PartGen: Part-level 3D Generation and Reconstruction with Multi-View Diffusion Models

Supplementary Material

This supplementary material contains the following parts:

- **Implementation Details.** Detailed descriptions of the network architectures, baselines for part completion, training and inference settings for all models used in PartGen are provided.
- **Additional Experiment Details.** We describe the detailed evaluation metrics employed in the experiments and provide additional experiments.
- **Additional Examples.** We include more outputs of our method, showcasing applications with part-aware text-to-3D, part-aware image-to-3D, real-world 3D decomposition, and iteratively adding parts.
- **Failure Case.** We analyse the modes of of failure of PartGen.
- **Discussion.** We discuss the compatibility of PartGen with other reconstruction models and potential alignment issues in part reassembling.
- **Ethics and Limitation.** We provide a discussion on the ethical considerations of data and usage, as well as the limitations of our method.

A. Implementation Details

We first provide network architectures and then we provide details of the training pipeline used in PartGen (Appendices A.3 to A.6). In addition, we provide the implementation details for the applications: for part composition (Appendix A.7) and for part editing (Appendix A.8).

A.1. Network architectures

Diffusion Model. The architecture of the diffusion model used in the paper is similar to [3], which contains an autoencoder to encode images and a U-Net for the denoising process. Specifically, the autoencoder compresses the images into 8 channels with a compression ratio of 8. For more details, please refer to [3].

Reconstruction Model. We use lightplaneLRM [1] as the reconstruction model with $128 \times 128 \times 128$ voxel grids. The Splatter and Renderer components both use 3-layer MLPs with a width of 32. For detailed implementation, please refer to [1].

A.2. Baseline for part completion

We compare our PartGen ($\hat{J} = \mathcal{B}(I \odot M, I)$) to four baselines: (1) directly use the reconstruction model Ψ to reconstruct the 3D part without completing the multi-view part image ($\hat{J} = I \odot M$); (2) fine-tune Φ to complete the multi-view part image without given the multi-view image of the whole object ($\hat{J} = \mathcal{B}(I \odot M)$); (3) fine-tune Φ to only complete 4 views one by one ($\hat{J}_v = \mathcal{B}(I_v \odot M_v, I_v)$); (4) use the groundtruth completed multi-view part image for 3D reconstruction ($\hat{J} = J$).

A.3. Text-to-multi-view generator

We fine-tune the text-to-multi-view generator starting with a pre-trained text-to-image diffusion model trained on billions of image-text pairs that uses an architecture and data similar to Emu [3]. We change the target image to a grid of 2×2 views as described in Section 3.5 following Instant 3D [7] via v-prediction [10] loss. The resolution of each view is 512×512, resulting in the total size of 1024×1024. To avoid the problem of the cluttered background mentioned in [7], we rescale the noise scheduler to force a zero terminal signal-to-noise ratio (SNR) following [8]. We use the DDPM scheduler with 1000 steps [4] for training. During the inference, we use DDIM [12] scheduler with 250 steps. The model is trained with 64 H100 GPUs with a total batch size of 512 and a learning rate 10^{-5} for 10k steps.

A.4. Image-to-multi-view generator

Building on the text-to-multi-view generator, we further fine-tune the model to accept images as input conditioning instead of text. The text condition is removed by setting it to a default null condition (an empty string). We concatenate the conditional image to the noised image along the spatial dimension, following [2]. Additionally, inspired by IP-adapter [16], we introduce another cross-attention layer into the diffusion model. The input image is first converted into tokens using CLIP [9], then reprojected into 157 tokens of dimension 1024 using a Perceiver-like architecture [5]. To train the model, we utilize all 140k 3D models of our data collection, selecting conditional images with random elevation and azimuth but fixed camera distance and field of view. We use the DDPM scheduler with 1000 steps [4], rescaled SNR, and v-prediction for training. Training is conducted with 64 H100 GPUs, a batch size of 512, and a learning rate of 10^{-5} over 15k steps.

A.5. Multi-view segmentation network

To obtain the multi-view segmentation network, we also fine-tune the pre-trained text-to-multi-view model. The input channels are expanded from 8 to 16 to accommodate the additional image input, where 8 corresponds to the la-

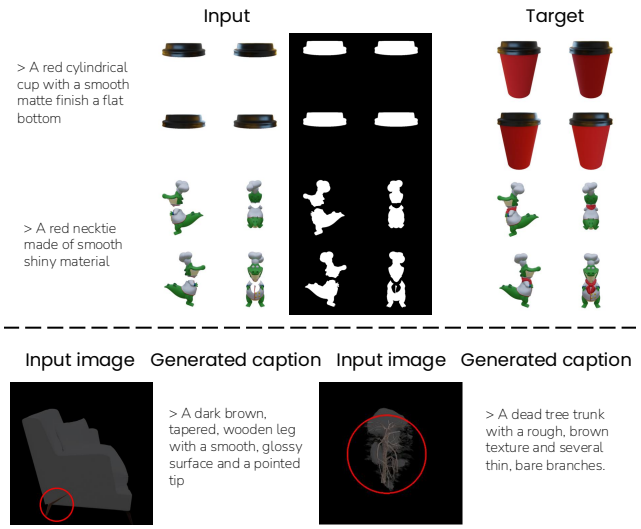

Figure A1. **3D part editing and captioning examples.** The top section illustrates training examples for the editing network, where a mask, a masked image, and text instructions are provided as conditioning to the diffusion network, which fills in the part based on the given textual input. The bottom section demonstrates the input for the part captioning pipeline. Here, a red circle and highlights are used to help the large vision-language model (LVLM) identify and annotate the specific part.

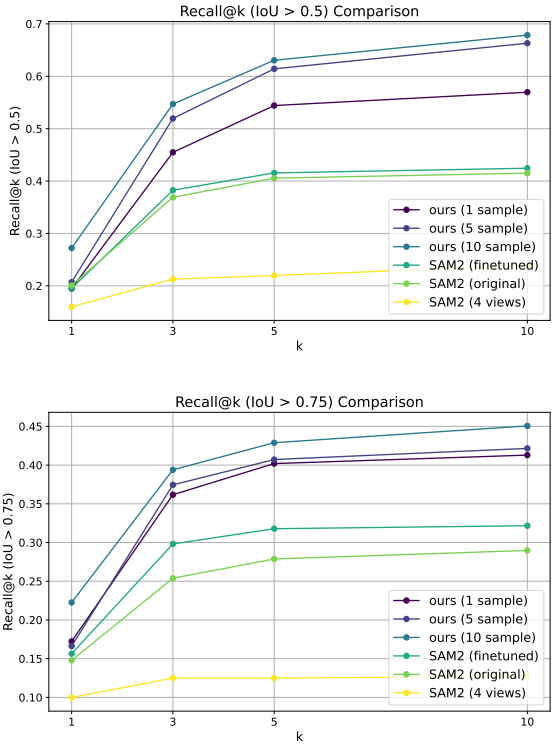

Figure A2. **Recall curve of different methods.** Our method achieve better performance comparing with SAM2 and its variants.

tent dimension of the VAE used in our network. We create segmentation-image pairs as inputs. The training setup follows a similar recipe to that of the image-to-multi-view generator, employing a DDPM scheduler, v-prediction, and rescaled SNR. The network is trained with 64 H100 GPUs, a batch size of 512, a learning rate of 10^{-5}, for 10k steps.

A.6. Multi-view completion network

The training strategy for the multi-view completion network mirrors that of the multi-view segmentation network, with the key difference in the input configuration. The number of input channels (in latent space) is increased to 25 by including the context image, masked image, and binary mask, where the mask remains a single unencoded channel. Example inputs are illustrated in Figure 5 of the main text. The network is trained with 64 H100 GPUs, a batch size of 512, a learning rate of 10^{-5}, and for approximately 10k steps.

A.7. Parts assembly

When compositing an object from its parts, we observed that simply combining the implicit neural fields of parts reconstructed by the Reconstruction Model (RM) in the rendering process with their respective spatial locations achieves satisfactory results.

To describe this formally, we first review the rendering function of LightplaneLRM [1] that we use as our reconstruction model. LightplaneLRM employs a generalized Emission-Absorption (EA) model for rendering, which calculates transmittance T_{ij}, representing the probability of a photon emitted at position x_{ij} (the j_{th} sampling point in the i_{th} ray) reaching the sensor. Then the rendered feature (*e.g.* color) v_i of ray r_i is computed as:

$$v_i = \sum_{j=1}^{R-1} (T_{i,j-1} - T_{i,j}) f_v(x_{ij})$$

where $f_v(x_{ij})$ denotes the feature of the 3D point x_{ij}; $T_{i,j} = \exp(-\sum_{k=0}^{j} \Delta \cdot \sigma(x_{ik}))$, where Δ is the distance between two sampled points and $\sigma(x_{ik})$ is the opacity at position x_{ik}, $T_{i,j-1} - T_{i,j}$ captures the visibility of the point.

Now we show how we generalise it to rendering N parts. Given feature functions $f_v^1, \ldots, f_v^N$ and their opacity functions $\sigma^1, \cdots, \sigma^N$, the rendered feature of a specific ray r_i becomes:

$$v_i = \sum_{j=1}^{R-1} \sum_{h=1}^{N} (\hat{T}_{i,j-1} - \hat{T}_{i,j}) w_{ij}^h \cdot f_v^h(x_{ij}).$$

where $w_{ij}^h = \sigma^h(x_{ij}) / \sum_{l=1}^{N} \sigma^l(x_{ij})$ is the weight of the feature $f_v^h(x_{ij})$ at x_{ij} for part h; $\hat{T}_{i,j} =$

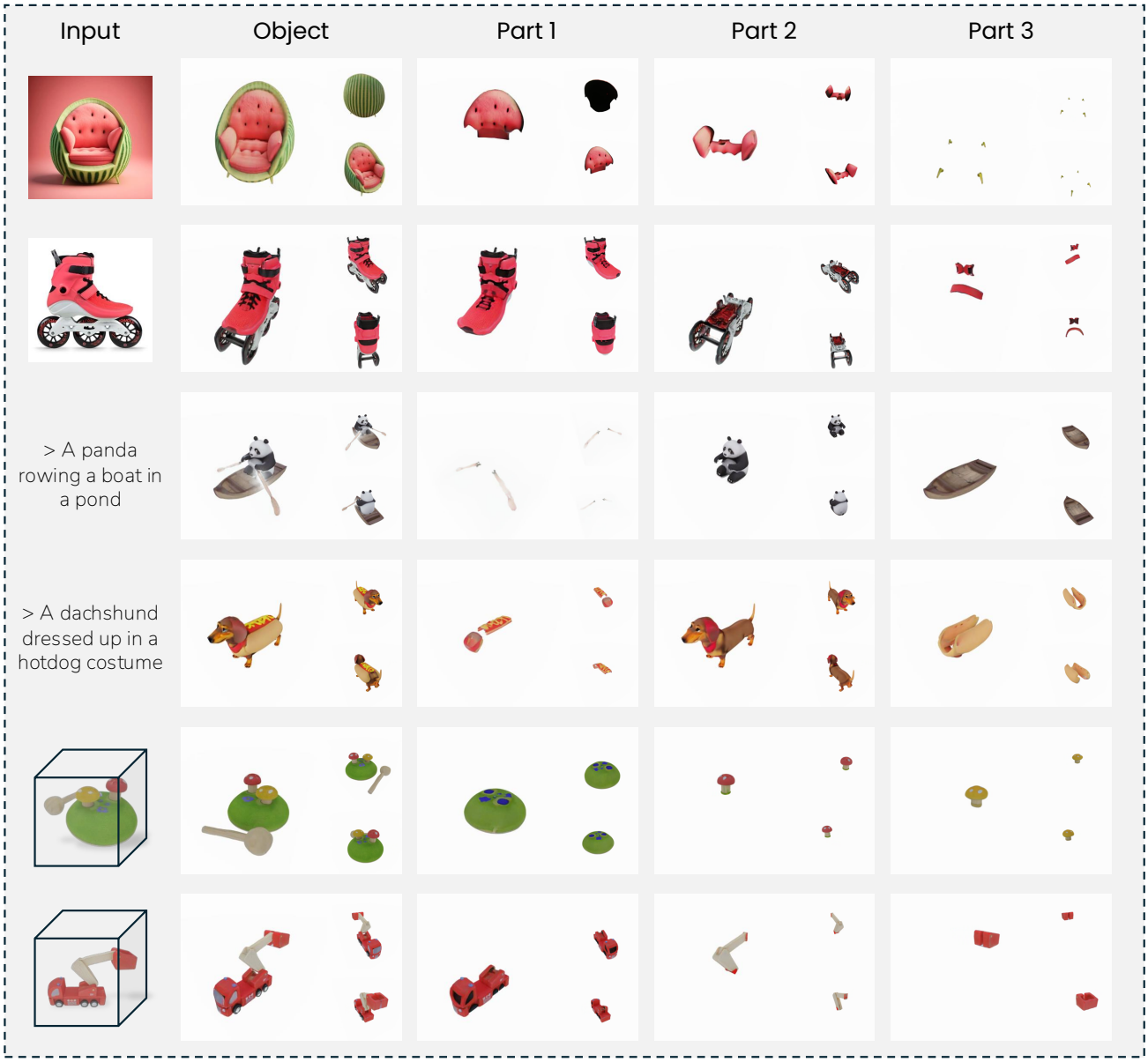

Figure A3. **More examples.** Additional examples illustrate that PartGen can process various modalities and effectively generate or reconstruct 3D objects with distinct parts.

$\exp(-\sum_{k=0}^{j}\sum_{h=1}^{N}\Delta\cdot\sigma^h(x_{ik}))$, Δ is the distance between two sampled points and $\sigma^h(x_{ik})$ is the opacity at position x_{ik} for part h, and $\hat{T}_{i,j-1}-\hat{T}_{i,j}$ is the visibility of the point.

A.8. 3D part editing

As shown in the main text and Figure 7, once 3D assets are generated or reconstructed as a composition of different parts through PartGen, specific parts can be edited using text instructions to achieve 3D part editing. To enable this, we fine-tune the text-to-multi-view generator using part multi-view images, masks, and text description pairs. Example of the training data are shown in Figure A1 (top). Notably, instead of supplying the mask for the part to be edited, we provide the mask of the remaining parts. This design choice encourages the editing network to imagine the part's shape without constraining the region where it has to project. The training recipe is similar to multi-view segmentation network.

To generate captions for different parts, we establish an annotation pipeline similar to the one used for captioning the whole object, where captions for various views are first generated using LLAMA3 and then summarized into a sin-

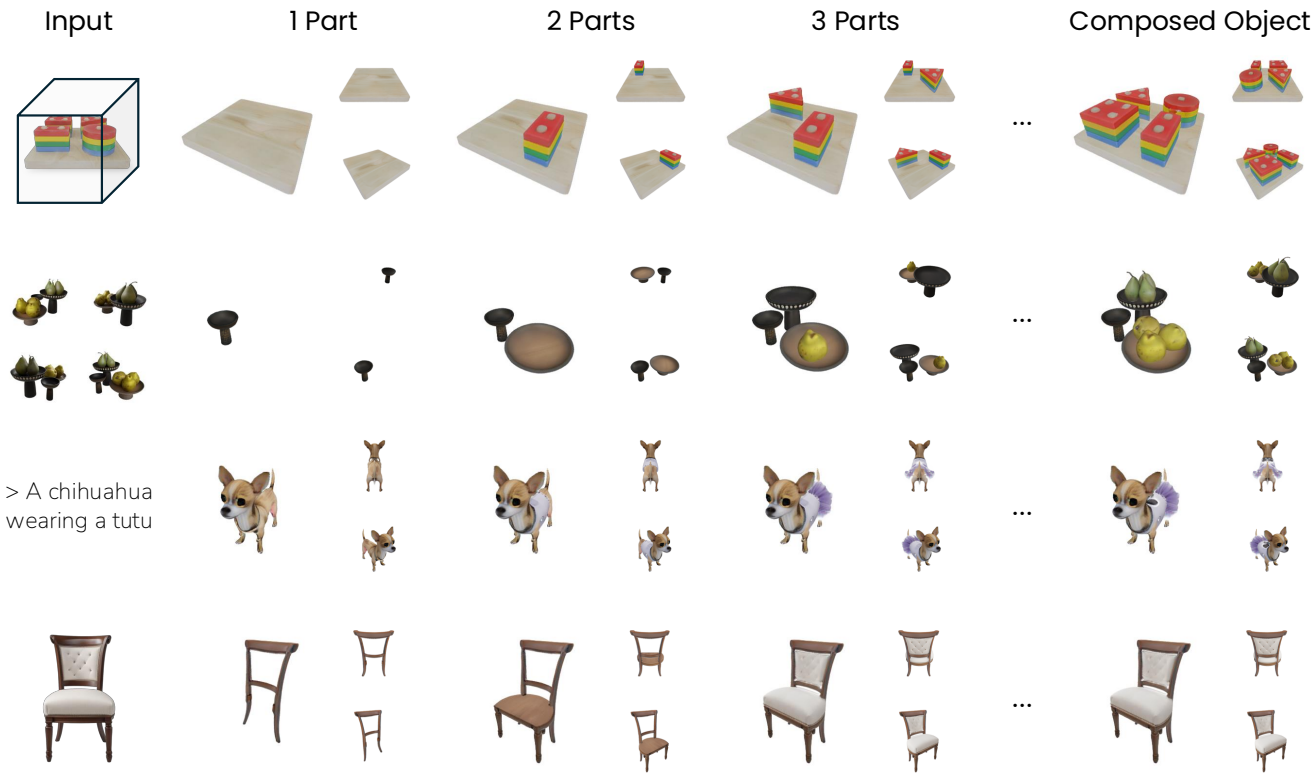

Figure A4. **Iteratively adding parts.** We show that users can iteratively add parts and combine the results of PartGen pipeline.

gle unified caption using LLAMA3 as well. The key challenge in this variant is that some parts are difficult to identify without knowing the context information of the object. We thus employ the technique inspired by [11]. Specifically, we use red annulet and alpha blending to emphasize the part being annotated. Example inputs and generated captions are shown in Figure A1 (bottom). The network is trained with 64 H100 GPUs, a batch size of 512, and the learning rate of 10^{-5} over 10,000 steps.

B. Additional Experiment Details

We provide a detailed explanation of the ranking rules applied to different methods and the formal definition of mean average precision (mAP) used in our evaluation protocol. Additionally, we report the recall at K in the automatic segmentation setting.

Ranking the parts. For evaluation using mAP and recall at K, it is necessary to rank the part proposal. For our method, we run the segmentation network several times and concatenate the results into an initial set $\mathcal{P}$ of segment proposals. Then, we assign to each segment $\hat{M} \in \mathcal{P}$ a reliability score based on how frequently it overlaps with similar segments in the list, *i.e.*,

$$s(\hat{M}) = \left| \left\{ \hat{M}' \in \mathcal{P} : m(\hat{M}', \hat{M}) > \frac{1}{2} \right\} \right|$$

where the *Intersection over Union* (IoU) [6] metric is given by:

$$m(\hat{M}, M) = \mathrm{IoU}(\hat{M}, M) = \frac{|\hat{M} \cap M| + \epsilon}{|\hat{M} \cup M| + \epsilon}.$$

The constant $\epsilon = 10^{-4}$ smooths the metric when both regions are empty, in which case $m(\phi, \phi) = 1$, and will be useful later.

Finally, we sort the regions M by decreasing score $s(M)$ and, scanning the list from high to low, we incrementally remove duplicates down the list if they overlap by more than $1/2$ with the regions selected so far. The final result is a ranked list of multi-view masks $\mathcal{M} = (\hat{M}_1, \dots, \hat{M}_N)$ where $N \leq |\mathcal{P}|$ and:

$$\forall i < j : \;\; s(\hat{M}_i) \geq s(\hat{M}_j) \;\wedge\; m(\hat{M}_i, \hat{M}_j) < \frac{1}{2}.$$

Other algorithms like SAM2 come with their own region reliability metric s, which we use for sorting. We otherwise apply non-maxima suppression to their ranked regions in the same way as ours.

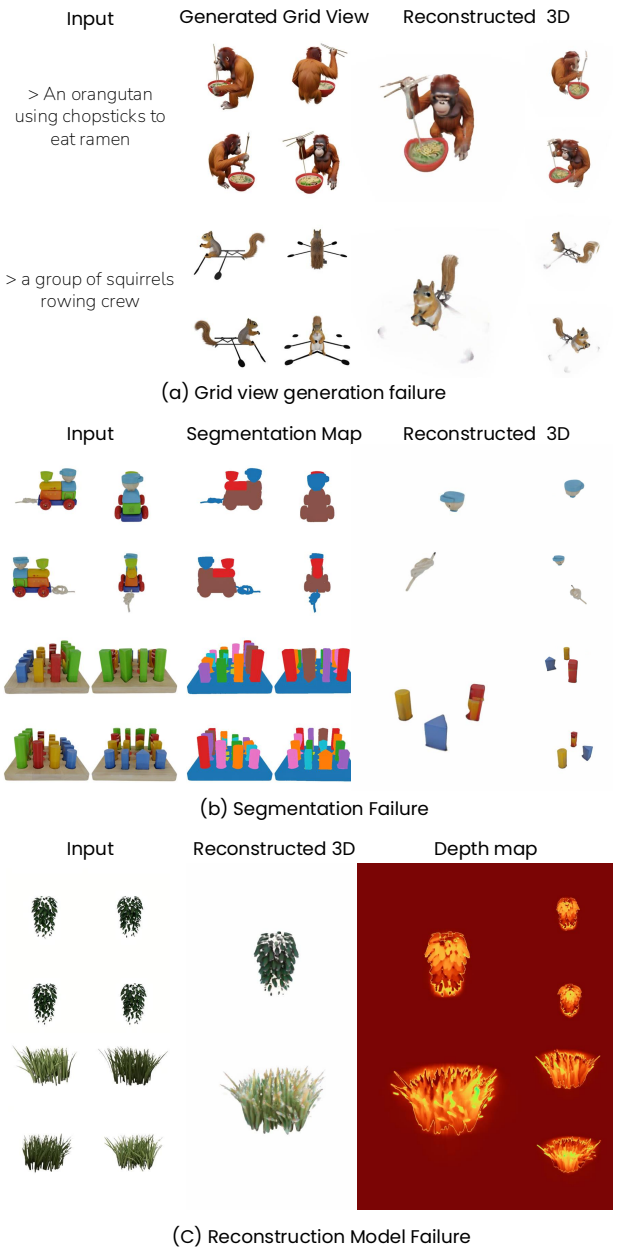

Figure A5. **Failure Cases.** (a) Multi-view grid generation failure, where the generated views lack 3D consistency. (b) Segmentation failure, where semantically distinct parts are incorrectly grouped together. (c) Reconstruction model failure, where the complex geometry of the input leads to inaccuracies in the depth map.

Computing mAP. The image I comes from an object $\mathbf{L}$ with parts $(\mathbf{S}^1, \dots, \mathbf{S}^S)$ from which we obtain the ground-truth part masks $\mathcal{S} = (M^1, \dots, M^S)$ as explained in Section 3.5 in the main text. We assign ground-truth segments to candidates following the procedure: we go through the list $\mathcal{M} = (\hat{M}_1, \dots, \hat{M}_N)$ and match the candidates one by one to the ground truth segment with the highest IOU, exclude that ground-truth segment, and continue traversing the candidate list. We measure the degree of overlap between a predicted segment and a ground truth segment as $m(\hat{M}, M) \in [0, 1]$. Given this metric, we then report the *mean Average Precision* (mAP) metric at different IoU thresholds τ. Recall that, based on this definition, computing the AP curve for a sample involves matching predicted segments to ground truth segments in ranking order, ensuring that each ground truth segment is matched only once, and considering any unmatched ground truth segments.

In more detail, we start by scanning the list of segments $\hat{M}_k$ in order $k = 1, 2, \dots$. Each time, we compare $\hat{M}_k$ to the ground truth segments $\mathcal{S}$ and define:

$$s^* = \operatorname*{argmax}_{s=1,\dots,S} m(\hat{M}_k, M_s).$$

If $m(\hat{M}_k, M_{s^*}) \geq \tau$, then we label the region M_s as retrieved by setting $y_k = 1$ and removing M_s from the list of ground truth segments not yet recalled by setting

$$\mathcal{S} \leftarrow \mathcal{S} \setminus \{M_{s^*}\}.$$

Otherwise, if $m(\hat{M}_k, M_{s^*}) < \tau$ or if $\mathcal{S}$ is empty, we set $y_k = 0$. We repeat this process for all k, which results in labels $(y_1, \dots, y_N) \in \{0, 1\}^N$. We then set the *average precision* (AP) at τ to be:

$$\mathrm{AP}(\mathcal{M}, \mathcal{S}; \tau) = \frac{1}{S} \sum_{k=1}^{N} \sum_{i=1}^{k} \frac{y_i y_k}{k}.$$

Note that this quantity is at most 1 because by construction $\sum_{i=1}^{N} y_i \leq S$ as we cannot match more proposal than there are ground truth regions. mAP is defined as the average of the AP over all test samples.

Computing recall at K. For a given sample, we define *recall at K* the curve

$$R(K; \mathcal{M}, \mathcal{S}, \tau) = \frac{1}{S} \sum_{s=1}^{S} \chi\left(\max_{k=1,\dots,K} m(\hat{M}_s, M_k) > \tau \right).$$

Hence, this is simply the fraction of ground truth segments recovered by looking up to position K in the ranked list of predicted segments. The results in Figure A2 demonstrate that our diffusion-based method outperforms SAM2 and its variants by a large margin and shows consistent improvement as the number of samples increases.

Seeded part segmentation. To evaluate *seeded part segmentation*, the assessment proceeds as before, except that a single ground truth part $\mathbf{S}$ and mask M is considered at a time, and the corresponding seed point $u \in M$ is passed to the algorithm $(\hat{M}_1, \dots, \hat{M}_K) = \mathcal{A}(I, u)$. Note that, because the problem is still ambiguous, it makes sense for the algorithm to still produce a ranked list of possible part segments.

C. Additional Examples

More application examples. We provide additional application examples in Figure A3, showcasing the versatility of our approach to varying input types. These include part-aware text-to-3D generation, where textual prompts guide the synthesis of 3D models with semantically distinct parts; part-aware image-to-3D generation, which reconstructs 3D objects from a single image while maintaining detailed part-level decomposition; and real-world 3D decomposition, where complex real-world objects are segmented into different parts. These examples demonstrate the broad applicability and robustness of PartGen in handling diverse inputs and scenarios.

Iteratively adding parts. As shown in Figure A4, we demonstrate the capability of our approach to compose a 3D object by iteratively adding individual parts to it. Starting with different inputs, users can seamlessly integrate additional parts step by step, maintaining consistency and coherence in the resulting 3D model. This process highlights the flexibility and modularity of our method, enabling fine-grained control over the composition of complex objects while preserving the semantic and structural integrity of the composition.

D. Failure Cases

As outlined in the method section, PartGen incorporates several steps, including multi-view grid generation, multi-view segmentation, multi-view part completion, and 3D part reconstruction. Failures at different stages will result in specific issues. For instance, as shown in Figure A5(a), failures in grid view generation can cause inconsistencies in 3D reconstruction, such as misrepresentations of the orangutan's hands or the squirrel's oars. The segmentation method can sometimes group distinct parts together, and limited, in our implementation, to objects containing no more than 10 parts, otherwise it merges different building blocks into a single part. Furthermore, highly complex input structures, such as dense grass and leaves, can lead to poor reconstruction outcomes, particularly in terms of depth quality, as illustrated in Figure A5(c).

E. Discussion

Compatibility with other reconstruction models. As claimed in the main text, our pipeline is designed to be compatible with various reconstruction models. In the current implementation, the multi-view diffusion models output four views at a fixed elevation ($20°$) and distinct azimuth angles ($0°, 90°, 180°, 270°$) and later the 3D parts and assets are reconstructed by lightplaneLRM. While other reconstruction methods may require fine-tuning to achieve optimal performance in this exact configuration, we demonstrate empirically (as shown in Figure A6) that even some open-source models [13–15] perform effectively for part-level reconstruction tasks without additional fine-tuning, and even when parts exhibit offsets from the center.

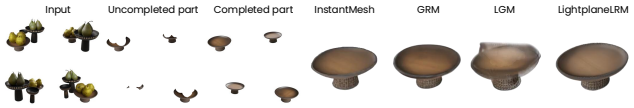

Figure A6. **Part reconstruction with different reconstruction models.** Results show that our pipeline is compatible with other reconstruction models without further fine-tuning.

Inter-part crossover and alignment. Results in Table 3 in the main text suggest that our current method inherently promotes effective part combinations because each part is reconstructed considering contextual information from the full object. Detailed assembly procedures supporting this claim are provided in Appendix A.7. Nevertheless, we acknowledge room for improvement. Enhanced part alignment and reduced crossover could potentially be achieved by training a Lightplane Reconstruction Model (LRM) to jointly reconstruct multiple parts simultaneously, leveraging multi-view inputs for all parts to generate interdependent 3D reconstructions. Additionally, jointly completing parts with 2D diffusion models is another promising approach currently under investigation. We expect these advancements will further refine the capabilities and insights of 3DGen.

F. Ethics and Limitation

Ethics. Our models are trained on datasets derived from artist-created 3D assets. These datasets may contain biases that could propagate into the outputs, potentially resulting in culturally insensitive or inappropriate content. To mitigate this, we strongly encourage users to implement safeguards and adhere to ethical guidelines when deploying PartGen in real-world applications.

Limitation. In this work, we focus primarily on object-level generation, leveraging artist-created 3D assets as our training dataset. However, this approach is heavily dependent on the quality and diversity of the dataset. Extending the method to scene-level generation and reconstruction is a promising direction but it will require further research and exploration.

References

[1] Ang Cao, Justin Johnson, Andrea Vedaldi, and David Novotny. Lightplane: Highly-scalable components for neural 3d fields. In *Proc. 3DV*, 2025. 1, 2

[2] Zheng Chong, Xiao Dong, Haoxiang Li, Shiyue Zhang, Wenqing Zhang, Xujie Zhang, Hanqing Zhao, and Xiaodan

Liang. Catvton: Concatenation is all you need for virtual try-on with diffusion models. *arXiv*, 2407.15886, 2024. 1

[3] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam S. Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramanathan, Zijian He, Peter Vajda, and Devi Parikh. Emu: Enhancing image generation models using photogenic needles in a haystack. *CoRR*, abs/2309.15807, 2023. 1

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. NeurIPS*, 2020. 1

[5] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *Proc. ICLR*, 2022. 1

[6] D. Larlus, G. Dorko, D. Jurie, and B. Triggs. Pascal visual object classes challenge. In *Selected Proceeding of the first PASCAL Challenges Workshop*, 2006. 4

[7] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3D: Fast text-to-3D with sparse-view generation and large reconstruction model. *Proc. ICLR*, 2024. 1

[8] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proc. WACV*, 2024. 1

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proc. ICML*, pages 8748–8763, 2021. 1

[10] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv*, 2202.00512, 2022. 1

[11] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proc. ICCV*, 2023. 4

[12] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proc. ICLR*, 2021. 1

[13] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. LGM: Large multi-view Gaussian model for high-resolution 3D content creation. *arXiv*, 2402.05054, 2024. 6

[14] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: efficient 3D mesh generation from a single image with sparse-view large reconstruction models. *arXiv*, 2404.07191, 2024.

[15] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large gaussian reconstruction model for efficient 3D reconstruction and generation. *arXiv*, 2403.14621, 2024. 6

[16] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv*, 2308.06721, 2023. 1