

EditHero: A Benchmark for Long-Horizon Part-Level 3D Editing and Vibe Modeling

Ruihan Yu^{1,2,*}, Yu-Ju Tsai^{3,4,*}, Muyao Niu^{1,2}, Runyi Li^{1,2}, Lian Fu^{1,2}, Hanqing Liu², Zheng-Hui Huang¹, Yonghao Yu¹, Sho Kuno^{1,2}, Ming-Hsuan Yang^{4,†}, Kaipeng Zhang^{1,†}, Zhixiang Wang^{1,†}

¹Alaya Lab ²The University of Tokyo ³Institute of Science Tokyo ⁴University of California, Merced

*Equal contribution, †Corresponding authors

3D editing methods are usually tested on a single edit, yet an asset is built through a long sequence of revisions, each of which must implement the requested change while leaving everything else unchanged. We introduce EditHero, to our knowledge the first benchmark for long-horizon, part-level 3D editing, with natural-language instructions and target images for both geometry and texture. A deterministic assembly engine produces the exact target after every edit, and every sequence is reviewed by hand. We use EditHero to compare 2 opposite approaches to 3D editing. Non-agentic methods operate top down, regenerating the object from a learned 3D representation and inferring what to keep. In contrast, LLM/VLM agents operate bottom up, editing through code that inspects the mesh and rewrites only the parts required by instructions. The non-agentic methods often miss the requested change and disturb regions that should stay fixed. Most LLMs follow instructions more closely, and all of them preserve the unedited parts better, but each of their edits takes minutes. We will release the engine and the edit sequences to support research on reliable iterative 3D editing.

Project page: <https://alaya-lab.github.io/EditHero/>

Code: <https://github.com/AlayaLab/EditHero>

Correspondence: Ming-Hsuan Yang, Kaipeng Zhang, Zhixiang Wang

Date: September 30, 2026



Figure 1 Editing is a sequence. The figurine goes through 5 edits, and every turn has to keep what the earlier turns built. It gains sunglasses (t_1), loses its cap (t_2), turns brown-haired (t_3), changes its outfit (t_4) and finally puts on a witch hat (t_5). The 2 plinths show 2 ways of getting there. A traditional, non-agentic method regenerates the whole figurine at every turn, so parts nobody asked to change can drift. An agentic method rewrites only the named part through code, which keeps the rest fixed, although the new parts come out simpler. Can either meet the demands of asset production, and how can we reach vibe modeling?

1 Introduction

A 3D artist rarely makes one edit and stops. Assets in art and game production evolve through successive editing and revisions: each turn of modification changes part of the current asset while preserving earlier work. The same expectation underlies the recent concept of “vibe modeling”, in which users shape assets through an iterative natural-language conversation [183, 213, 36]. Yet text- and image-guided 3D editing methods [98, 215, 198, 192] are typically evaluated on a single edit from a clean source. It remains unclear whether they can edit their own outputs repeatedly without losing earlier changes or degrading the asset.

Evaluating this ability requires a reference after every turn. Existing paired datasets [198, 215, 192] only provide isolated edits. Their targets come from either learned editing models or changes to poses and semantic parts. Neither design can tell us whether a method preserves earlier changes across a sequence of instructions.

We introduce EditHero, a benchmark of *part-level edit chains with exact ground truth* (Figure 2). The benchmark is built with the help of a data engine, which assembles library parts onto segmented host objects with 4 operations (**add**, **remove**, **replace** and **retexture**) and records every operation in a JSON file. Each entry is a single atomic operation, so replaying the file with the part library rebuilds the exact state after any turn. After human review, EditHero contains 457 chains and 2755 edits across 252 objects, with up to 30 turns per chain (Section 3).

We test 2 families of methods on this long-horizon 3D editing task, and they approach an edit in opposite ways. Non-agentic methods work top down. They regenerate the edited object from a learned 3D representation, such as the latents of a diffusion model, so every turn changes every corner of the asset, more or less. In contrast, LLM/VLM agents work bottom up. An LLM or VLM inspects the current mesh and writes code that changes only the named part. This keeps everything else untouched by construction and forces every new part to be built from primitives. We compare 4 non-agentic methods [192, 215, 198, 98] with 6 LLM/VLM agents under a *self-rollout* protocol (Section 4), in which every edit starts from the method’s own previous output. We also design 2 metrics to measure the differences between these methods, including instruction following (IF) inside the edit region and content consistency (CC) outside it.

The results indicate that agentic methods have superior editing ability. The non-agentic methods often miss the requested change as early as the first turn, and errors accumulate along a chain. The colors of their results also shift away from the target. The agents behave differently. Most agents follow instructions more closely than non-agentic methods, and all agents keep the untouched parts better. However, the agents’ speed is not so satisfying, as a single edit takes minutes rather than seconds (Sections 5 and 5.4). Our contributions are:

- **Generation engine.** We construct a data synthesis engine that produces part-level edit chains with an exact target after every turn.
- **Benchmark.** We propose *EditHero*, a human-reviewed benchmark for long-horizon part-level 3D editing.
- **Evaluation framework.** We evaluate 4 non-agentic methods and 6 LLM/VLM agents under self-rollout, with region-specific measures of instruction following and preservation.

2 Related Work

3D editing. 3D editing has followed the available generative models. Early methods optimized a NeRF or a Gaussian splat under a 2D diffusion prior [61, 28, 22], and later ones edit rendered views or an image and then reconstruct the object [7, 66]. Part-level and sketch-based methods let a user refine an object step by step [41, 157, 94, 97]. Native 3D generators such as TRELIS [199] made it possible to edit a 3D latent directly. Methods trained on paired edits feed the source latent to the generator through an added branch, much as ControlNet [233] conditions an image model. PartFlow [192] adds a source-control branch, and 3DEditFormer [198] uses dual-guidance attention. Training-free methods reuse a pretrained generator instead. VoxHammer [98] inverts the source and reuses its cached latents and attention features in the

preserved region, and Nano3D [215] follows the difference between the source-conditioned and target-conditioned flows, using the FlowEdit formulation [89]. We evaluate PartFlow, 3DEditFormer, VoxHammer and Nano3D as representatives. Each was proposed and evaluated for one edit at a time, and each uses generative latents to synthesize edited content. EditHero instead asks them to keep editing their own output, as an artist would, and measures what regeneration does to the parts not required by any instruction. Appendix A describes these methods in more detail and lists the others.

3D editing benchmarks. Most text- and image-guided 3D editing benchmarks score one instruction at a time. Some pair a source asset with a target asset, obtained through image-guided generation [198, 215, 126] or pose and part changes [198, 192, 145]. Others have no target asset and score region preservation and alignment to an edited image instead [98, 250, 53, 135], and scene-level benchmarks follow the same pattern [143, 115, 252]. The closest multi-step settings are agent benchmarks such as BlenderGym [58] and 3DCodeBench [52], which let an agent iterate toward one fixed goal. EZBlender [177] instead bundles several edits into one prompt. To our knowledge, EditHero is the first benchmark to evaluate different natural-language part edits in sequence on one asset, with an exact 3D reference after every turn (Appendix Table 4).

LLM/VLM agents for 3D generation and editing. LLM agents can modify 3D assets through code. They write programs or call tools to build scenes [166, 207] and edit Blender scenes from renders [69, 121]. Related systems generate objects and CAD programs [35, 83, 155]. Recent systems treat the program itself as the asset and revise it under visual feedback [109, 137, 183], aiming to keep edits local and preserve the rest of the state. Our data engine also records edits as operations that can be replayed to reconstruct each state. We evaluate such agents on the mesh, turn by turn and with the same metrics as the non-agentic methods (Section 5.4).

3 Benchmark Construction

Task. We hope to do multi-turn editing starting from a textured, part-segmented *host* object S_0 with text or image conditions. We define 4 operations, **add**, **remove**, **replace** and **retexture** (we even think that add and remove alone would be enough). Each turn specifies one operation on a part through a text instruction and a fixed-view render of the target state. An **add** attaches a part to a free contact face, and a **remove** deletes a named part, which may consist of several pieces. A **replace** swaps a named part for another at the same location, while a **retexture** changes how a part looks without touching its geometry. All other parts should remain unchanged. After each turn, the method returns a 3D object that becomes its input at the next turn. Text instructions may refer to earlier edits. For example, a part added at one turn can be removed a few turns later, so a method may want to keep track of what it has already built.

Exact ground truth by construction. Evaluating a chain requires a reliable target at every turn. Some existing paired datasets obtain their targets from image-guided generative pipelines [198, 215], and such a target can itself contain changes nobody asked for. We instead assemble each target from parts, much as CLEVR and Kubric generate their scenes programmatically [79, 57]. For an addition or replacement, the engine retrieves caption-matched parts from a library based on PartVerse-XL [40]. It then screens the candidates with a vision-language model (VLM) and tests their placement for contact, scale, connectivity, and interpenetration. Removal deletes the part occupying a named slot. For a retexture, the engine renders the slot and restyles the render with an image editing model (Qwen-Image-Edit [193]). The texturing module of TRELIS.2 [200] then textures the unchanged slot geometry from the restyled image. Each chain is stored as a JSON file that logs every accepted operation with its library part, pose and texture. Together with the part library, the log rebuilds any state S_k exactly and records what changed at each turn.

Human review. Reviewers inspected renders of every turn for incorrect parts, orientation, placement, scale, ambiguous instructions, and broken dependencies. They corrected flagged cases by editing the logged operations and rebuilding the affected turns. They also rewrote instructions drafted from part captions to describe the intended part, position, and orientation clearly. Appendix B describes the engine, the chain format and the review procedure.

Statistics. EditHero contains 457 chains and 2755 edits across 252 hosts (Figure 3). Chains have 3 to 30 turns, with a median of 6, and the median instruction is 17 words long. Six families group chains by their

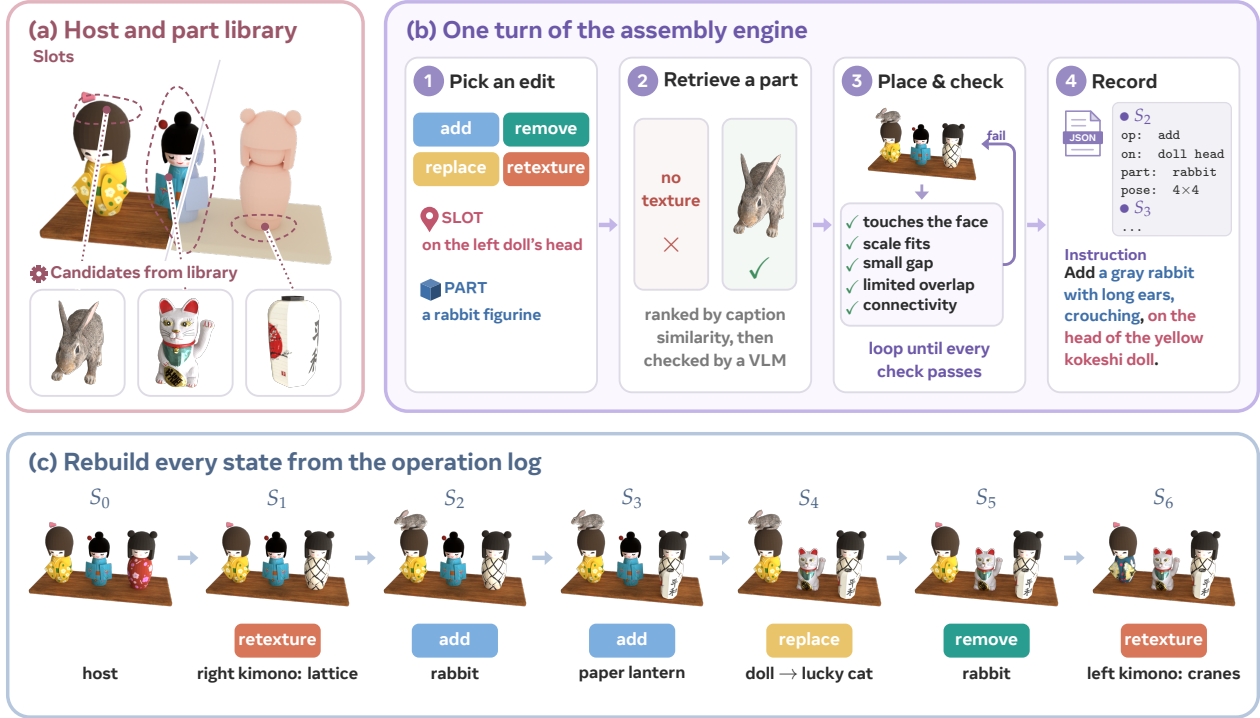


Figure 2 Constructing EditHero. (a) A host object is divided into named slots, and candidate parts are drawn from a captioned library. (b) The addition shown illustrates edit selection, part retrieval, visual screening and placement checks. Each accepted edit is recorded in the chain log, and reviewers refine the drafted instructions. (c) With the part library, the log rebuilds the exact target of any turn. A method under test never sees the log. It receives the instruction and the target render of each turn. The example combines recorded turns from 4 chains on the same host; all images are renders. Colors indicate the 4 edit operations.

allowed operations. Removal (R), replacement (P), and material (M) chains contain only their respective operations. Addition (A) chains primarily add parts. Geometry (G) chains combine additions, removals, and replacements, while geometry-and-material (X) chains also include retexturing. Appendix Table 4 compares EditHero with existing benchmarks.

4 Evaluation Protocol

Setting. We evaluate the editing methods under *self-rollout*, in which turn k takes the method’s own output from turn $k - 1$ as input. At each turn a method receives its current state, the instruction and the target render, and VoxHammer also receives a ground-truth edit mask. Ground-truth states are used only for scoring. All states of a chain share one coordinate frame and 4 fixed camera views, 1 conditioning view and 3 held-out views. Image metrics are averaged over the 4 views, and Appendix L reports the conditioning view and the held-out views separately. Scores are averaged over the turns a method completes. Appendix C provides details.

Design of metrics. Previous 3D editing evaluations follow 2 designs. When a target asset exists, they compare the edited object with it as a whole, using Chamfer distance, normal consistency, F-score and image similarity of the renders [198, 192, 126, 145]. Without a target asset, they measure how well the unedited region keeps the source object and judge the edit by its similarity to an edited image or to the text [98, 215, 250, 53, 135, 143]. Both designs score a single edit applied to a clean source object. In our chains, the input to each turn is the method’s own previous output, and every turn still has an exact target. Whole-object F-score can be a poor indicator of editing ability. Because each turn changes only a small region, simply returning the input already yields a high score. We therefore evaluate the edit region and the

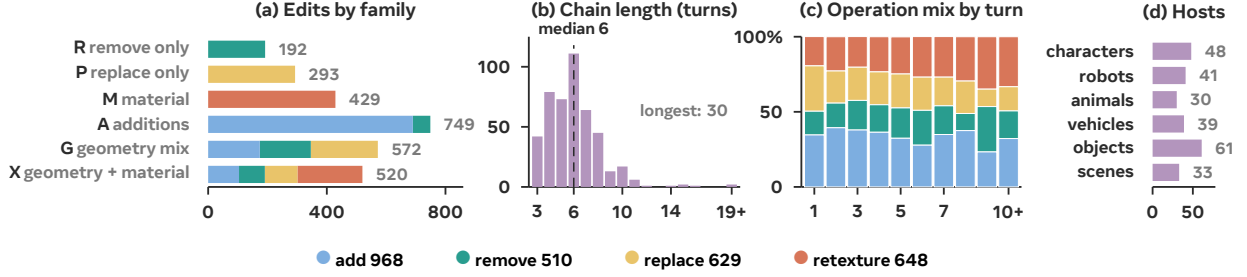


Figure 3 EditHero statistics. The benchmark contains 457 chains, 2755 edits, and 252 hosts. (a) Edits by chain family and operation. (b) Chain lengths, with lengths 19 and above pooled. (c) Share of each operation by turn index, with turns 10 and above pooled. (d) Host categories.

unchanged region separately. In addition, under self-rollout the method’s own states drift from the ground truth starting from the first turn. To isolate the effect of the current edit, we compare each output against the method’s own previous output rather than against the ground-truth previous state.

Two reference baselines. We use 2 baselines as reference points at opposite ends. The *no-operation (no-op) baseline* returns the initial object S_0 at every turn. It keeps everything and edits nothing, so a metric on which it scores well rewards preservation rather than editing. The *regeneration baseline* runs TRELLIS [199] on the target render of each turn, without the instruction or any previous state, and aligns the result to the ground-truth state of that turn by a similarity transform. It sees the target at every turn but keeps nothing from the previous output. A good editor should follow instructions better than the no-op baseline and keep the unedited parts better than regeneration. We also use the no-op baseline to check every metric, because a metric on which doing nothing scores well cannot measure editing.

Regions and region-restricted metrics. We define the edit region from the parts named by the instruction in the ground-truth states before and after turn k . Its complement is the unchanged region. Let $\mathcal{M}_k$ denote these parts and $S[\mathcal{M}_k]$ their geometry in state S , empty if they are absent. The union of their regions before and after the edit covers additions, removals, replacements, and retextures under one definition.

For 3D evaluation, let $\mathcal{V}(\cdot)$ voxelize a surface in the chain’s grid $\mathcal{G} = \{0, \dots, 63\}^3$, and let $\mathcal{D}(\cdot)$ dilate a set by one cell, clamped to the grid. We define

$$E_k = \mathcal{D}(\mathcal{V}(S_{k-1}[\mathcal{M}_k]) \cup \mathcal{V}(S_k[\mathcal{M}_k])), \quad U_k = \mathcal{G} \setminus E_k. \quad (1)$$

For a fixed camera view, let $\Pi^v(S; \mathcal{M}_k)$ be the depth-tested projection of the named parts, excluding pixels occluded by other geometry. With one-pixel dilation over the image domain Ω , the corresponding regions are

$$E_k^v = \mathcal{D}(\Pi^v(S_{k-1}; \mathcal{M}_k) \cup \Pi^v(S_k; \mathcal{M}_k)), \quad U_k^v = \Omega \setminus E_k^v. \quad (2)$$

These regions depend on part identity rather than image or voxel differences. Thus, lighting changes and retextures do not redefine where an edit is allowed, while unintended changes outside the named parts fall in the unchanged region. We then write $m_R(\cdot, \cdot)$ for a metric evaluated only in region R of Eq. (1) or (2). For 3D states X and Y , for example,

$$\text{IoUR}(X, Y) = \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|(\mathcal{V}(X) \cup \mathcal{V}(Y)) \cap R|}. \quad (3)$$

Surface samples are assigned to regions by their voxels, and image metrics aggregate per-pixel values within the specified region. We also evaluate the whole object by setting $R = \mathcal{G}$ or $R = \Omega$. We compare each output $\hat{S}_k$ with its target S_k in the edit region, the unchanged region and the whole object, using the fixed-view renders for the 2D comparisons. Table 1 summarizes the operators, and Appendix D defines them in full.

Instruction following. Target similarity in the edit region can be high simply because the named part was already there. We therefore also measure *instruction following* (IF) against the method’s own previous output. Under self-rollout this output can already differ from the ground truth, so IF counts only the change still

Table 1 Region-restricted evaluation metrics. X, Y are output and reference states; I, J are their fixed-view renders; and R is a region from Eq. (1) or (2). Full definitions appear in Appendix D.

3D			2D		
$\text{IoU}_R(X, Y)$	voxel IoU	↑	$\text{PSNR}_R(I, J)$	peak signal-to-noise ratio	↑
$\text{Recall}_R(X, Y)$	voxel recall of Y	↑	$\text{SSIM}_R(I, J)$	SSIM map [188]	↑
$\text{CD}_R(X, Y)$	Chamfer distance	↓	$\text{LPIPS}_R(I, J)$	LPIPS map [234]	↓
$\text{F}_R^\tau(X, Y)$	F-score at distance τ	↑	$\text{DINO}_R(I, J)$	DINOv2 cosine [139]	↑
			$\text{MAE}_R(I, J)$	mean absolute error	↓

required from it. Otherwise a part lost at an earlier turn would count as a successful removal now, and the no-op baseline would get credit for edits it never made. For additions, A_k contains required new voxels still missing before the turn. For removals, D_k contains voxels that must disappear but remain in the previous output:

$$\begin{aligned} \text{IF}_k^+ &= \text{Recall}_{A_k}(\hat{S}_k, S_k), & A_k &= (\mathcal{V}(S_k) \setminus \mathcal{V}(S_{k-1})) \setminus \mathcal{V}(\hat{S}_{k-1}) \subseteq E_k, \\ \text{IF}_k^- &= 1 - \text{Recall}_{D_k}(\hat{S}_k, S_{k-1}), & D_k &= (\mathcal{V}(S_{k-1}) \setminus \mathcal{V}(S_k)) \cap \mathcal{V}(\hat{S}_{k-1}) \subseteq E_k. \end{aligned} \quad (4)$$

IF is IF^+ for additions, IF^- for removals, and their harmonic mean for replacements. It is undefined when the relevant region is too small and for retextures, which require no geometric change. Appendix Table 6 reports how many turns have a defined IF.

Content consistency. Outside the edit region we measure what the current turn keeps from the method’s own previous output (CC^{prev}) and what is left of the initial host (CC^0):

$$\text{CC}_k^{\text{prev}} = \text{IoU}_{U_k}(\hat{S}_k, \hat{S}_{k-1}), \quad \text{CC}_k^0 = \text{IoU}_{U_k^0}(\hat{S}_k, S_0), \quad U_k^0 = \mathcal{G} \setminus \bigcup_{j \leq k} E_j. \quad (5)$$

The cumulative region excludes the named parts before and after every edit so far. By construction, the no-op baseline has $\text{IF} = 0$ and $\text{CC} = 1$. IF therefore cannot be won by doing nothing, and CC has to be read together with IF. Appendix C illustrates these choices with examples.

5 Experiments

5.1 Setup

We evaluate 4 non-agentic 3D methods, PartFlow [192], Nano3D [215], 3DEditFormer [198] and VoxHammer [98], with their released code, weights and sampling settings. Each of these 4 methods runs under self-rollout on all 457 chains (2755 turns) and receives, at each turn, its own previous output, the instruction and the target render in the fixed view of the chain. VoxHammer also receives the ground-truth edit mask. Nano3D has no material mode, so its material turns run in its replace mode. We also run the no-op and regeneration baselines of Section 4. Appendix F describes the method settings and execution coverage.

5.2 Main results

Table 2 reports geometry and image metrics in the edit region, the unchanged region and the whole object. The 2 baselines appear above the non-agentic methods.

Whole-object scores favor doing nothing. On the whole object, the no-op baseline beats every non-agentic method on F-score, LPIPS and PSNR. A single turn changes only a small part of the object, so returning the input already scores well. Only the edit region tells real edits apart from doing nothing. There, every method is closer to the target than the no-op baseline (LPIPS 0.474–0.523 against 0.570). The regeneration baseline sits at the other extreme. It rebuilds the whole object from the target render at every turn, so it has the highest IF (0.50), but its unchanged-region IoU with its own previous output is only 0.38. A good editor needs both high IF and good preservation, and no non-agentic method reaches regeneration on IF or the no-op baseline on preservation.

Table 2 Comparison of non-agentic methods and baselines. Self-rollout on 457 chains; means over completed turns with a defined score. IF: instruction following against the method’s own previous output. CC^{prev} : IoU of the unchanged region to the method’s own previous output. CD: Chamfer distance ($\times 100$). Drift: mean absolute change of the unchanged image region against the method’s own previous render. Best non-agentic method in bold.

Method	3D						2D				
	edit E_k		unchanged U_k		whole		edit	unchanged		whole	
	IF $\uparrow$	F $\uparrow$	$CC^{prev}\uparrow$	F $\uparrow$	F $\uparrow$	CD $\downarrow$	LPIPS $\downarrow$	LPIPS $\downarrow$	drift $\downarrow$	LPIPS $\downarrow$	PSNR $\uparrow$
No-op baseline	0.00	0.56	1.00	0.82	0.77	2.03	0.570	0.130	0.000	0.151	18.7
TRELLIS regen.	0.50	0.43	0.38	0.43	0.43	2.03	0.400	0.192	0.041	0.202	15.9
PartFlow	0.12	0.47	0.86	0.71	0.67	2.15	0.516	0.166	0.014	0.181	17.0
Nano3D	0.31	0.56	0.89	0.75	0.71	1.99	0.474	0.147	0.009	0.163	17.8
3DEditFormer	0.20	0.43	0.77	0.57	0.55	2.80	0.510	0.188	0.018	0.203	16.2
VoxHammer	0.32	0.38	0.71	0.54	0.52	2.69	0.523	0.205	0.022	0.220	15.2

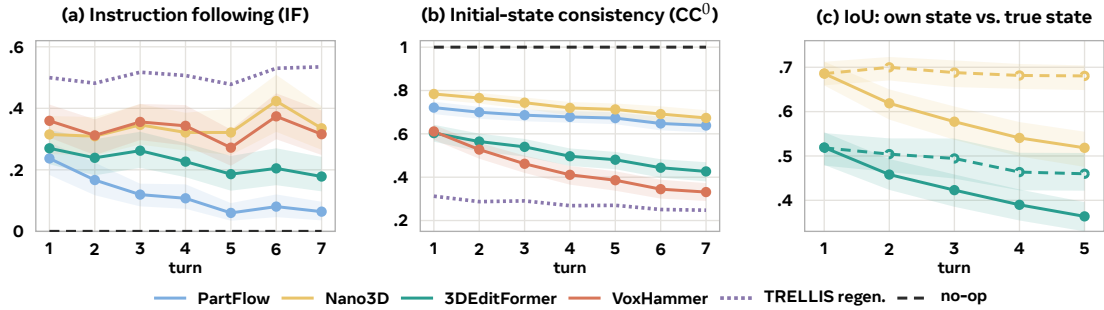


Figure 4 Along a chain. (a,b) Means on the same 135 chains with at least 7 turns, completed by all 4 non-agentic methods (103 hosts). Only defined scores enter each mean; shading shows 95% host-clustered bootstrap intervals. (a) IF relative to the method’s previous output. (b) CC^0 in regions never edited so far. (c) IoU to the target on the 147 chains of the reset diagnostic with at least 5 turns (Appendix H). Solid lines: each turn starts from the model’s own output. Dashed lines: each turn starts from the ground-truth previous state.

Instructions are followed poorly from the first turn. IF is only 0.12–0.32 over all turns, and it is already low at turn 1 (0.24–0.36), before any error can accumulate. Nano3D and VoxHammer follow instructions best, but VoxHammer keeps much less of the rest of the object (unchanged-region IoU 0.71 against 0.89).

Errors accumulate under self-rollout. Along a chain, IF stays low and all 4 methods keep losing the initial object (Figure 4a,b). From turn 1 to turn 7, CC^0 falls from 0.60–0.78 to 0.33–0.67, and VoxHammer loses the most. The reset diagnostic suggests that much of this loss is inherited, because each turn starts from the unintended changes of the previous one. When every turn instead starts from the ground-truth previous state, IoU to the target at turn 5 is 0.68 for Nano3D and 0.46 for 3DEditFormer, but under self-rollout it is only 0.52 and 0.36 (Figure 4c). Appendices G and I report more results.

5.3 Qualitative results

Figure 5 follows 6 replacements on a covered wagon from the comparison subset, in which the cover is replaced 4 times in a row. No non-agentic method follows this sequence. PartFlow keeps a blue rounded cover from turn 2 to turn 5, and the other 3 keep a dark cover and end with a dull olive instead of pale yellow. 3DEditFormer and VoxHammer also change the color of the wagon bed, which no instruction touches. The 3 LLMs shown (Opus 5.5, Astra and Fable 5.1) produce covers closer to the targets, and their errors stay on the cover. Appendix Figure 15 and Appendix M show more chains.

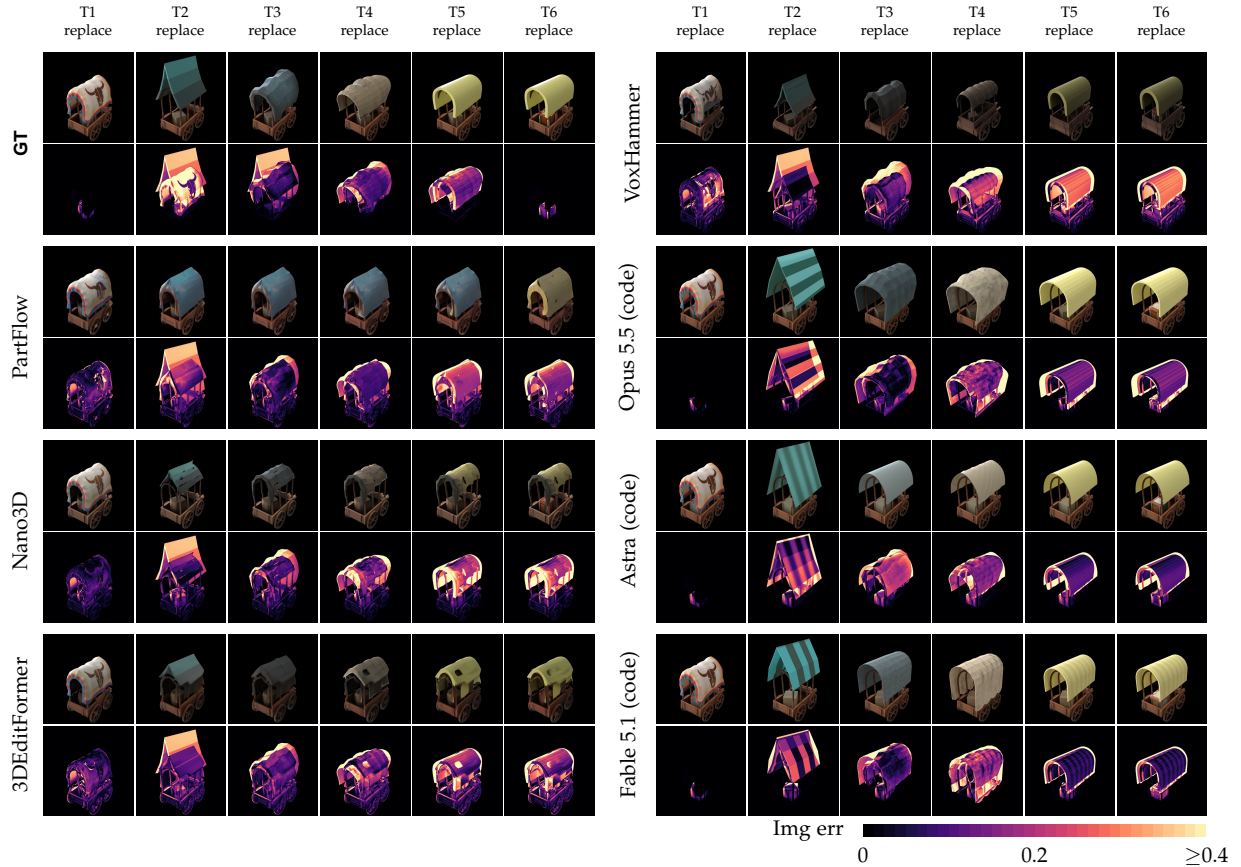


Figure 5 Six consecutive replacements on a covered wagon (turns 1–6). The barrel becomes a wooden crate, the cover becomes a striped peaked canopy and then gray, beige and pale yellow fabric covers, and the crate becomes a science-fiction storage box. Columns are turns. In each block, the upper row is the method’s own output under self-rollout, and the lower row is its per-pixel mean absolute RGB error to the ground truth of that turn. In the ground-truth block, the lower row shows the change from the previous turn. Left: the ground truth and 3 non-agentic methods. Right: VoxHammer and 3 agents (Section 5.4). Same camera and lighting throughout.

5.4 Comparison with LLM/VLM agents

An LLM agent inspects the current mesh, writes code to add, remove, replace or retexture parts, and checks rendered views before it commits an edit. We evaluate 6 models: GLM 5.3 Flash, DeepSeek V4.1 Flash, Sol 6, Astra, Fable 5.1 and Opus 5.5. Each chain is assigned to an independent agent using the same editing prompt. New parts must be built through code; external part libraries and generative 3D or image models are not allowed. Appendix J gives the setup.

We compare these agents with the 4 non-agentic methods and both baselines on a shared *comparison subset* of 55 chains (416 turns). We selected chains with clearly visible edits that resemble an artist’s revisions, covering all 6 families and a range of operation combinations. The editing methods use self-rollout. Table 3 reports the comparison, and Appendix N shows additional examples.

LLMs preserve more and usually follow instructions better. All 6 LLMs keep the rest of the object better than any non-agentic method (CC^{prev} 0.95–0.98 against at most 0.90). All except GLM 5.3 Flash also follow instructions better (IF up to 0.62 for Opus 5.5, against at most 0.36). This is precisely the advantage of the bottom-up approach: an agent modifies only the named part and leaves the remainder of the asset untouched by construction. Removal is the easiest operation, since code can simply delete a part (IF 0.88–0.99). Additions and replacements are harder (IF 0.10–0.49 and 0.17–0.51), because new geometry built from code often misses the target’s size or shape, as on the wagon cover in Figure 5.

Table 3 LLM/VLM agents, non-agentic methods and baselines. All rows use the 55 chains (416 turns) of the comparison subset, so the non-agentic rows differ from Table 2. *Completed / Total*: delivered meshes out of all scheduled turns. Means over completed turns with a defined value. *E*: edit region; *W*: whole object or image. Best method in bold, excluding baselines.

Method	Completed / Total	3D			2D		
		IF $\uparrow$	CC ^{prev} $\uparrow$	F _W $\uparrow$	LPIPS _E $\downarrow$	LPIPS _W $\downarrow$	PSNR _W $\uparrow$
No-op baseline	416/416	0.00	1.00	0.75	0.567	0.154	18.2
TRELLIS regen.	416/416	0.50	0.35	0.37	0.409	0.227	15.1
<i>Non-agentic methods</i>							
PartFlow	416/416	0.13	0.87	0.65	0.502	0.184	16.8
Nano3D	416/416	0.34	0.90	0.70	0.449	0.161	17.6
3DEditFormer	416/416	0.21	0.81	0.59	0.472	0.191	16.5
VoxHammer	377/416	0.36	0.73	0.54	0.491	0.212	15.7
<i>LLM/VLM agents</i>							
GLM 5.3 Flash	416/416	0.26	0.95	0.75	0.447	0.136	21.3
DeepSeek V4.1 Flash	416/416	0.42	0.96	0.78	0.366	0.106	24.0
Sol 6 (medium)	416/416	0.42	0.97	0.79	0.344	0.093	24.6
Astra (medium)	416/416	0.52	0.98	0.81	0.289	0.078	25.5
Fable 5.1	416/416	0.52	0.98	0.80	0.293	0.080	25.4
Opus 5.5	416/416	0.62	0.98	0.82	0.236	0.065	26.7

Unlike the non-agentic methods, every LLM also beats the no-op baseline on whole-object LPIPS (0.136 for GLM 5.3 Flash to 0.065 for Opus 5.5, against 0.154), and all of them deliver a mesh at every turn.

LLMs are slow. An LLM needs about 1.5–6 minutes and 6–16 model calls per edit (medians, Appendix J), while a non-agentic method takes 20–50 seconds on a local GPU. The best LLMs edit well, but they are still too slow for interactive use.

6 Conclusion

EditHero tests whether a 3D editing method can perform successive edits on an asset over many turns. Because its chains are assembled from parts, the correct state after every turn is known exactly, so we can score the requested change and the rest of the object separately. The non-agentic methods, which regenerate the whole object at every turn, often miss the edit from the first turn and let the unedited parts drift as the chain grows. On whole-object F-score, LPIPS and PSNR they even fall behind simply returning the input. LLM agents, which change only the parts required by instructions, keep the remainder of the asset intact, and most of them also follow instructions more closely. It seems that the agentic approach is naturally suited to editing. However, building new geometry from code remains hard, and an edit that takes minutes is still too slow for interactive work. It remains an open question whether we can find a proper combination of agentic and conventional methods that is both fast and accurate. We will release the data synthesis engine and the edit chains for further 3D editing research.

References

- [1] Jinxin Ai, Matthias Nießner, and Ziya Erkoç. Dreamedit3d: Personalization of multi-view diffusion models for 3d editing. *arXiv preprint arXiv:2605.16990*, 2026.
- [2] Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, et al. Self-consuming generative models go MAD. In *ICLR*, 2024.
- [3] Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Zaidi, Megan Langwasser, Wei Xu, and Matthew

- Gombolay. Generating cad code with vision-language models for 3d designs. *arXiv preprint arXiv:2410.05340*, 2024.
- [4] Mohammadmehdi Ataei, Farzaneh Askari, Kamal Rahimi Malekshan, and Pradeep Kumar Jayaraman. Zero-to-cad: Agentic synthesis of interpretable cad programs at million-scale without real data. *arXiv preprint arXiv:2604.24479*, 2026.
 - [5] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, 2022.
 - [6] Hmrishav Bandyopadhyay, Subhadeep Koley, Ayan Das, et al. Doodle your 3d: From abstract freehand sketches to precise 3d shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
 - [7] Amir Barda, Matheus Gadelha, Vladimir G. Kim, Noam Aigerman, Amit H. Bermano, and Thibault Groueix. Instant3dit: Multiview inpainting for fast editing of 3d objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
 - [8] Samyadeep Basu, Mehrdad Saberi, Shweta Bhardwaj, et al. EditVal: Benchmarking diffusion based text-guided image editing methods. *arXiv preprint arXiv:2310.02426*, 2023.
 - [9] Pradhaan S Bhat, Naveen Chandra R, Rishubh Parihar, et al. Thinking in boxes: 3d editing in real images made easy. *arXiv preprint arXiv:2606.20556*, 2026.
 - [10] Alexandre Binniger, Amir Hertz, Olga Sorkine-Hornung, Daniel Cohen-Or, and Raja Giryes. Sens: Part-aware sketch-based implicit neural shape modeling. In *Computer Graphics Forum (Proc. Eurographics)*, volume 43, 2024.
 - [11] Manuel Brack, Felix Friedrich, Katharina Kornmeier, et al. Ledits++: Limitless image editing using text-to-image models. In *CVPR*, 2024.
 - [12] Eric Brochu, Tyson Brochu, and Nando de Freitas. A bayesian interactive optimization approach to procedural animation design. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 103–112, 2010. doi: 10.2312/SCA/SCA10/103-112.
 - [13] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022.
 - [14] Weiwei Cai, Shuangkang Fang, Weicai Ye, et al. Native 3d editing with full attention. *arXiv preprint arXiv:2511.17501*, 2025.
 - [15] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. *arXiv preprint arXiv:2304.08465*, 2023.
 - [16] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.
 - [17] Chieh-Yun Chen, Min Shi, Gong Zhang, and Humphrey Shi. T2i-copilot: A training-free multi-agent text-to-image system for enhanced prompt interpretation and interactive generation. In *ICCV*, 2025.
 - [18] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023.
 - [19] Hansheng Chen, Ruoxi Shi, Yulin Liu, et al. Generic 3d diffusion adapter using controlled multi-view editing. *arXiv preprint arXiv:2403.12032*, 2024.
 - [20] Jiacheng Chen, Ramin Mehran, Xuhui Jia, Saining Xie, and Sanghyun Woo. Blenderfusion: 3d-grounded visual editing and generative compositing. *arXiv preprint arXiv:2506.17450*, 2025.
 - [21] Liyi Chen, Pengfei Wang, Guowen Zhang, Zhiyuan Ma, and Lei Zhang. Omni-3dedit: Generalized versatile 3d editing in one-pass. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

- [22] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. In *European Conference on Computer Vision (ECCV)*, 2024.
- [23] Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. Shap-editor: Instruction-guided latent 3d editing in seconds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [24] Minghao Chen, Roman Shapovalov, Iro Laina, et al. Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. In *CVPR*, 2025.
- [25] Minghao Chen, Jianyuan Wang, Roman Shapovalov, et al. Autopartgen: Autoregressive 3d part generation and discovery. *arXiv preprint arXiv:2507.13346*, 2025.
- [26] Sijin Chen, Xin Chen, Anqi Pang, et al. Meshxl: Neural coordinate field for generative 3d foundation models. In *NeurIPS*, 2024.
- [27] Tianyu Chen, Yasi Zhang, Zhi Zhang, et al. Edival-agent: An object-centric framework for automated, fine-grained evaluation of multi-turn editing. *arXiv preprint arXiv:2509.13399*, 2025.
- [28] Yiwen Chen, Zilong Chen, Chi Zhang, et al. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [29] Yiwen Chen, Yikai Wang, Yihao Luo, et al. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. *arXiv preprint arXiv:2408.02555*, 2024.
- [30] Yiwen Chen, Tong He, Di Huang, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. In *ICLR*, 2025.
- [31] Zilong Chen, Yikai Wang, Wenqiang Sun, Feng Wang, Yiwen Chen, and Huaping Liu. Meshgen: Generating pbr textured mesh with render-enhanced auto-encoder and generative data augmentation. In *CVPR*, 2025.
- [32] Yankuan Chi, Xiang Li, Zixuan Huang, and James M. Rehg. Vinedresser3d: Agentic text-guided 3d editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [33] Hai Ci, Ziheng Peng, Pei Yang, Yingxin Xuan, and Mike Zheng Shou. Diffseg30k: A multi-turn diffusion editing benchmark for localized aigc detection. *arXiv preprint arXiv:2511.19111*, 2025.
- [34] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022.
- [35] Bingquan Dai, Li Ray Luo, Qihong Tang, et al. Meshcoder: Llm-powered structured mesh code generation from point clouds. *arXiv preprint arXiv:2508.14879*, 2025.
- [36] Fernanda De La Torre, Cathy Mengying Fang, Han Huang, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. Llmr: Real-time prompting of interactive worlds using large language models. In *CHI*, 2024.
- [37] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, et al. ProcTHOR: Large-scale embodied AI using procedural generation. In *NeurIPS*, 2022.
- [38] Matt Deitke, Ruoshi Liu, Matthew Wallingford, et al. Objaverse-XL: A universe of 10m+ 3D objects. In *NeurIPS*, 2023.
- [39] Chaorui Deng, Deyao Zhu, Kunchang Li, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.
- [40] Lihe Ding, Shaocong Dong, Yaokun Li, Chenjian Gao, Xiao Chen, Rui Han, Yihao Kuang, Hong Zhang, Bo Huang, Zhanpeng Huang, Zibin Wang, Dan Xu, and Tianfan Xue. Fullpart: Generating each 3d part at full resolution. *arXiv preprint arXiv:2510.26140*, 2025.
- [41] Shaocong Dong, Lihe Ding, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. Interactive3d: Create what you want by interactive 3d generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- [42] Wenqi Dong, Bangbang Yang, Lin Ma, et al. Coin3d: Controllable and interactive 3d assets generation with proxy-guided conditioning. In *ACM SIGGRAPH Conference Papers*, 2024.
- [43] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1979. ISSN 0090-5364. doi: 10.1214/aos/1176344552.
- [44] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. *arXiv preprint arXiv:1612.00603*, 2016.
- [45] Rongyao Fang, Chengqi Duan, Kun Wang, et al. Got: Unleashing reasoning capability of multimodal large language model for visual generation and editing. *arXiv preprint arXiv:2503.10639*, 2025.
- [46] Shuangkang Fang, Yufeng Wang, Yi-Hsuan Tsai, et al. Chat-edit-3d++: Interactive 3d and 4d scene editing via large language models. *arXiv preprint arXiv:2608.29137*, 2026.
- [47] Fan Fei, Jiajun Tang, Fei-Peng Tian, Boxin Shi, and Ping Tan. Pature: Efficient pbr texture generation on packed views with visual autoregressive models. *Computational Visual Media*, 2026.
- [48] Haitang Feng, Xinkai Chen, Jie Liu, et al. Objfiller3d: Scaling 3d object inpainting to dense multi-view consistency. *arXiv preprint arXiv:2508.18271*, 2025.
- [49] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, et al. Layoutgpt: Compositional visual planning and generation with large language models. In *NeurIPS*, 2023.
- [50] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *ICLR*, 2024.
- [51] Rinon Gal, Yuval Alaluf, Yuval Atzmon, et al. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.
- [52] Yipeng Gao, Lei Shu, Genzhi Ye, Xi Xiong, Ameesh Makadia, Meiqi Guo, Laurent Itti, and Jindong Chen. 3dcodebench: Benchmarking agentic procedural 3d modeling via code. *arXiv preprint arXiv:2606.01057*, 2026.
- [53] Inbar Gat, Dana Cohen-Bar, Guy Levy, Elad Richardson, and Daniel Cohen-Or. Shapeup: Scalable image-conditioned 3d editing. In *ACM SIGGRAPH Conference Papers*, 2026.
- [54] Yuying Ge, Sijie Zhao, Chen Li, Yixiao Ge, and Ying Shan. SEED-Data-Edit technical report: A hybrid dataset for instructional image editing. *arXiv preprint arXiv:2405.04007*, 2024.
- [55] Zigang Geng, Binxin Yang, Tiankai Hang, et al. Instructdiffusion: A generalist modeling interface for vision tasks. *arXiv preprint arXiv:2309.03895*, 2023.
- [56] Javier Gonzalez, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential bayesian optimization. *arXiv preprint arXiv:1704.03651*, 2017.
- [57] Klaus Greff, Francois Belletti, Lucas Beyer, et al. Kubric: A scalable dataset generator. In *CVPR*, 2022.
- [58] Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. Blendergym: Benchmarking foundational model systems for graphics editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [59] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [60] Zekun Hao, David W. Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024.
- [61] Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [62] Yi He, Jiangming Wang, Xinyu Wang, et al. Geoedit: Geometry-aware object editing via dual-branch denoising. In *ECCV*, 2026.

- [63] Zebin He, Mingxin Yang, Shuhui Yang, et al. Materialmvp: Illumination-invariant material generation via multi-view pbr diffusion. *arXiv preprint arXiv:2503.10289*, 2025.
- [64] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.
- [65] Yicong Hong, Kai Zhang, Jiuxiang Gu, et al. Lrm: Large reconstruction model for single image to 3d. In *ICLR*, 2024.
- [66] Teng-Fang Hsiao, Bo-Kai Ruan, Yu-Lun Liu, and Hong-Han Shuai. Vecset-edit: Unleashing pre-trained lrm for mesh editing from single image. In *ACM SIGGRAPH Conference Papers*, 2026.
- [67] Shimin Hu, Yuanyi Wei, Fei Zha, Yudong Guo, and Juyong Zhang. Easy3e: Feed-forward 3d asset editing via rectified voxel flow. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [68] Ziniu Hu, Ahmet Iscen, Aashi Jain, et al. Scenecraft: An llm agent for synthesizing 3d scene as blender code. *arXiv preprint arXiv:2403.01248*, 2024.
- [69] Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. *arXiv preprint arXiv:2404.17672*, 2024.
- [70] Jiahui Huang, Yasi Zhang, Tianyu Chen, et al. MT-EditFlow: Reinforcement learning for multi-turn image editing with flow matching. *arXiv preprint arXiv:2606.01985*, 2026.
- [71] Xin Huang, Tengfei Wang, Ziwei Liu, and Qing Wang. Material anything: Generating materials for any 3d object via diffusion. *arXiv preprint arXiv:2411.15138*, 2024.
- [72] Yinming Huang, Shuyuan Tu, Xi Yan, et al. Agentic visual generation: From generative models to agentic control. *arXiv preprint arXiv:2609.06758*, 2026.
- [73] Yuzhou Huang, Liangbin Xie, Xintao Wang, et al. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. *arXiv preprint arXiv:2312.06739*, 2023.
- [74] Zijian Huang, Jay Zhangjie Wu, Zian Wang, et al. Ape: Agentic prompt enhancer for image generation and editing. *arXiv preprint arXiv:2606.00204*, 2026.
- [75] Mude Hui, Siwei Yang, Bingchen Zhao, et al. Hq-edit: A high-quality dataset for instruction-based image editing. *arXiv preprint arXiv:2404.09990*, 2024.
- [76] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993. ISSN 0162-8828. doi: 10.1109/34.232073.
- [77] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH*, pages 409–416, 1999.
- [78] img2threejs contributors. img2threejs: Image to procedural Three.js. <https://github.com/img2threejs/img2threejs>, 2026. Upstream v2.0.0; used with our source-editing and Blender-rendering adaptations.
- [79] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.
- [80] R. Kenny Jones, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapelib: Designing a library of programmatic 3d shape abstractions with large language models. *arXiv preprint arXiv:2502.08884*, 2025.
- [81] Bahjat Kawar, Shiran Zada, Oran Lang, et al. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276*, 2022.
- [82] Umar Khalid, Hasan Iqbal, Nazmul Karim, Jing Hua, and Chen Chen. Latenteditor: Text driven local editing of 3d scenes. In *European Conference on Computer Vision (ECCV)*, 2024.

- [83] Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad models from beginner-to-expert level text prompts. In *NeurIPS*, 2024.
- [84] Hyunwoo Kim, Itai Lang, Noam Aigerman, Thibault Groueix, Vladimir G. Kim, and Rana Hanocka. Meshup: Multi-target mesh deformation via blended score distillation. In *International Conference on 3D Vision (3DV)*, 2025.
- [85] Jeonghwan Kim, Yushi Lan, Armando Fortes, Yongwei Chen, and Xingang Pan. Fastmesh: Efficient artistic mesh generation via component decoupling. In *3DV*, 2026.
- [86] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017. ISSN 1557-7368. doi: 10.1145/3072959.3073599.
- [87] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4):48:1–48:11, 2017. doi: 10.1145/3072959.3073598.
- [88] Yuki Koyama, Issei Sato, and Masataka Goto. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics*, 2020.
- [89] Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *ICCV*, 2025.
- [90] Black Forest Labs, Stephen Batifol, Andreas Blattmann, et al. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.
- [91] Long Le, Jason Xie, William Liang, et al. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *ICLR*, 2025.
- [92] Seunggywan Lee, Hwanhee Jung, Byoungsoo Koh, Qixing Huang, Sangho Yoon, and Sangpil Kim. Pasta: Part-aware sketch-to-3d shape generation with text-aligned prior. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [93] Bo Li, Jiahao Kang, Yubo Ma, et al. Sketchfacegs: Real-time sketch-driven face editing and generation with gaussian splatting. In *CVPR*, 2026.
- [94] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Sketch2cad: Sequential cad modeling by sketching in context. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, volume 39, 2020.
- [95] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2cad: Parsing freehand drawings into cad commands. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 41, 2022.
- [96] Chenxi Li, Weijie Wang, Qiang Li, Bruno Lepri, Nicu Sebe, and Weizhi Nie. Freeinsert: Disentangled text-guided object insertion in 3d gaussian scene without spatial priors. *ACM Multimedia*, 2025.
- [97] Haoxuan Li, Ziya Erkoç, Lei Li, et al. Meshpad: Interactive sketch-conditioned artist-reminiscent mesh generation and editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [98] Lin Li, Zehuan Huang, Haoran Feng, et al. Voxhammer: Training-free precise and coherent 3d editing in native 3d space. In *International Conference on 3D Vision (3DV)*, 2026.
- [99] Peng Li, Suizhi Ma, Jialiang Chen, et al. Cmd: Controllable multiview diffusion for 3d editing and progressive generation. In *ACM SIGGRAPH Conference Papers*, 2025.
- [100] Sikuang Li, Chen Yang, Jiemin Fang, et al. Sculpt: Subtractive composition for 3d part generation. *arXiv preprint arXiv:2608.13541*, 2026.
- [101] Weiyu Li, Xuanyang Zhang, Zheng Sun, et al. Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets. *arXiv preprint arXiv:2505.07747*, 2025.
- [102] Weiyu Li, Antoine Toisoul, Tom Monnier, et al. Meshflow: Efficient artistic mesh generation via meshvae and flow-based diffusion transformer. In *CVPR*, 2026.

- [103] Yangguang Li, Zi-Xin Zou, Zexiang Liu, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *arXiv preprint arXiv:2502.06608*, 2025.
- [104] Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. In *NeurIPS*, 2025.
- [105] Zihan Liang, Jiahao Sun, and Haoran Ma. An llm-lvlm driven agent for iterative and fine-grained image editing. *arXiv preprint arXiv:2508.17435*, 2025.
- [106] Manwen Liao, Xinyu Lian, Jian Mao, et al. Megaparts: Scaling part-aware 3d object generation to 300 parts via token-efficient autoregressive modeling. *arXiv preprint arXiv:2608.14783*, 2026.
- [107] Bin Lin, Zongjian Li, Xinhua Cheng, et al. Uniworld-v1: High-resolution semantic encoders for unified visual understanding and generation. *arXiv preprint arXiv:2506.03147*, 2025.
- [108] Chen-Hsuan Lin, Jun Gao, Luming Tang, et al. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023.
- [109] Youtian Lin, Yikang Yang, Zhanpeng Hu, et al. Procedura: Agentic 3d modeling with procedural control. *arXiv preprint arXiv:2608.26238*, 2026.
- [110] Yuchen Lin, Chenguo Lin, Panwang Pan, et al. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. In *NeurIPS*, 2025.
- [111] Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *CVPR*, 2025.
- [112] Chenxi Liu, Selena Ling, and Alec Jacobson. Gimbo: Interactive generative image model merging via bayesian optimization. In *SIGGRAPH*, 2026.
- [113] Chunjiang Liu, Xiaoyuan Wang, Haoyu Chen, Yizhou Zhao, Ming-Hsuan Yang, and László A. Jeni. Simworlds: A multi-agent system for dynamic 3d scene creation. *arXiv preprint arXiv:2607.01766*, 2026.
- [114] Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. Sketchdream: Sketch-based text-to-3d generation and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 43(4), 2024.
- [115] Jiageng Liu, Weijie Lyu, Xueting Li, Yejie Guo, and Ming-Hsuan Yang. Edit3r: Instant 3d scene editing from sparse unposed images. *arXiv preprint arXiv:2512.25071*, 2025.
- [116] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, et al. PartField: Learning 3D feature fields for part segmentation and beyond. In *ICCV*, 2025.
- [117] Ruoshi Liu, Rundui Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023.
- [118] Shiyu Liu, Yucheng Han, Peng Xing, et al. Step1X-Edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.
- [119] Zichen Liu, Yue Yu, Hao Ouyang, et al. Magicquill: An intelligent interactive image editing system. In *CVPR*, 2025.
- [120] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *CVPR*, 2024.
- [121] Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. Ll3m: Large language 3d modelers. *arXiv preprint arXiv:2508.08228*, 2025.
- [122] Changfeng Ma, Yang Li, Xinhao Yan, et al. P3-SAM: Native 3D part segmentation. *arXiv preprint arXiv:2509.06784*, 2025.
- [123] Shichao Ma, Yunhe Guo, Jiahao Su, Qihe Huang, Zhengyang Zhou, and Yang Wang. Talk2image: A multi-agent system for multi-turn image generation and editing. *arXiv preprint arXiv:2508.06916*, 2025.
- [124] Yiwei Ma, Jiayi Ji, Ke Ye, et al. I2EBench: A comprehensive benchmark for instruction-based image editing. In *NeurIPS*, 2024.

- [125] Yiwei Ma, Ke Ye, Weihuang Lin, et al. An extensive benchmark for single-round and multi-round instruction-based image editing. *International Journal of Computer Vision*, 2026.
- [126] Ziqi Ma, Hongqiao Chen, Yisong Yue, and Georgia Gkioxari. Feedforward 3D editing via text-steerable image-to-3D. *arXiv preprint arXiv:2512.13678*, 2025.
- [127] Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, et al. Cad-assistant: Tool-augmented vllms as generic cad task solvers. *arXiv preprint arXiv:2412.13810*, 2024.
- [128] Chenlin Meng, Yutong He, Yang Song, et al. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.
- [129] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.
- [130] Aryan Mikaeili, Or Perel, Mehdi Safaei, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [131] Kaichun Mo, Shilin Zhu, Angel X. Chang, et al. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *CVPR*, 2019.
- [132] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. *arXiv preprint arXiv:2211.09794*, 2022.
- [133] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020.
- [134] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: Designing freeform surfaces with 3d curves. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), 2007.
- [135] Rui Nie, Chuang Wang, Haitao Zhou, et al. EditFlow3D: Automated local editing of 3D assets with trajectory preservation. *arXiv preprint arXiv:2608.03179*, 2026.
- [136] Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *ICLR*, 2024.
- [137] Yansong Ning, Jingwen Ye, Zhongkai Wu, et al. Vibeworliding: Can multimodal agents construct 3d open worlds end-to-end? *arXiv preprint arXiv:2608.15265*, 2026.
- [138] Nimra Noor, Muhammad Bilal, Abdullah Hussain, and Hassan Baig. Nova3d: Code-native generation of programmable 3d assets. *arXiv preprint arXiv:2607.22738*, 2026.
- [139] Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [140] Bo Pang, Jiaqi Pan, Xiaocheng Zhang, Jiacheng Xu, Guoping Wang, and Peng-Shuai Wang. Visculpt: Visual-centric agentic geometry editing. *arXiv preprint arXiv:2608.24169*, 2026.
- [141] Maria Parelli, Michael Oechsle, Michael Niemeyer, Federico Tombari, and Andreas Geiger. 3d-latte: Latent space 3d editing from textual instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [142] Geon Yeong Park, Roman Shapovalov, Rakesh Ranjan, Jong Chul Ye, Andrea Vedaldi, and Thu Nguyen-Phuoc. Meshregen: A unified 3d geometry regeneration framework. *arXiv preprint arXiv:2604.28134*, 2026.
- [143] Sariah Patro, Arjun Mehra, and Nikhil Bhatia. What makes a 3d scene editable? a factorized benchmark of fidelity, locality, consistency, and preservation. *arXiv preprint arXiv:2609.14899*, 2026.
- [144] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.
- [145] Tobias Preintner, Yunfei Deng, Phillip Müller, et al. 3dmorph: Single-image-guided local 3d shape editing and morphing. In *International Joint Conference on Neural Networks (IJCNN)*, 2026.

- [146] Zhangyang Qi, Yunhan Yang, Mengchen Zhang, et al. Tailor3d: Customized 3d assets editing and generation with dual-side images. *arXiv preprint arXiv:2407.06191*, 2024.
- [147] Yusu Qian, Eli Bocek-Rivele, Liangchen Song, et al. Pico-banana-400k: A large-scale dataset for text-guided image editing. *arXiv preprint arXiv:2510.19808*, 2025.
- [148] Yansong Qu, Dian Chen, Xinyang Li, et al. Drag your gaussian: Effective drag-based editing with score distillation for 3d gaussian splatting. In *ACM SIGGRAPH Conference Papers*, 2025.
- [149] Qwen Team. Qwen3-VL-32B-Instruct. <https://huggingface.co/Qwen/Qwen3-VL-32B-Instruct>, 2025. Official model card.
- [150] Alexander Raistrick, Lahav Lipson, Zeyu Ma, et al. Infinite photorealistic worlds using procedural generation. In *CVPR*, 2023.
- [151] Rajalaxmi Rajagopalan, Debottam Dutta, Yu-Lin Wei, and Romit Roy Choudhury. Personalized image generation via human-in-the-loop bayesian optimization. *arXiv preprint arXiv:2602.02388*, 2026.
- [152] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.
- [153] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. *arXiv preprint arXiv:2302.01721*, 2023.
- [154] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023.
- [155] Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-recode: Reverse engineering cad code from point clouds. *arXiv preprint arXiv:2412.14042*, 2024.
- [156] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [157] Etai Sella, Hao Phung, Nitay Amiel, Or Litany, Or Patashnik, and Hadar Averbuch-Elor. Prox-e: Fine-grained 3d shape editing via primitive-based abstractions. In *ACM SIGGRAPH Conference Papers*, 2026.
- [158] Mingqi Shao, Feng Xiong, Zhaoxu Sun, and Mu Xu. Mvpainter: Accurate and detailed 3d texture generation via multi-view diffusion with geometric control. *arXiv preprint arXiv:2505.12635*, 2025.
- [159] Shelly Sheynin, Adam Polyak, Uriel Singer, et al. Emu Edit: Precise image editing via recognition and generation tasks. In *CVPR*, 2024.
- [160] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024.
- [161] Zhenyu Shu, Junlong Yu, Kai Chao, Shiqing Xin, and Ligang Liu. Gaussedit: Adaptive 3d scene editing with text and image prompts. *IEEE Transactions on Visualization and Computer Graphics*, 2025.
- [162] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, et al. Meshgpt: Generating triangle meshes with decoder-only transformers. In *CVPR*, 2024.
- [163] Yawar Siddiqui, Tom Monnier, Filippos Kokkinos, et al. Meta 3d assetgen: Text-to-mesh generation with high-quality geometry, texture, and pbr materials. *arXiv preprint arXiv:2407.02445*, 2024.
- [164] Habib Slim, Shariq Farooq Bhat, Mohamed Elhoseiny, Yifan Wang, and Mike Roberts. Compose: Compositional synthesis and editing of 3d shapes via part-aware control. *arXiv preprint arXiv:2605.19350*, 2026.
- [165] Gaochao Song, Zibo Zhao, Haohan Weng, Jingbo Zeng, Rongfei Jia, and Shenghua Gao. Topology-preserved auto-regressive mesh generation in the manner of weaving silk. In *ICLR*, 2026.

- [166] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945*, 2023.
- [167] Abdel Aziz Taha and Allan Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC Medical Imaging*, 15(1), 2015. ISSN 1471-2342. doi: 10.1186/s12880-015-0068-x.
- [168] Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. Scenemotifcoder: Example-driven visual program learning for generating 3d object arrangements. In *3DV*, 2025.
- [169] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, et al. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. In *ICLR*, 2025.
- [170] Jiaxiang Tang, Ruijie Lu, Zhaoshuo Li, et al. Efficient part-level 3d object generation via dual volume packing. In *NeurIPS*, 2025.
- [171] Kenan Tang, Praveen Arunshankar, Andong Hua, Anthony Yang, and Yao Qin. Banana100: Breaking NR-IQA metrics by 100 iterative image replications with Nano Banana Pro. *arXiv preprint arXiv:2604.03400*, 2026.
- [172] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3D reconstruction networks learn? In *CVPR*, 2019.
- [173] Tencent Hunyuan3D Team. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025.
- [174] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. *arXiv preprint arXiv:2211.12572*, 2022.
- [175] Chunshi Wang, Haohan Weng, Junliang Ye, et al. Polyflow: Continuous topology embedding flow matching for artist-style mesh generation. *arXiv preprint arXiv:2606.30673*, 2026.
- [176] Chunshi Wang, Junliang Ye, Yunhan Yang, et al. Part-x-mlm: Part-aware 3d multimodal large language model. In *ICLR*, 2026.
- [177] Hao Wang, Wenhui Zhu, Shao Tang, Zhipeng Wang, Xuanzhao Dong, Xin Li, Xiwen Chen, Ashish Bastola, Xinhao Huang, Yalin Wang, and Abolfazl Razi. Ezblender: Efficient 3d editing with plan-and-react agent. *arXiv preprint arXiv:2601.07143*, 2026.
- [178] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [179] Penghao Wang, Yiyang He, Xin Lv, et al. PartNeXt: A next-generation dataset for fine-grained and hierarchical 3D part understanding. In *NeurIPS Datasets and Benchmarks Track*, 2025.
- [180] Rui-Huan Wang, Si-Tong Wei, Jia-Qi He, Heng-Yi Wei, Baoquan Chen, and Peng-Shuai Wang. adsl: Agentic 3d creation via joint agent-program design. *arXiv preprint arXiv:2608.17975*, 2026.
- [181] Su Wang, Chitwan Saharia, Ceslee Montgomery, et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *CVPR*, 2023.
- [182] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*, 2020.
- [183] Xinjie Wang, Liu Liu, Taojun Ding, Andrew Choi, Chaodong Huang, Mengao Zhao, Ziang Li, Jackson Jiang, Chunlei Yu, Shengxiang Liu, Wei Xu, and Zhizhong Su. Embodiedgen v2: An agentic, simulation-ready 3d world engine for embodied ai. *arXiv preprint arXiv:2607.07459*, 2026.
- [184] Yuxuan Wang, Xuanyu Yi, Haohan Weng, et al. Nautilus: Locality-aware autoencoder for scalable mesh generation. In *ICCV*, 2025.

- [185] Zhengyi Wang, Cheng Lu, Yikai Wang, et al. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2023.
- [186] Zhengyi Wang, Jonathan Lorraine, Yikai Wang, et al. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024.
- [187] Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. Genartist: Multimodal llm as an agent for unified image generation and editing. In *NeurIPS*, 2024.
- [188] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.
- [189] Cong Wei, Zheyang Xiong, Weiming Ren, Xinrun Du, Ge Zhang, and Wenhui Chen. Omniedit: Building image editing generalist models through specialist supervision. *arXiv preprint arXiv:2411.07199*, 2024.
- [190] Haohan Weng, Yikai Wang, Tong Zhang, C. L. Philip Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. In *ICLR*, 2025.
- [191] Haohan Weng, Zibo Zhao, Biwen Lei, et al. Scaling mesh generation via compressive tokenization. In *CVPR*, 2025.
- [192] Jiawei Weng, Saining Zhang, Zhenxin Diao, et al. Feedforward 3d editing learns from semantic-part transformation. *arXiv preprint arXiv:2605.27351*, 2026.
- [193] Chenfei Wu, Jiahao Li, Jingren Zhou, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.
- [194] Jing Wu, Jia-Wang Bian, Xinghui Li, et al. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European Conference on Computer Vision (ECCV)*, 2024.
- [195] Qiucheng Wu, Jing Shi, Simon Jenni, et al. Retouchiq: Mllm agents for instruction-based image retouching with generalist reward. *arXiv preprint arXiv:2602.17558*, 2026.
- [196] Shuang Wu, Youtian Lin, Feihu Zhang, et al. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. In *NeurIPS*, 2024.
- [197] Yongliang Wu, Zonghui Li, Xinting Hu, et al. Kris-bench: Benchmarking next-level intelligent image editing models. *arXiv preprint arXiv:2505.16707*, 2025.
- [198] Ruihao Xia, Yang Tang, and Pan Zhou. Towards scalable and consistent 3d editing. *arXiv preprint arXiv:2510.02994*, 2025.
- [199] Jianfeng Xiang, Zelong Lv, Sicheng Xu, et al. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, 2025.
- [200] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, et al. Native and compact structured latents for 3d generation. In *CVPR*, 2026.
- [201] Shitao Xiao, Yueze Wang, Junjie Zhou, et al. Omnigen: Unified image generation. *arXiv preprint arXiv:2409.11340*, 2024.
- [202] Bojun Xiong, Jialun Liu, Jiakui Hu, et al. Texgaussian: Generating high-quality pbr material via octree-based 3d gaussian splatting. In *CVPR*, 2025.
- [203] Hang Xu, Xiaoxiao Ma, Guohui Zhang, et al. AnchorEdit: Maintaining temporal consistency in multi-turn image editing via causal memory. *arXiv preprint arXiv:2606.11751*, 2026.
- [204] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.
- [205] Xiangyuan Xue, Zeyu Lu, Di Huang, Zidong Wang, Wanli Ouyang, and Lei Bai. Comfybench: Benchmarking llm-based agents in comfyui for autonomously designing collaborative ai systems. *arXiv preprint arXiv:2409.01392*, 2024.

- [206] Xinhao Yan, Jiachen Xu, Yang Li, et al. X-Part: High fidelity and structure coherent shape decomposition. *arXiv preprint arXiv:2509.08643*, 2025.
- [207] Yue Yang, Fan-Yun Sun, Luca Weihs, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *CVPR*, 2024.
- [208] Yujia Yang, Yuanxiang Wang, Zhenyu Guan, et al. Omni iie bench: Benchmarking the practical capabilities of image editing models. *arXiv preprint arXiv:2603.16944*, 2026.
- [209] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, et al. SAMPart3D: Segment any part in 3D objects. *arXiv preprint arXiv:2411.07184*, 2024.
- [210] Yunhan Yang, Yuan-Chen Guo, Yukun Huang, et al. HoloPart: Generative 3D part amodal segmentation. *arXiv preprint arXiv:2504.07943*, 2025.
- [211] Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, et al. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. In *SIGGRAPH Asia*, 2025.
- [212] Chongjie Ye, Yushuang Wu, Ziteng Lu, et al. Hi3dgen: High-fidelity 3d geometry generation from images via normal bridging. *arXiv preprint arXiv:2503.22236*, 2025.
- [213] Junliang Ye, Zhengyi Wang, Ruowen Zhao, Shenghao Xie, and Jun Zhu. Shapellm-omni: A native multimodal llm for 3d generation and understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- [214] Junliang Ye, Kenkun Liu, Guocun Wang, et al. Hunyuan3d-buffalo 1.0: A unified multimodal model for scalable 3d generation, understanding, and editing. *arXiv preprint arXiv:2608.02711*, 2026.
- [215] Junliang Ye, Shenghao Xie, Ruowen Zhao, et al. Nano3d: A training-free approach for efficient 3d editing without masks. In *International Conference on Learning Representations (ICLR)*, 2026.
- [216] Ruijie Ye, Jiayi Zhang, Zhuoxin Liu, et al. Agent banana: High-fidelity image editing with agentic thinking and tooling. *arXiv preprint arXiv:2602.09084*, 2026.
- [217] Yang Ye, Xianyi He, Zongjian Li, et al. ImgEdit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025.
- [218] Yuxiao Ye, Haoran He, Fangyuan Kong, et al. Edit-r2: Context-aware reinforcement learning for multi-turn image editing. *arXiv preprint arXiv:2606.05950*, 2026.
- [219] Yu-Ying Yeh, Jia-Bin Huang, Changil Kim, et al. Texturedreamer: Image-guided texture synthesis through geometry-aware diffusion. *arXiv preprint arXiv:2401.09416*, 2024.
- [220] Kyeongmin Yeo, Yunhong Min, Jaihoon Kim, and Minhyuk Sung. Matlat: Material latent space for pbr texture generation. *arXiv preprint arXiv:2512.17302*, 2025.
- [221] Youtan Yin, Yanning Zhou, Jiacheng Wei, et al. Editverse3d: High-quality 3d object editing with region-aware learning. In *European Conference on Computer Vision (ECCV)*, 2026.
- [222] Qifan Yu, Wei Chow, Zhongqi Yue, et al. Anyedit: Mastering unified high-quality image editing for any idea. In *CVPR*, 2025.
- [223] Ruihan Yu, Lian Fu, Muyao Niu, et al. Kaininja: Extending native 3d generators to the part level. *arXiv preprint arXiv:2609.15659*, 2026.
- [224] Zeqing Yuan, Haoxuan Lan, Qiang Zou, and Junbo Zhao. 3d-premise: Can large language models generate 3d shapes with sharp features and parametric control? *arXiv preprint arXiv:2401.06437*, 2024.
- [225] Xianfang Zeng, Xin Chen, Zhongqi Qi, et al. Paint3d: Paint anything 3d with lighting-less texture diffusion models. *arXiv preprint arXiv:2312.13913*, 2023.
- [226] Zilai Zeng, Mingdeng Cao, Zijie Li, et al. Towards robust sequential decomposition for complex image editing. In *CVPR*, 2026.

- [227] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (SIGGRAPH)*, 2023.
- [228] Chuanrui Zhang, Minghan Qin, Yuang Wang, Baifeng Xie, Hang Li, and Ziwei Wang. Simart: Decomposing monolithic meshes into sim-ready articulated assets via mllm. *arXiv preprint arXiv:2603.23386*, 2026.
- [229] Haochen Zhang, Animesh Sinha, Felix Juefei-Xu, et al. Non-markov multi-round conversational image generation with history-conditioned mllms. *arXiv preprint arXiv:2601.20911*, 2026.
- [230] Kai Zhang, Lingbo Mo, Wenhui Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023.
- [231] Longwen Zhang, Ziyu Wang, Qixuan Zhang, et al. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (SIGGRAPH)*, 2024.
- [232] Longwen Zhang, Qixuan Zhang, Haoran Jiang, et al. Bang: Dividing 3d assets via generative exploded dynamics. *ACM Transactions on Graphics (SIGGRAPH)*, 2025.
- [233] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.
- [234] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- [235] Xuying Zhang, Yutong Liu, Yangguang Li, et al. Tar3d: Creating high-quality 3d assets via next-part prediction. In *ICCV*, 2025.
- [236] Yang Zhang, Xiukun Wei, and Xueru Zhang. When and how human curation backfires: Preference alignment under multi-model self-consuming loop. *arXiv preprint arXiv:2605.29267*, 2026.
- [237] Yudi Zhang, Yeming Geng, and Lei Zhang. Scribblesense: Generative scribble-based texture editing with intent prediction. *IEEE Transactions on Visualization and Computer Graphics*, 2026.
- [238] Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. In *NeurIPS*, 2025.
- [239] Haozhe Zhao, Xiaojuan Ma, Liang Chen, et al. Ultraedit: Instruction-based fine-grained image editing at scale. In *NeurIPS*, 2024.
- [240] Ruowen Zhao, Junliang Ye, Zhengyi Wang, et al. Deepmesh: Auto-regressive artist-mesh creation with reinforcement learning. In *ICCV*, 2025.
- [241] Wang Zhao, Yan-Pei Cao, Jiale Xu, Yuejiang Dong, and Ying Shan. Assembler: Scalable 3d part assembly via anchor point diffusion. *arXiv preprint arXiv:2506.17074*, 2025.
- [242] Xiangyu Zhao, Peiyuan Zhang, Kexian Tang, et al. Envisioning beyond the pixels: Benchmarking reasoning-informed visual editing. *arXiv preprint arXiv:2504.02826*, 2025.
- [243] Yiran Zhao, Yaoqi Ye, Xiang Liu, Michael Qizhe Shieh, and Trung Bui. Imageedit-r1: Boosting multi-agent image editing via reinforcement learning. *arXiv preprint arXiv:2603.08059*, 2026.
- [244] Yuming Zhao, Zangyueyang Xian, Qijian Zhang, et al. Seamflow: Structure-aware flow matching on edge probabilities for artist-like uv unwrapping. In *SIGGRAPH Asia*, 2026.
- [245] Zibo Zhao, Wen Liu, Xin Chen, et al. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *NeurIPS*, 2023.
- [246] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.
- [247] Wentao Zheng and Ancong Wu. Pwm-artgen: Part world model for articulated object generation. *arXiv preprint arXiv:2607.02045*, 2026.

- [248] Jie Zhou, Zhongjin Luo, Qian Yu, Xiaoguang Han, and Hongbo Fu. Ga-sketching: Shape modeling from multi-view sketching with geometry-aligned deep implicit functions. *arXiv preprint arXiv:2309.05946*, 2023.
- [249] Matt Zhou, Ruining Li, Xiaoyang Lyu, et al. Articraft: An agentic system for scalable articulated 3D asset generation. *arXiv preprint arXiv:2605.15187*, 2026.
- [250] Zhenglin Zhou, Fan Ma, Chengzhuo Gui, et al. Anchorflow: Training-free 3d editing via latent anchor-aligned flows. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [251] Zijun Zhou, Yingying Deng, Xiangyu He, Weiming Dong, and Fan Tang. Multi-turn consistent image editing. *arXiv preprint arXiv:2505.04320*, 2025.
- [252] Xinnan Zhu, Ruijie Xu, Jiayu Ying, et al. Jointedit3d: Feed-forward 3d scene editing in a unified latent space. *arXiv preprint arXiv:2606.13345*, 2026.
- [253] Yiheng Zhu, Kangle Deng, Jean-Philippe Fauconnier, et al. Cubepart: An open-vocabulary part-controllable 3d generator. In *SIGGRAPH*, 2026.
- [254] Zixin Zhu, Haoxiang Li, Xuelu Feng, He Wu, Chunming Qiao, and Junsong Yuan. Georemove: Removing objects and their causal visual artifacts. In *NeurIPS*, 2025.
- [255] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia Conference Papers*, 2023.
- [256] Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics (SIGGRAPH)*, 2024.

Appendix

A Extended related work

3D generation. An influential line of text-to-3D generation optimizes a 3D representation against a 2D diffusion prior [144, 108, 185], with related approaches using multi-view diffusion [117, 160, 120] and feed-forward reconstruction [65, 204]. Current systems train a diffusion or flow model directly on a native 3D latent, which may be a vector set [227, 245, 231, 103, 246, 173, 101] or a triplane [196], or a sparse or structured voxel grid [199, 200, 104, 212]. All 4 non-agentic methods we evaluate build on the structured latent of TRELIS [199]. A separate line generates low-polygon, artist-style meshes autoregressively, one face or one token at a time [133, 162, 26, 30, 29, 190, 186, 169, 191, 240, 111, 184, 165, 85, 60] or by flow matching in a mesh latent space [102, 175]. These methods target compact mesh structure rather than instruction-guided changes to an existing asset. TAR3D [235] autoregressively predicts quantized triplane tokens, while SeamFlow [244] predicts cuts for UV unwrapping on an existing mesh. Part-aware modeling includes segmentation [209, 116, 122], generative decomposition and completion [210, 232, 206, 100], and generation of objects as sets of parts [24, 110, 211, 170, 25, 223, 253, 106]. Assembler [241] places input parts into a complete object, while SIMART and PWM-ArtGen address articulated assets [228, 247]. Texture synthesis is a line of work of its own [153, 18, 225, 219, 158], including PBR material generation [163, 71, 63, 31, 47, 220, 202]. EditHero applies retexturing to one part while requiring the rest of the object to remain unchanged. CMD [99] supports progressive generation and local editing by conditioning on known parts. Its progressive examples illustrate construction, whereas EditHero supplies a benchmark of different instructions and exact reference states along each chain. EditHero uses the parts of an object both as the unit of editing and to build the exact ground truth.

Additional 3D editing methods. The methods we evaluate come from the 2 main families built on native 3D latents. Trained methods learn from paired edits. Much as ControlNet [233] conditions an image model, they feed the source latent to the generator through an added control branch. PartFlow [192] does this with a source-control branch, and 3DEditFormer [198] does it with dual-guidance attention [see also 14, 53, 221]. Training-free methods reuse pretrained generators without edit-specific training. Inversion-based methods such as VoxHammer [98] invert the source along the sampling trajectory and reuse its cached latents and attention keys and values in the preserved region. Inversion-free methods follow the difference between the source-conditioned and target-conditioned flows. Nano3D [215] does this with the FlowEdit formulation [89], and related variants follow the same idea [250, 135, 67]. Single-view regeneration from the target render, which we report as a reference, is the limiting case of regenerating the object from an edited image. Further approaches are discussed below. Early editing methods modify a neural field or a Gaussian splat with a 2D diffusion prior [61, 156, 255, 28, 178, 82, 256, 194, 22], and some of them take sketches [130, 114] or drag handles [148] as conditions instead of text. Recent methods also use mesh reconstruction or native 3D latents [23, 19, 146, 84, 7, 141, 250, 53, 66, 32, 21, 221, 214, 1, 142]. Other systems edit Gaussian scenes or scene atlases [161, 46]. Object insertion in 3D scenes and 3D object inpainting are also studied separately [96, 48]. Part-level and interactive methods let a user act on one part at a time [41, 42, 176, 157, 164], and sketch-based systems have supported iterative refinement since Teddy [77, 134, 94, 95, 59, 10, 248, 6, 97, 92, 93, 237]. Sketch2CAD [94] and MeshPad [97] support sequential sketch-based modeling. Their evaluations focus on sketch interpretation and interactive modeling, whereas EditHero evaluates a sequence of natural-language part edits against exact 3D references.

2D image editing and its evaluation. Instruction-based image editing developed from inversion, personalization and attention control [128, 5, 64, 51, 154, 132, 34, 81, 174, 15, 233, 11] to models trained on edit pairs [13, 159, 55, 50, 239, 75, 222, 189, 147] and to unified generators that edit in context [201, 238, 118, 90, 193, 39]. Geometry-aware image editing uses 3D cues for spatial transformations or object removal [9, 62, 254], and BlenderFusion [20] combines 3D scene manipulation with generative image compositing. Before diffusion, iterative editing was studied as human-in-the-loop optimization, in which Bayesian optimization searches the parameters of an image or an animation under human preference feedback [12, 56, 87, 88]. Recent work applies the same idea to generative image search or adapter merging [151, 112]. These systems study iterative improvement under human feedback. Benchmarks for image editing [181, 230, 8, 124, 217, 242, 197, 54]

established that “did it edit” and “did it leave the rest alone” must be scored separately, and that similarity to the source mixes up the requested change with drift. Multi-turn image evaluations use reference images or a VLM judge [230, 251, 125, 70, 226, 208, 27]. DiffSeg30k [33] instead uses multi-turn edits to benchmark localized detection of generated content. MagicBrush provides intermediate target images reviewed by human annotators [230]. EditHero instead supplies complete 3D reference states that can be rebuilt deterministically, with the edited parts known at every turn. Multi-turn methods respond with memory of earlier turns or reinforcement learning over the sequence [229, 203, 218]. Repeated image replication can accumulate artifacts that no-reference quality metrics fail to capture [171]. Related work studies different sources of accumulated error: repeated training on synthetic outputs [2, 236] and exposure bias within diffusion sampling [136]. We study successive 3D edits at inference time, where holes, loose pieces and displacement can be computed directly.

Additional LLM/VLM and agentic methods. LLMs can build 3D content by writing procedural code or calling tools. They do this for scenes and layouts [166, 36, 68, 207, 49, 168] and for editing an existing scene in Blender from renders [69, 121, 177, 140]. Related code-based systems generate objects and CAD programs [224, 80, 35, 83, 155, 3, 127, 91]. Recent systems treat the program as the asset itself, with an agent that plans, writes and revises procedural code under visual feedback [109, 180, 138, 113, 137, 4]. EmbodiedGen V2 [183] describes this as vibe coding of 3D worlds through a dialogue of instructions, aiming to make local edits while preserving the rest of the state. EditHero measures this workflow. Agentic pipelines exist for images as well [187, 119, 17, 205, 123, 216, 243, 105, 195, 74, 72]. Related VLM-guided editing models and unified generators use learned multimodal reasoning [73, 45, 107]. Benchmarks for these agents [58, 52] score the final state against one goal.

Ground truth by construction. Part-annotated collections [16, 131, 179, 40] and procedural asset generators [249, 150] provide objects and parts for assembly. EditHero builds its chains with a program, like synthetic benchmarks whose targets are produced by a program [79, 57, 37] and unlike annotated datasets. The assembly engine that produces each state is deterministic, and rebuilding a turn from its logged operations reproduces the state exactly. The engine does not guarantee quality, so every chain went through manual review (Appendix B).

Comparison with existing benchmarks. 3DEditVerse [198] combines pose changes with image-guided appearance edits, while Nano3D-Edit-100k [215] uses a native 3D editing method guided by edited images. Uni3DEdit-Bench [192] uses semantic-part transformations, and Delta3D [145] removes parts from CAD assemblies. Steer3D’s EDIT3D-BENCH [126] and the editing subset of 3D-Alpaca [213] reconstruct edited images into target assets. Edit3D-Bench [98], Eval3DEdit [250], BenchUp [53] and EditFlow-Bench [135] have no target asset, so they score region preservation and alignment to an edited image instead. Scene-level benchmarks follow the same pattern [143, 115, 252]. Each sample specifies a single editing goal, represented by an asset or an edited image. BlenderGym [58] and 3DCodeBench [52] let an agent iterate toward one fixed goal, and EZBlender [177] bundles several edits into one prompt. Table 4 places EditHero next to these benchmarks. The table compares target construction, instruction sequences, intermediate references and evaluation on each method’s own outputs.

B Benchmark construction details

Why assemble the targets? Assembling a target needs a suitable part and a plausible placement. The part has to make sense on the object (for example, a hat on a head or a barrel on a porch) and be attached where a person would attach it. Accepted placements must also satisfy explicit geometric constraints and stay reproducible after a repair. We therefore combine deterministic placement checks with manual review of every chain.

Hosts and slots. Our primary source library is PartVerse-XL, released with FullPart [40], whose underlying objects come from Objaverse-XL [38]. We use its textured part meshes and supplied part decomposition. For retrieval, we generate part descriptions with Qwen3-VL-32B-Instruct [149]. These descriptions are separate

Table 4 EditHero compared with existing editing benchmarks. *Target*: how the reference state is obtained (2D lift: an edited image reconstructed as a 3D target; *native edit*: a native 3D editing method guided by an edited image; *none*: no target asset, scored against an edited image). *Turns*: 1, one instruction per sample; *goal*, one target task, the agent may iterate; *bundle*, several edits given in one prompt; *sequence*, different instructions applied in order. *Ref.*: a reference is supplied after each instruction in a multi-turn sequence; a dash also covers single-turn or single-goal settings where this property does not apply. *Own*: turn k is applied to the method’s output of turn $k - 1$.

Benchmark	Domain	Target	Turns	Ref.	Own
MagicBrush [230]	2D image	annotated image	sequence (few)	✓	✓
3DEditVerse [198]	3D	pose edits / 2D lift	1	–	–
Nano3D-Edit-100k [215]	3D	native edit	1	–	–
Uni3DEdit-Bench [192]	3D	semantic-part edits	1	–	–
EDIT3D-BENCH (Steer3D) [126]	3D	2D lift	1	–	–
Delta3D [145]	3D	part removal	1	–	–
3D-Alpaca [213]	3D	2D lift	1	–	–
Edit3D-Bench [98], Eval3DEdit [250]	3D	none	1	–	–
BenchUp [53], EditFlow-Bench [135]	3D	none	1	–	–
BlenderGym [58], 3DCodeBench [52]	3D scene, code	goal state	goal	–	✓
EZBlender [177]	3D scene	–	bundle	–	–
EditHero (ours)	3D, part-level	program-built state	sequence (3–30)	✓	✓

from the editing instructions that people later review. Each chain starts from a textured, part-segmented object, which we call the *host*. Its parts are grouped into a small number of *slots*. A slot is a sub-assembly, such as a head, an arm or a porch canopy, that an instruction can name and that is large enough to see in a render. Slots are proposed automatically from the part contact graph and the part captions, and then accepted, merged or renamed by a reviewer looking at highlighted renders. Reviewers also correct the proposed slot names so they describe the grouped parts clearly.

Operations. *Add* places a new part from the library on a free face of an existing part, *remove* deletes the part or part group occupying a named slot, *replace* swaps the part in a slot for a retrieved one on the same contact face, and *retexture* repaints a slot and leaves its geometry exactly as it was. The new texture comes from a render of the slot restyled by Qwen-Image-Edit [193] and is applied to the unchanged slot geometry by the texturing module of TRELIS.2 [200]. Before a support part is replaced, its attachments are removed in separate turns, and they are reattached in separate turns as well. Each step thus stays an explicit operation on a named part or part group. Instructions use natural-language part descriptions and never contain internal slot identifiers or library metadata.

Placement. Placement is deterministic, as in procedural datasets whose labels come from the generating program [79, 57, 150]. For a replacement or an add-on part, the engine retrieves candidates by the similarity of their caption embeddings to the requested part [182, 152]. It computes an initial pose from the host’s contact face (a planar, axial or point mate, depending on the slot), scales the part so that its mount matches the face (between 0.75 and 2 times its original size) and snaps it into contact. Automatic candidate screening then applies 4 checks. A contact frame must exist, the remaining gap must be below 0.01 of the object diagonal, the object must stay one connected piece under that tolerance, and the interpenetration fraction must stay below 0.02. The automatic search rejects candidates that fail a check and tries the next one. Human review then assesses whether the assembly is visually appropriate, including intentional overlaps or disconnected components in stylized hosts.

Chain recipe. Each chain is stored as a JSON file that names the host and has one entry per turn. An entry gives the slot, the library part for an addition or a replacement, the accepted pose and, for a retexture, the generated texture. Slots that a turn does not name stay as they were. With the part library, the file rebuilds

any state $S_k = o_k(\dots o_1(S_0))$ on its own, without repeating retrieval or texture generation. A repair edits the entry of one turn, and that turn and the later ones are rebuilt and checked again.

Human review and repair. Every turn was rendered from several views before and after the edit and inspected. Reviewers flagged wrong part semantics (a foot placed as a hand), wrong orientation or mounting, implausible scale, instructions that gave away the source or were ambiguous, and dependency errors (an attachment silently carried along when its parent was replaced). Reviewers corrected poses in the operation log, saved revised geometry or materials as separate assets when needed, and rebuilt the affected turns. Original source assets were retained. Turns that could not be fixed were cut, and the rest of the chain was rebuilt from that point.

Families. Chains are grouped by the operations they contain. Family *A* holds mostly additions. Families *R* and *P* contain only removals and replacements, respectively. Family *M* contains only material edits. Family *G* mixes geometry operations, and family *X* mixes geometry and material operations. The benchmark contains 141 *A*, 72 *M*, 45 *R*, 54 *P*, 85 *G* and 60 *X* chains. Taking the 6 families together, we try to keep the share of each operation stable and even from turn to turn. Figure 3(c) counts turns 1 to 9 one by one and puts the few edits from turn 10 onward into one bar.

C Evaluation protocol details

Self-rollout. We give each method the initial mesh and, at turn k , the instruction together with the conditions it needs, such as a target image or a mask. Its input at turn k is its own output at turn $k - 1$. The main evaluation never resets the state to the ground truth, and Appendix H reports a separate diagnostic that does. If a method fails a turn, for example with an empty or unloadable output, the rest of that chain is blocked and the chain counts as a failure. This rule is the same for the non-agentic methods and the LLM/VLM agents, and we never restart from the ground truth to rescue a chain. Infrastructure faults are handled separately (Appendix F).

Conditioning. Every method receives the same target image at a given turn, rendered with the chain’s fixed conditioning camera, and VoxHammer also receives an edit mask. The target image shows what the object should look like after the edit.

A shared coordinate frame and fixed views. We measure all states of a chain, ground truth and outputs alike, in one frame normalized by the bounding box of all its ground-truth states. We render them with one camera rig fitted to the bounding sphere of those states. The rig has 4 views 90° apart at one elevation, a fixed focal length and fixed lighting. If we re-centered every turn or every part, a part that moved would look as if it had stayed in place, and a new camera at every turn would make image metrics incomparable across turns. One default direction for all chains, however, can hide the edited part, since a small addition on the far side of the object barely shows in the conditioning view. We therefore choose the direction per chain. For every turn, the edited region is the set of part nodes whose geometry or material differs between the previous and the current ground-truth state. We fix for the chain the direction that makes its worst turn most visible (Appendix E). To estimate visibility, we cast rays from surface samples of that region toward each candidate direction against the full state. Because the rig is fitted to all states of the chain, its framing reflects how large the object will become. No method receives the rig as geometry, and every method still sees the object in its own frame, fixed at the initial state.

Why these definitions. The definitions follow the design of Section 4, and Figure 6 shows examples of the problems they avoid.

Score the edit and the rest separately. A turn changes only a small part of an object. On the whole object, returning the initial object can therefore score higher than doing the requested edit on a state that has already drifted (Figure 6b). We therefore split every metric into E_k , U_k and the whole object. We take the regions from the named parts rather than from voxel or pixel differences, because a retexture leaves the ground-truth geometry unchanged even when a method changes it (Figure 6d). Using the named parts in

both the old and the new state covers all 4 operations with one definition. The one-cell dilation absorbs a part that lands one cell off, so a small misplacement lowers IF but is not counted a second time as damage in CC.

Measure each turn against the method’s own previous output. Suppose a chain removes a hat at turn 2 and adds it back at turn 5. If we took the required addition from the ground truth, that is, the voxels present at turn 5 and absent at turn 4, the no-op output would already contain the hat and get full credit, although it never did anything. We instead take the required addition from the method’s own previous output. The no-op output then needs nothing and gets no credit, and a method that did remove the hat gets credit only when it puts the hat back. For the same reason, a part lost at an earlier turn does not count as a successful removal now (Figure 6a). IF also catches a replacement that only deletes the old part. Its IF is close to 0 because the new part is missing, while its F-score to the target is within 0.02 of the no-op baseline (Figure 6c). CC^{prev} uses the same anchor and asks whether this turn disturbed what it should keep. CC^0 compares with the initial object and shows how far the untouched content has drifted over the whole chain.

Read every score next to the references. The no-op baseline has $IF = 0$ and $CC = 1$, and the regeneration baseline has a high IF and a low CC. A good editor should get close to both at once. We still report the target similarity $m_{E_k}(\hat{S}_k, S_k)$, because prior work uses it. Read next to IF, it separates looking like the target from actually making the change at this turn.

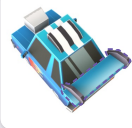
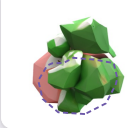
Ground truth		Model, self-rollout		Target-based scores	EditHero
S_{k-1}	S_k	$\hat{S}_{k-1}$	$\hat{S}_k$		
a Remove the ammunition crate.					
				✗ removal 1.00 credits a removal never made	✓ IF⁻ not scored the crate was gone two turns earlier
b Remove the wooden crate.					
				✗ IoU 0.39 below the no-op baseline (0.97)	✓ IF 1.00 CC^{prev} 0.86 crate removed; some drift remains
c Replace the car’s rear wing with a fox tail.					
				✗ F 0.87 deleting the wing ≈ no-op (0.88)	✓ IF 0.03 fox tail missing (IF ⁺ 0.02)
d Make the largest leaf plate glossy red.					
				✗ edit region empty a retexture changes no voxel	✓ CC^{prev} 0.05 the whole object was regenerated

Figure 6 Why the metrics are defined on regions and on the model’s own state. Four turns from the evaluation, in the conditioning view. Outlines mark the edited parts before and after the turn, dashed on the model’s own states. For each turn, the left card shows how a target-similarity or ground-truth-based score can mislead, and the right card shows the corresponding IF or CC score (Eqs. (1)–(5)). The tag on the right names the design choice the turn motivates.

D Metric implementation

3D operators. For object states X (output) and Y (reference), $\mathcal{V}(\cdot)$ is the surface voxelization into $\mathcal{G}$, $\mathcal{Q}(\cdot)$ is a dense set of surface samples, and $d(p, Y) = \min_{q \in \mathcal{Q}(Y)} \|p - q\|$ is the distance from a point to the reference surface (searched over all of Y , not only inside R). A sample belongs to R when its voxel does.

$$\text{IoU}_R(X, Y) = \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|(\mathcal{V}(X) \cup \mathcal{V}(Y)) \cap R|}, \quad (6)$$

$$\text{Recall}_R(X, Y) = \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|\mathcal{V}(Y) \cap R|}.$$

$$\text{CD}_R(X, Y) = \frac{1}{2} \left(\text{mean}_{p \in \mathcal{Q}(X) \cap R} d(p, Y) + \text{mean}_{q \in \mathcal{Q}(Y) \cap R} d(q, X) \right). \quad (7)$$

$$\text{F}_R^\tau(X, Y) = \frac{2 \text{Pr Re}}{\text{Pr} + \text{Re}},$$

$$\text{Pr} = \text{mean}_{p \in \mathcal{Q}(X) \cap R} \mathbb{I}[d(p, Y) < \tau], \quad (8)$$

$$\text{Re} = \text{mean}_{q \in \mathcal{Q}(Y) \cap R} \mathbb{I}[d(q, X) < \tau].$$

2D operators. For renders I (output) and J (reference) in the same fixed view, with RGB values in $[0, 1]$,

$$\text{MAE}_R(I, J) = \text{mean}_{p \in R} \frac{1}{3} \|I(p) - J(p)\|_1, \quad (9)$$

$$\text{PSNR}_R(I, J) = -10 \log_{10} \left(\text{mean}_{p \in R} \frac{1}{3} \|I(p) - J(p)\|_2^2 \right),$$

and $\text{SSIM}_R(I, J)$, $\text{LPIPS}_R(I, J)$ are the SSIM [188] and LPIPS [234] maps of (I, J) averaged over the pixels of R . $\text{DINO}_R(I, J)$ is the cosine similarity of the DINOv2 patch features of I and J , averaged over the patches whose center lies in R .

Constants. Voxelization uses the 64^3 grid of the chain frame. Chamfer distance and F-score use 20,000 surface samples per mesh, and a sample belongs to a region when its voxel does. Nearest neighbors are searched over the whole reference, distances are in units of the chain side, and the F-score threshold is $\tau = 0.01$. IF^+ (IF^-) is defined only when A_k (D_k) has at least 8 voxels and at least a quarter of the ground-truth addition (removal) $\mathcal{V}(S_k) \setminus \mathcal{V}(S_{k-1})$ ($\mathcal{V}(S_{k-1}) \setminus \mathcal{V}(S_k)$). Otherwise that turn has no IF. Defined-turn counts for the 2 main comparisons are in Table 6. Image metrics use the 4 fixed views, composited on black, with the LPIPS map of AlexNet [234], the SSIM map [188] and the patch features of DINOv2 ViT-B/14 [139] (a 16×16 patch grid), each averaged over the region. Target-similarity metrics compare with the ground truth, and image drift compares consecutive outputs.

E Camera search

For each chain and each candidate camera direction, we sample points on the edited region of every turn and cast rays toward the camera against the full state. The visible projected area, weighted by the cosine to the view direction and corrected for depth, is divided by the frame area at the object center. The chosen direction is the one whose worst turn is most visible, with a small preference for the default direction. The estimate correlates with the measured pixel change of the rendered views at Spearman $\rho = 0.95$.

F Experimental setup and coverage

We evaluate PartFlow [192], Nano3D [215], 3DEditFormer [198], and VoxHammer [98] with their released code and weights. Each method is wrapped in an adapter that only supplies inputs and records outputs. The sampling code, seeds and hyperparameters are the authors’ own. PartFlow, Nano3D and 3DEditFormer take the current state and the target render. VoxHammer also takes the ground-truth edit mask. Nano3D’s official interface has add, remove and replace modes but no material mode. With the official routing, Nano3D rejects a material instruction and the rest of the chain is blocked, which would happen on all 132 chains that contain one. We therefore route every material instruction to the replace mode. The adapter passes the unchanged instruction text and the target render exactly as for a replace turn, and the output is scored like any other turn. The Nano3D rows use this routing on those 132 chains and the official routing on the other chains. All methods are scored with the same fixed-camera protocol of Section 4. The meshes of the 4 methods, exported through TRELIS, carry no metallic value and would render as dark metal, so we render them as non-metallic. The agents’ outputs of Section 5.4 set their own materials and are rendered as exported.

The TRELIS regeneration baseline runs the released TRELIS-image-large pipeline [199] on the target render of every turn, in the same fixed view that the non-agentic methods receive. It sees neither the instruction nor its previous output, so its turns are independent. Its output lies in the normalized frame of TRELIS. One view does not fix the orientation in that frame, so on some turns the object comes out turned by a multiple of 90° . We therefore align every turn to the ground-truth state of the same turn with a yaw search followed by a similarity ICP. The ground-truth geometry is used only for this alignment. It completes all 2755 turns of the 457 chains.

Table 5 reports, per method, the chains completed, the completed turns, the failed or subsequently blocked turns, and the failure categories. Infrastructure faults, such as a time-out or a lost node, were handled by rerunning the whole chain from its first turn, and the decision to rerun never depended on the outputs. These faults do not appear in the table. The remaining failures occurred during model execution. VoxHammer produces an empty sparse structure on 39 chains, and Nano3D writes no edited voxels on 2 chains. Nano3D would also fail on every material instruction without this adaptation, but with the replace routing above it completes all chains except those 2. We keep these as failures rather than drop the chains, because a method that cannot run a turn has not completed the chain.

Table 5 Execution coverage on the 457 chains. Completion means the chain ran to the end, not that the edits were correct. Conditions: *image* = current state and target render; + *mask* = also the ground-truth edit mask. The Turns column reports completed turns out of all scheduled turns. Failed / blocked includes the failure turn and later turns that could not run.

Method	Cond.	Chains	Turns	Failed / blocked	Failure category
PartFlow	image	457/457	2755/2755	0	–
Nano3D	image	455/457	2749/2755	6	no edited voxels (2)
3DEditFormer	image	457/457	2755/2755	0	–
VoxHammer	image + mask	418/457	2569/2755	186	empty sparse output (39)

G Whole-object metrics, per-turn curves and end state

Table 7 reports whole-object and preservation metrics in 2 groups. The first group uses the chains that all 4 non-agentic methods complete, so every method is scored on the same turns. The second uses all completed turns of the methods that complete almost every chain. The metric means in the 2 groups agree to within 0.015. The no-op baseline has a higher surface IoU to the target than every method (0.71 against at most 0.58 on these chains) and a lower LPIPS to the target render. Nano3D has the highest IoU and preservation of the 4, and its IF is close to VoxHammer’s. VoxHammer gets its IF while keeping only 0.45 of the untouched

Table 6 Defined-turn counts for IF, by operation and comparison set. Retexture turns have no geometry IF and are excluded. These counts differ from execution coverage.

Method	All defined	Add	Remove	Replace
<i>Main evaluation (457 chains)</i>				
PartFlow	1880	960	416	504
Nano3D	1929	961	438	530
3DEditFormer	1895	966	414	515
VoxHammer	1750	894	375	481
No-op baseline	1815	946	386	483
TRELLIS regen.	1798	967	352	479
<i>Agentic/non-agentic comparison (55 chains)</i>				
PartFlow	270	125	55	90
Nano3D	274	125	57	92
3DEditFormer	278	125	56	97
VoxHammer	248	122	50	76
No-op baseline	243	119	46	78
TRELLIS regen.	259	125	51	83
GLM 5.3 Flash	236	125	42	69
DeepSeek V4.1 Flash	268	125	54	89
Sol 6 (medium)	269	125	57	87
Astra (medium)	288	125	65	98
Fable 5.1	286	125	60	101
Opus 5.5	300	125	71	104

initial object (CC^0). The TRELLIS regeneration baseline, which builds every turn from the target render alone, sits at the other extreme. It has the highest IF on these chains (0.51) but also the lowest IoU (0.30) and the lowest preservation (CC^{prev} 0.39).

Figure 7 and Table 8 follow the metrics turn by turn on the 416 chains completed by all 4 non-agentic methods, with 95% intervals from a bootstrap [43] over hosts (chains built on the same host are not independent). Target similarity drops as the chain goes on. The no-op baseline also declines as the targets move away from the initial object, but it stays above all 4 methods. The TRELLIS regeneration baseline, whose turns are independent, stays almost flat (IoU 0.31 at turn 1 and 0.27 at turn 7). These curves alone cannot tell whether later targets are harder or the errors are inherited, and Appendix H tests this. After turn 7 the set shrinks to 80 chains at turn 8 and we do not draw conclusions from it.

End state. The per-turn curves stop at turn 7 and average over chains of different lengths, so they do not show where a chain ends up. Table 9 scores the last turn of every chain of this set, grouped by chain length, next to the first turn of the same chains. All 4 methods lose similarity and preservation from the first to the last turn in every length group, and they lose more on the longest chains. The IoU of PartFlow falls by 0.10 on chains of 3 turns and by 0.20 on chains of 11 turns or more. The CC^0 of 3DEditFormer falls by 0.06 and 0.41 on the same groups, and that of VoxHammer by 0.11 and 0.39. The IoU of the no-op baseline falls with length too, from 0.83–0.88 at the first turn to 0.61–0.78 at the end, because the target moves away from the initial object. End-state IoU should be read against this baseline, and none of the 4 non-agentic methods beats it on mean end-state IoU in any length group. The TRELLIS regeneration baseline, which never carries a state over, has an IoU of roughly 0.3 at both the first and last turns.

H Own state versus ground-truth state

The decline under self-rollout has 2 possible causes. Later instructions may be harder, or errors in the previous output may make them harder to execute. To tell them apart we rerun Nano3D and 3DEditFormer

Table 7 Whole-object metrics of the self-rollout runs. Upper block: the 416 chains that all 4 non-agentic methods complete, so all rows share the same turns. Lower block: all completed turns of the 457 chains, for the methods with full or near-full coverage. Bold: best non-agentic method in the upper block.

Method	n	IoU $\uparrow$	IF $\uparrow$	CC $^{\text{prev}}\uparrow$	CC $^0\uparrow$	LPIPS $\downarrow$	drift $\downarrow$
<i>416 chains completed by every non-agentic method</i>							
PartFlow	2472	0.516	0.126	0.862	0.674	0.177	0.014
Nano3D	2472	0.584	0.317	0.887	0.730	0.160	0.009
3DEditFormer	2472	0.414	0.202	0.774	0.490	0.198	0.018
VoxHammer	2472	0.380	0.322	0.714	0.452	0.220	0.022
No-op baseline	2472	0.712	0.000	1.000	1.000	0.144	0.000
TRELLIS regen.	2472	0.296	0.508	0.385	0.282	0.202	0.041
<i>All completed turns (methods with full or near-full coverage)</i>							
PartFlow	2755	0.507	0.122	0.862	0.666	0.181	0.014
Nano3D	2749	0.575	0.312	0.887	0.718	0.163	0.009
3DEditFormer	2755	0.408	0.197	0.772	0.481	0.203	0.018
No-op baseline	2755	0.700	0.000	1.000	1.000	0.151	0.000

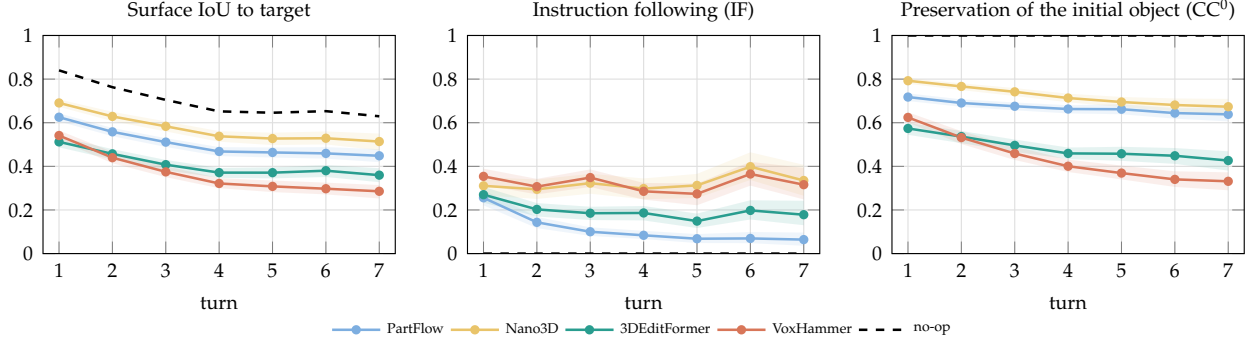


Figure 7 Per-turn metrics on the 416 chains that all 4 non-agentic methods complete (228 hosts). Shaded: 95% host-clustered bootstrap intervals. The dashed line is the no-op baseline, which has IF = 0 and CC⁰ = 1 by definition. Its IoU compares the unchanged initial object with the current target.

on 200 chains and change one thing. The input state at turn k is the ground-truth state S_{k-1} instead of the model’s own $\hat{S}_{k-1}$, with the same instruction, target render, camera and seed. The ground-truth states are encoded into each method’s latent space with the same pipeline that encodes its initial object. Under this condition the turns are independent. Table 10 pairs the 2 input conditions on the same chains at each turn. Figure 4(c) instead uses a fixed subset of 147 chains with at least 5 turns, so its means differ slightly from those in the table.

In this diagnostic, much of the decline comes from the inherited state. Under self-rollout, IoU to the target falls from turn 1 to turn 5 (0.68 to 0.52 for Nano3D and 0.51 to 0.36 for 3DEditFormer). With the ground-truth input it stays at 0.68 for Nano3D and falls only to 0.46 for 3DEditFormer. Instruction following is higher with the ground-truth input at every turn after the first, by 0.06 to 0.08 for Nano3D and by 0.08 to 0.19 for 3DEditFormer (95% host-clustered intervals exclude zero at turns 2 to 5). For this diagnostic, CC $_{\Delta}^{\text{prev}}$ measures preservation outside the one-cell-dilated voxel difference between the 2 ground-truth states, rather than the part-based region of Eq. (5). Preservation of the previous state moves the other way. Over turns 2 to 7, for Nano3D it is 0.90–0.94 under self-rollout and 0.71–0.75 with the ground-truth input, and for 3DEditFormer it is 0.80–0.83 against 0.46–0.56. In other words, a method keeps its own output well but disturbs a state it did not produce, as it also does at turn 1. A high CC $^{\text{prev}}$ under self-rollout only means that the method keeps its own state, which can already be wrong.

Table 8 Per-turn means within the matched cohort (turns 1–7), using chains that reach each turn and excluding undefined scores. TRELLIS regen.: the regeneration baseline, whose turns are independent.

Turn <i>n</i> chains	1 416	2 416	3 416	4 375	5 302	6 233	7 135
<i>IoU</i>							
PartFlow	0.625	0.558	0.511	0.468	0.464	0.459	0.448
Nano3D	0.691	0.629	0.583	0.538	0.527	0.529	0.514
3DEditFormer	0.512	0.457	0.408	0.371	0.371	0.380	0.360
VoxHammer	0.541	0.440	0.375	0.321	0.308	0.297	0.286
No-op baseline	0.840	0.763	0.705	0.652	0.646	0.653	0.630
TRELLIS regen.	0.306	0.300	0.299	0.294	0.292	0.285	0.274
<i>IF</i>							
PartFlow	0.256	0.143	0.100	0.084	0.068	0.070	0.064
Nano3D	0.311	0.295	0.323	0.298	0.313	0.398	0.335
3DEditFormer	0.270	0.203	0.185	0.186	0.149	0.198	0.178
VoxHammer	0.354	0.307	0.349	0.285	0.274	0.365	0.316
No-op baseline	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TRELLIS regen.	0.536	0.472	0.519	0.494	0.484	0.534	0.535
<i>CC⁰</i>							
PartFlow	0.718	0.690	0.676	0.663	0.662	0.644	0.638
Nano3D	0.793	0.766	0.742	0.713	0.695	0.681	0.673
3DEditFormer	0.574	0.536	0.496	0.459	0.458	0.448	0.427
VoxHammer	0.624	0.532	0.459	0.400	0.369	0.340	0.331
No-op baseline	1.000	1.000	1.000	1.000	1.000	1.000	1.000
TRELLIS regen.	0.303	0.294	0.292	0.277	0.275	0.257	0.248

I Material edits

Geometry metrics do not register material edits, so we measure them in images. For every retexture turn we take the pixels where the ground-truth render changes. This changed-pixel diagnostic uses a narrower region than the part-based definition in Section 4. We then measure how much each method’s own consecutive renders change there, relative to the ground-truth change (Table 11). PartFlow, 3DEditFormer and VoxHammer produce 0.68–0.98 times the ground-truth image-change magnitude and bring the result closer to the target. Nano3D, whose material turns run in the replace mode, changes the region by 0.31 times the ground-truth image change, and its distance to the target is close to that of its own unedited previous output. It completes all chains containing retexture turns and has the highest whole-object IoU on those chains (0.67). This does not show material accuracy, since retexturing leaves the ground-truth geometry unchanged. Figure 9 shows what the other methods do instead. They change the appearance of untouched parts too, so their errors spread over the whole object instead of staying on the named part. Their colors also shift away from the target.

J LLM/VLM setup and cost

Setup. Opus 5.5 and Fable 5.1 run in Claude Code, Astra and Sol 6 run as Codex subagents, and GLM 5.3 Flash and DeepSeek V4.1 Flash run through OpenRouter in our API tool loop, which dispatches the tools a model requests and returns their results to the conversation. We evaluate each model with the tools provided by its runtime. Each chain is edited by its own agent, which starts a fresh session and shares no memory with other chains. Each agent receives the source mesh, the chain’s instructions and target renders, and the camera parameters. It edits its own previous output and saves a full mesh after every turn. All models use the same base editing prompt (Appendix P) and our adaptation of img2threejs v2.0.0 [78], which

Table 9 End-state evaluation on the same chains, by chain length: IoU to the target and CC^0 at the first turn $\rightarrow$ at the last turn of the chain (means over the chains in the group; the first and last groups are small).

Chain length	3 turns (41 chains)		4–6 turns (240 chains)		7–10 turns (124 chains)		11–30 turns (11 chains)	
Method	IoU		IoU		IoU		IoU	
	CC^0		CC^0		CC^0		CC^0	
PartFlow	0.67 $\rightarrow$ 0.57	0.71 $\rightarrow$ 0.69	0.61 $\rightarrow$ 0.43	0.72 $\rightarrow$ 0.63	0.63 $\rightarrow$ 0.43	0.72 $\rightarrow$ 0.62	0.67 $\rightarrow$ 0.47	0.72 $\rightarrow$ 0.57
Nano3D	0.74 $\rightarrow$ 0.68	0.79 $\rightarrow$ 0.76	0.68 $\rightarrow$ 0.50	0.80 $\rightarrow$ 0.68	0.69 $\rightarrow$ 0.50	0.78 $\rightarrow$ 0.64	0.75 $\rightarrow$ 0.50	0.81 $\rightarrow$ 0.58
3DEditFormer	0.55 $\rightarrow$ 0.46	0.59 $\rightarrow$ 0.53	0.49 $\rightarrow$ 0.33	0.55 $\rightarrow$ 0.40	0.54 $\rightarrow$ 0.36	0.60 $\rightarrow$ 0.42	0.58 $\rightarrow$ 0.24	0.65 $\rightarrow$ 0.24
VoxHammer	0.58 $\rightarrow$ 0.47	0.62 $\rightarrow$ 0.51	0.54 $\rightarrow$ 0.28	0.63 $\rightarrow$ 0.35	0.54 $\rightarrow$ 0.29	0.61 $\rightarrow$ 0.33	0.57 $\rightarrow$ 0.22	0.61 $\rightarrow$ 0.22
No-op baseline	0.88 $\rightarrow$ 0.78	1.00 $\rightarrow$ 1.00	0.83 $\rightarrow$ 0.61	1.00 $\rightarrow$ 1.00	0.85 $\rightarrow$ 0.62	1.00 $\rightarrow$ 1.00	0.86 $\rightarrow$ 0.64	1.00 $\rightarrow$ 1.00
TRELLIS regen.	0.29 $\rightarrow$ 0.31	0.28 $\rightarrow$ 0.30	0.31 $\rightarrow$ 0.30	0.30 $\rightarrow$ 0.28	0.31 $\rightarrow$ 0.28	0.32 $\rightarrow$ 0.25	0.28 $\rightarrow$ 0.28	0.28 $\rightarrow$ 0.21

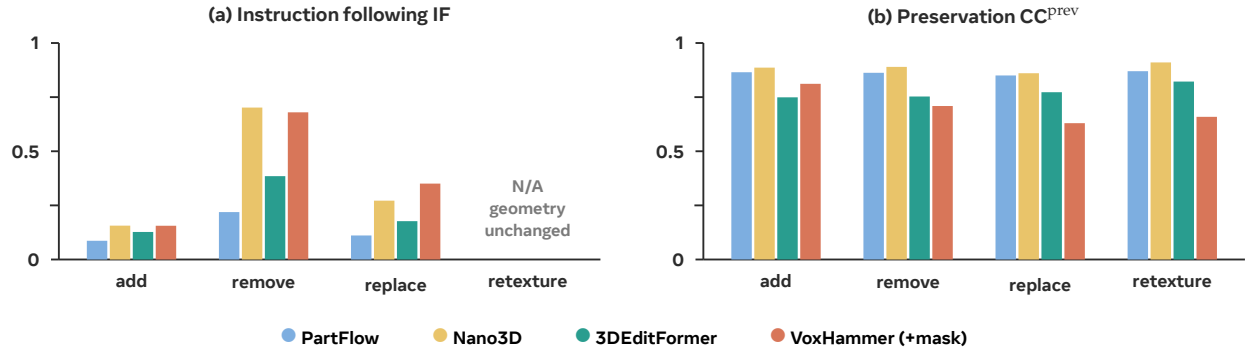


Figure 8 Metrics by operation. Means over completed turns with defined scores. Retexture IF is undefined (N/A, not zero). Coverage and conditioning differ by method, so the methods are not compared on the same turns here.

retains staged construction and render-and-compare refinement while editing an existing mesh and using Blender for rendering. New parts are built in code, removals delete the named part, and retextures change its material while keeping its geometry. External assets, hidden ground-truth meshes, and generative 3D or image models are prohibited.

Budgets. Opus 5.5 and Fable 5.1 were asked to finish a chain in about 3 hours. GLM 5.3 Flash and DeepSeek V4.1 Flash had 200 model responses and 3 hours per chain, followed by a completion pass without a cap. Astra and Sol 6 use medium reasoning effort without a time cap. We report the cost they actually used.

Coverage and failures. A turn counts as delivered when its mesh exists, is not empty and loads. Otherwise the turn fails and the rest of the chain is blocked, as for the non-agentic methods.

Cost. Table 12 reports the amortized cost per edit for all 6 LLMs. We divide each chain’s recorded time, tokens and model calls by its number of delivered turns, then take the median across chains. Opus 5.5 uses 149 s per edit (90th percentile 291 s) and 10.9 model calls, most of whose input is cached context.

K Extended metrics and held-out views

Tables 13 and 14 add metrics that the main tables leave out, computed on the same runs. For geometry we report the symmetric Chamfer distance [44] and the 95th percentile of the two-sided nearest-neighbor distance (HD95 [76, 167]), which is more sensitive to large surface errors than the mean distance. We also add normal consistency (NC [129]) and the F-score at 2 thresholds [86, 172]. The same HD95 and NC are also restricted to an untouched region (subscript u), which here lies outside the dilated voxel difference $\mathcal{V}(S_0) \triangle \mathcal{V}(S_k)$ between the initial and the target state. This region can include parts that were edited and

Table 10 Self-rollout versus ground-truth input, paired per turn. Turn 1 has the same input in both conditions. *self-rollout*: the model consumes its own previous output; *ground-truth input*: it consumes the ground-truth state. Turn k averages the chains with at least k turns (200 at turns 1–3, 181 at turn 4, 147 at turn 5, 115 at turn 6 and 65 at turn 7), the same chains in both conditions.

Metric	State	1	2	3	4	5	6	7
<i>Nano3D</i>								
IoU	self-rollout	0.684	0.619	0.575	0.529	0.518	0.525	0.534
	ground-truth input	0.684	0.694	0.676	0.666	0.681	0.694	0.725
IF	self-rollout	0.335	0.320	0.306	0.291	0.277	0.365	0.387
	ground-truth input	0.337	0.378	0.364	0.356	0.337	0.424	0.466
$CC_{\Delta}^{\text{prev}}$	self-rollout	0.790	0.903	0.906	0.896	0.899	0.900	0.936
	ground-truth input	0.789	0.750	0.730	0.714	0.727	0.739	0.751
<i>3DEditFormer</i>								
IoU	self-rollout	0.506	0.446	0.405	0.367	0.364	0.383	0.380
	ground-truth input	0.506	0.482	0.471	0.431	0.460	0.502	0.545
IF	self-rollout	0.282	0.197	0.201	0.172	0.152	0.184	0.185
	ground-truth input	0.284	0.289	0.279	0.316	0.321	0.361	0.379
$CC_{\Delta}^{\text{prev}}$	self-rollout	0.561	0.799	0.810	0.813	0.826	0.821	0.834
	ground-truth input	0.560	0.515	0.502	0.456	0.486	0.531	0.564

Table 11 Change inside the ground-truth changed-pixel region on retexture turns. Views count turn $\times$ view pairs. A view is skipped when the ground-truth change covers fewer than 50 pixels, and VoxHammer has fewer views because some of its chains stop early. Own change / GT change compares a method’s consecutive renders with the ground-truth change. Residuals are mean absolute error to the target render inside the region, for the method’s render (*method*) and for its previous output (*no edit*).

Method	Views	Own change / GT change	Residual to target	
			method	no edit
PartFlow	2563	0.68	0.187	0.213
Nano3D	2563	0.31	0.209	0.210
3DEditFormer	2563	0.80	0.186	0.219
VoxHammer	2384	0.98	0.216	0.225

later restored, whereas the cumulative part-based region of Eq. (5) excludes them. The last geometry column ($\Delta\text{frag.}$) is the median difference in the number of small connected components (below 0.5% of surface area) between the prediction and ground truth. Negative values mean fewer such components, which is not always a better reconstruction. For images we report PSNR, LPIPS [234], silhouette IoU from the alpha channel and DINOv2 ViT-B/14 cosine similarity [139] to the target render. PSNR, LPIPS and silhouette IoU are given separately for the conditioning view (subscript 0), which every image-conditioned method sees, and for the 3 held-out views 90° apart (subscript 1-3), which are not provided as target images. $\text{DINO}_{\text{retex}}$ is the similarity in the conditioning view on retexture turns only. All numbers are means over completed turns, except $\Delta\text{frag.}$, which is a median.

The 4 non-agentic methods show little average gap between conditioning and held-out views (PSNR within 0.5 dB, LPIPS within 0.01). In the untouched region the no-op baseline has an HD95 of 0.009, and the 4 methods have values 3 to 8 times as large. On retexture turns, DINO similarity is close to the no-op baseline (0.851) for PartFlow (0.854), Nano3D (0.843) and 3DEditFormer (0.847), and lower for VoxHammer (0.804).

Table 12 Cost per delivered edit. Non-agentic methods: median time per completed turn on one local H100 (90th percentile in parentheses). All agent costs are chain totals divided by delivered turns, summarized by the median across 55 chains (90th percentile for time). Input tokens include cached context. Agent time includes model and tool latency and is not comparable to local GPU time.

Method	time per turn (s)	tokens per turn (in / out)	calls per turn
PartFlow	26 (32)	–	–
Nano3D	22 (29)	–	–
3DEditFormer	20 (26)	–	–
VoxHammer	49 (80)	–	–
GLM 5.3 Flash	358 (619)	680k / 4.66k	12.8
DeepSeek V4.1 Flash	358 (1036)	1697k / 14.13k	15.9
Sol 6 (medium)	87 (183)	848k / 2.67k	7.8
Astra (medium)	95 (143)	325k / 2.36k	5.8
Fable 5.1	258 (557)	1555k / 18.26k	9.3
Opus 5.5	149 (291)	1216k / 9.20k	10.9

Table 13 Extended geometry metrics over completed turns on the 457 chains. Subscript u: untouched region; Δ frag.: median fragment-count difference relative to ground truth.

Method	CD↓	HD95↓	NC↑	F _{.01} ↑	F _{.02} ↑	HD95 _u ↓	NC _u ↑	Δ frag.
PartFlow	0.022	0.114	0.81	0.67	0.80	0.039	0.87	-17.0
Nano3D	0.020	0.100	0.83	0.71	0.83	0.034	0.87	-17.0
3DEditFormer	0.028	0.117	0.75	0.55	0.72	0.065	0.79	-17.0
VoxHammer	0.027	0.122	0.75	0.52	0.69	0.072	0.79	-17.0
No-op baseline	0.020	0.122	0.85	0.77	0.84	0.009	0.93	0.0

L Conditioning and held-out views by region

The main tables average every image metric over the 4 fixed views. Tables 15 and 16 split that average by view and by region on the comparison subset of Section 5.4, so the LLM/VLM agents appear next to the non-agentic methods. The conditioning view is the one in which each method receives the target render, and no method ever sees the 3 held-out views. Most methods score a little better in the conditioning view, and the gap is widest in the edit region. There Opus 5.5 reaches an LPIPS of 0.215 in the conditioning view and 0.244 in the held-out views, while the non-agentic methods lose at most 0.016. The no-op baseline goes the other way. The camera direction of each chain is chosen so that every edit is visible in the conditioning view, which makes an unedited object look worse there than from the side. The order of the methods is almost the same in both kinds of view, with only 2 neighboring pairs swapping places, so averaging the 4 views does not favor a method that fits the view it was shown.

Table 14 Multi-view image metrics over completed turns. Subscript 0: conditioning view; 1-3: held-out views; retex: retexture turns only.

Method	PSNR ₀	PSNR ₁₋₃	LPIPS ₀ ↓	LPIPS ₁₋₃ ↓	Sil ₀ ↑	Sil ₁₋₃ ↑	DINO ₁₋₃ ↑	DINO _{retex} ↑
PartFlow	17.2	17.0	0.180	0.182	0.78	0.79	0.753	0.854
Nano3D	17.8	17.7	0.163	0.163	0.82	0.81	0.765	0.843
3DEditFormer	16.5	16.1	0.197	0.205	0.73	0.73	0.752	0.847
VoxHammer	15.3	15.2	0.221	0.219	0.72	0.72	0.737	0.804
No-op baseline	18.1	18.9	0.161	0.147	0.79	0.80	0.752	0.851

Table 15 LPIPS by view and region on the comparison subset. Cond.: the conditioning view; held-out: mean of the 3 other fixed views. Means over each method’s completed turns. Best method (excluding the 2 baselines) in bold.

Method	Edit region		Unchanged region		Whole object	
	Cond.↓	Held-out↓	Cond.↓	Held-out↓	Cond.↓	Held-out↓
No-op baseline	0.592	0.558	0.140	0.128	0.165	0.151
TRELLIS regen.	0.377	0.420	0.188	0.225	0.199	0.236
<i>Non-agentic methods</i>						
PartFlow	0.497	0.503	0.161	0.169	0.178	0.186
Nano3D	0.439	0.453	0.138	0.145	0.156	0.163
3DEditFormer	0.462	0.475	0.162	0.180	0.177	0.195
VoxHammer	0.479	0.495	0.195	0.199	0.209	0.214
<i>LLM/VLM agents</i>						
GLM 5.3 Flash	0.458	0.443	0.123	0.117	0.141	0.134
DeepSeek V4.1 Flash	0.356	0.369	0.087	0.093	0.102	0.108
Sol 6 (medium)	0.336	0.347	0.077	0.080	0.091	0.094
Astra (medium)	0.279	0.293	0.063	0.066	0.076	0.079
Fable 5.1	0.276	0.299	0.063	0.069	0.075	0.082
Opus 5.5	0.215	0.244	0.048	0.056	0.058	0.067

Table 16 PSNR (dB) by view and region on the comparison subset, as in Table 15.

Method	Edit region		Unchanged region		Whole object	
	Cond.↑	Held-out↑	Cond.↑	Held-out↑	Cond.↑	Held-out↑
No-op baseline	10.6	10.4	23.0	23.8	17.5	18.4
TRELLIS regen.	13.6	12.0	16.5	15.3	15.9	14.8
<i>Non-agentic methods</i>						
PartFlow	10.4	9.9	18.3	18.0	17.0	16.8
Nano3D	12.1	11.4	19.2	19.0	17.7	17.6
3DEditFormer	10.9	10.2	18.2	17.4	17.0	16.3
VoxHammer	11.0	10.3	16.5	16.4	15.7	15.6
<i>LLM/VLM agents</i>						
GLM 5.3 Flash	16.9	16.4	24.5	25.2	20.8	21.4
DeepSeek V4.1 Flash	20.2	19.3	27.2	27.6	23.9	24.1
Sol 6 (medium)	20.6	19.5	28.2	28.3	24.5	24.6
Astra (medium)	21.4	20.6	28.9	29.2	25.3	25.6
Fable 5.1	21.6	20.4	29.0	28.9	25.3	25.4
Opus 5.5	23.3	21.8	30.3	29.9	26.8	26.6

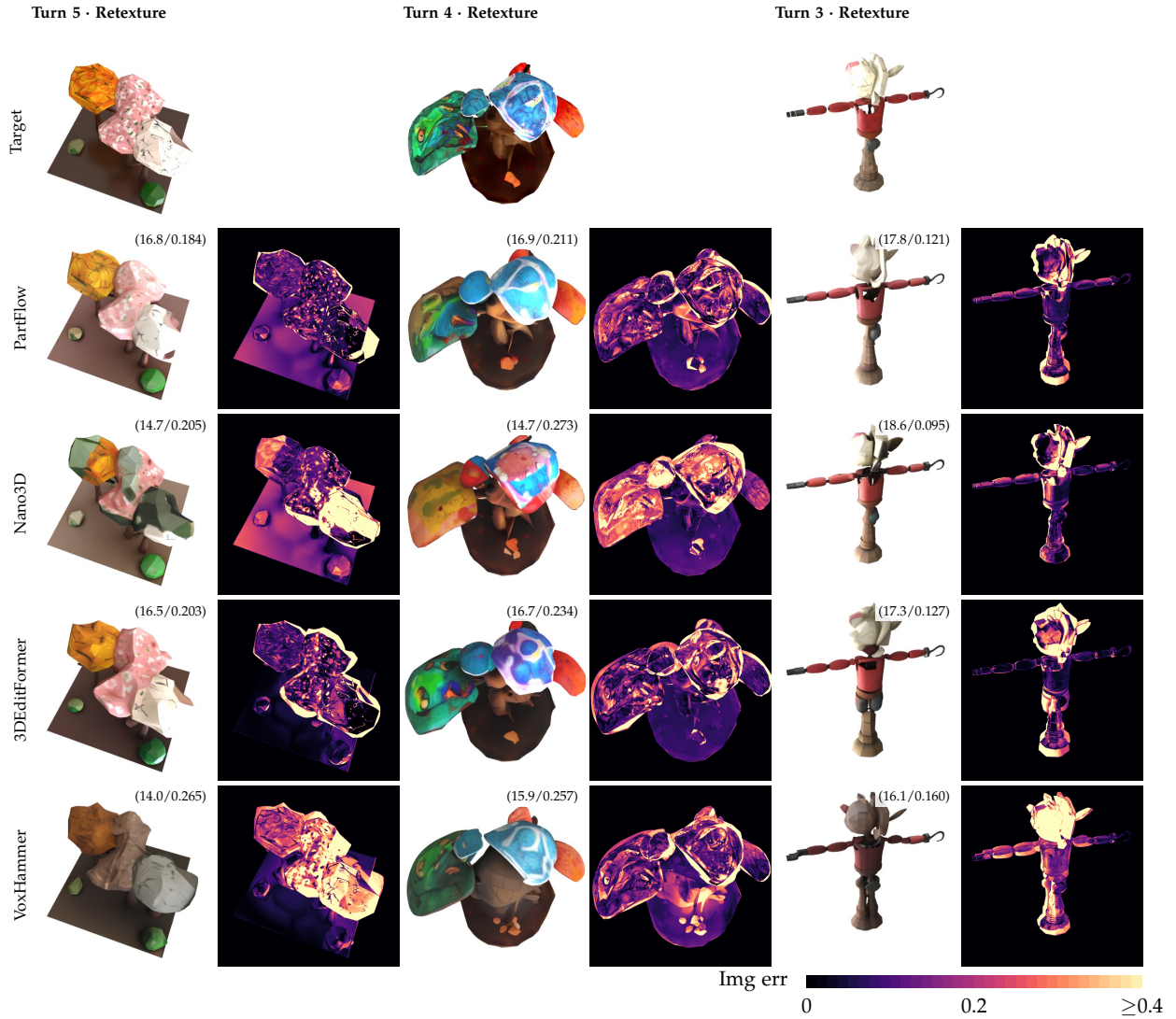


Figure 9 Retexture turns from 3 chains in the conditioning view. For each method the left tile is its render and the right tile its per-pixel error to the target render (mean absolute RGB difference; color bar at the bottom right). The numbers in the top-right corner are (PSNR in dB / LPIPS) of the render against the target view. The ground truth changes only the appearance of the named part, while the errors of the 4 methods spread over the whole object.

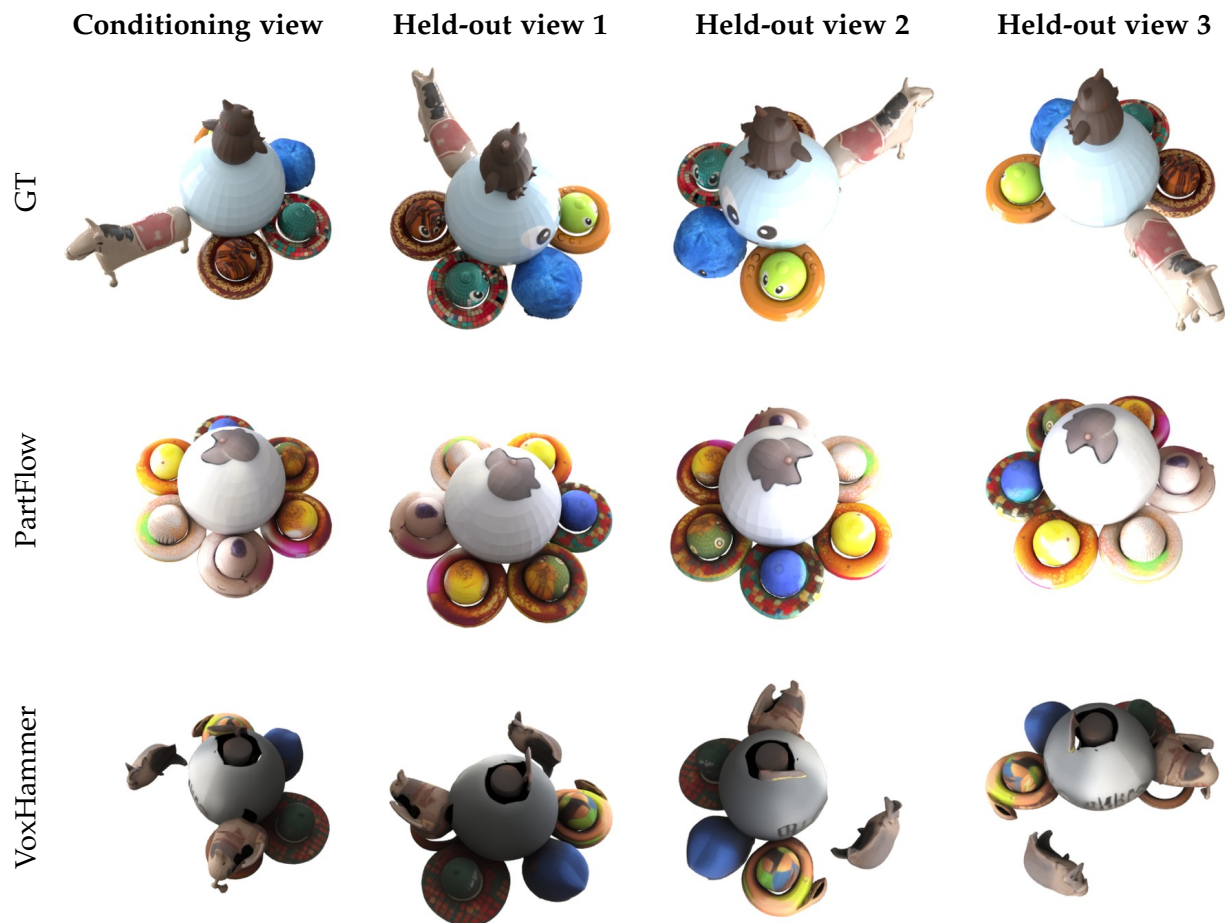


Figure 10 Conditioning and held-out views. GT, PartFlow and VoxHammer at turn 6 of one chain, in the conditioning view and the 3 held-out views.

M More qualitative comparisons

Figures 11 and 12 follow 2 more chains. One is a table setting whose edits are mostly material changes, and the other is a house whose porch gains and loses parts. Rows are methods and columns are the first 5 turns. Under each output is its per-pixel error to the ground truth of that turn (mean absolute RGB difference).

N Qualitative comparison of LLM/VLM agents and non-agentic methods

Figures 13–15 show 3 chains of the comparison subset with every method side by side. The 4 non-agentic methods come first, followed by the 6 agents in the order of Table 3. Columns are the first 5 turns and rows are methods. Every tile is the method’s own output after that turn under self-rollout, rendered from the same camera as the ground truth. The non-agentic methods drift in color and shape even on turns that ask for a small change, while most agents keep their changes on the named part.

O Qualitative examples of chains

Figures 16–28 show ground-truth chains from EditHero. Each page uses the same grid of 5 columns and 4 rows, with equally sized cards for the source and every edited state. Each chain stays on one page and reads from left to right, continuing on the next row when needed. The operation and an instruction excerpt appear below each state. Geometry-only examples use one flat color per slot. Examples with material edits and those grouped by subject keep the assets’ own textures.

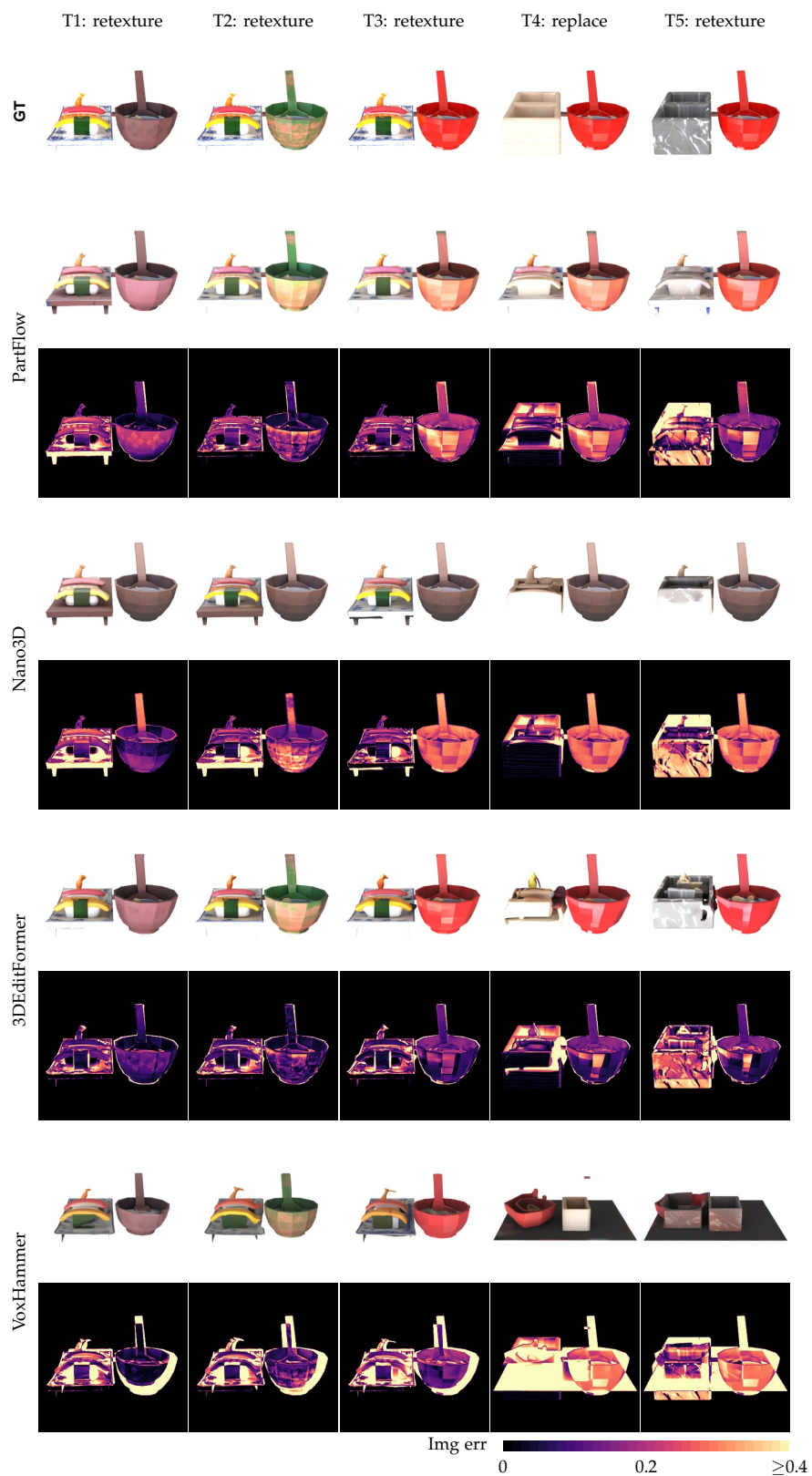


Figure 11 A table setting (turns 1–5): turns 1–3 and 5 change the material of the sushi tray, the soup bowl or the compartment tray, and turn 4 replaces the sushi tray and the sushi with a wooden compartment tray. Shown in the conditioning view.

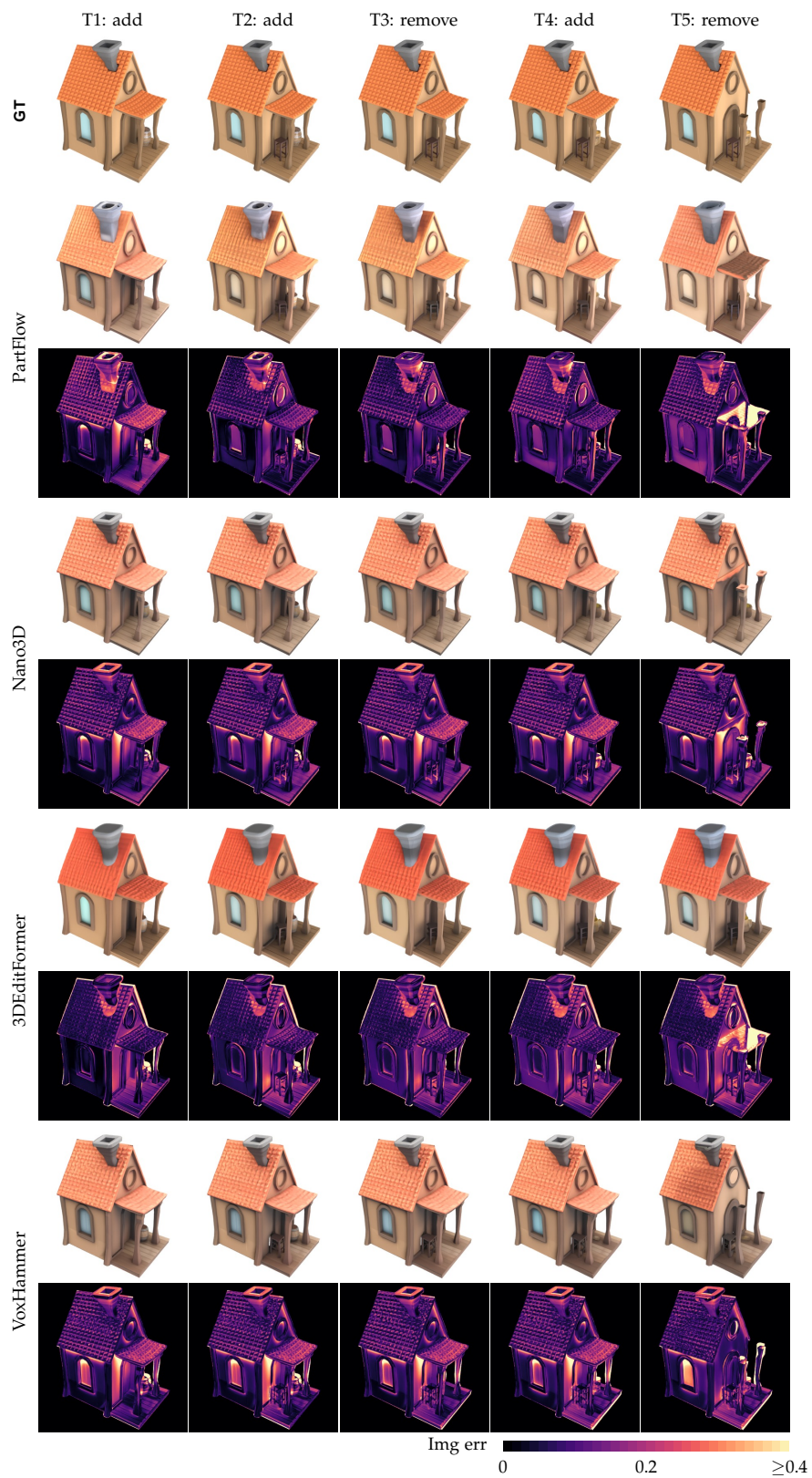


Figure 12 A house (turns 1–5): 3 additions and 2 removals on the porch. Shown from a held-out view other than the conditioning view.

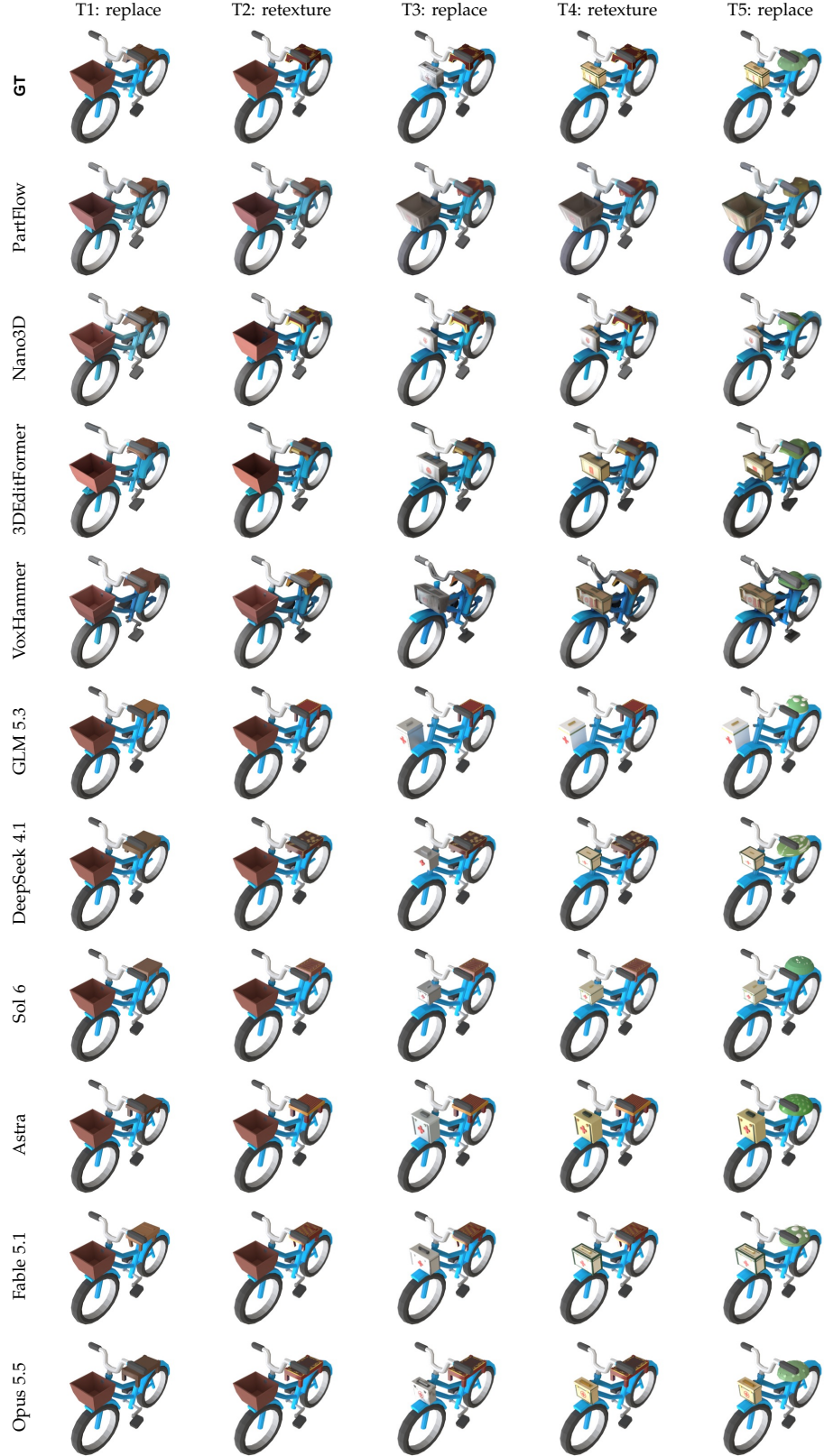


Figure 13 A bicycle (turns 1–5): the saddle is replaced by a wooden stool and repainted, the basket is replaced by a medical case and repainted, and the stool is replaced by a mushroom cap. Shown in the conditioning view.

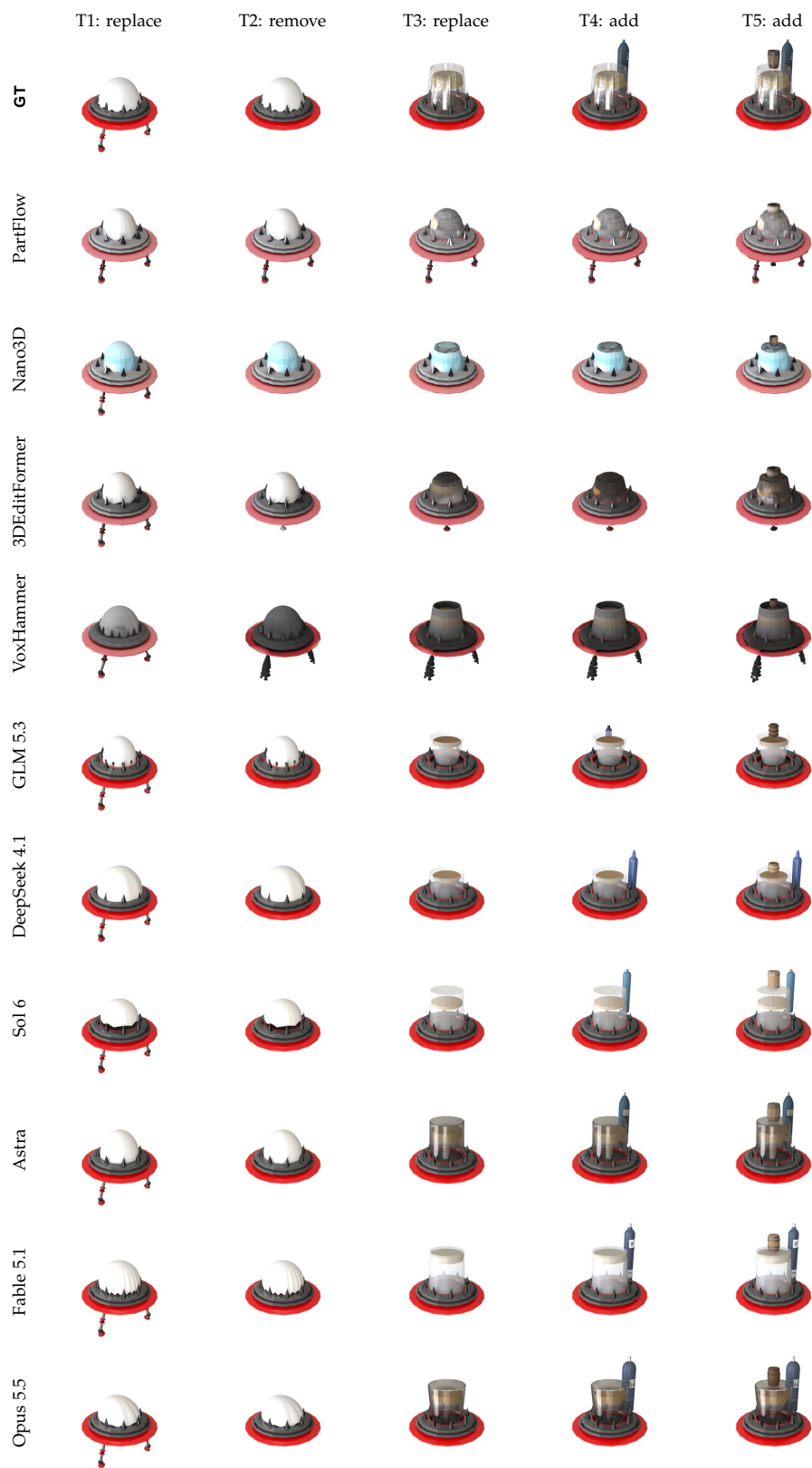
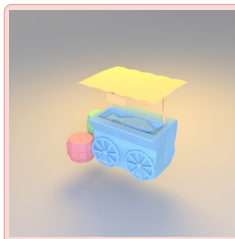


Figure 14 A flying saucer (turns 1–5): the dome is replaced by a parachute canopy, the landing legs are removed, the canopy is replaced by a transparent cylinder, and a gas cylinder and a barrel are added. Shown from a held-out view other than the conditioning view.



Figure 15 A character (turns 1–5, all 4 operations): the green cap is removed, the legs are replaced by legs in black thigh-high socks, a yellow hard hat is added, the arms are retextured with red and white candy stripes, and the legs are retextured as honey-oak wood. Shown from a held-out view other than the conditioning view.

A wooden fish cart with a striped canopy and barrels.



Source

A wooden fish cart with a striped canopy and barrels



Turn 1 · Add

Add a wooden barrel with metal hoops on the top...



Turn 2 · Add

Add a wooden bucket with a rope handle on...



Turn 3 · Add

Add the fruit body of a small, round, green and...



Turn 4 · Add

Add a bunch of yellow bananas on the canopy...

A wooden fish cart with a striped canopy and barrels.



Turn 5 · Add

Add a red apple on top of the shorter wooden barrel.



Turn 6 · Add

Add a single red onion on the canopy top, next to...



Turn 7 · Add

Add a stack of white octagonal bowls with...



Turn 8 · Add

Add a burlap sack filled with yellow grain on...

A red and blue robotic quadruped with wheel-like feet.



Source

A red and blue robotic quadruped with...



Turn 1 · Remove

Remove the machine's rear leg, the blue jointed...



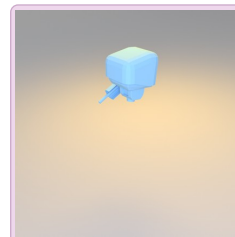
Turn 2 · Remove

Remove the machine's front leg, the blue...



Turn 3 · Remove

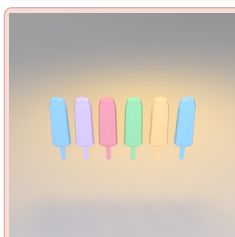
Remove the cylindrical metal motor drum...



Turn 4 · Remove

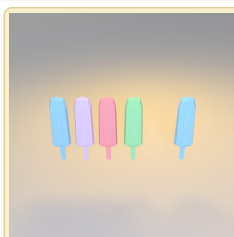
Remove the remaining dark bracket beneath...

A row of six colorful popsicles with wooden sticks.



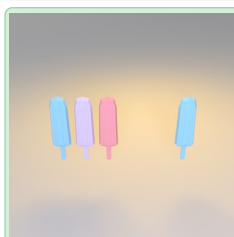
Source

A row of six colorful popsicles with wooden...



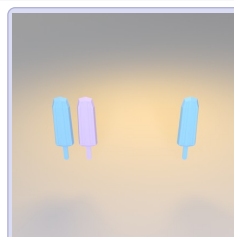
Turn 1 · Remove

Remove the orange popsicle.



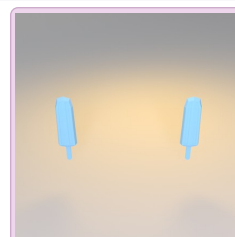
Turn 2 · Remove

Remove the yellow popsicle.



Turn 3 · Remove

Remove the green popsicle.



Turn 4 · Remove

Remove the blue popsicle.

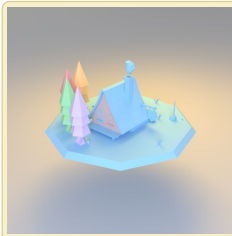
Figure 16 Add-only and remove-only chains.

A low-poly 3D model of a cozy cabin in a forest setting.



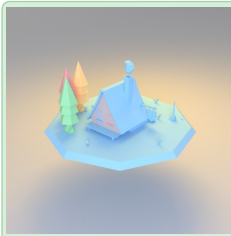
Source

A low-poly 3D model of a cozy cabin in a forest...



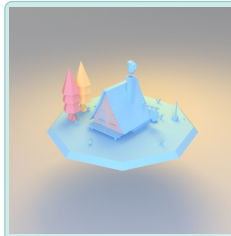
Turn 1 · Remove

Remove the small stone well with its wooden...



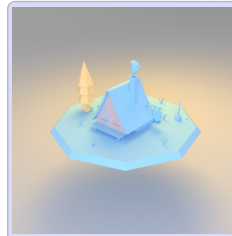
Turn 2 · Remove

Remove the tree closest to the cabin's rear gable...



Turn 3 · Remove

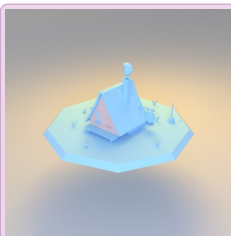
Remove the tree farthest from the front door...



Turn 4 · Remove

Remove the tree farthest from the front door...

A low-poly 3D model of a cozy cabin in a forest setting.



Turn 5 · Remove

Remove the last remaining tree beside the cabin.

A classic black kerosene lantern with a greenish-yellow glass globe.



Source

A classic black kerosene lantern with a...



Turn 1 · Replace

Replace the lantern's cream glass globe with...



Turn 2 · Replace

Replace the lantern's dark blue vented top cap...



Turn 3 · Replace

Replace the clear glass globe with a white...



Turn 4 · Replace

Replace the light grey chimney pipe with a...

A classic black kerosene lantern with a greenish-yellow glass globe.



Turn 5 · Replace

Replace the weathered zinc chimney cap with a...



Turn 6 · Replace

Replace the white mesh lampshade with a green...

Figure 17 Removal and replacement chains.

A set of three Japanese kokeshi dolls on a wooden base.



Source

A set of three Japanese kokeshi dolls on a...



Turn 1 · Replace

Replace the right kokeshi doll with a snowman...



Turn 2 · Replace

Replace the kokeshi doll on the left with a small...



Turn 3 · Replace

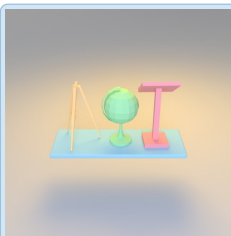
Replace the middle wooden doll with a...



Turn 4 · Replace

Replace the snowman on the right with a wooden...

A set of three Japanese kokeshi dolls on a wooden base.



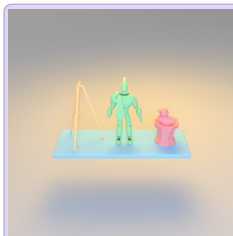
Turn 5 · Replace

Replace the lucky cat in the middle with a small...



Turn 6 · Replace

Replace the wooden book stand on the left with a...



Turn 7 · Replace

Replace the globe in the middle with a small...



Turn 8 · Replace

Replace the wooden tripod stand on the...

A wooden block-style robot with a cube head and articulated limbs.



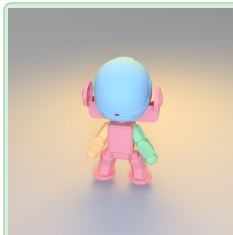
Source

A wooden block-style robot with a cube head...



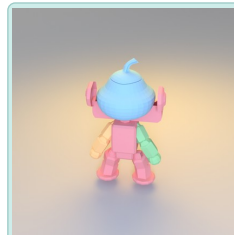
Turn 1 · Replace

Replace the figure's head with a vintage kerosene...



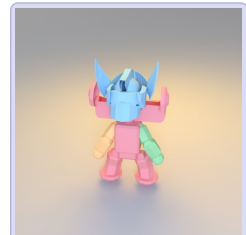
Turn 2 · Replace

Replace the lantern globe with a blue and red...



Turn 3 · Replace

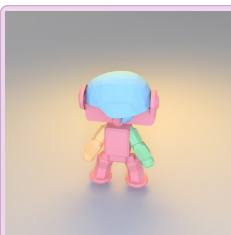
Replace the blue and red helmet dome with a...



Turn 4 · Replace

Replace the jack-o'-lantern head with a horned...

A wooden block-style robot with a cube head and articulated limbs.



Turn 5 · Replace

Replace the horned helmet with an upright grey...

Figure 18 Replacement chains.

A decorative wooden treasure chest with golden accents and a gemstone.



Source

A decorative wooden treasure chest with...



Turn 1 · Add

Add a dark brown treasure sack with a...



Turn 2 · Remove

Remove the lid gem of the object.



Turn 3 · Add

Add a stone guardian lion statue at the center of...



Turn 4 · Add

Add a stack of golden coins on the book cover...

A decorative wooden treasure chest with golden accents and a gemstone.



Turn 5 · Add

Add a golden bird statue on the book cover, on...

A whimsical fantasy-style house with arched windows and a thatched roof.



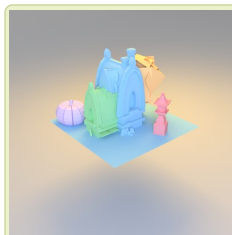
Source

A whimsical fantasy-style house with arched...



Turn 1 · Add

Add a stone lantern with a green tiled roof and a...



Turn 2 · Add

Add a large orange pumpkin with a green...



Turn 3 · Add

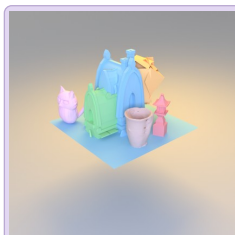
Add a weathered terracotta pot with a...



Turn 4 · Remove

Remove the orange pumpkin with the green...

A whimsical fantasy-style house with arched windows and a thatched roof.



Turn 5 · Add

Add a black cat with glowing green eyes...



Turn 6 · Remove

Remove the weathered terracotta pot with the...

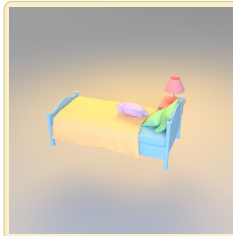
Figure 19 Add-and-remove chains.

A wooden single bed with a green duvet and a bedside table with a lamp.



Source

A wooden single bed with a green duvet and a...



Turn 1 • Add

Add two white and gray throw pillows on the...



Turn 2 • Add

Add a plush toy cat on the duvet, facing the foot of...



Turn 3 • Remove

Remove the two white and gray throw pillows...



Turn 4 • Add

Add a baby deer plush with white spots...

A vintage-style desktop fan with a metal cage and three blades.



Source

A vintage-style desktop fan with a metal cage...



Turn 1 • Replace

Replace the black flared fan stand with a chunky...



Turn 2 • Replace

Replace the caged fan head with a black...



Turn 3 • Replace

Replace the fan rotor with a red three-bladed...



Turn 4 • Replace

Replace the base with an upright chicken foot...

A vintage-style desktop fan with a metal cage and three blades.



Turn 5 • Replace

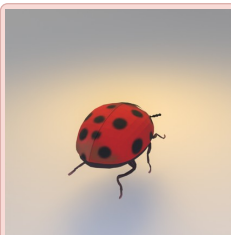
Replace the bird's foot with a weathered...



Turn 6 • Remove

Remove the weathered sheet-metal dome base.

A red ladybug with black spots and black legs and head.



Source

A red ladybug with black spots and black legs...



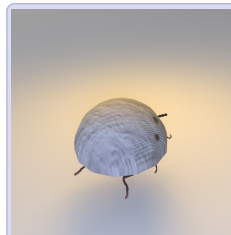
Turn 1 • Replace

Replace the ladybug's red spotted wing case with...



Turn 2 • Add

Add an upright wooden cask with vertical staves...



Turn 3 • Remove

Remove the wooden cask from the top of the...



Turn 4 • Replace

Replace the beetle's pale gray dome shell with a...

Figure 20 Add-and-remove and mixed-edit chains.

A 3D model of a Japanese sushi meal set with soup bowl and chopsticks.



Source

A 3D model of a Japanese sushi meal set with...



Turn 1 · Retexture

Change the material of the dark red soup bowl and...



Turn 2 · Retexture

Change the material of the dark red tray holding...



Turn 3 · Retexture

Change the material of the pale ash wood sushi...



Turn 4 · Replace

Replace the cream crackle-glaze soup bowl...

A 3D model of a Japanese sushi meal set with soup bowl and chopsticks.



Turn 5 · Retexture

Change the material of the white footed bowl to...



Turn 6 · Retexture

Change the material of the glossy black sushi tray...

A yellow and brown three-wheeled motorized rickshaw with a black canopy.



Source

A yellow and brown three-wheeled...



Turn 1 · Retexture

Change the material of the boxy passenger cabin to...



Turn 2 · Replace

Replace the left rear wheel with a wheel that has a...



Turn 3 · Retexture

Change the material of the flat canopy roof above...



Turn 4 · Add

Add an ivory-and-cobalt floral enamel jar on top...

A yellow and brown three-wheeled motorized rickshaw with a black canopy.



Turn 5 · Retexture

Refinish the motorcycle front in...



Turn 6 · Remove

Remove the red and white striped canvas awning...

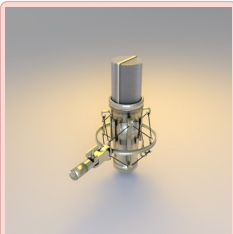


Turn 7 · Replace

Replace the white-tire left rear wheel with a...

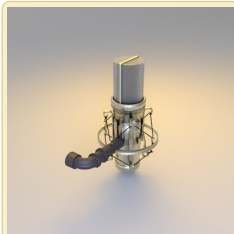
Figure 21 Mixed-edit chains.

A studio condenser microphone with a shock mount and XLR connector.



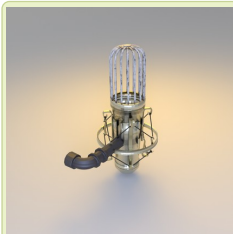
Source

A studio condenser microphone with a...



Turn 1 · Replace

Replace the connector on the side of the...



Turn 2 · Replace

Replace the microphone's mesh grille head with a...



Turn 3 · Retexture

Refinish the bent connector in copper...



Turn 4 · Retexture

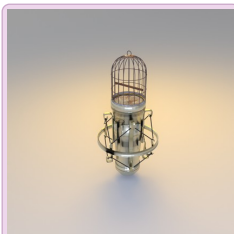
Give the grille navy enamel bars, fine gold...

A studio condenser microphone with a shock mount and XLR connector.



Turn 5 · Replace

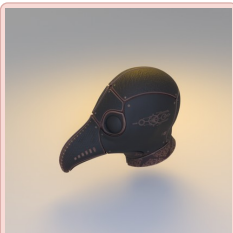
Replace the navy-and-brass grille...



Turn 6 · Remove

Remove the copper-and-brass...

A black leather plague doctor mask with a long beak and decorative stitching.



Source

A black leather plague doctor mask with a long...



Turn 1 · Replace

Replace the beaked mask's top panel with a...



Turn 2 · Retexture

Change the material of the tortoise shell to...



Turn 3 · Retexture

Change the material of the mask's beak to bone...



Turn 4 · Replace

Replace the aged bronze turtle shell with a red...

A black leather plague doctor mask with a long beak and decorative stitching.



Turn 5 · Retexture

Change the material of the porcelain beak to deep...



Turn 6 · Remove

Remove the red square lid.

Figure 22 Chains with all four operations.

A wooden barrel drum with a decorative stand and ornate metal accents.



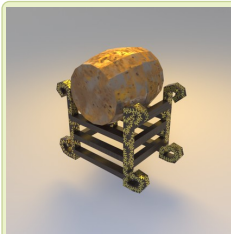
Source

A wooden barrel drum with a decorative stand...



Turn 1 · Retexture

Change the material of the barrel drum resting in...



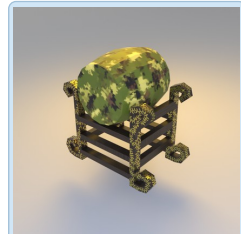
Turn 2 · Retexture

Change the material of the black and white...



Turn 3 · Retexture

Give the drum a scarlet festival lacquer shell...



Turn 4 · Retexture

Give the drum an olive-green and brown...

A wooden barrel drum with a decorative stand and ornate metal accents.



Turn 5 · Retexture

Change the material of the camouflage-patterned...



Turn 6 · Retexture

Change the material of the honey brown wicker...

A low-poly 3D tree with a brown trunk and green leaf clusters.



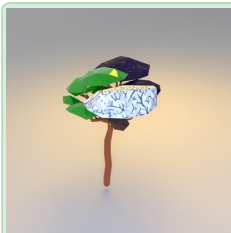
Source

A low-poly 3D tree with a brown trunk and green...



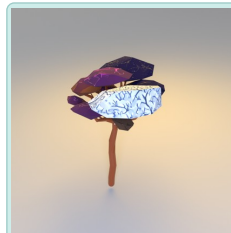
Turn 1 · Retexture

Change this canopy cluster to silvery-white...



Turn 2 · Retexture

Change this canopy cluster to midnight-blue...



Turn 3 · Retexture

Change this canopy cluster to dark...



Turn 4 · Retexture

Change this canopy cluster to charcoal-black...

A low-poly 3D tree with a brown trunk and green leaf clusters.



Turn 5 · Retexture

Change this canopy cluster to deep...

Figure 23 Retexture-only chains.

A small 3D model of a bonsai tree with red-orange leaves on a dark base.



Source

A small 3D model of a bonsai tree with...



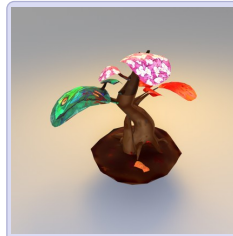
Turn 1 · Retexture

Change the low side branch's leaves to...



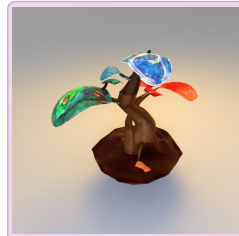
Turn 2 · Retexture

Give the tree's top leaves a pink cherry-blossom...



Turn 3 · Retexture

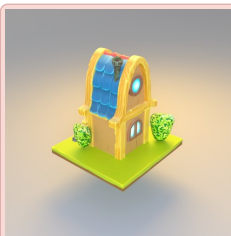
Give the low side leaves an emerald...



Turn 4 · Retexture

Give the tree's top leaves a sapphire-blue skin with...

A whimsical cartoon-style house with a curved roof and glowing window.



Source

A whimsical cartoon-style house with a curved...



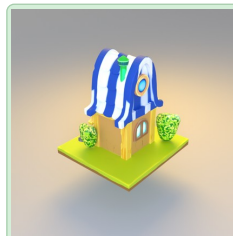
Turn 1 · Retexture

Change the material of the arched roof of the little...



Turn 2 · Retexture

Change the material of the cottage's terracotta tiled...



Turn 3 · Retexture

Change the material of the cottage's mossy grey...



Turn 4 · Retexture

Change the material of the cottage's blue and white...

A whimsical cartoon-style house with a curved roof and glowing window.



Turn 5 · Retexture

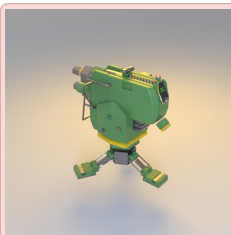
Change the material of the cottage's black roof to...



Turn 6 · Retexture

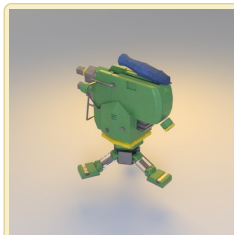
Change only the roof shingles to an amethyst...

A green tripod-mounted robotic turret with multiple weapon barrels.



Source

A green tripod-mounted robotic turret with...



Turn 1 · Replace

Replace the turret's upper secondary barrel with a...



Turn 2 · Add

Add a rectangular metal box to the side of the...



Turn 3 · Replace

Replace the tripod with a low metallic pedestal.



Turn 4 · Remove

Remove the rectangular metal box from the turret.

Figure 24 Retexture and geometry-edit examples.

A small wooden house with a tiled roof and chimney.



Source

A small wooden house with a tiled roof and ...



Turn 1 · Add

Add a small green potted plant on the porch deck ...



Turn 2 · Add

Add an orange pumpkin on the porch deck, next ...



Turn 3 · Remove

Remove the small tiled awning above the porch ...



Turn 4 · Add

Add a green pine tree with a brown trunk on ...

A small wooden house with a tiled roof and chimney.



Turn 5 · Remove

Remove the green potted plant from the porch deck.



Turn 6 · Add

Add a cartoon-style red mushroom with white ...

A modern computer desk with a monitor, keyboard, and two towers.



Source

A modern computer desk with a monitor ...



Turn 1 · Replace

Replace the flat monitor on the desk with a black ...



Turn 2 · Replace

Replace the small amplifier beside the ...



Turn 3 · Replace

Replace the blue computer tower standing under ...



Turn 4 · Add

Place the orange folding chair in front of the ...

A modern computer desk with a monitor, keyboard, and two towers.



Turn 5 · Replace

Replace the ribbed olive-green cylinder on ...



Turn 6 · Replace

Replace the black curved-screen monitor ...

Figure 25 Scene editing.

A red vintage-style tractor with large rear wheels and a small front cab.



Source

A red vintage-style tractor with large rear wheels...



Turn 1 · Retexture

Refinish the tractor hood in cream-and-barn-red...



Turn 2 · Retexture

Refinish the left rear tire's rubber in an icy tread...



Turn 3 · Retexture

Refinish the two front tires' outer rubber with...



Turn 4 · Retexture

Change the material of the tractor's right rear...

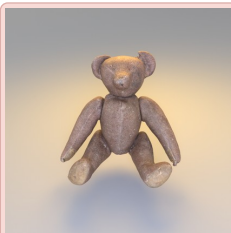
A red vintage-style tractor with large rear wheels and a small front cab.



Turn 5 · Retexture

Refinish the hood in petrol-green explorer...

A small clay teddy bear figurine in a seated pose.



Source

A small clay teddy bear figurine in a seated pose



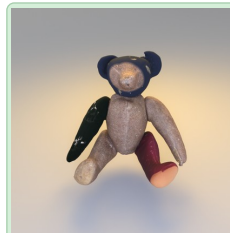
Turn 1 · Retexture

Give the teddy head midnight-navy fabric...



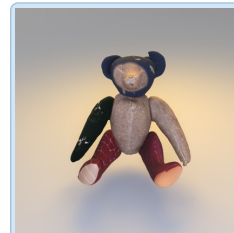
Turn 2 · Retexture

Give this teddy leg burgundy fabric with...



Turn 3 · Retexture

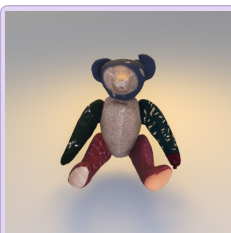
Give this teddy arm deep forest-green fabric with...



Turn 4 · Retexture

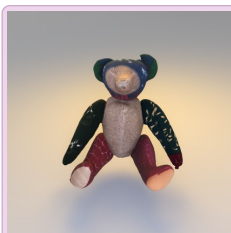
Give this teddy leg burgundy fabric with...

A small clay teddy bear figurine in a seated pose.



Turn 5 · Retexture

Give this teddy arm deep teal-green fabric with...



Turn 6 · Retexture

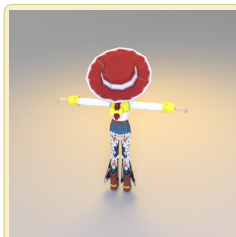
Give the teddy head navy-blue fabric with...

Figure 26 Scene and character editing.

A 3D model of a female cowboy character with a red hat and yellow shirt.



Source
A 3D model of a female cowboy character with...



Turn 1 · Retexture
Give the denim trouser panels a deep navy...



Turn 2 · Retexture
Change the figure's hat to a smooth jet-black...

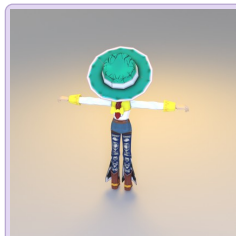


Turn 3 · Retexture
Give the denim trouser panels a deep...



Turn 4 · Retexture
Give the hat deep teal leather with dark...

A 3D model of a female cowboy character with a red hat and yellow shirt.



Turn 5 · Retexture
Give the denim trouser panels a deep navy...



Turn 6 · Retexture
Change the turquoise hat to a warm...

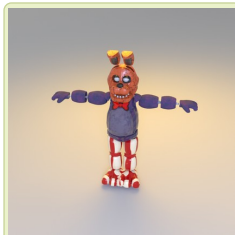
A purple robotic bunny figure with a red bow tie and articulated limbs.



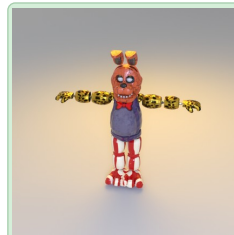
Source
A purple robotic bunny figure with a red bow...



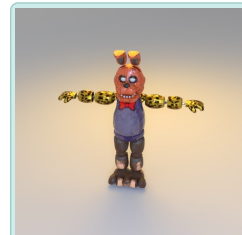
Turn 1 · Retexture
Change the material of the rabbit's legs and feet to...



Turn 2 · Retexture
Change the material of the purple rabbit's head...

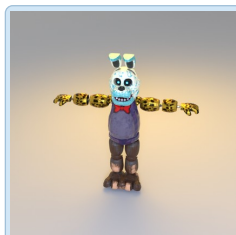


Turn 3 · Retexture
Change the material of the rabbit's outstretched...

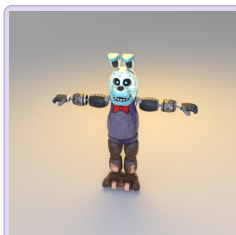


Turn 4 · Retexture
Change the material of the red-and-cream striped...

A purple robotic bunny figure with a red bow tie and articulated limbs.



Turn 5 · Retexture
Give the head creamy ivory and turquoise...



Turn 6 · Retexture
Give the arms and gloves deep-navy leather with...



Turn 7 · Retexture
Change the material of the brown leather legs to...

Figure 27 Character editing.

A wooden shelf holding various glass bottles and a funnel.



Source

A wooden shelf holding various glass bottles...



Turn 1 · Remove

Remove the tall conical glass flask with the...



Turn 2 · Add

Add a small hourglass with a brown wooden...



Turn 3 · Remove

Remove the two small glass bottles standing at...



Turn 4 · Add

Add a potted plant with dark gray angular...

A wooden shelf holding various glass bottles and a funnel.



Turn 5 · Replace

Replace the large smoky glass jar topped by a...



Turn 6 · Remove

Remove the small hourglass with the...



Turn 7 · Add

Add a stone candle stand with four yellow...

An orange and black forklift model with a cab and lifting forks.



Source

An orange and black forklift model with a...



Turn 1 · Retexture

Change the material of the forklift's rear wheels...



Turn 2 · Retexture

Change the material of the forklift's engine hood to...



Turn 3 · Retexture

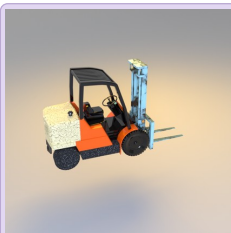
Change the material of the forklift's mast and forks...



Turn 4 · Retexture

Change the material of the mud-caked rear wheels...

An orange and black forklift model with a cab and lifting forks.



Turn 5 · Retexture

Change the material of the camouflaged engine...



Turn 6 · Retexture

Refinish the mast in worn safety-yellow livery...

Figure 28 Object editing.

P Prompts

The templates below are used for material restyling and for drafting replacement instructions, which reviewers then revised (Section 3). An excerpt of the shared agent prompt is listed last.

Retexturing material redraw prompt

Change the material to <material_description>. Keep the shape, geometry and camera angle exactly the same. White background.

Replacement instruction template

Replace the <current_part_description> with <new_part_description>.

LLM/VLM task prompt (excerpt). Each agent receives the task below, followed by the modeling skill and rendering and resource rules. These will be released with the benchmark. We quote the main task and the retexture rule, with omissions marked.

[...] Edit your own previous saved output, starting with source.glb. Infer each operation and region from text/image; preserve unaffected parts. [...] Remove is direct physical deletion; add/replace retain the original construction and review loop; retexture directly edits only the target material/texture while preserving geometry and placement, with render-and-compare iteration. No external assets, hidden GT models/passes or generative geometry/image models. [...] Attempt the requested actual edit on every turn within the finite budget. Deliver the best runnable edited candidate even if quality gates fail; label it partial and continue from it. [...]

- Retexture: directly edit the named part's material, texture, or shader parameters to match the instruction and target appearance. Preserve its geometry, topology, UV coordinates, node names, transforms, placement, and attachment relationships. [...] Patterned finishes: either (a) project the target image onto the part [...]; or (b) generate the pattern procedurally in code [...] evaluated at each surface point's 3D position [...] rather than in UV space. Either route is acceptable; choose by render-and-compare at the evaluation camera and match the target colours.