addition, our computational cost increases linearly with the number of parts. We plan to explore ways to accelerate the inference in future work.

E USE OF LARGE LANGUAGE MODELS

During the writing of this paper, we employed a large language model (GPT-4) solely to improve the clarity and fluency of the original text we had drafted. The model was used as a tool for language polishing and did not contribute to the conceptual development, technical execution, experimental analysis, or scientific conclusions presented in this work