From One to More: Contextual Part Latents for 3D Generation

Supplementary Material

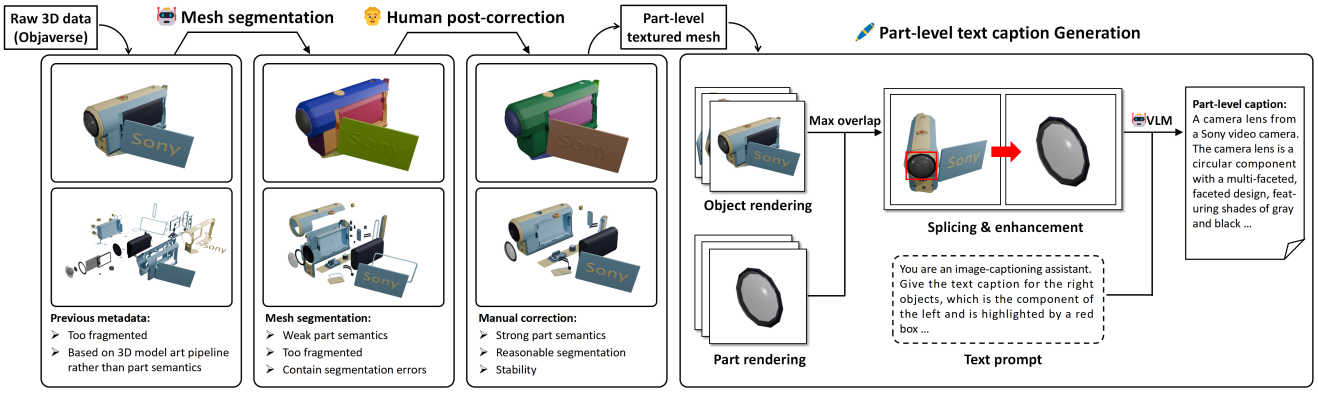

Figure 8. PartVerse dataset processing pipeline. We follow the pipeline of "raw data - mesh segment algorithm - human post correction - generate text caption" to produce part-level data.

A. 3D part VAE fine-tuning

Interestingly, we observe that the original pre-trained holistic 3D VAE from CraftMan already exhibits some capability to encode part meshes, reducing the need for extensive part data during VAE fine-tuning. To further enhance the part mesh encoding, we fine-tune the CraftMan's VAE [55] on our PartVerse data. Specifically, we sample points $P \in \mathbb{R}^{S\times 3}$ and normals $Q \in \mathbb{R}^{S\times 3}$ on the surface of the part mesh, then use a transformer encoder $\mathcal{E}_{3D}$ to extract a fixed length latent set following [55]:

$$\boldsymbol{L}_{3D} = \mathcal{E}_{3D}(P, Q) \in \mathbb{R}^{T\times D} \tag{14}$$

Furthermore, to enable differentiable rendering for the 3D latents, we modify the output layer of the VAE's decoder $\mathcal{D}_{3D}$ to predict Flexicube [41] parameters given a query grid point $p \in \mathbb{R}^3$:

$$(\alpha, \beta, \gamma, \delta) = \mathcal{D}_{3D}(L_{3D}, p), \tag{15}$$

where $\alpha \in \mathbb{R}^8_{>0}, \beta \in \mathbb{R}^{12}_{>0}, \gamma \in \mathbb{R}_{>0}, \delta \in \mathbb{R}^3$ are Flexicube parameters. Then we can extract a triangle mesh via Dual Marching Cubes from predicted parameters on each grid and render the mesh into depth and normals:

$$I_{\text{depth}}, I_{\text{normal}} = \text{Render}(\text{DualMC}(\alpha, \beta, \gamma, \delta)), \tag{16}$$

Therefore, we supervise the 3D part VAE fine-tuning with the following losses:

$$\begin{aligned}\text{Loss}_{\text{VAE}} = \lambda_1 \cdot \text{Loss}_{L1}(I_{\text{depth}}, I_{\text{gt depth}}) \\ + \lambda_2 \cdot \text{Loss}_{L1}(I_{\text{normal}}, I_{\text{gt normal}}) \\ + \lambda_3 \cdot \text{Loss}_{KL}(L_{3D}),\end{aligned} \tag{17}$$

where Loss_{KL} is a KL reguarlization.

B. Dataset

As demonstrated in Fig. 8 and main paper, we use segmentation algorithms to process the raw 3D data and then perform human-post annotation to obtain the 3D textured part mesh. In addition, we have also provided additional annotations for part-level text captions.

Part-level text caption. We also created textual descriptions for each part, covering appearance features, shape characteristics, and part-whole relationships. The process starts by rendering multi-view images of both complete objects and individual parts. We select the view showing maximum visible overlap between part and whole, then combine the object image with the highlighted part using a bounding box. This composite image is input to a vision-language model (VLM) to generate descriptions of the component and its relationship to the complete object.

C. More comparsions with part-based methods

As shown in Fig. 9, we compared our CoPart with segmentation-driven methods Part123 and PartGen. Our method has richer details and reasonable geometry. Moreover, since each part is generated as a complete 3D object in our method, there will be no loss of details for the contact surface between two parts.

D. Application Details

D.1. Mini-scene Generation

Our approach extends naturally to mini-scene generation, as a mini scene can be represented as a layout comprising sets of bounding boxes. This aligns with our bounding box-guided

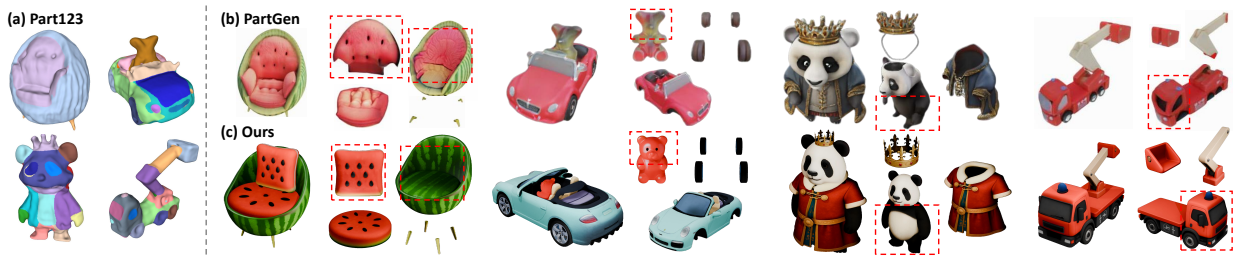

Figure 9. Comparisons with segmentation-driven methods Part123 and PartGen. PartGen results are from their paper and project website.

Figure 10. Segmentation and annotation results.

part generation task, where each part corresponds to an object. Furthermore, our training data includes mini-scenes, which enhances *CoPart*'s ability to understand and generate such scenes. Consequently, we can generate mini scenes directly by specifying bounding boxes and text prompts, as illustrated in Fig. 5 (b).

D.2. Long part sequence sampling

GPU memory constraints limit the maximum number of parts N that can be processed during training. To generate a long sequence of parts to create a complex and intricate object, we adopt a strategy similar to the one described in Section 4.1. Specifically, to generate N(> 8) parts given N condition 3D bounding boxes, we first generate 8 biggest parts according to the box sizes since these parts can represent the overall shape. Then, we regard the generation of the remaining parts as a task of part editing. To generate extra part latents, we manually select and fix M (typically 5-7) already sampled part latents while generating other (8 - M) latents from pure noises under their bounding box conditions. This allows us to generate longer sequences without exceeding memory constraints, as demonstrated at the top of Fig. 11.

E. Limitations

CoPart achieves high-quality part-based 3D generation by training on the new proposed Partverse dataset. But we also find its deficiencies.

Memory constraints. *CoPart* can not simultaneously generate a long sequence of parts due to GPU memory limitation and adopt a sub-optimal sampling strategy instead in Sec. 4.3. In further work, we can adapt the auto-regressive paradigm to generate long sequence parts without sacrificing consistency.

Text prompt accuracy. Some parts in our data are hard to annotate text prompts for VLM, introducing language hallucination in the data. To improve text label accuracy, we will explore using the video language model to understand parts better.

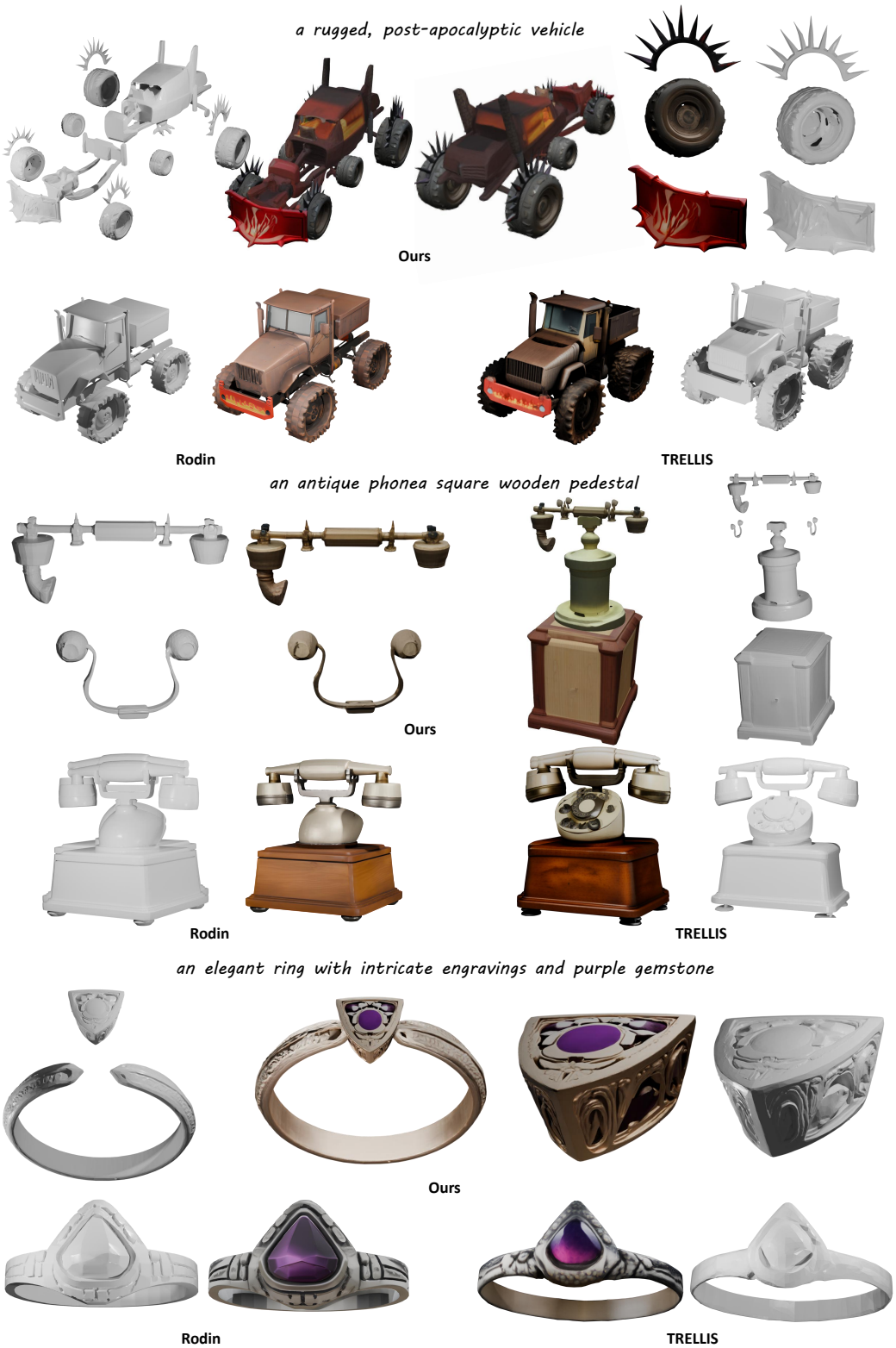

Figure 11. More results compared with state-of-the-art 3D generators.

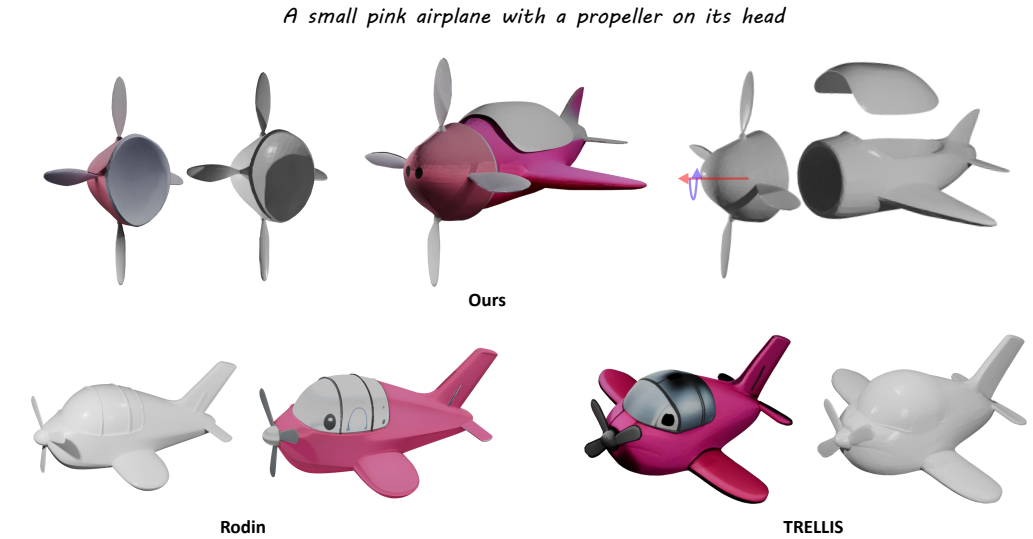

Figure 12. More results compared with state-of-the-art 3D generators.

Figure 13. Data example from Partverse.