

SPACEFLOW: Locally Controllable 3D Generation

Neil De La Fuente^{1*}, Joan Lafuente Baeza^{1*}, Mukhammadali Sayfiddinov^{1*}, Felicia Scharitzer^{1*}, Marc Pollefeys¹, Ata Çelen¹, Sayan Deb Sarkar^{2†}, Elisabetta Fedele^{1†}

¹ETH Zürich, ²Stanford University *Equal Contribution †Equal Supervision

Current 3D generation methods lack explicit local control: geometric adherence is often defined by a global control strength, and appearance cannot be specified locally. We present SPACEFLOW, a training-free pipeline for locally controllable 3D generation from text descriptions and a collection of geometric primitives. Each primitive serves as a proxy for an object part and is assigned a local control level, enabling users to specify whether regions should strictly follow the input shape or allow generative completion. During structure generation, we enforce these spatial constraints within the generative flow process. For appearance synthesis, the generated structure is segmented and matched to the primitives. Each generated part is conditioned only on its assigned text or image cue, thereby limiting cross-part leakage. Regional geometry metrics demonstrate that SPACEFLOW preserves the specified geometry in high-control regions and enables plausible shape variation in low-control areas. A user study further indicates that the resulting balance between geometric fidelity and generative freedom remains competitive in overall quality. When evaluating appearance on fixed geometry, text-conditioned routing achieves state-of-the-art prompt faithfulness and color/material accuracy. Qualitative results additionally show localized routing of image cues.

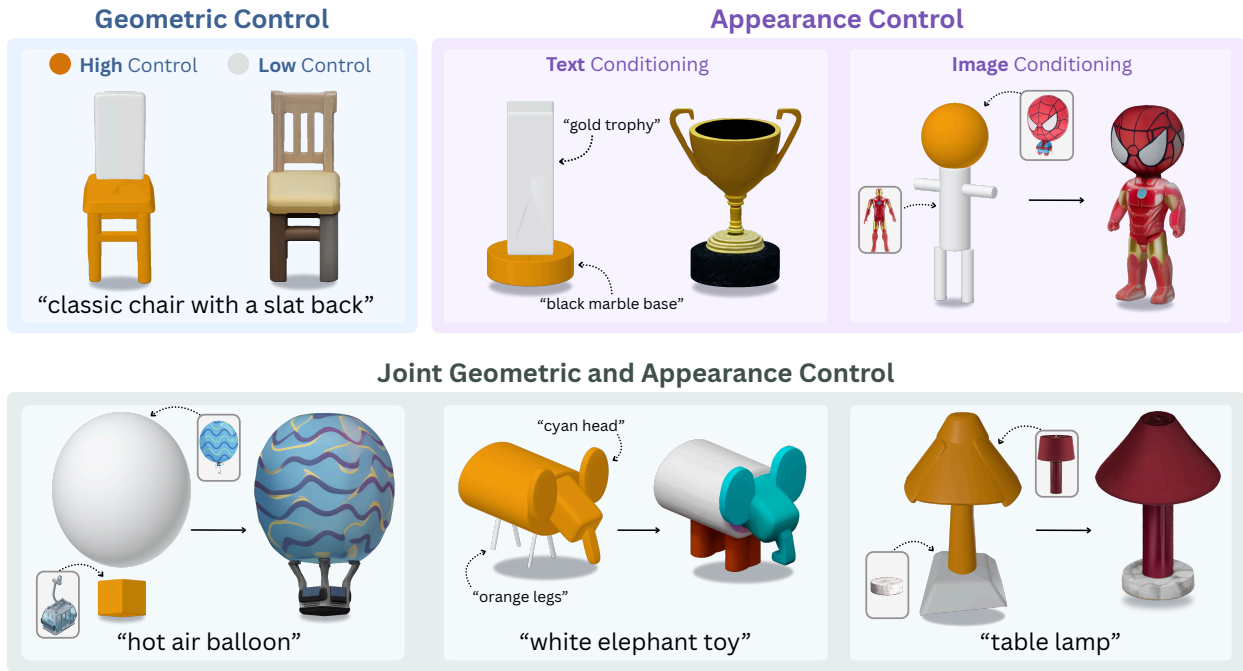


Figure 1 Overview of Local Geometric and Appearance Control Capabilities. *Top left (Geometric Control):* SPACEFLOW assigns different control strengths to editable geometric inputs: orange regions (strong geometric control) preserve the specified geometry, whereas grey regions (weak geometric control) allow prompt-driven completion. *Top right (Appearance Control):* Local text or image cues are bound to their target primitives, localizing distinct materials on specified regions, and combining visual references across different parts. *Bottom (Joint Geometric and Appearance Control):* Additional examples show localized shape and appearance control.

1 Introduction

Recent text- and image-conditioned 3D generative models can synthesize high-quality textured assets across increasingly diverse object categories [1–8]. These advances dramatically lower the barrier from a visual idea or reference to an initial 3D candidate asset, with applications in games, immersive media, and digital content creation [9–13]. Although the quality of assets generated from a single image or text prompt continues to improve, integrating such generators into artistic workflows requires more interactive and controllable guidance mechanisms [14–17].

In practice, asset creation is iterative [18, 19] and part-specific [20–24], rather than one-shot: creators establish coarse structure, inspect generated candidates, retain some design decisions, and revise others [25]. Some parts may have to follow an intended shape, whereas others are left unconstrained, relying on the generative prior [26]. Supporting this process requires control not only over *what* is generated, but also over *where* each instruction applies and *how strictly* it should be followed [20, 27, 28].

To support this need, recent methods augment general-purpose 3D generators [3, 29] with controllable geometric or appearance guidance [28, 30]. On one hand, proxy-guided approaches [28, 31] enable *geometric* conditioning by guiding 3D generation with an input 3D shape. However, they apply this conditioning *globally*, such that all regions are equally constrained by the input geometry. On the other hand, other approaches introduce test-time, part-aware guidance for *appearance* transfer [30] or localized editing of implicit representations [32], while leaving geometry generation unconstrained or operating post-hoc on pre-existing representations. Appearance guidance is also limited to a single global condition, preventing different cues from being assigned to different regions. Existing approaches therefore lack a unified mechanism for local control over both geometry and appearance.

We propose SPACEFLOW, a unified approach for local control of both geometry and appearance during 3D generation, as shown in Fig. 1. In practice, the user specifies the geometry and scene layout as an explicit geometric input, decomposed into an arbitrary set of local parts that can be addressed independently. For geometric conditioning, we assign a local control level to each part: higher levels enforce closer adherence to the specified geometry, whereas lower levels leave more freedom to the generative prior. The formulation is agnostic to how individual parts are represented, ranging from coarse geometric primitives to detailed meshes. Following [28], we represent parts as superquadric primitives [33], which provide a compact, expressive, and easily editable representation. These can either be defined manually or initialized from existing geometry using shape decomposition methods [34, 35].

We use the same part decomposition for appearance conditioning. Specifically, the user can assign a distinct appearance condition to each part, allowing different regions to be controlled independently. We achieve this by restricting cross-attention to the condition associated with each part, while retaining global self-attention across all tokens. This enables distinct local appearances while preserving coherence and smooth blending across the generated asset.

We evaluate the geometric and appearance components separately and demonstrate their combined capabilities on diverse multipart assets. Regional geometry metrics show that SPACEFLOW preserves strongly constrained regions more faithfully than weak global conditioning, while allowing greater variation in weakly constrained regions than strong global conditioning. A user study further supports this balance between geometric fidelity and generative freedom. On fixed geometry, our VLM-as-a-judge evaluation of localized text-conditioned appearance shows higher prompt faithfulness and color/material accuracy than the evaluated TRELLIS and GuideFlow3D variants [3, 30]. We additionally demonstrate localized image-conditioned appearance through qualitative results.

In summary, our technical contributions are:

- An approach to locally control the strength of *geometric* conditioning for 3D generation.
- An approach to locally condition the *appearance* of generated 3D assets.
- A *unified framework* combining both approaches to enable local geometric and appearance control for 3D generation.

2 Related Work

3D Asset Generation. Generative 3D modeling has advanced rapidly in output fidelity and representation flexibility, from explicit point clouds [36] and implicit functions [7] to compact latent-space approaches [37–39]. Recently, large-scale generators have achieved strong fidelity: CLAY [1] leverages multi-resolution latent spaces, Hunyuan3D [2] scales diffusion for high-resolution textured assets, SAM 3D [29] enables single-image 3D reconstruction, and TRELLIS [3] introduces structured latents factoring geometry and appearance on a sparse voxel grid. However, these models condition on a global text or image prompt and lack fine-grained, localized part control.

Spatially Controllable 3D Generation. Spatially controllable generation conditions on explicit 3D inputs, such as voxels, primitives, proxy shapes, or meshes, rather than ambiguous text or images [28, 39–41]. Existing frameworks either fine-tune generative models to accept structural conditions [31, 41, 42] at additional training cost, or enforce test-time constraints via optimization and latent projection [28, 40] without retraining. Closest to our structural setting, SpaceControl [28] injects user-specified geometry into a pretrained generator’s latent space with a control strength that trades geometric fidelity for generative realism. However, since this trade-off is applied uniformly across the entire asset, it cannot handle scenarios where designated regions must strictly preserve input geometry while others are left unconstrained for generative completion [28].

Appearance Control for 3D Assets. Appearance control has been widely explored across mesh texturing [43–47], radiance fields, and 3D Gaussian splats [48–50]. These methods primarily formulate appearance control as post-hoc texturing or stylization of an existing 3D asset, whereas our focus is localized appearance conditioning during the generative flow process [43, 44, 48, 49]. GuideFlow3D [30] provided training-free guidance of a pretrained rectified-flow model using a self-similarity loss for robust appearance transfer. However, it transfers a single appearance source globally across the generated asset. In contrast, SPACEFLOW routes independent visual conditions to specific semantic parts, eliminating cross-part leakage.

Part-Aware Generation and Editing. A complementary line of work introduces part-aware generators, which incorporate explicit part structure through hierarchical graphs and programs [21, 24], implicit or radiance-field representations, and diffusion over point or latent spaces [22, 23, 26, 51], including recent part-level reconstruction and synthesis methods [20, 42, 52–54]. In contrast, SPACEFLOW does not require the generated asset to be represented as parts; it uses primitives and part assignments only as localized control signals for a pretrained generator.

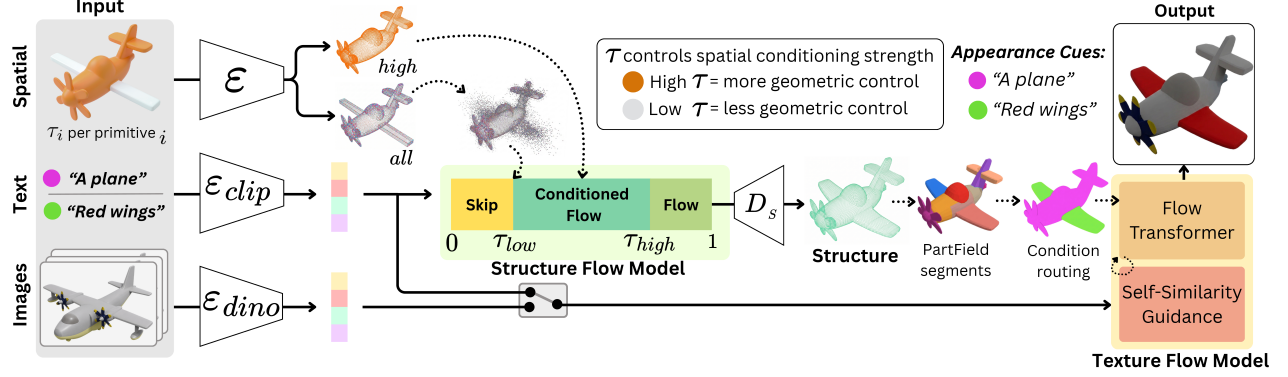


Figure 2 SPACEFLOW pipeline. Given editable geometric controls with local geometric conditioning strengths τ_i , SPACEFLOW encodes the high-control regions and injects them into the sparse-structure flow process over a controlled time interval. The resulting structure is segmented with PartField [55], then per-part text or image conditions are routed to each part, and the latent is finally passed to an appearance flow model and refined with self-similarity guidance, producing a 3D asset with localized geometric and appearance control.

3 Method

3.1 Problem Formulation

Given a text prompt p and a user-edited set of deformable primitives $\mathcal{S} = \{S_i\}_{i=1}^N$, our goal is to generate a textured 3D asset $X = (G, T)$ whose geometry and appearance adhere to localized spatial controls. Each primitive S_i is parameterized by a control strength $\tau_i \in \{\tau_{\text{low}}, \tau_{\text{high}}\}$ and an optional appearance cue $a_i \in \mathcal{A}$ such as an image or text prompt. The control strengths τ_i determine the local trade-off between geometric fidelity and generative freedom. τ_{high} promotes strict adherence to the input, whereas τ_{low} relaxes the spatial constraints to allow for flexible modeling using a generative prior. As shown in Fig. 3, assigning control independently per primitive enables the trade-off to vary spatially across the asset. Furthermore, when an appearance cue a_i is available, it only influences the appearance of its corresponding primitive region S_i .

3.2 Preliminaries

TRELLIS [3] is a pretrained 3D generator based on structured latents (SLAT), which attach local feature vectors to the active voxels of a sparse 3D grid. It generates assets in two stages using rectified flow transformers: first establishing the coarse sparse voxel structure, and then synthesizing the local geometry and appearance latents on the active voxels. Finally, different decoders are applied to map SLAT to diverse 3D representations [56–58] of high quality.

SpaceControl [28] is a training-free method built on TRELLIS that enables conditioning asset generation on geometric inputs. Given a user-specified geometric control input $\mathcal{S}$ (primitives or meshes), it encodes the geometry into the structure latent space via TRELLIS’s pretrained VAE encoder to obtain a clean latent $\mathbf{z}_1$. Instead of sampling from pure Gaussian noise, SpaceControl perturbs $\mathbf{z}_1$ to an intermediate flow timestep τ , producing a partially noisy latent $\mathbf{z}_\tau$, and starts denoising from this state. The timestep τ serves as a global control parameter governing the trade-off between geometric adherence and generative freedom across the entire asset.

3.3 Method Overview

We propose SPACEFLOW, a training-free framework built upon TRELLIS [3]. As shown in Fig. 2, our method takes as input a set of editable superquadric control shapes with per-primitive geometric conditioning strengths τ_i , alongside optional appearance cues a_i (text descriptors or reference images) for localized appearance control.

The primitives and their corresponding control levels τ_i first guide the Structure Flow Model (Sec. 3.4) during sparse-structure generation. The resulting sparse structure is then passed to the appearance stage. We extract and cluster PartField [55] features on this structure to segment the geometry, match each segment to its respective input primitive, and route the local appearance conditions to their target regions in the Appearance Flow Model (Sec. 3.5). Finally, self-similarity guidance [30] is applied to ensure coherent appearance and smooth cross-part transitions.

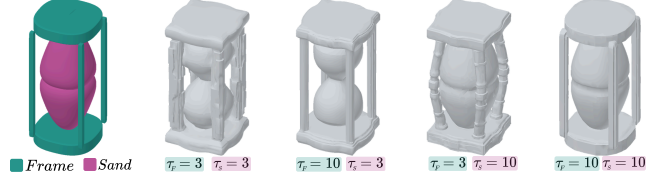


Figure 3 Local spatial control. Starting from the same "hourglass" superquadric set and prompt, high control ($\tau=10$) preserves the corresponding part’s input geometry, while low control ($\tau=3$) allows generative variation. The four settings showcase independent control of the frame and sand.

3.4 Local Geometric Control

To implement local geometric control during structure generation, we partition the user-defined primitives $\mathcal{S}$ according to their assigned control levels τ_i :

$$\mathcal{S}_r = \{S_i \in \mathcal{S} \mid \tau_i = \tau_r\}, \quad r \in \{\text{low}, \text{high}\}, \quad (1)$$

where τ_{low} and τ_{high} denote the endpoints of the flow interval over which spatial guidance is enforced.

First, we convert the complete set $\mathcal{S}$ and the high-control subset $\mathcal{S}_{\text{high}}$ into meshes independently. Both meshes are encoded into the structure latent space via the TRELLIS VAE to obtain clean latents $\mathbf{z}_1^{\text{all}}$ and $\mathbf{z}_1^{\text{high}}$. Additionally, we construct a binary spatial mask $\mathbf{M}_{\text{low}}$ by generating a bounding box around $\mathcal{S}_{\text{low}}$ and downsampling the region to match the dimensions of the latent grid. Following [28], we perturb $\mathbf{z}_1^{\text{all}}$ with noise at timestep τ_{low} to obtain $\mathbf{z}_{\tau_{\text{low}}}^{\text{all}}$ and begin the denoising process, thus establishing the coarse layout of the more flexible, low-control components.

To preserve the precise geometry and structural details of high-control regions, we inject the clean high-control latent $\mathbf{z}_1^{\text{high}}$ at each denoising step t . Using the low-control mask $\mathbf{M}_{\text{low}}$, we compute a spatially masked weighted average to update the latent state $\mathbf{z}'_t$:

$$\mathbf{z}'_t = (1 - \mathbf{M}_{\text{low}}) \odot (\alpha \mathbf{z}_1^{\text{high}} + (1 - \alpha) \mathbf{z}_t) + \mathbf{M}_{\text{low}} \odot \mathbf{z}_t \quad (2)$$

where $\mathbf{z}_t$ is the intermediate denoised prediction, $\odot$ denotes the element-wise product, and α is the blending weight.

We leverage a RePaint-inspired resampling strategy [59] over K iterations for a smooth transition between high- and low-control regions. In each iteration, we perturb the blended state from t_{i+1} back to step t_i following the noise schedule, denoise forward to t_{i+1} in a single solver step, and re-apply the masked blending in Eq. 2. This iterative resampling is performed across the interval $[\tau_{\text{low}}, \tau_{\text{high}}]$, corresponding to the *Conditioned Flow* phase shown in Fig. 2.

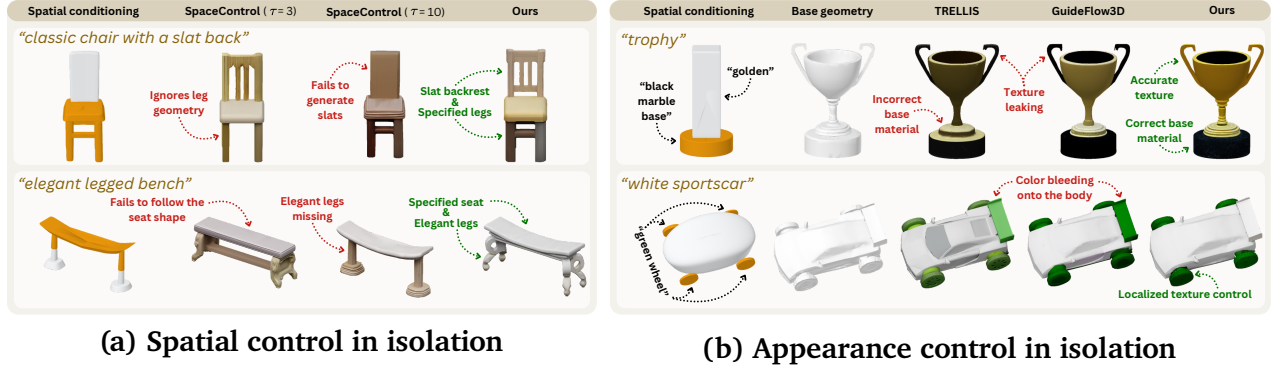


Figure 4 Spatial and appearance control in isolation. (a) **Spatial control.** Uniform control with SpaceControl either loses specified structure ($\tau=3$) or constrains the full object ($\tau=10$), whereas SPACEFLOW preserves the high-control regions and allows the generative prior to complete the low-control regions. (b) **Appearance control on fixed geometry.** Global conditioning can miss part-level cues or spread them beyond their targets; SPACEFLOW routes each color or material cue to its target part while retaining coherent textures.

3.5 Local Appearance Routing and Guidance

The appearance stage takes as input the generated sparse structure and the local appearance cues $\mathcal{A} = \{a_i\}_{i=1}^N$, which default to the global text prompt p when unspecified. To ensure each local cue affects only its designated primitive S_i , we propose *local appearance routing*. Specifically, we define a routing volume Φ that maps every appearance-latent voxel to its corresponding condition a_i , dictating which prompt guides each spatial region along the flow trajectory.

Constructing Φ requires establishing semantic correspondences between the generated structure and initial primitives. Direct per-voxel geometric assignment is unreliable as low-control regions can be reshaped by the generative prior, causing drift from their original boundaries. To resolve this, we extract PartField [55] descriptors from the generated structure, cluster the active voxels via K -means, and assign each semantic cluster as a unit to its closest input primitive. This produces a geometrically adaptive part-consistent routing volume, accommodating for structural reinterpretation. Details on cluster-to-primitive assignment and small-part handling are described in Supp.

We build on the optimization-guided rectified flow mechanism in [30], extending its global appearance guidance with the routing volume Φ . Each cue a_i is encoded into an embedding $\mathbf{e}_i$. Within every cross-attention layer, each latent voxel attends only to the embedding selected by Φ . This confines each cue to its user-intended region.

Cross-attention routing alone does not completely isolate local cues because subsequent self-attention can inadvertently propagate and blend appearance features across distinct parts [60, 61]. We inject a soft self-attention bias modulated by Φ . For any query-key pair sharing the same assigned condition, we add a positive bias of $\log \beta$ ($\beta > 1$) to their raw attention logit [62]. This prioritizes intra-part feature propagation without blocking interactions across the rest of the asset.

To further encourage a coherent appearance within semantic parts, we optimize a test-time supervised contrastive loss over the PartField semantic clusters $\{\ell_j\}$:

$$\mathcal{L} = -\frac{1}{|\mathcal{V}|} \sum_j \log \frac{\sum_{k: \ell_k = \ell_j, k \neq j} \exp(s_{jk}/\tau_c)}{\sum_{k \neq j} \exp(s_{jk}/\tau_c)}, \quad (3)$$

where $\mathcal{V}$ is the set of structure voxels, s_{jk} is the cosine similarity between voxel latents, and τ_c is the temperature. During inference, we perform test-time optimization directly on the intermediate appearance latents without modifying the pretrained network weights. Specifically, at each guidance

step, we compute $\nabla_{\mathbf{z}} \mathcal{L}$ and update the latent state. This pulls together features within the same semantic part while pushing distinct parts apart. To maintain optimal visual fidelity, we track the latents along the guidance trajectory and select the lowest-loss state for final decoding. Full implementation details, routing hyperparameters, and optimizer configurations are provided in Supp.

4 Experimental Results

Evaluation. We evaluate each control component separately to isolate their effects: Sec. 4.1 analyzes local geometric control, while Sec. 4.2 evaluates localized appearance conditioning.

Dataset. To evaluate localized control, we constructed a dataset of 83 3D assets covering diverse object categories (similar to Objaverse [63]). Each asset is defined by superquadric primitives with designated high-control ($\tau_i=10$) and low-control ($\tau_i=3$) regions, alongside local appearance prompts a_i . We refer to Supp. for more details.

4.1 Geometric Control

We evaluate our localized geometric control formulation ($\tau_{\text{high}}=10$ and $\tau_{\text{low}}=3$) in direct comparison to SpaceControl with uniform global guidance scales ($\tau=3$ or $\tau=10$). Specifically, $\tau=3$ allows the generative model substantial freedom to generate natural, plausible geometry while retaining a small level of structural guidance, whereas $\tau=10$ enforces strict geometric adherence to the input shapes.

4.1.1 Quantitative Results

We evaluate the performance of SPACEFLOW in local geometric control using *GPT 5.6 Sol* as an automated VLM judge. We assess prompt and geometric fidelity (PGF), which captures high-control shape adherence and low-control prompt completion, alongside realism and overall preference. Each comparison is evaluated in three passes with random ordering and aggregated by majority vote.

Tab. 1 shows the head-to-head win rates against global guidance baselines. These results illustrate the trade-off of uniform guidance: it forces a single compromise across the entire asset. A high global scale ($\tau=10$) enforces geometry at the expense of realism, whereas a low scale ($\tau=3$) allows generative freedom but weakens input adherence.

PGF results demonstrate that SPACEFLOW reliably follows the high-control input geometry while using the generative prior to match the prompt in low-control regions. The *Overall* metric confirms that, when evaluated on the complete input, our method is preferred against both baselines.

Baseline	Prompt and geometric fidelity		Realism		Overall	
	Win rate	95% CI	Win rate	95% CI	Win rate	95% CI
SpaceControl ($\tau=3$)	78.3%	[68.3, 85.8]	43.4%	[33.2, 54.1]	75.9%	[65.7, 83.8]
SpaceControl ($\tau=10$)	66.3%	[55.6, 75.5]	85.5%	[76.4, 91.5]	68.7%	[58.1, 77.6]

Table 1 Localized geometric control. Win rates (%) of SPACEFLOW ($\tau_{\text{low}}=3$, $\tau_{\text{high}}=10$) against SpaceControl [28] using a uniform global control strength. Brackets report 95% Wilson CIs over 83 assets.

4.1.2 Geometric and Feature-Space Evaluation

To complement the VLM evaluation, we quantify spatial fidelity, regional feature deviation, and semantic alignment using three metrics:

(a) Regional Chamfer Distance. We evaluate spatial fidelity to the input primitives by computing the Chamfer Distance (CD) separately on the designated high-control (CD_{high}) and low-control (CD_{low}) regions. Specifically, we partition mesh faces by mapping each face centroid to the nearest input primitive surface, inheriting its assigned control level. All meshes are normalized to a unit cube, and we report the symmetric mean-squared Chamfer distance ($\times 10^3$). To quantify the difference in control enforcement across regions, we compute the regional delta $\Delta CD = CD_{\text{low}} - CD_{\text{high}}$, where a higher positive value indicates that the model effectively differentiates control strengths, preserving specified geometry more strictly in high-control regions than in low-control ones.

(b) Feature-Based Deviation. As a complementary measure of local control, we compare each source voxel structure V^s , for $s \in \{\text{local-}\tau, \tau=3\}$, with a uniformly guided $\tau=10$ reference V^{10} . This reference remains close to the input while providing an object-like surface on which PartField descriptors are meaningful; direct adherence to the superquadric proxy is measured separately by Chamfer distance. Each active voxel $v \in V^s$ has a spatial coordinate x_v and an ℓ_2 -normalized PartField feature z_v^s sampled at x_v . We first match it to its nearest reference voxel:

$$u^*(v) = \underset{u \in V^{10}}{\operatorname{argmin}} \|x_v - x_u\|_2. \quad (4)$$

We then compute the cosine distance in PartField space:

$$d_{\text{feat}}^{s \rightarrow 10}(v) = 1 - \mathbf{z}_v^s \cdot \mathbf{z}_{u^*(v)}^{10}. \quad (5)$$

As for regional Chamfer distance, each voxel inherits the high- or low-control label of its nearest input primitive. Let V_r^s denote the source voxels assigned to region $r \in \{\text{high}, \text{low}\}$. Their mean feature distance is

$$D_r^{s \rightarrow 10} = \frac{1}{|V_r^s|} \sum_{v \in V_r^s} d_{\text{feat}}^{s \rightarrow 10}(v). \quad (6)$$

Finally, the feature delta $\Delta D^{s \rightarrow 10} = D_{\text{low}}^{s \rightarrow 10} - D_{\text{high}}^{s \rightarrow 10}$ is positive when low-control features deviate more from the reference than high-control features, enabling direct comparison with SpaceControl.

(c) Text Alignment. In order to verify semantic fidelity to the text prompt, we compute the CLIP similarity between rendered views of the generated assets and the input text.

Results. Tab. 2 reports additional metrics evaluating our method against SpaceControl baselines under uniform global guidance ($\tau=3$ and $\tau=10$). The regional Chamfer Distance on high-control areas (CD_{high}) demonstrates that SPACEFLOW provides significantly stronger geometric adherence than $\tau=3$, approaching the strict compliance of the $\tau=10$ baseline.

The regional deltas show that while uniform SpaceControl baselines exhibit virtually no difference in the level of control

Method	$CD_{\text{high}} \downarrow$	$\Delta CD \uparrow$	$\Delta D^{s \rightarrow 10} \uparrow$	CLIP $\uparrow$
SpaceControl ($\tau=3$)	15.99	-0.14	0.01	0.232
SpaceControl ($\tau=10$)	0.94	0.11	0.00 [†]	0.226
SPACEFLOW (<i>Ours</i>)	3.65	6.05	0.15	0.233

Table 2 Quantitative evaluation of spatial control selectivity and semantic alignment. We report the regional Chamfer Distance delta ($\Delta CD = CD_{\text{low}} - CD_{\text{high}}, \times 10^3$), the feature deviation delta to the reference ($\Delta D^{s \rightarrow 10} = D_{\text{low}}^{s \rightarrow 10} - D_{\text{high}}^{s \rightarrow 10}$), and CLIP text-image similarity. Positive Δ values indicate effective localized control selectivity between high- and low-control regions. [†] ΔD is 0.00 by definition for $\tau=10$ as $D^{10 \rightarrow 10}=0$.

across regions ($\Delta CD \approx 0$ and $\Delta D^{s \rightarrow 10} \approx 0$), SPACEFLOW achieves a substantial positive delta in both physical geometry (ΔCD) and feature space ($\Delta D^{s \rightarrow 10}$). This confirms that our method effectively leverages the two control levels, performing more constrained generation in high-control than low-control regions. Furthermore, CLIP similarity remains comparable across methods, suggesting that this selectivity does not reduce semantic alignment. Together, these objective metrics ground the VLM judge’s findings, demonstrating that localized control successfully bypasses the rigid trade-off of uniform guidance across the entire asset.

4.1.3 Human Evaluation

To complement our quantitative evaluation, we conducted a pairwise user study assessing the perceptual quality of the generated assets. We compared SPACEFLOW against SpaceControl baselines under uniform guidance ($\tau = 3$ and $\tau = 10$). For each scene, three variants (SPACEFLOW, $\tau = 3$, and $\tau = 10$) were generated from identical text prompts and superquadric inputs. Participants inspected interactive 3D model pairs side-by-side and evaluated them based on shape fidelity, realism, and overall preference, yielding a total of 337 trials across 29 evaluators.

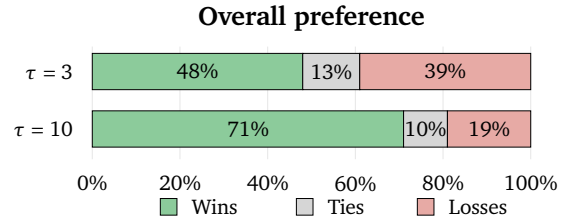


Figure 6 User study showing overall preference for SPACEFLOW over SpaceControl [28] with uniform $\tau = 3$ and $\tau = 10$ guidance across 337 evaluation trials.

As shown in Fig. 6, SPACEFLOW remains competitive in overall preference against low guidance ($\tau = 3$), with a 48% win rate, and is strongly preferred to the high-guidance baseline ($\tau = 10$), with a 71% win rate. Overall, the study shows that local control avoids the rigid trade-off of uniform global guidance while maintaining overall quality.

4.2 Appearance Control

We assess localized appearance control using *GPT 5.6 Sol* as an automated vision-language judge, evaluating multiple rendered views of each asset.

Text-conditioned appearance. We evaluate on 83 text-conditioned assets across three variants sharing an identical base structure: (i) **TRELLIS**, applying the appearance flow model on fixed geometry without guidance or routing; (ii) **GuideFlow3D**, employing self-similarity guidance without routing; and (iii) **SPACEFLOW** (ours), adding part-wise condition routing to self-similarity guidance (Sec. 3.5).

Baseline	Prompt faithfulness		Texture detail		Overall	
	Win rate	95% CI	Win rate	95% CI	Win rate	95% CI
TRELLIS	63.9%	[53.1, 73.4]	31.3%	[22.4, 41.9]	59.0%	[48.3, 69.0]
GuideFlow3D	58.5%	[47.7, 68.6]	37.5%	[27.7, 48.5]	57.3%	[46.5, 67.5]

Table 3 Pairwise appearance evaluation. Win rates (%) of SPACEFLOW against fixed-structure appearance baselines. Brackets report 95% Wilson CIs over the set of test assets.

The input specifies per-primitive requests, such as an orange backrest or a red cushion, consumed by all methods. We perform two complementary evaluations: First, an *absolute rating* on a 1–10 scale across

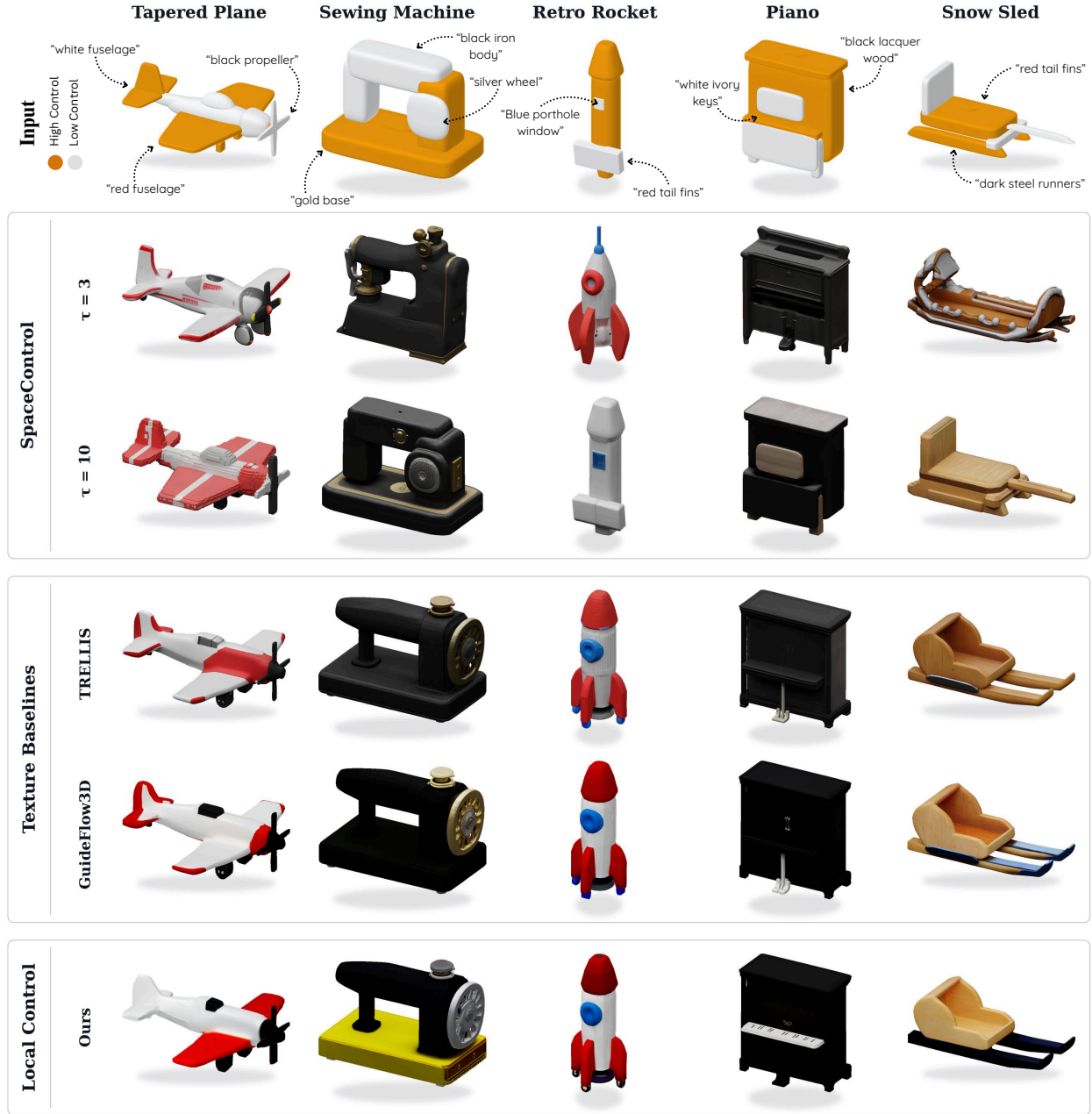


Figure 5 Qualitative comparison of spatial and appearance control across diverse assets. *Control (top):* Input superquadric proxies with designated local control strengths (orange: high control $\tau_i = 10$; gray: low control $\tau_i = 3$) and localized part-level appearance prompts. *SpaceControl [28]:* Baselines under uniform control strengths ($\tau = 3$ vs. $\tau = 10$), illustrating the trade-off between generative reinterpretation and strict adherence to the proxy geometry. *Texture Baselines:* Appearance baselines (TRELLIS [3] and GuideFlow3D [30]) built directly on our generated structure to isolate texture synthesis. *Local Control (Ours):* SPACEFLOW provides localized geometric control (generating text-aligned geometric details in low-control regions while preserving high-control shapes) alongside accurate part-level texture routing.

four metrics: *prompt faithfulness* (evaluating localized per-primitive adherence), *color and material accuracy* (evaluating the correctness of colors and surface finishes relative to the prompt), *texture detail*, and *overall preference* (see Supp. for details). We report the mean scores in Tab. 4. Second, analogous to our geometric evaluation, we conduct a *pairwise head-to-head evaluation* to measure win rates across prompt faithfulness, texture detail, and overall preference.

In pairwise comparisons (Tab. 3), SPACEFLOW shows an advantage on prompt-faithfulness and overall preference (considering the input requirements) over TRELIS [3] and GuideFlow3D [30]. This shows that part-wise routing improves prompt fidelity by ensuring that localized appearance cues are faithfully reflected in the final asset and eliminating cross-part color bleeding. Conversely, both baselines are favored on texture detail, indicating that localized routing prioritizes part-level prompt adherence at some cost to unconstrained texture quality.

The complementary absolute ratings in Tab. 4 show that SPACEFLOW’s gains in localized prompt fidelity and material accuracy compensate for its lower unconstrained texture uniformity, yielding the highest overall score. Further, since all three variants share the same base structure, Tab. 4 isolates the improvement from self-similarity guidance from TRELIS [3] to GuideFlow3D [30], and the additional effect of localized condition routing in SPACEFLOW.

Image-conditioned appearance. In the *image* regime, appearance is dictated by reference style images. Because baseline methods cannot process per-part image conditions, SPACEFLOW is scored standalone on *local routing accuracy*, *style fidelity*, *texture detail*, and *overall preference* (see Supp. for details). SPACEFLOW obtains a style fidelity of 6.53, a local routing accuracy of 7.07, texture detail of 6.33, and an overall score of 6.73, confirming that our framework effectively transfers visual style references to their designated regions.

4.3 Qualitative Results

As shown in Fig. 4, existing methods use global control, forcing the entire object to follow either geometric fidelity ($\tau = 10$) or generative freedom ($\tau = 3$), while global appearance cues may leak across parts. SPACEFLOW avoids this compromise by assigning control strengths per region and routing appearance cues to their corresponding parts.

Fig. 5 presents a diverse gallery of multi-part objects. Under our joint control, high-control primitives (orange) strictly guide the asset geometry in the specified area, while low-control proxies (gray) are completed with text-aligned geometric details (e.g., the rocket tail fins). To benchmark each modality without confounding factors, the figure isolates both axes: SpaceControl baselines demonstrate the structural failure modes of uniform τ under identical texturing, whereas TRELIS and GuideFlow3D show texture leakage across our fixed geometry. In contrast, SPACEFLOW consistently preserves user-specified geometry while cleanly localizing textures to their target parts (see Supp. for more qualitative results).

Method	PF $\uparrow$	CM $\uparrow$	TD $\uparrow$	Overall $\uparrow$
TRELIS	6.13	5.79	6.87	6.42
GuideFlow3D	6.40	5.96	6.76	6.46
SPACEFLOW (ours)	7.07	6.43	6.49	6.78

Table 4 Quantitative evaluation of appearance control. Mean scores (1–10) averaged over 83 cases: prompt faithfulness (PF), color/material accuracy (CM), texture detail (TD), and overall. All variants share the same structure; metrics are appearance-only.

5 Conclusion

We introduced SPACEFLOW, an inference-time framework for localized geometry and appearance control in 3D asset generation. Editable geometric inputs define the controlled regions, each associated with a geometric control level and a text or image appearance cue. During structure generation, the control levels modulate spatial guidance, allowing constrained regions to follow the input geometry more closely while leaving others unconstrained, relying on the generative prior. During appearance synthesis, the framework matches the input regions to generated parts and routes each cue only to its corresponding part, supporting localized conditioning while limiting cross-part leakage.

Evaluations combining vision-language judge assessments, user studies, and regional metrics demonstrate that SPACEFLOW successfully overcomes the limitations of uniform global guidance, enabling localized geometric control. Furthermore, localized appearance routing consistently outperforms existing unrouted baselines in prompt faithfulness. Overall, SPACEFLOW enables users to specify not only *what* to generate, but precisely *where* each spatial constraint and appearance cue should apply.

6 Acknowledgements

Neil de la Fuente is funded by the Rafael del Pino Foundation. Sayan Deb Sarkar is partly funded by a Google XR grant on 4D understanding.

References

- [1] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM TOG*, 2024. 2, 3
- [2] Tencent Hunyuan3D Team. Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation, 2025. 3
- [3] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D Latents for Scalable and Versatile 3D Generation. In *CVPR*, 2025. 2, 3, 4, 5, 10, 11
- [4] 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. In *ECCV*, 2024.
- [5] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D Using 2D Diffusion. In *ICLR*, 2023.
- [6] 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, 2024.
- [7] Heewoo Jun and Alex Nichol. Shap-E: Generating Conditional 3D Implicit Functions, 2023. 3
- [8] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast Single Image to 3D Objects with Consistent Multi-View Generation and 3D Diffusion. In *CVPR*, 2024. 2
- [9] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation. In *CVPR*, 2023. 2
- [10] Tianrun Chen, Chaotao Ding, Shangzhan Zhang, Chunan Yu, Ying Zang, Zejian Li, Sida Peng, and Lingyun Sun. Rapid 3D Model Generation with Intuitive 3D Input. In *CVPR*, 2024.

- [11] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-It-3D: High-fidelity 3D Creation from A Single Image with Diffusion Prior. In *ICCV*, 2023.
- [12] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *ICCV*, 2023.
- [13] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3D: Hierarchical 3D generation with bootstrapped diffusion prior. In *ICLR*, 2024. [2](#)
- [14] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. CraftsMan3D: High-fidelity Mesh Generation with 3D Native Diffusion and Interactive Geometry Refiner. In *CVPR*, 2025. [2](#)
- [15] Aditya Sanghi, Pradeep Kumar Jayaraman, Arianna Rampini, Joseph Lambourne, Hooman Shayani, Evan Atherton, and Saeid Asgari Taghanaki. Sketch-A-Shape: Zero-Shot Sketch-to-3D Shape Generation, 2023.
- [16] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In *ICCV*, 2023.
- [17] Gang Li, Heliang Zheng, Chaoyue Wang, Chang Li, Chang Wen Zheng, and Dacheng Tao. 3DDesigner: Towards Photorealistic 3D Object Generation and Editing with Text-guided Diffusion Models, 2022. [2](#)
- [18] Haoxuan Li, Ziya Erkoç, Lei Li, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshPad: Interactive Sketch-Conditioned Artist-Reminiscent Mesh Generation and Editing. In *ICCV*, 2025. [2](#)
- [19] Meng Yuan, Dawei Lin, Hongxia Xie, Tieru Wu, and Rui Ma. CAD-Refiner: A Unified Framework for CAD Generation and Iterative Editing. In *CVPR*, 2026. [2](#)
- [20] Minghao Chen, Roman Shapovalov, Iro Laina, Tom Monnier, Jianyuan Wang, David Novotny, and Andrea Vedaldi. PartGen: Part-level 3D Generation and Reconstruction with Multi-view Diffusion Models. In *CVPR*, 2025. [2](#), [3](#)
- [21] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM TOG*, 2019. [3](#)
- [22] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. *ACM TOG*, 2022. [3](#)
- [23] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. SALAD: Part-Level Latent Diffusion for 3D Shape Generation and Manipulation. In *ICCV*, 2023. [3](#)
- [24] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis. *ACM TOG*, 2020. [2](#), [3](#)
- [25] Chuan Fang, Yuan Dong, Kunming Luo, Xiaotao Hu, Rakesh Shrestha, and Ping Tan. Ctrl-Room: Controllable text-to-3d room meshes generation with layout constraints. In *Int. Conf. 3D Vision (3DV)*, 2025. [2](#)
- [26] Kiyohiro Nakayama, Mikaela Angelina Uy, Jiahui Huang, Shi-Min Hu, Ke Li, and Leonidas Guibas. DiffFacto: Controllable Part-Based 3D Point Cloud Generation with Cross Diffusion. In *ICCV*, 2023. [2](#), [3](#)
- [27] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3D: Progressively local editing for text-to-3D content creation with complex semantic prompts. In *ICLR*, 2024. [2](#)
- [28] Elisabetta Fedele, Francis Engelmann, Ian Huang, Or Litany, Marc Pollefeys, and Leonidas Guibas. Space-Control: Introducing test-time spatial control to 3D generative modeling. In *ICLR*, 2026. [2](#), [3](#), [4](#), [5](#), [7](#), [9](#), [10](#)
- [29] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jia-Wei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong

- Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3Dfy Anything in Images. In *CVPR*, 2026. [2](#), [3](#)
- [30] Sayan Deb Sarkar, Sinisa Stekovic, Vincent Lepetit, and Iro Armeni. GuideFlow3D: Optimization-Guided Rectified Flow For 3D Appearance Transfer. In *NeurIPS*, 2025. [2](#), [3](#), [5](#), [6](#), [10](#), [11](#)
- [31] Etai Sella, Gal Fiebelman, Noam Atia, and Hadar Averbuch-Elor. Spice·E: Structural Priors in 3D Diffusion using Cross-Entity Attention. In *ACM SIGGRAPH Conference Papers*, 2024. [2](#), [3](#)
- [32] Hyeonseop Song, Seokhun Choi, Hoseok Do, Chul Lee, and Taehyeong Kim. Blending-NeRF: Text-Driven Localized Editing in Neural Radiance Fields. In *ICCV*, 2023. [2](#)
- [33] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids. In *CVPR*, 2019. [2](#)
- [34] Elisabetta Fedele, Boyang Sun, Leonidas Guibas, Marc Pollefeys, and Francis Engelmann. SuperDec: 3D Scene Decomposition with Superquadric Primitives. In *ICCV*, 2025. [2](#)
- [35] Gabriel Tavernini, Elisabetta Fedele, Tiago Novello, Leonidas Guibas, Marc Pollefeys, and Francis Engelmann. SuperFlex: Deformable Superquadrics for Point Cloud Decomposition. In *ECCV*, 2026. [2](#)
- [36] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A System for Generating 3D Point Clouds from Complex Prompts, 2022. [3](#)
- [37] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent Point Diffusion Models for 3D Shape Generation. In *NeurIPS*, 2022. [3](#)
- [38] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *ACM TOG*, 2023.
- [39] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G. Schwing, and Liang-Yan Gui. SDFusion: Multimodal 3D Shape Completion, Reconstruction, and Generation. In *CVPR*, 2023. [3](#)
- [40] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. In *CVPR*, 2023. [3](#)
- [41] Wenqi Dong, Bangbang Yang, Lin Ma, Xiao Liu, Liyuan Cui, Hujun Bao, Yuewen Ma, and Zhaopeng Cui. Coin3D: Controllable and Interactive 3D Assets Generation with Proxy-Guided Conditioning. In *ACM SIGGRAPH Conference Papers*, 2024. [3](#)
- [42] Peng Li, Suizhi Ma, Jialiang Chen, Yuan Liu, Congyi Zhang, Wei Xue, Wenhan Luo, Alla Sheffer, Wenping Wang, and Yike Guo. CMD: Controllable multiview diffusion for 3d editing and progressive generation. In *ACM SIGGRAPH Conference Papers*, 2025. [3](#)
- [43] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2Mesh: Text-Driven Neural Stylization for Meshes. In *CVPR*, 2022. [3](#)
- [44] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. TEXTure: Text-Guided Texturing of 3D Shapes. In *ACM SIGGRAPH Conference Papers*, 2023. [3](#)
- [45] Sai Raj Kishore Perla, Yizhi Wang, Ali Mahdavi-Amiri, and Hao Zhang. EASI-Tex: Edge-Aware Mesh Texturing from Single Image. *ACM TOG*, 2024.
- [46] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3D: Paint anything 3D with lighting-less texture diffusion models. In *CVPR*, 2024.
- [47] Dana Cohen-Bar, Daniel Cohen-Or, Gal Chechik, and Yoni Kasten. TriTex: Learning Texture from a Single Mesh via Triplane Semantic Features. In *CVPR*, 2025. [3](#)
- [48] Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric Xing. StyleRF: Zero-shot 3D Style Transfer of Neural Radiance Fields. In *CVPR*, 2023. [3](#)

- [49] Kunhao Liu, Fangneng Zhan, Muyu Xu, Christian Theobalt, Ling Shao, and Shijian Lu. StyleGaussian: Instant 3D Style Transfer with Gaussian Splatting, 2024. [3](#)
- [50] Abhishek Saroha, Mariia Gladkova, Cecilia Curreli, Dominik Muhle, Tarun Yenamandra, and Daniel Cremers. Gaussian Splatting in Style. In *DAGM German Conference on Pattern Recognition*, 2024. [3](#)
- [51] Konstantinos Tertikas, Despoina Paschalidou, Boxiao Pan, Jeong Joon Park, Mikaela Angelina Uy, Ioannis Emiris, Yannis Avrithis, and Leonidas Guibas. Generating Part-Aware Editable 3D Shapes without 3D Supervision. In *CVPR*, 2023. [3](#)
- [52] Hyeongjin Nam, Donghwan Kim, Gyeongsik Moon, and Kyoung Mu Lee. PARTE: Part-guided texturing for 3D human reconstruction from a single image. In *ICCV*, 2025. [3](#)
- [53] Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Feng Yiqiang, Yadong Mu, and Katerina Fragkiadaki. PartCrafter: Structured 3D Mesh Generation via Compositional Latent Diffusion Transformers. In *NeurIPS*, 2025.
- [54] Habib Slim, Shariq Farooq Bhat, Mohamed Elhoseiny, Yifan Wang, and Mike Roberts. CompoSE: Compositional Synthesis and Editing of 3D Shapes via Part-Aware Control, 2026. [3](#)
- [55] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. PartField: Learning 3D Feature Fields for Part Segmentation and Beyond, 2025. [4](#), [5](#), [6](#)
- [56] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. [4](#)
- [57] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 2023.
- [58] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. *ACM TOG*, 2023. [4](#)
- [59] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. In *CVPR*, 2022. [5](#)
- [60] Ruichen Wang, Zekang Chen, Chen Chen, Jian Ma, Haonan Lu, and Xiaodong Lin. Compositional text-to-image synthesis with attention map control of diffusion models. In *AAAI*, 2024. [6](#)
- [61] Dong Huk Park, Grace Luo, Clayton Toste, Samaneh Azadi, Xihui Liu, Maka Karalashvili, Anna Rohrbach, and Trevor Darrell. Shape-guided diffusion with inside-outside attention. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. [6](#)
- [62] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, 2021. [6](#)
- [63] 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*, 2023. [7](#)

Appendix

This supplementary material contains:

- *Structure generation and guidance architecture (Sec. S1) and the appearance generation pipeline (Sec. S2).*
- *Superquadric primitives and their mathematical formulation (Sec. S3).*
- *Details of the 83-asset evaluation dataset (Sec. S4).*
- *Vision-language-model evaluation, including structure-control evaluation, pairwise and standalone appearance evaluation, judge configuration, and validation (Sec. S5).*
- *User-study details and protocol (Sec. S6).*
- *Additional analyses, including sensitivity to local control strength, spatial feature-distance visualization, and primitive-to-part assignment visualizations (Sec. S7).*
- *Limitations (Sec. S8).*
- *Additional qualitative results for image-conditioned and text-conditioned appearance control (Sec. S9).*
- *Implementation details (Sec. S10).*
- *Interactive examples in an accompanying HTML gallery, with rotatable 3D input geometries and generated results that can be inspected from arbitrary viewpoints.*

S1 Structure Generation and Guidance Architecture

Tab. S1 details the hyperparameters used across our spatial control evaluations. In addition, we present a schematic of our Conditioned Flow formulation, which is applied between τ_{low} and τ_{high} , in Fig. S1. As illustrated, this mechanism blends the flow-predicted latent with the high-control target latent inside the designated control regions while allowing the low-control regions to generate freely. The combined latent is then updated through a feedback loop that adds noise back to the target timestep. This refinement process is repeated K times before the model proceeds to the next generation step. At each of the K resampling iterations, a fresh noise sample is drawn independently.

To match the notation of the main paper, K denotes the number of conditioned-flow resampling iterations. We use K_{PF} to denote the number of PartField semantic clusters.

Latent Space Mask Generation. To spatially constrain the conditioning mechanism, we construct a discrete 3D binary mask from the input primitives. For low-control regions, a single bounding box is fitted around all low-control superquadrics with 7% diagonal padding. This continuous representation is first voxelized at a high resolution of 64^3 , then downsampled to the 16^3 latent space resolution via $4 \times 4 \times 4$ average pooling, and finally binarized with a threshold of > 0 (marking any latent cell with non-zero occupancy). The resulting binary mask directly dictates where the latent flow blending is applied.

S2 Appearance Generation Pipeline

Tab. S2 details the hyperparameters used across our appearance control evaluations, and Fig. S3 gives an overview of the part-wise appearance pipeline.

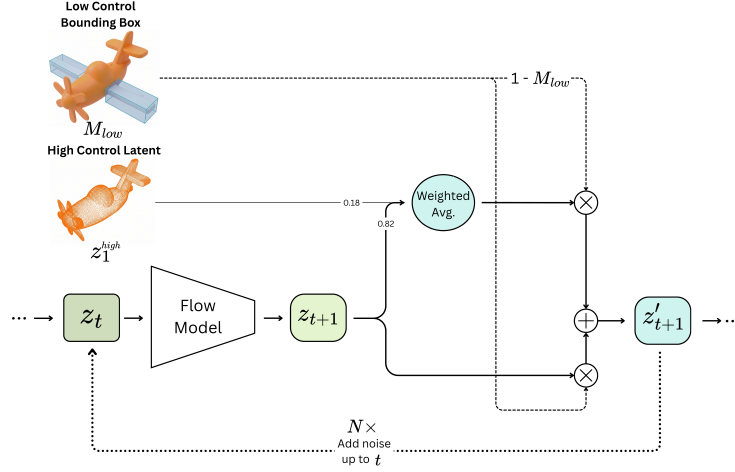


Figure S1 Illustration of a single Conditioned Flow iteration. The process receives the low-control region mask M_{low} (derived from the bounding boxes), the high-control target latent z_1^{high} , and the current latent state z_t . The flow model predicts the next state z_{t+1} . Inside the high-control region (masked by $1 - M_{low}$), a weighted average of the target latent z_1^{high} and the predicted latent z_{t+1} is computed. This is combined with the unconstrained prediction z_{t+1} in the low-control region (masked by M_{low}) to form z'_{t+1} . Finally, noise is added to z'_{t+1} up to timestep t (repeated K times) to update the latent z_t for the subsequent step.

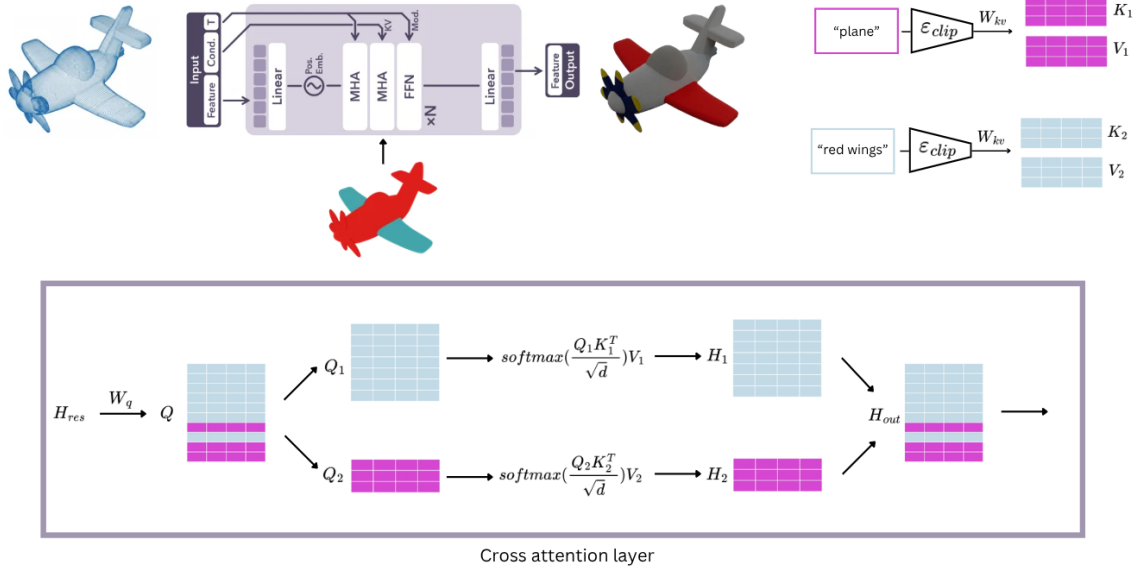


Figure S2 Routed cross-attention. Each text or image cue is encoded by a frozen CLIP (text) or DINOv2 (image) encoder into its own key/value pair K_i, V_i . Within every cross-attention layer of the appearance transformer, the latent queries Q are partitioned by the routing volume so that the voxels of a part attend only to their assigned cue—e.g. “red wings” routes to the wings while the global “plane” prompt covers the remaining voxels—and the per-region outputs H_i are recombined into H_{out} . This confines each appearance cue to its target part and reduces cross-part texture leakage.

The appearance stage operates on the sparse structure produced by the first stage. We decode the structure into a mesh and extract a PartField feature representation. Features sampled at active structure voxels are clustered into semantic regions, and each cluster is matched to its closest input primitive. This produces a routing volume that remains part-consistent even when low-control geometry

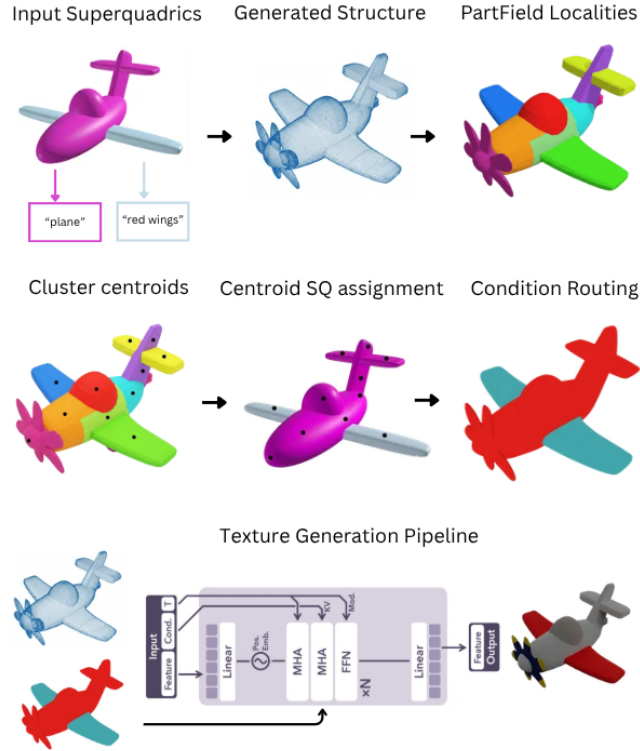


Figure S3 Part-wise appearance pipeline. From the generated sparse structure and the part-wise text or image cues, we decode the structure into a mesh and extract a PartField triplane feature field. Active structure voxels are clustered into semantic localities; each cluster centroid is assigned to its nearest input superquadric, and the resulting map is rasterized into a condition-routing volume that stays part-consistent even where low-control geometry drifts from the scaffold. The appearance rectified-flow transformer then synthesizes the SLAT appearance latents under this routing.

deviates from the original superquadric geometric input.

Each global or part-specific text or image condition is encoded into a separate embedding. Within the appearance-flow transformer, every latent voxel attends only to the embedding assigned by the routing volume; unassigned voxels attend to the global condition. This routed cross-attention, illustrated in Fig. S2, restricts each cue to its intended region. During sampling, we additionally optimize a part-aware self-similarity objective that encourages coherent appearance within each semantic part and separation across different parts.

Routing-volume construction. We decode the generated sparse structure into a mesh and extract a PartFieldtriplane feature field $\mathcal{P}$. At each active structure voxel $\mathbf{v}_j$ on the 64^3 feature-sampling grid, we sample a descriptor $\mathbf{f}_j$ and cluster the descriptors with K_{PF} -means, obtaining semantic labels ℓ_j . For each cluster C_m , we select the active voxel $\mathbf{c}_m$ nearest the cluster mean as its representative, where $m \in \{1, \dots, K_{\text{PF}}\}$. We voxelize the surface of each input primitive S_i into a set $\mathcal{B}_i$ and assign the complete cluster to the closest primitive:

$$a(m) = \arg \min_i \min_{\mathbf{b} \in \mathcal{B}_i} \|\mathbf{c}_m - \mathbf{b}\|_2. \quad (\text{S1})$$

Each active voxel inherits the primitive assignment $a(\ell_j)$ of its cluster, so all voxels in the same semantic cluster are routed consistently. We rasterize these inherited assignments on the structure voxel grid and aggregate them onto the coarser 32^3 appearance-latent grid to obtain the routing volume Φ . After downsampling, we count how many latent cells are assigned to each primitive. Thin or small primitives, such as wheels, handles, or chair legs, may receive too few cells because multiple fine voxels are merged into a single coarse latent cell. When this support falls below a minimum threshold, we recompute the affected mask directly on the latent grid by assigning each latent-cell center to its nearest voxelized primitive. We then dilate the recovered mask by one-cell neighborhood step, ensuring that the corresponding local appearance cue has enough latent cells to influence the generated texture while remaining spatially close to the intended primitive.

Attention routing. The global condition and every primitive-specific text or image cue are encoded separately, using the encoders listed in Tab. S2. In each cross-attention layer, Φ selects the condition embedding attended to by every latent voxel; voxels without a local assignment use the global condition. For the self-attention bias, we add $\log \beta$ to the attention logit of each query-key pair whose voxels share the same routed condition. This is equivalent to multiplying its pre-softmax attention weight by β and preserves soft interactions with the rest of the object.

Guidance optimization. At each conditioned appearance-flow step, we evaluate the supervised contrastive objective in Equation S2. To avoid materializing the full $|\mathcal{V}| \times |\mathcal{V}|$ similarity matrix, pairwise similarities are computed in chunks, with the chunk size selected according to the number of active voxels. We then take one AdamW step on the appearance latents. The lowest-loss latent state encountered along the guidance trajectory is retained, denormalized, and decoded into the final textured asset. Setting the guidance weight to zero disables the contrastive update and recovers part-routed sampling alone. Tab. S2 reports the grid resolutions, K_{PF} , β , chunk sizes, optimizer settings, and number of appearance-flow steps.

$$\mathcal{L} = -\frac{1}{|\mathcal{V}|} \sum_j \log \frac{\sum_{k: \ell_k = \ell_j, k \neq j} \exp(s_{jk}/\tau_c)}{\sum_{k \neq j} \exp(s_{jk}/\tau_c)}, \quad (\text{S2})$$

where $\mathcal{V}$ is the set of structure voxels, s_{jk} is the cosine similarity between voxel latents, and τ_c is the temperature. Voxels with no same-cluster neighbors are excluded from the mean through a validity mask.

Setting	Value
Structure backbone	TRELLIS text sparse-structure flow model
Structure latent tensor shape	$8 \times 16 \times 16 \times 16$
Local control strengths	$\tau_{\text{low}} = 3, \tau_{\text{high}} = 10$
High-control blending weight (α)	0.18 for local- τ
Number of structure flow steps	12
Flow solver	Euler flow-matching sampler with classifier-free guidance interval
CFG strength and interval	7.5, active for $t \in [0.5, 0.95]$
Time rescaling	3.0
Resampling iterations K	10

Table S1 Structure-generation settings used in our spatial-control experiments.

Setting	Value
Appearance backbone	TRELLIS SLAT flow model and decoders
Conditioning used in quantitative evaluation	Global and part-specific text prompts
Text-condition encoder	CLIP ViT-L/14
Image-condition encoder	DINOv2 ViT-L/14, used for image-conditioned standalone evaluation and qualitative examples
Part representation	PartField triplanes
PartField descriptor dimension	448
Feature-sampling grid	Generated sparse voxels on a $64 \times 64 \times 64$ grid
Local routing volume	$32 \times 32 \times 32$, only for local appearance prompts
Clustering method	K_{PF} -means on PartField voxel descriptors
Number of PartField clusters K_{PF}	30
Self-attention boost β	2.5
Guidance optimizer	AdamW
Guidance weight	1.0
Optimizer learning rate	5×10^{-4}
Similarity-computation chunk size	Automatic: 1024, 512, or 256 depending on voxel count
Number of appearance-flow steps	300
Contrastive temperature τ_c	1.0

Table S2 Appearance-generation and self-similarity guidance settings. Text-conditioned appearance is evaluated pairwise and with standalone VLM scores; image-conditioned appearance is evaluated standalone using the criteria in Fig. S18 and is additionally illustrated qualitatively.

S3 Superquadric Primitives

Superquadrics are compact geometric primitives capable of representing a continuous spectrum of shapes, such as ellipsoids, cylinders, rounded boxes, and cuboids. A standard superquadric is defined by its scale $\mathbf{s} = (s_x, s_y, s_z)^\top \in \mathbb{R}_+^3$ and shape exponents $\epsilon = (\epsilon_1, \epsilon_2)^\top \in \mathbb{R}_+^2$, which govern its roundness and squareness along the vertical and horizontal cross-sections.

For a 3D point $\mathbf{x} \in \mathbb{R}^3$, let $\mathbf{u} = (u_x, u_y, u_z)^\top = \mathbf{R}^\top(\mathbf{x} - \mathbf{t})$ represent its coordinates in the primitive’s local frame, transformed by rotation $\mathbf{R} \in \text{SO}(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$. The volume bounded by the superquadric is implicitly defined by the inside-outside function $F(\mathbf{x}) \leq 1$:

$$F(\mathbf{x}) = \left(\left| \frac{u_x}{s_x} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{u_y}{s_y} \right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left| \frac{u_z}{s_z} \right|^{\frac{2}{\epsilon_1}}, \quad (\text{S3})$$

where $F(\mathbf{x}) < 1$ corresponds to points inside the primitive, $F(\mathbf{x}) = 1$ on its surface, and $F(\mathbf{x}) > 1$ outside.

Deformable Superquadrics. We use the deformable superquadric representation of SuperFlex.¹ For a world-space point $\mathbf{x}$, let $\mathbf{q} = \mathbf{R}^\top(\mathbf{x} - \mathbf{t})$ denote its coordinates in the primitive’s local frame. Let $F_{\text{can}}(\mathbf{v}; \mathbf{s}, \epsilon)$ denote the right-hand side of Eq. (S3) with $\mathbf{u}$ replaced by $\mathbf{v}$. The deformed inside–outside function is

$$F_{\text{def}}(\mathbf{x}) = F_{\text{can}}\left(\mathcal{D}_{\boldsymbol{\tau}, \boldsymbol{\beta}}^{-1}(\mathbf{q}); \mathbf{s}, \epsilon\right), \quad (\text{S4})$$

where $\boldsymbol{\tau} = (\tau_x, \tau_y)$ controls tapering and $\boldsymbol{\beta}_a = (k_a, \alpha_a)$ controls bending about local axis $a \in \{x, y, z\}$. The inverse deformation follows the composition

$$\mathcal{D}_{\boldsymbol{\tau}, \boldsymbol{\beta}}^{-1} = \mathcal{T}_{\boldsymbol{\tau}}^{-1} \circ \mathcal{B}_{y, \boldsymbol{\beta}_y}^{-1} \circ \mathcal{B}_{x, \boldsymbol{\beta}_x}^{-1} \circ \mathcal{B}_{z, \boldsymbol{\beta}_z}^{-1}. \quad (\text{S5})$$

Thus, inverse bending is applied about z , then x , then y , followed by inverse tapering. Equivalently, the forward map tapers the canonical primitive first and then bends it about y , x , and z . We use $\tau \in (-1, 1)^2$, $k_a \geq 0$, and $\alpha_a \in [0, 2\pi)$, with angles measured in radians. The exact tapering and bending maps, axis conventions, parameterization, and admissible curvature range are specified by the linked implementation.

In our pipeline, these deformable primitives serve as compact, editable spatial control regions rather than final surface meshes.

S4 Dataset Details

We constructed a dataset of 83 3D assets covering diverse object categories. Each asset is defined by superquadric primitives with designated high-control and low-control regions, alongside local appearance prompts a_i .

The benchmark combines manually authored configurations designed to reflect realistic use cases with assets generated via a semi-automated pipeline in which an LLM (*ChatGPT 5.6 Sol*) proposed candidate primitive layouts, local control assignments, and appearance cues. All assets were validated and, if needed, refined by human annotators. The validation criteria were the following:

(i) *meaningful control contrast*, ensuring high-control regions specify well-defined structural elements while low-control regions remain intentionally coarse to test generative completion rather than trivial generation. (ii) *spatial validity*, confirming that low-control primitives are not geometrically contained by high-control ones. (iii) *appearance fidelity*, verifying that per-part appearance prompts are coherent, unambiguous, and accurately mapped to target primitives.

Object Categories. In Tab. S3, we report the distribution of object categories in the benchmark along with representative examples. The dataset spans 7 broad semantic categories covering everyday manufactured goods, furniture, vehicles, electronics, and natural/decorative objects, with primitive complexity ranging from 2 to 18 parts per asset.

Dataset Samples. Fig. S4 shows a representative selection of assets from our 83-asset benchmark across diverse categories.

¹<https://github.com/GabrielTavernini/superflex>

Category	Count	Share	Representative Examples
Vehicles & Transport	15	18.1%	Airplane, Space Shuttle, Satellite, Sailboat, F1 Car, Truck
Electronics, Audio & Optics	15	18.1%	Telescope, Gramophone, Piano, Radio, Microscope, Clocks
Toys, Nature & Decor	14	16.9%	Tin Robot, Wooden Dolls, Bonsai, Cactus, Elephant, Trophy
Furniture & Seating	13	15.7%	Office Chair, Bar Stool, Armchairs, Park Bench, Pool Table
Tools & Utility	10	12.0%	Fire Hydrant, Hammer, Sewing Machine, Toolbox, Iron, Bucket
Lighting & Lanterns	8	9.6%	Desk Lamps, Chandelier, Stone Lantern, Candle Lantern
Kitchenware & Tableware	8	9.6%	Teapot, Moka Pot, Mug, Stand Mixer, Frying Pan, Spoon
Total	83	100.0%	

Table S3 Object category distribution in the evaluation benchmark ($N = 83$). The benchmark spans 7 broad semantic categories designed to evaluate multi-part geometric control and local appearance conditioning.

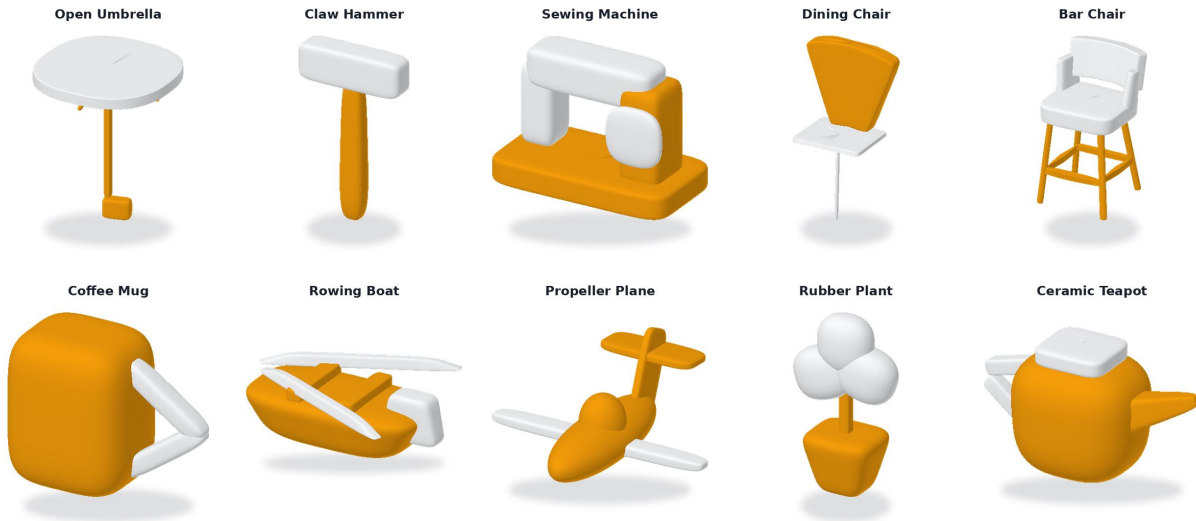


Figure S4 Dataset assets gallery. Ten representative assets from our 83-asset benchmark. Orange primitives designate high-control regions requiring strict geometric preservation, while white/gray primitives designate low-control regions intended for generative completion. Subtitles denote the target object concept.

S5 Vision-Language-Model Evaluation

We evaluate localized geometric structure control and appearance generation using vision-language-model judges (*GPT-5.6 Sol*). Each evaluation instance presents the judge with two fixed views of a generated asset ($\sim 210^\circ$ apart) alongside the corresponding input conditions (superquadric proxies or text/image references). All evaluations are blinded: judges receive no method identifiers, filenames, or cross-sample scores.

For reproducibility, we detail below the exact prompts, rendering procedures, inference parameters used across our structure and appearance evaluations.

S5.1 Structure Control Evaluation

To isolate geometric control from appearance quality, we render the generated assets as untextured shapes at the same scale and alignment as the input superquadrics. This prevents appearance quality from masking geometric defects.

The judge is presented with the input control shape, where region colors provide control-level labels:

- **Orange (High Control, $\tau = 10$):** The output geometry must strictly preserve the shape, proportions, and placement of these primitives.
- **White/Gray (Weak Control, $\tau = 3$):** The generator is expected to rely on the generative prior to generate a plausible geometry that matches the text prompt.

Pairwise Evaluation. We conduct head-to-head comparisons across all 83 dataset assets between SPACEFLOW (local guidance with $\tau = 10$ in high-control and $\tau = 3$ in low-control regions) and SpaceControl uniform guidance baselines ($\tau = 3$ and $\tau = 10$). The judge evaluates control and prompt adherence, realism, and overall preference. The exact prompt layout and instructions are detailed in Fig. S15.

S5.2 Appearance Evaluation: Pairwise Evaluation

For text-guided texture synthesis, we perform pairwise comparisons between SPACEFLOW (part-level condition routing) and appearance baselines (TRELIS and GuideFlow3D). All methods generate textures on identical base geometry generated with SPACEFLOW to strictly isolate appearance generation. The complete pairwise appearance instructions are shown in Fig. S16.

S5.3 Appearance Evaluation: Standalone Prompts

In addition to pairwise comparisons, we score text- and image-conditioned appearance generations on absolute 1–10 scales.

Text-Conditioned Evaluation. Measures prompt faithfulness, color/material accuracy, texture detail, and overall appearance quality (Fig. S17).

Image-Conditioned Evaluation. Measures reference-style fidelity, local routing accuracy, texture detail, and overall appearance quality (Fig. S18).

S5.4 VLM-Judge Configuration and Validation

Inference. The *GPT-5.6 Sol* model enforces a fixed default temperature of $T = 1.0$. To eliminate stochastic judge noise, evaluations are conducted over three independent passes. For head-to-head comparisons:

- A/B presentation order is independently randomized across passes
- Final pairwise preferences are decided by majority vote across three passes (three-way ties marked undecided), yielding 0.898 average inter-pass agreement.

Validation Probes. We verify judge reliability with two calibration checks:

1. **Mismatched Controls:** Scoring outputs against incorrect geometric inputs reduced adherence scores by 4.43 points, confirming sensitivity to geometric errors.
2. **Identical Pairs:** Comparing identical renders yielded 100% “Equal” ratings.

Tab. S4 summarizes the overall judge configuration, inference hyperparameters, and calibration metrics.

Setting	Value / Specification
Judge Model	<i>GPT-5.6 Sol</i>
Sampling Temperature	Default ($T = 1.0$)
Number of Passes	3 independent passes per condition
Pairwise Aggregation	2-of-3 majority vote (re-salted slot order)
Standalone Aggregation	Mean across 3 passes
Inter-Pass Agreement	Pairwise agreement: 0.898 (Spearman)
Rendered Input Views	2 fixed views ($\sim 210^\circ$ azimuth separation)
Evaluated Modalities	Structure (untextured GLB) & Appearance (textured)

Table S4 VLM-judge configuration and evaluation settings. Summary of model parameters, multi-pass aggregation rules, and rendering specifications.

S6 User Study Details

This section provides additional details on the user study interface and evaluation protocol. Fig. S5 shows the interface presented to participants during the evaluation trials.

When participants entered the user study website, they saw a visual explanation of the task SPACEFLOW solves. Afterwards, for each task, participants were shown: (i) the target text prompt (to assess realism of the generated asset); (ii) the input geometric control shape with high-control regions highlighted in orange (to evaluate shape fidelity); and (iii) two randomized, anonymous generated samples. Both the input shape and candidate samples were interactive 3D assets inspectable from any viewpoint.

Participants were then asked to choose between *Sample A*, *Sample B*, or *No difference* for the following question: “Which object looks better overall?”

S7 Additional Ablation Study

We first isolate the spatial- and appearance-control components to illustrate their individual effects. We then present additional examples using the full SPACEFLOW pipeline, in which spatial control and part-specific appearance conditions are applied simultaneously.

S7.1 Sensitivity to Local Control Strength

The main paper evaluates local geometric control using $\tau_{\text{low}} = 3$ and $\tau_{\text{high}} = 10$. While this setting demonstrates the contrast between high- and low-control regions, it does not characterize how the output changes between the two control endpoints.

To quantify the effect of varying local guidance strength, we evaluate performance across $\tau_{\text{low}} \in \{1, 3, 5, 7, 9\}$ on the 83-asset dataset, keeping $\tau_{\text{high}} = 10$, $\alpha = 0.18$, random seeds, and the resampling schedule fixed. Following the regional Chamfer evaluation in the main text, generated mesh faces are partitioned into high- and low-control regions based on their nearest labeled input primitive surface. We compute the mean-squared Chamfer distance ($\times 10^3$) between each region and its corresponding input primitives, reporting the asset-wise mean.

As shown in Fig. S6, a lower τ_{low} gives the generator substantial freedom to deviate from the input primitives, while increasing τ_{low} progressively constrains the shape to the input geometry until both curves converge at $\tau_{\text{low}} \approx \tau_{\text{high}}$. Notably, deviation in the high-control regions also decreases slightly as τ_{low} increases, despite τ_{high} being held constant. This reflects the global nature of the generative flow

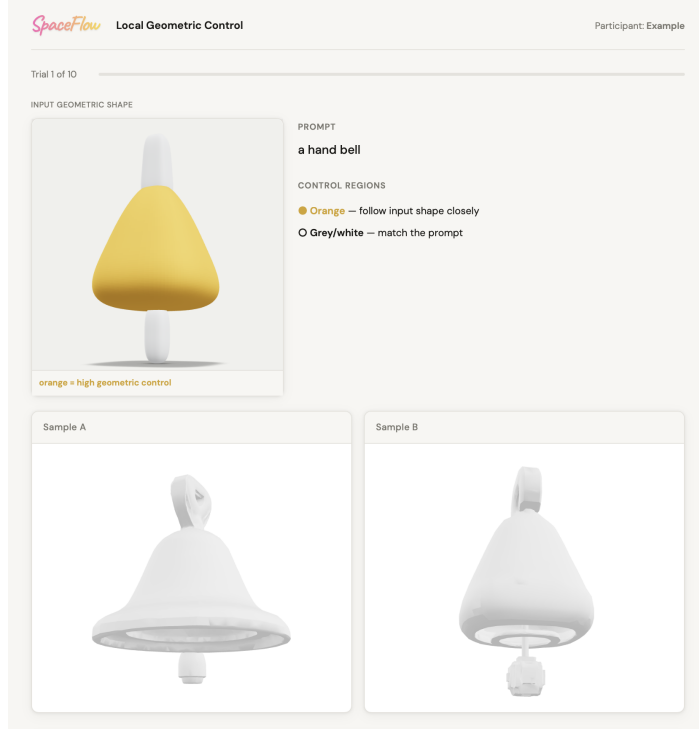


Figure S5 User study interface screenshot. Screenshot of the evaluation interface to study participants during a trial. The interface displays the text prompt, the input geometric control shape with color-coded region guidelines, and two randomized, anonymous outputs for comparison.

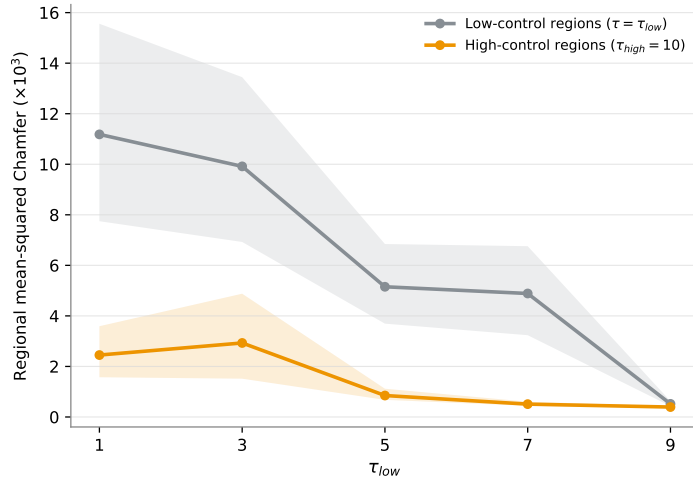


Figure S6 Geometric deviation across local control strengths. We fix the high-control strength at $\tau_{high} = 10$ and vary the low-control strength over $\tau_{low} \in \{1, 3, 5, 7, 9\}$. We report the regional symmetric mean-squared Chamfer distance to the input primitive surfaces ($\times 10^3$; lower is better), averaged asset-wise over the 83-asset dataset. Lines show the mean and shaded bands indicate 95% bootstrap confidence intervals. Low-control regions (gray) deviate most under weak control, and the gap to high-control regions (orange) closes as the two control strengths converge.

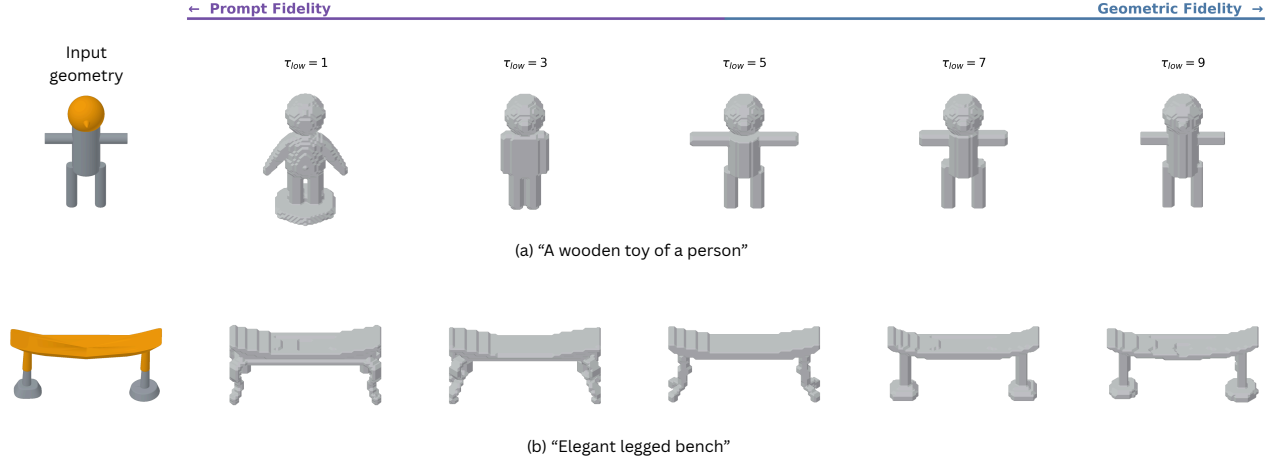


Figure S7 Qualitative local control strength continuum. We fix the protected primitives at $\tau_{\text{high}} = 10$ and vary the editable-region strength over $\tau_{\text{low}} \in \{1, 3, 5, 7, 9\}$. Orange input primitives denote high control and gray input primitives denote low control. Lower values permit greater prompt-driven geometric freedom, whereas increasing τ_{low} progressively strengthens adherence to the input geometry. Outputs are shown without texture to isolate geometric behavior.

backbone: large shape modifications in low-control regions exert a slight geometric pull across part boundaries onto neighboring high-control regions.

Additionally, the higher mean and variance observed in the high-control region at low τ_{low} are partly due to a boundary effect of our metric definition. Under low control, the model can freely generate novel parts that may emerge near the boundary between regions. Because mesh faces are partitioned according to their nearest input primitive, some of these newly generated structures may be mapped to adjacent high-control primitives despite having no counterpart in the geometric input, thereby increasing the regional Chamfer distance. As τ_{low} increases, unconstrained generation is reduced, lowering both the metric mean and variance.

Fig. S7 illustrates this behavior for two representative inputs. At low τ_{low} , the model has greater freedom to reinterpret the geometric input according to the prompt. At higher values, the output increasingly follows the input geometry, providing a continuous transition from prompt-driven generation to geometric fidelity.

S7.2 Spatial Feature-Distance Visualization

The feature-distance metric reported in the main paper measures regional deviation from an asset generated with uniform high-control. For each generated voxel, we find its nearest spatial match in the reference generated with $\tau = 10$ and compute the cosine distance between their normalized PartField features.

Fig. S8 visualizes this metric for a representative example. Larger distances occur primarily in the low-control region, while the high-control region remains closer to the reference, illustrating that SPACEFLOW localizes generative variation to regions where control is relaxed.

S7.3 Primitive-to-Part Assignment Visualizations

In Fig. S9, we visualize local appearance routing. In each pair, the left image is the geometric input and the right image is the derived routing volume from generated structure in 32^3 voxel space; colored

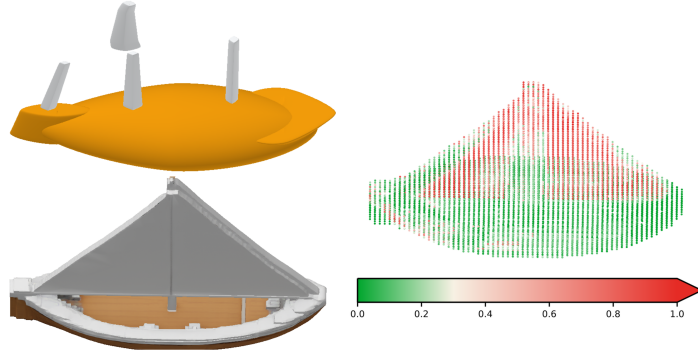


Figure S8 Visualization of spatial feature distance. Left: input superquadrics (top; orange denotes high control and gray denotes low control) and the asset generated by SPACEFLOW (bottom). Right: voxel-wise cosine distance between the generated asset’s PartField features and those of the nearest spatial match in the uniformly controlled ($\tau = 10$) reference. Feature deviations concentrate primarily in the low-control region.

regions attend to local appearance cues and gray regions attend to the global condition. We use $K_{PF} = 30$ PartField clusters.

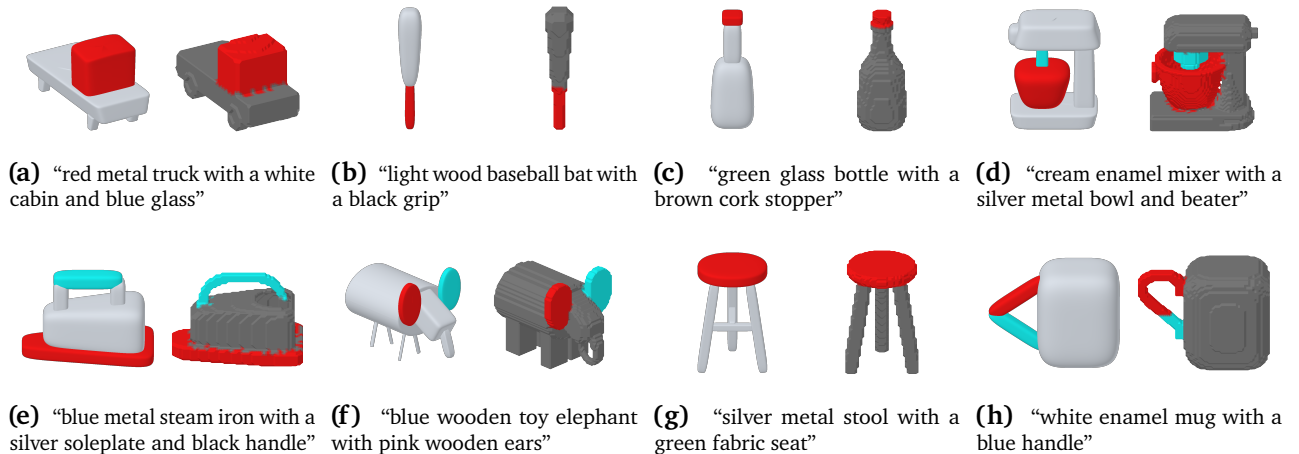


Figure S9 Primitive-to-part appearance routing. In each pair (a–h), we show the input structure (left) and the resulting 32^3 routing volume (right). The mapping is defined color-wise: regions sharing the same color between the input primitives and the generated structure represent a 1-to-1 mapping to that condition (with gray assigned to the global prompt). Subcaptions provide the flattened prompt containing all localized appearance cues.

S8 Limitations

Semantic-Geometric Conflicts. A primary failure mode in our geometric control module occurs when the text prompt contradicts the input geometry (e.g., requesting a “*standing person*” over a “*bench*” structure). In such cases, the text-conditioning pathway attempts to generate human features while the geometric guidance enforces the shape of the high-control regions. Because the underlying generator lacks a joint prior for geometric and textual combinations, the model struggles to reconcile both signals. This typically results in unnatural geometric distortions as the model attempts to synthesize the text-described asset around the constrained high-control regions (see Fig. S10).

Misrouted Appearance Cues. Our appearance module inherits the granularity of the underlying part

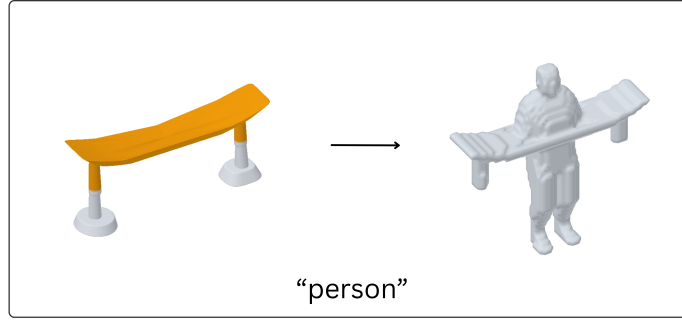


Figure S10 Semantic-geometric conflict. Synthesizing a “person” over an input “bench” structure (left). Lacking a joint prior for conflicting modalities, the model generates unnatural distortions by embedding the bench’s high-control geometry inside the generated “person” (right).

decomposition: the routing volume is derived from PartField clusters in 32^3 voxel space, and each cluster is assigned to a single appearance condition. When a cluster boundary does not coincide with the true material boundary of the generated asset, the corresponding appearance cue is applied to the wrong voxels, and the error is directly visible in the final texture. Fig. S11 shows two representative cases. In the frying pan, the routing volume assigns the pan body and the handle to different conditions, but a thin band of handle voxels leaks along the rim; the wooden handle appearance consequently bleeds onto the metal pan body. In the chair, the seat cluster does not fully cover the cushion surface, so only part of the seat receives the intended upholstery condition while the remaining voxels are textured by the competing condition, yielding a visibly discontinuous seat. Both failures are localized to the routing stage rather than the generator: the geometry is faithful to the input primitives, and only the region-to-condition assignment is incorrect. Finer part decompositions or soft, distance-weighted routing boundaries are a natural direction for mitigating this effect.

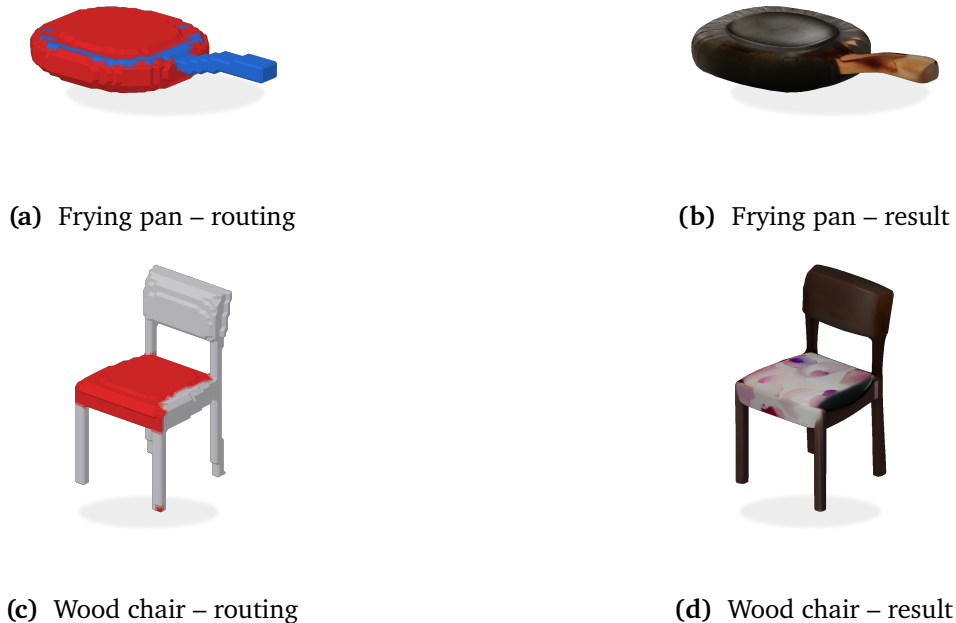


Figure S11 Failure cases of local appearance routing. For each asset we show the derived routing volume (left; colors denote distinct appearance conditions) and the generated result (right). **(a–b)** The handle cluster wraps around the rim, so the wooden handle appearance bleeds onto the metal pan body. **(c–d)** The seat cluster covers only part of the cushion, leaving the rest textured by the competing condition.

S9 Additional Qualitative Results

Fig. S12 shows additional qualitative results for image-conditioned appearance control. In each example, the geometric input is paired with two reference images that specify the material or style for designated parts (indicated by arrows). Across diverse object categories, SPACEFLOW faithfully transfers these appearance cues to the corresponding regions while preserving the underlying geometry.



Figure S12 Additional qualitative results for image-conditioned appearance control. Each panel shows an input geometric structure (left), the object-level text prompt (bottom), and two reference images specifying the desired material, texture or style for the object or for particular parts. Dotted arrows indicate the part-level associations between the reference images and the corresponding target regions. We recommend zooming in to inspect the fine-grained structural and appearance details; for example, in the metal-toolbox example, the generated asset includes the small front latches visible in the reference image and renders them with a consistent metallic appearance. These examples illustrate how SPACEFLOW transfers reference-driven appearance cues to the corresponding regions while retaining the overall structure of the input object.

Additionally, Figs. S13 and S14 show a gallery of text-conditioned qualitative examples of SPACEFLOW spanning diverse object categories, varying numbers of geometric primitives, and different localized color and material requests. For each object, we pair the input geometric structure (with high-control regions in orange and low-control regions in gray) with the generated asset rendered from the same viewpoint. For interactive examples, we refer to the attached HTML gallery, which includes a broader set of examples of joint geometric and appearance control. In addition to that, we share an accompanying video that demonstrates the complete workflow and control capabilities in action using our own UI.

S10 Implementation Details

Our implementation is developed in PyTorch and builds upon the SpaceControl² and GuideFlow3D³ codebases. We use the publicly released TRELLIS-XL⁴ for all pretrained encoders, flow transformers, and decoders, which remain frozen because SPACEFLOW is training-free. CLIP⁵ and DINOv2⁶ encode the text and reference-image conditions, respectively. All experiments were conducted on a single NVIDIA GeForce RTX 5060 Ti GPU. The structure and appearance models were evaluated using the inference settings reported in Tables S1 and S2.

²<https://github.com/spacecontrol3d/spacecontrol>

³<https://github.com/GradientSpaces/GuideFlow3D>

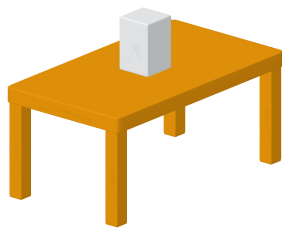
⁴<https://github.com/microsoft/TRELLIS>

⁵<https://github.com/openai/CLIP>

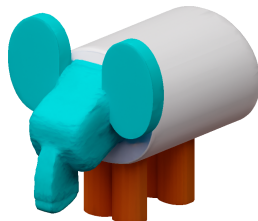
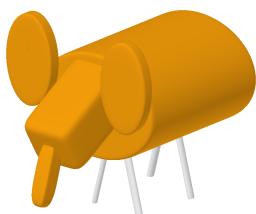
⁶<https://github.com/facebookresearch/dinov2>



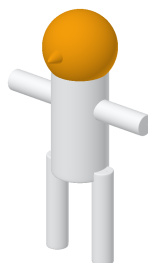
(a) “wood chair with straight legs and a red cushioned seat and orange backrest”



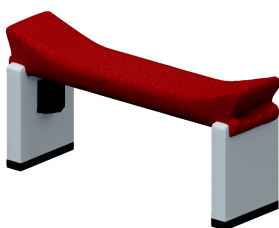
(b) “A table with a plant”



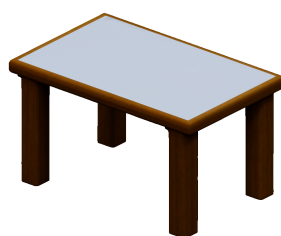
(c) “white elephant toy with orange legs and cyan head”



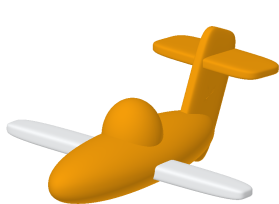
(d) “A wooden toy of a person with a yellow plastic head”



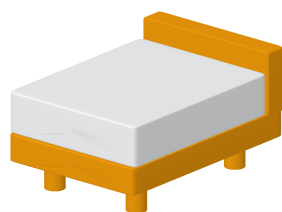
(e) “A white bench with a red velvet seat and a black bottom on the legs”



(f) “A glass table with wooden legs”

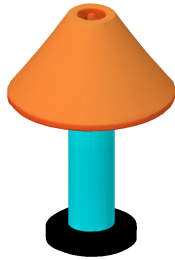


(g) “A plane with the main wings red”

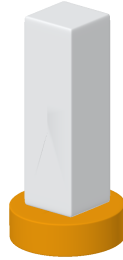


(h) “A bed with a green velvet duvet, pillows and an oak bed frame”

Figure S13 Extended qualitative gallery (Part 1). Text-conditioned SPACEFlow results across diverse object categories. Each cell pairs the input superquadric scaffold (left; orange = high-control, gray = low-control regions) with the generated asset (right) rendered from the same viewpoint.



(a) “orange lamp with a cyan pole and a rectangular black base”



(b) “A gold trophy with a black marble base”



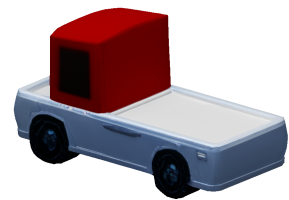
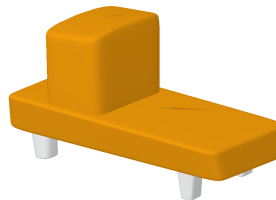
(c) “A white marble pool table with red billiard cloth”



(d) “A white house with a triangular brick roof”



(e) “white chair with a blue backrest”



(f) “A silver pick up truck with a red cab and red wheels”



(g) “blue classic chair with a black slat back”



(h) “A brown leather stool with white horizontal slats connecting the legs”

Figure S14 Extended qualitative gallery (Part 2). Additional text-conditioned SPACEFLOW results. Each cell pairs the input superquadric scaffold (left; orange = high-control, gray = low-control regions) with the generated asset (right) rendered from the same viewpoint.

[SYSTEM INSTRUCTION]

You are an expert judge of locally controlled 3D shape generation.

First you are shown an **INPUT CONTROL SHAPE** built from coarse primitives. Its colours are region LABELS, not a texture target:

- **ORANGE** marks high-control regions. The generated object must follow these parts closely in **SHAPE**.
- **WHITE / GREY** marks weak-control regions. The generator is free to reshape these into whatever the text prompt needs, and doing so is the **DESIRED** behaviour, not an error.

Then you are shown the generated object's **STRUCTURE ONLY**: the raw generated geometry, rendered untextured as a plain uniform grey shape. It carries no colour or material of its own, so the orange/white labels of the control shape have no counterpart in it and you must never look for one. Work out which part of the output corresponds to which part of the control shape from **POSITION** and **GEOMETRY** alone.

The two region types are judged by opposite standards, and this is the single most important thing to get right:

- An orange region that changed shape is a **FAILURE**.
- A white/grey region that changed shape is a **SUCCESS**. The whole point is that the generator turns those coarse blobs into real, detailed parts. Never treat 'the output no longer looks like the input primitives here' as an error in a white/grey region.

Judge **GEOMETRY ONLY**: proportions, part placement, silhouette, structural plausibility. Completely **IGNORE** colour, material, texture sharpness, lighting, shadows and background.

You are shown two rendered views of each object. They are the **SAME** object from two camera angles roughly 210 degrees apart, **NOT two different objects**. A feature may therefore be visible in only one of the two views; judge the object as a whole and never penalise a feature merely for being out of frame in the other view.

Sample A and Sample B were generated from the **SAME** prompt and the **SAME** input control shape. Compare them on geometry alone.

For each question choose **exactly one** of: **A, B, Equal**. Use Equal only when there is genuinely no meaningful difference - do not use it to avoid deciding.

- q1_control_prompt: Which object best matches the prompt in the white/grey weak-control regions while also closely following the orange high-control input shape?
- q2_realism: Which object looks more realistic as a real-world object?
- q3_overall: Which object is better overall as a controlled 3D generation?

Return **ONLY** a single JSON object, no prose, no markdown fences, with exactly these string keys: {"q1_control_prompt": "A|B|Equal", "q2_realism": "A|B|Equal", "q3_overall": "A|B|Equal"}

[USER DATA]

Target description: [Shape Prompt]

INPUT CONTROL SHAPE (two views):

[Image: Superquadric Proxy View 1] [Image: Superquadric Proxy View 2]

Sample A (two views), untextured grey geometry:

[Image: Sample A View 1] [Image: Sample A View 2]

Sample B (two views), untextured grey geometry:

[Image: Sample B View 1] [Image: Sample B View 2]

Figure S15 Prompt template for pairwise structure evaluation. The judge evaluates two untextured geometries against the input control shape, with randomized A/B sample ordering across passes.

[SYSTEM INSTRUCTION]

You are an expert judge of 3D appearance and texture generation.

You are shown a generated 3D object that was textured to satisfy a **TARGET DESCRIPTION**. The description may name different materials for different parts; getting each material onto the correct part is the central thing being tested.

Judge **APPEARANCE ONLY**: colour, material, and how cleanly the texture sits on the surface. Completely **IGNORE** the underlying geometry, the object's proportions, lighting, shadows, background and camera angle. If the shape itself is odd, that is not an appearance failure.

You are shown two rendered views of each object. They are the **SAME object** from two camera angles roughly 210 degrees apart, **NOT two different objects**. A feature may therefore be visible in only one of the two views; judge the object as a whole and never penalise a feature merely for being out of frame in the other view.

Sample A and Sample B were generated from the **SAME prompt**. They also share the same underlying geometry, so do not look for shape differences - look at colour, material and texture.

For each question choose **exactly one** of: **A, B, Equal**. Use Equal only when there is genuinely no meaningful difference - do not use it to avoid deciding.

- q1_prompt_faithfulness: Which object's appearance better matches the target description, including getting the **right material onto the right part**?
- q2_texture_quality: Which object has the cleaner, sharper, less artifacted surface appearance?
- q3_overall: Considering both faithfulness to the requested appearance and surface texture quality, which object is preferred overall?

Return **ONLY** a single JSON object, no prose, no markdown fences, with exactly these string keys: {"q1_prompt_faithfulness": "A|B|Equal", "q2_texture_quality": "A|B|Equal", "q3_overall": "A|B|Equal"}

[USER DATA]

Target description: [Global object prompt and part-specific requests]

Sample A (two views):

[Image: Sample A Textured View 1] [Image: Sample A Textured View 2]

Sample B (two views):

[Image: Sample B Textured View 1] [Image: Sample B Textured View 2]

Figure S16 Prompt template for pairwise appearance evaluation. Blinded comparison of textured outputs on identical geometry to measure prompt faithfulness and surface quality.

[SYSTEM INSTRUCTION]

You are an expert judge of 3D texture / appearance transfer. You are shown two rendered views of a single textured 3D object **SAME OBJECT** from different camera angles, not different objects. The two views look at **OPPOSITE sides**, so a textured region may appear in only ONE view -- judge the object as a whole and do NOT penalise a feature merely for being out of frame in the other view. Base your scores on colour, material and pattern transfer onto the correct parts; **IGNORE** lighting, shadows, background and which side faces the camera. You may also first be shown a 'structure reference': clean, untextured renders of the same geometry from the same angles, provided only so you can recognise the object's parts -- **do NOT score the geometry itself**. Imagine the full 3D object and judge how the texture behaves across its surfaces.

The texture was generated to satisfy a **TEXT** description. Judge how well the rendered object matches that description.

Score **EACH** of the following criteria as an integer from 1 to 10, using the **FULL** range and these anchors:

- 1-2 = the target colour/material/pattern is absent or plainly wrong;
- 3-4 = barely present or mostly wrong;
- 5-6 = roughly right but with clear, obvious errors;
- 7-8 = clearly matches the target, only minor issues;
- 9-10 = near-perfect match.

Judge the object on its own merits, not relative to anything. Avoid defaulting to the extremes (all 1s or all 9s) -- reserve those for cases that truly warrant them.

1. **prompt_faithfulness**: How well the texture matches what the prompt asked for, **INCLUDING per-part / local requests** (e.g. 'orange backrest', 'red cushion'). Penalise missing or misplaced local colours/materials.
2. **color_material_accuracy**: Correctness of colours and material feel (matte / glossy / wood / metal / fabric) relative to the prompt.
3. **texture_detail_quality**: Sharpness and cleanliness of local texture detail; absence of noise, blur, smearing or obvious artifacts.
4. **overall**: Overall texture quality considering all of the above.

Return **ONLY** a single JSON object, no prose, no markdown fences, with exactly these integer keys: {"prompt_faithfulness": <1-10>, "color_material_accuracy": <1-10>, "texture_detail_quality": <1-10>, "overall": <1-10>}

[USER DATA]

Target description: [Global object prompt and all part-specific color/material requests]
Structure reference - the untextured geometry the texture was applied to (same shape and camera angles as the output below; for understanding the object's parts, NOT to be scored):
[Image: Untextured Render View 1] [Image: Untextured Render View 2]
Rendered output view(s) of the result:
[Image: Textured Render View 1] [Image: Textured Render View 2]

Figure S17 Prompt template for text-conditioned standalone VLM evaluation. Complete system instructions and input layout used to score part-wise text appearance generation.

[SYSTEM INSTRUCTION]

You are an expert judge of 3D texture / appearance transfer. You are shown one or two rendered views of a single textured 3D object (the **SAME object** from different camera angles, not different objects). The two views look at **OPPOSITE sides**, so a textured region may appear in only ONE view -- judge the object as a whole and do NOT penalise a feature merely for being out of frame in the other view. Base your scores on colour, material and pattern transfer onto the correct parts; **IGNORE** lighting, shadows, background and which side faces the camera. You may also first be shown a 'structure reference': clean, untextured renders of the same geometry from the same angles, provided only so you can recognise the object's parts -- **do NOT score the geometry itself**. Imagine the full 3D object and judge how the texture behaves across its surfaces.

The texture was generated to imitate **REFERENCE STYLE IMAGES**, and the **TARGET DESCRIPTION** states which reference applies to which part: the **GLOBAL** reference image gives the overall appearance and each **LOCAL** reference image gives the style of a specific part. Judge how faithfully the rendered object reproduces that target, using the description and the reference images together (this is also what local_routing_accuracy checks). The reference images show a material/style, **NOT the target shape** -- ignore the shape in the reference.

Score **EACH** of the following criteria as an integer from 1 to 10, using the **FULL** range and these anchors:

- 1-2 = the target colour/material/pattern is absent or plainly wrong;
- 3-4 = barely present or mostly wrong;
- 5-6 = roughly right but with clear, obvious errors;
- 7-8 = clearly matches the target, only minor issues;
- 9-10 = near-perfect match.

Judge the object on its own merits, not relative to anything. Avoid defaulting to the extremes (all 1s or all 9s) -- reserve those for cases that truly warrant them.

1. style_fidelity: How faithfully the output reproduces the reference style image's colour, material and pattern/motif.
2. local_routing_accuracy: Whether the correct reference texture landed on the correct part of the object (when multiple reference images / per-part references are given).
3. texture_detail_quality: Sharpness and cleanliness of local texture detail; absence of noise, blur, smearing or obvious artifacts.
4. overall: Overall texture quality considering all of the above.

Return **ONLY** a single JSON object, no prose, no markdown fences, with exactly these integer keys: {"style_fidelity": <1-10>, "local_routing_accuracy": <1-10>, "texture_detail_quality": <1-10>, "overall": <1-10>}

[USER DATA]

Target description: [Short text description mapping target parts to the reference styles]

Global reference image (overall appearance): [Image: Global Reference Style]

Local reference image (specific part): [Image: Local Reference Style 1] ...

Structure reference - the untextured geometry the texture was applied to (same shape and camera angles as the output below; for understanding the object's parts, NOT to be scored):

[Image: Untextured Render View 1] [Image: Untextured Render View 2]

Rendered output view(s) of the result:

[Image: Textured Render View 1] [Image: Textured Render View 2]

Figure S18 Prompt template for image-conditioned standalone VLM evaluation. Layout provided to the VLM judge to score local reference image style transfer and spatial routing accuracy.