1 Introduction

Processing 3D objects, including generating them based on a textual description or an image, is an important aspect of Spatial Intelligence. Current 3D generators often treat objects or even entire scenes as monolithic shells. However, many applications require modeling their compositional structure, decomposing them into well-defined 3D parts to enable reasoning or manipulation at a finer granularity, such as applying textures and materials to each part separately. More specifically, a character in a video game should be decomposable into different parts to support animation or allow the game software to swap clothes or accessories. Windows and doors in the 3D model of a house need to be separate entities to allow user interaction, such as opening or closing them. Similarly, the design of a machine must consist of distinct parts to be functional (e.g., the cogs in a clock) or to enable 3D printing or other kinds of CNC manufacturing.

In this paper, we address the problem of generating 3D objects with a *compositional structure*. We introduce *AutoPartGen*, a new autoregressive model that can *directly* generate a 3D object part by part, building on a powerful latent 3D representation. AutoPartGen is robust, flexible, and scalable. As shown in Fig. 1, AutoPartGen can be applied, either independently or in combination with other models, to generate compositional 3D objects, scenes, or even cities, starting from 3D models, images, or text prompts. AutoPartGen solves three key problems to enable such applications: (i) object-to-parts, where it decomposes an existing 3D object into meaningful parts; (ii) image-to-parts, where the model generates 3D parts from an unstructured input image; and (iii) masks-to-parts, where users can provide 2D part masks to guide the generation. In the first two scenarios, AutoPartGen *automatically* predicts semantically meaningful 3D parts without requiring part annotations. In the third scenario, user-provided masks offer fine-grained control over the model partitioning.

Our autoregressive approach has two key benefits. First, it models the joint distribution over the object parts, ensuring that they fit together cohesively. Second, it enables the model to generate a variable number of parts, which is crucial since the number of parts is not fixed or known a priori.

We build AutoPartGen on recent advances in latent 3D representations and parameterize the 3D surface $\boldsymbol{x} \subset \mathbb{R}^3$ of the object using a latent code vector $\boldsymbol{z} \in \mathbb{R}^d$. We use the 3DShape2VecSet representation [62, 30, 63] and observe, for the first time, that this representation is inherently compositional. Specifically, we show that the concatenation $\boldsymbol{z} = \boldsymbol{z}^{(1)} \oplus \boldsymbol{z}^{(2)}$ of two codes $\boldsymbol{z}^{(1)}$ and $\boldsymbol{z}^{(2)}$ decodes into the union $\boldsymbol{x} = \boldsymbol{x}^{(1)} \cup \boldsymbol{x}^{(2)}$ of the corresponding surfaces $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$.

Based on this insight, we propose generating a sequence of latent codes $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(K)}$, each decoding into a corresponding 3D part $\boldsymbol{x}^{(k)}$. Crucially, the generation of each part is conditioned on an overall latent representation of the target 3D object $\boldsymbol{x}$ as well as on all previously generated parts $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k-1)}$. This conditioning improves the consistency of the generated parts, meaning that they better fit each other compared to the output of the methods that extract parts independently [5, 58].

As noted in [5], object decomposition is an ambiguous problem. For example, a chair might be decomposed into few high-level parts (e.g., seat, back, legs) or a more granular set of components (e.g., individual leg segments, cushion, backrest slats). This choice typically depends on the application or the preferences of the 3D artist creating the asset. We address this ambiguity by making the 3D autoregressive model *stochastic*, using denoising diffusion to generate the next part vector $\boldsymbol{z}^{(k)}$ based on the previously generated parts $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(k-1)}$ and the available evidence (i.e., the full 3D object, an image of the object, or 2D part masks, depending on the application). Importantly, we train a *single* diffusion model capable of handling all three cases.

We evaluate AutoPartGen against state-of-the-art part-aware 3D generators. Compared to the recent PartGen [5], AutoPartGen is easier to implement and maintain (as it does not require training several multi-view image generators) and more accurate. Compared to HoloPart [58], a method that completes a *pre-segmented* outer surface of a mesh to form 3D parts, AutoPartGen is more accurate and significantly more capable, as it can automatically discover parts and reconstruct them from either a 2D image or a "shell" 3D object, optionally guided by 2D masks, not requiring any 3D annotation.

2 Related Work

Generating a 3D object from a single image, or even just text, faces an obvious challenge: the 3D object contains significantly more information than the image or the text. This is similar to the

problem of generating images or videos from text, and it is solved by learning a prior, or conditional distribution, from billions of data samples. However, data of this size is unavailable for 3D objects. Authors address this problem by involving 2D image or video generators in the 3D generation process. We distinguish two main camps: multi-view direct and single-view latent 3D generation.

Multi-view 3D Generation. In multi-view 3D generation, one asks the image generator to do most of the lifting, generating several views of the 3D objects, and thus simplifying extracting a 3D object from them. First, this was done using Score Distillation Sampling (SDS) [41], an idea explored extensively in follow-ups like GET3D [13], ProlificDreamer [52], Dream Gaussians [49], Lucid Dreamer [9] which seek to achieve multi-view consistency via iterative (and slow) optimization of a radiance field (NeRF [43] or 3DGS [23]). A significant innovation, pioneered by UpFusion [22], 3DiM [53], Zero-1-to-3 [35] and MVDream [44], was to fine-tune the image generator to directly produce multiple consistent views of the object. By making the image generator more 3D aware, 3D reconstruction becomes simpler, as noted in InstantMesh [55], GRM [56], and others [54].

Single-view Latent 3D Generation. The alternative approach is to start from a single image of the object and directly reconstruct the 3D object from it. Because single-view reconstruction is extremely ambiguous, this requires to learn a reconstruction function. This was the path taken, for example, by LRM [19] and others [17, 47]. However, their deterministic reconstruction model cannot cope well with this ambiguity and often produces blurry outputs. Much better results were recently obtained by switching to stochastic 3D reconstruction based on *latent* diffusion. Some of the best single-image 3D reconstructors are based on the 3DShapeToVecSet [62] latent representation (also similar to Michelangelo's [64]). Building on it, CLAY [63], DreamCraft3D [46], CraftsMan [29], TripoSG [30], and others [10, 57, 60] are able to generate highly detailed and accurate 3D shapes. We build on this representation as well and show that it also supports compositionality very effectively.

Composable 3D Generation. Approaches to composable 3D generation typically start by decomposing objects into constituent parts. One common strategy represents objects as mixtures of primitives, often without semantic labels. For instance, SIF [16] models object occupancy using mixtures of 3D Gaussians. LDIF [15] represents shapes as a set of local deep implicit functions (DIFs), spatially arranged and blended using a template of Gaussian primitives. Methods such as Neural Template [20] and SPAGHETTI [1] achieve decompositions through auto-decoding. SALAD [28] utilizes SPAGHETTI for diffusion-based generation. PartNeRF [50] expands this concept by employing mixtures of NeRFs. NeuForm [31] and DifFacto [38] specifically target part-level controllability. DBW [37] uses textured superquadrics to decompose scenes. In contrast, another research direction emphasizes explicitly semantic parts. PartSLIP [34] and PartSLIP++ [67] segment objects into semantic components from point clouds using vision-language models. Part123 [33] adapts techniques from scene-level approaches like Contrastive Lift [2] to reconstruct object parts. PartSDF [48] learns latent codes for parts using an auto-decoder and then uses SALAD for part prediction. Comboverse [8] leverages single-view inpainting model and 3D generator for composable 3D generation with spatial-aware SDS optimization. HoloPart [58], a recent work, starts from the shell of a 3D object and a part-level segmentation for it and performs 3D amodal part completion.

The work most related to ours is PartGen [5]. This squarely sits on the 'multi-view direct' camp (see above). It uses multi-view diffusion models for segmentation and completion of compositional 3D objects from diverse modalities.

3D Segmentation. 3D parts can be obtained by segmenting a 3D object (although the resulting parts will generally be incomplete). Some approaches for semantic 3D segmentation such as [65, 27, 51, 24, 2] used neural fields to 'fuse' 2D semantic features in 3D. Contrastive Lift [2] introduced a slow-fast contrastive clustering scheme for 3D instance segmentation. Recent methods such as [25, 61, 42, 3] integrate SAM [26] and often CLIP to model multi-scale concepts, where LangSplat explicitly encodes scale information and N2F2 learns to bind concepts to scales automatically. Neural Part Priors [4] used learned priors for test-time decomposition. Additionally, efforts to develop 3D 'foundation' models [66, 7] are enabling zero-shot point cloud segmentation across diverse domains.

3 Method

Let $\boldsymbol{x} \subset \mathbb{R}^3$ be a *3D object* given by a surface embedded in $\mathbb{R}^3$. We assume that the object is *compositional*, meaning that it can be expressed as the union $\boldsymbol{x} = \bigcup_{k=1}^{K} \boldsymbol{x}^{(k)}$ of K disjoint *parts*