EDITHERO: A BENCHMARK FOR LONG-HORIZON PART-LEVEL 3D EDITING AND VIBE MODELING

Anonymous authors
Paper under double-blind review

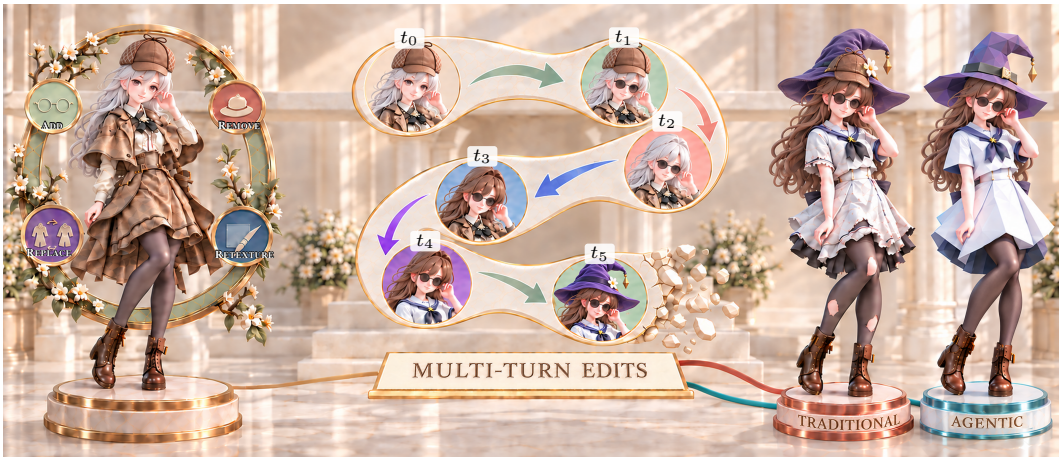

Figure 1: **Editing is a sequence.** The initial figurine (t_0) undergoes five successive edits: adding sunglasses (t_1), removing the cap (t_2), changing the hair from white to brown (t_3), replacing the outfit (t_4), and adding a witch hat (t_5). Each turn must satisfy the new instruction while preserving prior edits, as required in iterative asset creation.

ABSTRACT

3D editing methods are typically evaluated one edit at a time, although creating an asset requires a sequence of revisions. A method must follow each new instruction while preserving earlier edits and untouched regions, yet this ability has not been systematically measured. We introduce EditHero, to our knowledge the first benchmark for long-horizon 3D editing from image and natural-language instructions covering both geometry and texture. A deterministic assembly engine produces the target after every edit, and manual review verifies each sequence. Evaluating four 3D editing methods and six large language models (LLMs) reveals that conventional methods often miss the requested changes and alter regions meant to be preserved. Most LLMs that edit through code follow instructions more accurately, and all of them preserve unchanged regions well, but their time and token costs limit interactive use. We will release the synthesis engine and edit sequences to support research on reliable iterative 3D editing.

1 INTRODUCTION

A 3D artist rarely makes one edit and stops. Assets in art and game production evolve through successive revisions: each instruction changes part of the current asset while preserving earlier work. The same expectation underlies "vibe modeling", in which users shape assets through an iterative natural-language conversation (Wang et al., 2026e; Ye et al., 2025b; De La Torre et al., 2024). Yet text- and image-guided 3D editing methods (Li et al., 2026b; Ye et al., 2026b; Xia et al., 2025; Weng et al., 2026; Hu et al., 2026) 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.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Evaluating this ability requires a reference after every turn. Existing paired datasets (Xia et al., 2025; Ye et al., 2026b; Weng et al., 2026) contain isolated edits rather than a sequence with a reference after every instruction. Their targets come from learned editors or from changes to poses and semantic parts, and neither kind measures how well a method preserves earlier work across a sequence of different instructions.

We introduce EditHero, a benchmark of *part-level edit chains with exact ground truth* (Figure 2). A data engine assembles library parts onto segmented host objects using four operations (add, remove, replace and retexture) and logs each operation in a JSON file. Together with the part library, this log rebuilds the exact state after every turn. EditHero contains 485 chains and 2965 edits across 255 objects, with up to 30 turns per chain (Section 3).

We evaluate methods under a *self-rollout* protocol (Section 4): each edit starts from the method's own previous output. Region-specific metrics assess instruction following (IF) in the edited region and content consistency (CC) elsewhere. We test four conventional editing methods (Weng et al., 2026; Ye et al., 2026b; Xia et al., 2025; Li et al., 2026b) and six large language models (LLMs) that edit through code (Sections 5 and 5.4). Conventional methods often fail to make the requested change even at the first turn and alter regions meant to remain unchanged. Their preservation of the initial object worsens over successive turns. In a diagnostic on two methods, starting each turn from the ground-truth previous state removes much of the decline in target similarity. Most code-editing LLMs follow instructions more accurately, and all of them preserve unchanged regions well, but the median amortized cost ranges from about 1.5 to 6 minutes and 6 to 16 model calls per edit. 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 sequential, long-horizon, and part-level 3D editing.
- **Evaluation framework.** We build a self-rollout evaluation of four conventional methods and six code-editing LLMs, with region-specific measures of instruction following and preservation.

2 Related Work

3D editing. 3D editing has followed the available generative models. Early editors optimized a NeRF or a Gaussian splat under a 2D diffusion prior (Haque et al., 2023; Chen et al., 2024e;b), and later ones edit rendered views or an image and then reconstruct the object (Barda et al., 2025; Hsiao et al., 2026). Part-level and sketch-based editors let a user refine an object step by step (Dong et al., 2024a; Sella et al., 2026; Li et al., 2020; 2025b). Native 3D generators such as TRELLIS (Xiang et al., 2025) made it possible to edit a 3D latent directly. Trained editors feed the source latent to the generator through an added branch, much as ControlNet (Zhang et al., 2023c) conditions an image model. PartFlow (Weng et al., 2026) adds a source-control branch, and 3DEditFormer (Xia et al., 2025) uses dual-guidance attention. Training-free editors reuse a pretrained generator instead. VoxHammer (Li et al., 2026b) inverts the source and reuses its cached latents and attention features in the preserved region, and Nano3D (Ye et al., 2026b) follows the difference between the source-conditioned and target-conditioned flows, following FlowEdit (Kulikov et al., 2025). We evaluate these four methods as representatives. Each was proposed and evaluated for one edit at a time, and each uses generative latents to synthesize edited content. EditHero places them in a setting that reflects iterative asset creation, a sequence of edits on their own output, and measures what regeneration does to the parts that no instruction names. 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 (Xia et al., 2025; Ye et al., 2026b; Ma et al., 2025c) or pose and part changes (Xia et al., 2025; Weng et al., 2026; Preintner et al., 2026). Others have no target asset and score region preservation and alignment to an edited image instead (Li et al., 2026b; Zhou et al., 2026b; Gat et al., 2026; Nie et al., 2026), and scene-level benchmarks follow the same pattern (Patro et al., 2026; Liu et al., 2025a; Zhu et al., 2026a). The closest multi-step settings are agent benchmarks such as BlenderGym (Gu et al., 2025) and 3DCodeBench (Gao et al., 2026), which let an agent iterate toward one

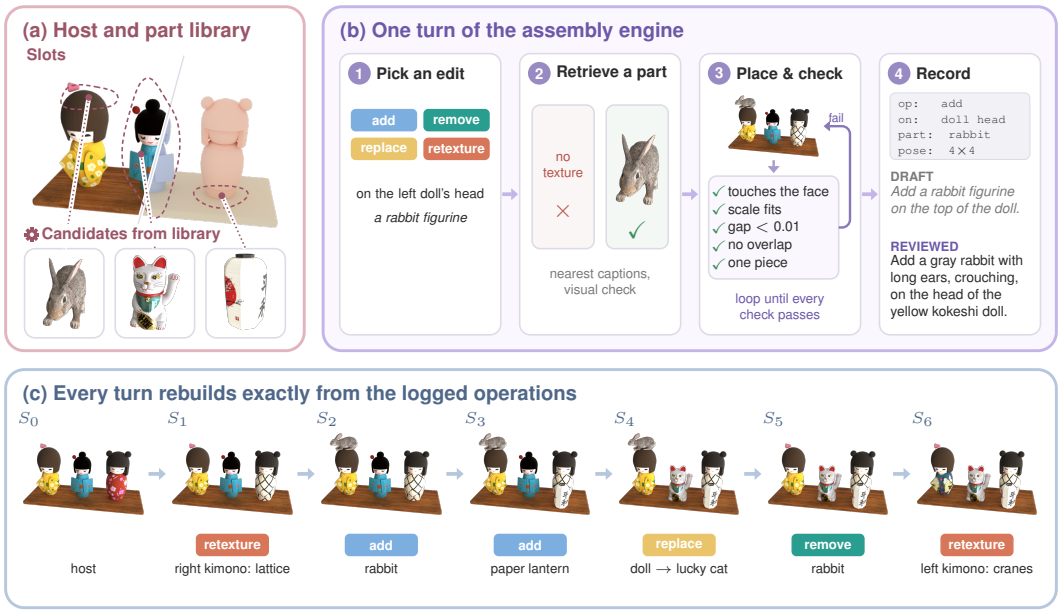

Figure 2: **Constructing EditHero.** (a) A host object is divided into named slots, and candidate parts are drawn from a captioned library. (b) For each turn, the engine selects an edit and its target part, retrieves matching parts, screens them visually, and tests their placement against geometric constraints. Each accepted edit is logged with its library part and pose, 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 four chains on the same host; all images are renders. Colors indicate the four edit operations.

fixed goal, and EZBlender (Wang et al., 2026c), which 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).

Agentic generation and editing by code. Editing by code is the main alternative to regeneration. LLMs write code or call tools to build scenes (Sun et al., 2023; Yang et al., 2024a) and edit Blender scenes from renders (Huang et al., 2024a; Lu et al., 2025). Related systems generate objects and CAD programs (Dai et al., 2025; Khan et al., 2024; Rukhovich et al., 2024). Recent systems treat the program itself as the asset and revise it under visual feedback (Lin et al., 2026; Ning et al., 2026; Wang et al., 2026e), aiming to keep edits local and preserve the rest of the state. Our data engine rests on the same idea, since every state of a chain is rebuilt from a log of assembly operations. We also evaluate LLMs that edit the mesh by code, turn by turn and with the same metrics as the conventional methods (Section 5.4).

3 Benchmark Construction

Task. A chain begins with a textured, part-segmented *host* object S_0 and applies n instructions in order. Each instruction specifies one operation on a part and includes a fixed-view render of the target state. An *add* attaches a part to an available contact face; a *remove* deletes a named part, which may comprise several pieces; a *replace* swaps a named part for another at the same location; and a *retexture* changes a part's appearance without changing 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. Instructions may refer to earlier edits: a newly added part might later be removed, for example. Success therefore requires maintaining the asset's evolving state throughout the chain.

Exact ground truth by construction. Evaluating a chain requires a reliable target at every turn. Some existing paired datasets use image-guided generative pipelines for their targets (Xia et al.,

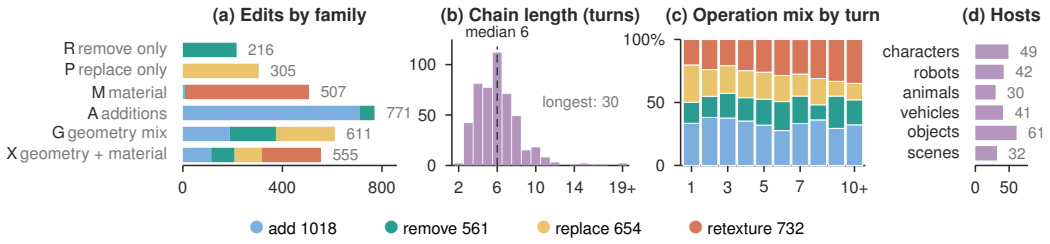

Figure 3: **EditHero statistics.** The benchmark contains 485 chains, 2965 edits, and 255 hosts. (a) Edits by chain family and operation. (b) Chain lengths, with lengths 19 and above pooled. (c) Operation frequencies by turn index; all four operations appear in each displayed bin (turns 1–9 and pooled turns 10+). (d) Host categories.

2025; Ye et al., 2026b); the resulting target can itself contain unintended changes. We instead construct each target through part assembly, in the spirit of programmatically generated benchmarks (Johnson et al., 2017; Greff et al., 2022). For an addition or replacement, the engine retrieves caption-matched parts from a library based on PartVerse-XL (Ding et al., 2025), screens candidates with a VLM agent, and tests their placement for contact, scale, connectivity, and interpenetration. Removal deletes a named slot. For a retexture, the engine renders the slot, restyles the render with an image editing model (Qwen-Image-Edit (Wu et al., 2025a)) and textures the unchanged slot geometry from the restyled image with the texturing module of TRELLIS.2 (Xiang et al., 2026). 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 485 chains and 2965 edits across 255 hosts (Figure 3). Chains have 2 to 30 turns, with a median of six; the median instruction is 17 words long. Six families group chains by their allowed operations. Removal (R), replacement (P), and material (M) chains focus on their respective operations, although two material chains also include addition and removal. Addition (A) chains primarily add parts. Geometry (G) chains combine additions, removals, and replacements, while geometry-and-material (X) chains also include retexturing. All four operations appear at each of turns 1–9 and in the pooled group of later turns. A visibility check retains 457 chains and 2750 turns for evaluation (Section 4). Appendix Table 4 compares EditHero with existing benchmarks.

4 Evaluation Protocol

Setting. We evaluate every method under *self-rollout*: turn k takes the method's output from turn $k-1$ as input. In this main evaluation, ground-truth states are never fed back, and a failed turn ends the chain. Each method receives the instruction and its required conditioning inputs, taken from the ground truth: a target-view render and, where applicable, a ground-truth edit mask. Ground-truth and predicted states share a coordinate frame and a rig of four fixed views for each chain, one of which is the conditioning view. We choose its direction so that every edit is visible and keep a chain only if each of its turns changes at least 0.5% of the frame. Three borderline chains were kept after visual inspection. This leaves 457 chains and 2750 turns. Appendix C provides details.

Edit and unchanged regions. 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 $\mathcal{G} = \{0, \ldots, 63\}^3$, and let $\mathcal{D}(\cdot)$ dilate a set by one cell, clamped to the grid. We define

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

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 image domain Ω, the corresponding regions are

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

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 stray changes outside the named parts fall in the unchanged region.

Region-restricted metrics. We write $m_R(\cdot, \cdot)$ for a metric evaluated only in region R of Eq. equation 1 or equation 2. For 3D states X and Y, for example,

$$\mathrm{IoU}_R(X, Y) = \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|(\mathcal{V}(X) \cup \mathcal{V}(Y)) \cap R|}. \tag{3}$$

Surface samples are assigned to regions by their voxels; image metrics aggregate per-pixel values within the specified region. We also evaluate the whole object by setting $R = \mathcal{G}$ or $R = \Omega$. Table 1 summarizes the operators; Appendix D defines them in full.

Target similarity and instruction following. We compare each output $\hat{S}_k$ with its target S_k in the edit region, unchanged region, and whole object; 2D comparisons use their fixed-view renders. Because high target similarity can reflect parts already present, we also report *instruction following* (IF) relative to the method's own previous output. Under self-rollout the method's states differ from the ground truth from the first turn, so IF and content consistency are anchored to the method's own previous output rather than to the ground-truth previous state. Against the ground truth, a part that the method lost at an earlier turn would count as a successful removal, and the no-op baseline would receive 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:

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

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. We measure preservation outside the current edit both relative to the method's previous output and relative to the initial host:

$$\mathrm{CC}_k^{\mathrm{prev}} = \mathrm{IoU}_{U_k}\left(\hat{S}_k, \hat{S}_{k-1}\right), \qquad \mathrm{CC}_k^0 = \mathrm{IoU}_{U_k^0}\left(\hat{S}_k, S_0\right), \quad U_k^0 = \mathcal{G} \setminus \textstyle\bigcup_{j \le k} E_j. \tag{5}$$

The cumulative region excludes every part intentionally edited so far. Appendix C illustrates these choices with evaluation examples.

Image changes and baseline. For images, we measure unchanged-region drift as $\mathrm{MAE}_{U_k^v}\left(\hat{I}_k^v, \hat{I}_{k-1}^v\right)$. On retexture turns, we compare the magnitude of the predicted change in the edit region with that of the target: $\mathrm{MAE}_{E_k^v}\left(\hat{I}_k^v, \hat{I}_{k-1}^v\right) / \mathrm{MAE}_{E_k^v}\left(I_k^v, I_{k-1}^v\right)$. A no-op baseline returns S_0 at every turn. Its IF is zero whenever defined and its CC is one, and its target similarity provides a reference for interpreting method scores. We average metrics over successful turns and report coverage and failures alongside them (Appendix F).

5 Experiments

5.1 Setup

We evaluate four conventional 3D editing methods, PartFlow (Weng et al., 2026), Nano3D (Ye et al., 2026b), 3DEditFormer (Xia et al., 2025) and VoxHammer (Li et al., 2026b), with their released code,

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. equation 1 or equation 2. Full definitions appear in Appendix D.

3D			2D		
$\mathrm{IoU}_R(X,Y)$	voxel IoU	↑	$\mathrm{PSNR}_R(I,J)$	peak signal-to-noise ratio	↑
$\mathrm{Recall}_R(X,Y)$	voxel recall of Y	↑	$\mathrm{SSIM}_R(I,J)$	SSIM map (Wang et al., 2004)	↑
$\mathrm{CD}_R(X,Y)$	Chamfer distance	↓	$\mathrm{LPIPS}_R(I,J)$	LPIPS map (Zhang et al., 2018)	↓
$\mathrm{F}^{\tau}_R(X,Y)$	F-score at distance τ	↑	$\mathrm{DINO}_R(I,J)$	DINOv2 cosine (Oquab et al., 2023)	↑
			$\mathrm{MAE}_R(I,J)$	mean absolute error	↓

Table 2: **Quantitative results of conventional editing methods under self-rollout.** Means over all successful turns of the 457 evaluation chains. IF: instruction following, anchored to the model's own previous state; $\mathrm{CC}^{\mathrm{prev}}$: IoU of the unchanged region to the model's own previous state; F: F-score to the target state; CD: Chamfer distance ($\times 100$); drift: mean absolute change of the unchanged image region against the model's own previous render. Reference rows: the no-op baseline returns the initial object at every turn; TRELLIS regeneration builds each turn from the target render alone and is aligned to the target state. Best conventional method in bold.

	3D						2D				
	edit E_k		unchanged U_k		whole		edit	unchanged		whole	
Method	IF↑	F↑	$\mathrm{CC}^{\mathrm{prev}}$ ↑	F↑	F↑	CD↓	LPIPS↓	LPIPS↓	drift↓	LPIPS↓	PSNR↑
No-op baseline	0.00	0.56	1.00	0.82	0.77	2.03	0.569	0.130	0.000	0.151	18.7
TRELLIS regen.	0.50	0.43	0.38	0.43	0.43	2.03	0.400	0.193	0.041	0.202	15.8
PartFlow	0.12	0.47	0.86	0.70	0.67	2.16	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**	**17.7**
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	0.522	0.205	0.022	0.220	15.2

weights and sampling settings. Every method runs under self-rollout on the 457 evaluation chains (2750 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. Next to the no-op baseline we report TRELLIS single-view regeneration (Xiang et al., 2025) as a second reference. At every turn it generates a whole object from the target render alone, without the instruction or any previous state, and its output is aligned to the ground-truth state of the same turn by a similarity transform. Adapters, conditioning, the routing and the execution coverage of every method are in Appendix F.

5.2 Main Results

Table 2 reports selected geometry and image metrics from Section 4 in the edit region, the unchanged region and on the whole object, in 3D and in the fixed views, with the two reference rows on top.

Whole-object similarity is dominated by what does not change. On the whole object the no-op baseline beats every conventional method on F-score (F 0.77 against at most 0.71) and on image similarity (LPIPS 0.151 against at least 0.163, PSNR 18.7 against at most 17.7 dB). Most of an object does not change at any single turn, so returning the initial object is a strong strategy under these whole-object measures. Restricting the score to the edit region does not separate them in 3D either. The no-op F-score there (0.56) equals that of the best method, because the edit region includes the part as it was before the edit. IF and image similarity in the edit region separate real edits from the no-op output, since every method is closer to the target there than the no-op baseline (LPIPS 0.474–0.522 against 0.569). The regeneration reference shows why instruction following has to be read together with preservation. It rebuilds the whole object from the target render and reaches the highest IF (0.50) and the lowest edit-region LPIPS (0.400). Its unchanged-region IoU with its own previous output is only 0.38, and it has the lowest whole-object F-score (0.43). An editor has to do both at once, yet none of the four conventional methods matches regeneration on IF or the no-op baseline on preservation.

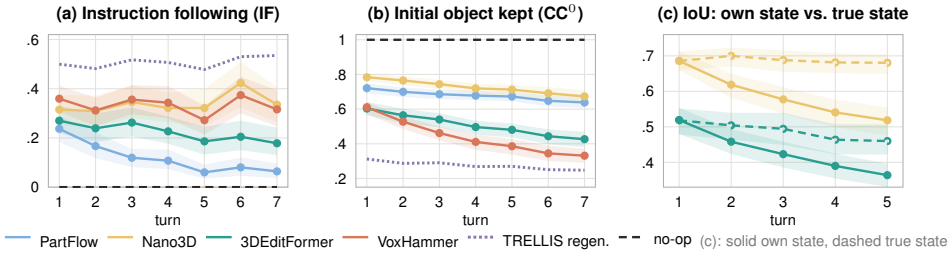

Figure 4: **Along a chain.** Means within a fixed set of 135 chains at every turn (chains that every conventional method completes and that have at least seven turns, 103 hosts); only defined scores enter each mean; shaded: 95% host-clustered bootstrap intervals. (a) Instruction following, relative to the model's own previous state. (b) Preservation of the initial object in the untouched region. The no-op baseline has IF $= 0$ and $\mathrm{CC}^0 = 1$; the regeneration reference rebuilds each turn independently from the target render. (c) IoU to the target when each turn starts from the model's own output (solid) or from the ground-truth previous state (dashed), on the 147 chains of the reset diagnostic with at least five turns (Appendix H).

Instructions are followed poorly from the first turn. IF measures the fraction of the required change that a method makes relative to its own previous state. It is 0.12–0.32 over all turns, and it is already low at turn 1, before any error can accumulate. At that turn it is 0.24 for PartFlow, 0.32 for Nano3D, 0.27 for 3DEditFormer and 0.36 for VoxHammer on the chains of Figure 4. The two methods with the highest IF differ in what they disturb on the way. VoxHammer keeps an IoU of only 0.71 of the unchanged region to its own previous state, whereas Nano3D keeps 0.89.

Errors accumulate under self-rollout. Figure 4 follows the same 135 chains from turn 1 to turn 7. Instruction following remains limited: it ranges from 0.31 to 0.42 for Nano3D and from 0.27 to 0.37 for VoxHammer, and it falls from 0.24 to 0.06 for PartFlow and from 0.27 to 0.18 for 3DEditFormer. Preservation of the initial object declines over the chain for all four methods. CC^0 falls from 0.60–0.78 to 0.33–0.67, and VoxHammer loses the most (0.61 to 0.33). Independent regeneration also loses some preservation as the target sequence changes. The decline could come from harder later instructions or from a worse inherited state, and resetting the input state to the ground truth helps distinguish the two (Figure 4c, Appendix H). When every turn starts from the true previous state, IoU to the target stays at 0.68 for Nano3D and falls from 0.52 to 0.46 for 3DEditFormer. Under self-rollout the same values at turn 5 are 0.52 and 0.36, so inherited errors account for much of the decline. End-state scores by chain length, the metrics by operation and the analysis of material edits, which geometry cannot see, are in Appendices G and I.

5.3 Qualitative results

Figure 5 follows the first six edits of a covered wagon from the LLM set (Section 5.4), all of them replacements, with every output rendered from the same camera as its ground truth. The crate at turn 1 and the storage box at turn 6 are small, so most of the visible change comes from the cover, which is replaced four times in a row. No conventional method follows this sequence. PartFlow keeps a blue rounded cover from turn 2 to turn 5 instead of the striped peaked canopy and the gray, beige and pale yellow covers that follow. Nano3D, 3DEditFormer and VoxHammer produce a peaked canopy at turn 2 but then keep a dark cover, and by turn 5 they reach only a dull olive instead of pale yellow. 3DEditFormer and VoxHammer also darken the wagon bed, which no instruction touches. The three LLMs (Opus 5.5, Astra and Fable 5.1) replace the cover with the requested shape and color at every turn, and their largest errors are concentrated on the cover. Appendix Figure 16 shows a character chain with all four operations, Appendix Figures 12 and 13 add a table setting and a house, and Appendix L shows more chains.

5.4 Editing by code

Conventional methods regenerate the object at every turn. Code-editing agents instead inspect the current mesh, write code to build, remove, or repaint the named part, and compare a Blender render with the target view before committing the edit. We evaluate six models: Opus 5.5, Fable 5.1,

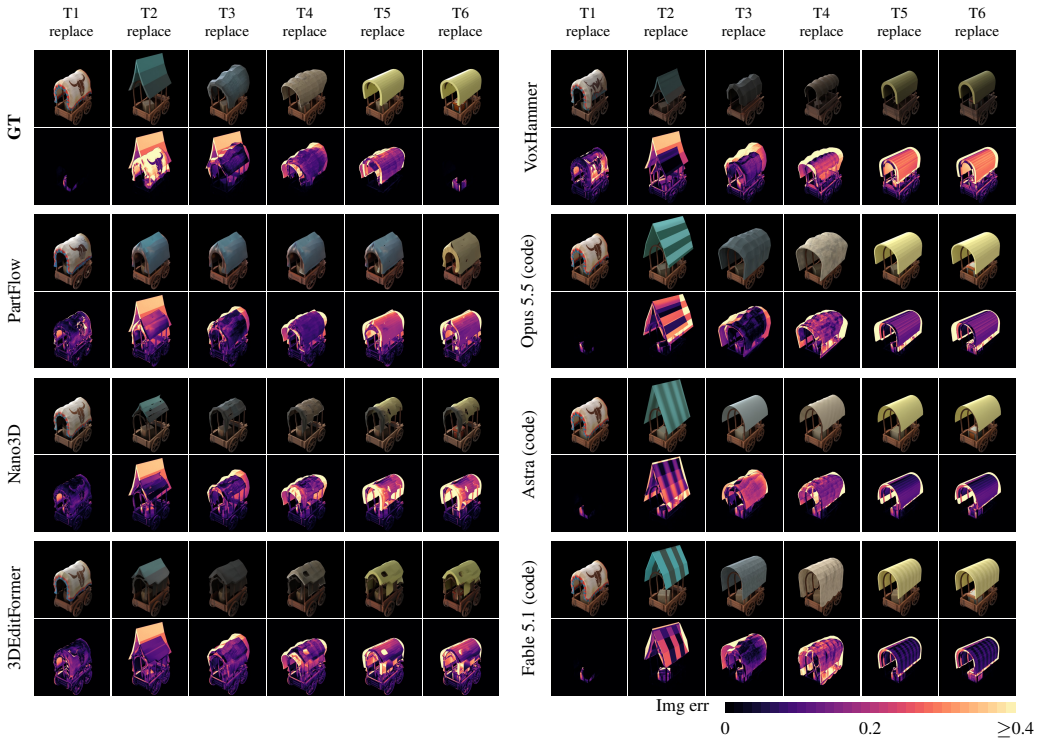

Figure 5: **Six consecutive replacements on a covered wagon** (turns 1–6 of a chain in the LLM set): the barrel becomes a wooden crate, the cover is replaced by a striped peaked canopy and then by gray, beige and pale yellow fabric covers, and the crate is replaced by a science-fiction storage box. Columns are turns. Each block shows a method's own output after turn k under self-rollout (upper row) and its per-pixel difference to the ground truth of turn k (lower row; mean absolute RGB difference); for the ground truth the lower row is the change from turn $k-1$. Left: the ground truth and three conventional methods; right: VoxHammer and three LLMs that edit by code (Section 5.4). Same camera and lighting for every block.

Astra, Sol 6, GLM 5.3 Flash and DeepSeek V4.1 Flash. All receive the same base editing prompt and edit every chain with an independent agent, but they run in different agent frameworks, so each comparison is between a model together with its framework (Appendix J). None has access to a part library, retrieval system, or generative 3D or image model, so new parts must be built through code. We evaluate them on a subset of the evaluation chains, which we call the *LLM set*. We selected its 55 chains (416 turns) by hand, keeping chains whose edits are clearly visible and resemble the edits an artist would make, while covering all six families and as many combinations of operations as possible. All methods run under self-rollout on this set. Table 3 reports their results, and Appendix M provides additional side-by-side examples.

Code editing improves preservation and, for most models, instruction following. Most code-editing agents follow instructions better than the conventional methods, and all of them preserve the rest of the object far better. IF ranges from 0.26 for GLM 5.3 Flash to 0.62 for Opus 5.5, with 0.52 for Fable 5.1, 0.52 for Astra, 0.42 for Sol 6 and 0.42 for DeepSeek V4.1 Flash, against at most 0.36 for a conventional method; only GLM stays below that level. CC^{prev} lies between 0.95 and 0.98 for all six models, against at most 0.90 for the conventional methods. Removals are the most reliable operation (IF from 0.88 to 0.99), because code can delete a named part directly. Additions and replacements are harder (IF from 0.10 to 0.49 and from 0.17 to 0.51), because parts built from primitives often miss the target's size or depth. The covered wagon in Figure 5 illustrates these localized errors.

The gains extend to whole-object similarity. The best conventional-method LPIPS is 0.161, compared with 0.154 for no-op, whereas all six models improve on no-op, from 0.136 for GLM 5.3 Flash to 0.065 for Opus 5.5. All six models deliver all 416 turns. Opus 5.5 is closer to the target

Table 3: **Quantitative results of code-editing LLMs.** Code-editing LLMs and conventional methods on the LLM set, the 55 chains (416 turns) of Section 5.4. The rows of the conventional methods are computed on these chains and therefore differ from Table 2. *Delivered / Tried* counts completed turns among those attempted; a missing, empty, or unloadable output ends the chain. Scores are averaged over each method's delivered turns, as in Table 2; 2D metrics use four fixed views. Bold marks the best method, excluding reference rows.

		3D			2D		
Method	Delivered / Tried	IF↑	CC^{prev} ↑	F_W ↑	$LPIPS_E$ ↓	$LPIPS_W$ ↓	$PSNR_W$ ↑
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
Conventional methods							
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
LLMs that edit by code							
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
Opus 5.5	416/416	**0.62**	**0.98**	**0.82**	**0.236**	**0.065**	**26.7**
Fable 5.1	416/416	0.52	**0.98**	0.80	0.293	0.080	25.4

than no-op on all 55 chains, Fable 5.1 on 54, Astra on 53, Sol 6 and DeepSeek V4.1 Flash on 51 and 51, and GLM 5.3 Flash on 35.

Latency limits interactive use. Across the six models, the median amortized cost is about 1.5–6 minutes and 6–16 model calls per edit (Appendix J). Conventional methods take 20–50 seconds per successful turn on a local GPU, while agent time also includes remote model and tool latency. Code editing gives the strongest instruction following and preservation among the evaluated methods, but its latency remains a barrier to interactive editing.

6 Conclusion

EditHero measures whether a 3D editing method can keep editing its own output. Its chains are assembled from parts, so the correct state after every turn is known exactly, and its metrics separate the region an instruction changes from the region it leaves alone, and every score is read next to a no-op baseline and a regeneration reference. Measured this way, the conventional methods follow instructions poorly from the first turn and change what they should keep, and their preservation declines over successive turns. Their whole-object F-score, LPIPS and PSNR are also worse than those of the no-op baseline. LLMs that edit by code change mostly what they are asked to change and are closer to the target than the unchanged object on mean whole-object LPIPS in the code-editing comparison, but at a time and token cost that is still too high for interactive editing. We will release the data synthesis engine and the chains.

USE OF AI

Generative models are part of the data engine. A vision-language model (VLM) agent rejects implausible part candidates during retrieval, and the engine drafts instructions from part captions. For retexture turns, the engine renders the slot, restyles the render with an image editing model (Qwen-Image-Edit (Wu et al., 2025a)) and textures the unchanged slot geometry from the restyled image with the texturing module of TRELLIS.2 (Xiang et al., 2026) (Section 3). Every turn of every chain was inspected by a person. The reference states are deterministically assembled from stored library parts and textures, including the generated ones. Section 5.4 evaluates large language models (LLMs) that edit by code. VLMs/LLMs were also used to draft, edit and polish the text of this paper, and for retrieval and discovery, such as finding related work. The authors checked the code, the reported numbers, the text and every citation, and take full responsibility for them. Figure 1 is an AI-generated conceptual illustration, not benchmark evidence. Other figures show benchmark renders, method diagrams, or measured results.

REFERENCES

Jinxin Ai, Matthias Nießner, and Ziya Erkoç. Dreamedit3d: Personalization of multi-view diffusion models for 3d editing. *arXiv preprint arXiv:2605.16990*, 2026. 24

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, et al. Self-consuming generative models go MAD. In *ICLR*, 2024. 25

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. 25

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. 25

Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, 2022. 24

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. 24

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. 2, 24

Samyadeep Basu, Mehrdad Saberi, Shweta Bhardwaj, et al. EditVal: Benchmarking diffusion based text-guided image editing methods. *arXiv preprint arXiv:2310.02426*, 2023. 24

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. 24

Alexandre Binninger, 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. 24

Manuel Brack, Felix Friedrich, Katharina Kornmeier, et al. Ledits++: Limitless image editing using text-to-image models. In *CVPR*, 2024. 24

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)*, pp. 103–112, 2010. doi: 10.2312/SCA/SCA10/103-112. 24

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022. 24

Weiwei Cai, Shuangkang Fang, Weicai Ye, et al. Native 3d editing with full attention. *arXiv preprint arXiv:2511.17501*, 2025. 24

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. 24

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 25

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*, 2025a. 25

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. 23

Hansheng Chen, Ruoxi Shi, Yulin Liu, et al. Generic 3d diffusion adapter using controlled multi-view editing. *arXiv preprint arXiv:2403.12032*, 2024a. 24

Jiacheng Chen, Ramin Mehran, Xuhui Jia, Saining Xie, and Sanghyun Woo. Blenderfusion: 3d-grounded visual editing and generative compositing. *arXiv preprint arXiv:2506.17450*, 2025b. 24

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. 24

Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. In *European Conference on Computer Vision (ECCV)*, 2024b. 2, 24

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)*, 2024c. 24

Minghao Chen, Roman Shapovalov, Iro Laina, et al. Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. In *CVPR*, 2025c. 23

Minghao Chen, Jianyuan Wang, Roman Shapovalov, et al. Autopartgen: Autogressive 3d part generation and discovery. *arXiv preprint arXiv:2507.13346*, 2025d. 23

Sijin Chen, Xin Chen, Anqi Pang, et al. Meshxl: Neural coordinate field for generative 3d foundation models. In *NeurIPS*, 2024d. 23

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*, 2025e. 25

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)*, 2024e. 2, 24

Yiwen Chen, Yikai Wang, Yihao Luo, et al. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. *arXiv preprint arXiv:2408.02555*, 2024f. 23

Yiwen Chen, Tong He, Di Huang, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. In *ICLR*, 2025f. 23

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*, 2025g. 24

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. 24

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. 25

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. 24

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. 3, 25

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. 1, 25

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, et al. ProcTHOR: Large-scale embodied AI using procedural generation. In *NeurIPS*, 2022. 25

Matt Deitke, Ruoshi Liu, Matthew Wallingford, et al. Objaverse-XL: A universe of 10m+ 3D objects. In *NeurIPS*, 2023. 26

Chaorui Deng, Deyao Zhu, Kunchang Li, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025. 24

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. 4, 25, 26

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)*, 2024a. 2, 24

Wenqi Dong, Bangbang Yang, Lin Ma, et al. Coin3d: Controllable and interactive 3d assets generation with proxy-guided conditioning. In *ACM SIGGRAPH Conference Papers*, 2024b. 24

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1979. ISSN 0090-5364. doi: 10.1214/aos/1176344552. 31

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. 36

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. 25

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. 24

Fan Fei, Jiajun Tang, Fei-Peng Tian, Boxin Shi, and Ping Tan. Pacture: Efficient pbr texture generation on packed views with visual autoregressive models. *Computational Visual Media*, 2026. 24

Haitang Feng, Xinkai Chen, Jie Liu, et al. Objfiller3d: Scaling 3d object inpainting to dense multi-view consistency. *arXiv preprint arXiv:2508.18271*, 2025. 24

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, et al. Layoutgpt: Compositional visual planning and generation with large language models. In *NeurIPS*, 2023. 25

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. 24

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. 24

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. 2, 25, 26

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. 2, 24, 25, 26

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. 24

Zigang Geng, Binxin Yang, Tiankai Hang, et al. Instructdiffusion: A generalist modeling interface for vision tasks. *arXiv preprint arXiv:2309.03895*, 2023. 24

Javier Gonzalez, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential bayesian optimization. *arXiv preprint arXiv:1704.03651*, 2017. 24

Klaus Greff, Francois Belletti, Lucas Beyer, et al. Kubric: A scalable dataset generator. In *CVPR*, 2022. 4, 25, 26

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. 2, 25, 26

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. 24

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. 23

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. 2, 24

Yi He, Jiangming Wang, Xinyu Wang, et al. Geoedit: Geometry-aware object editing via dual-branch denoising. In *ECCV*, 2026. 24

Zebin He, Mingxin Yang, Shuhui Yang, et al. Materialmvp: Illumination-invariant material generation via multi-view pbr diffusion. *arXiv preprint arXiv:2503.10289*, 2025. 24

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. 24

Yicong Hong, Kai Zhang, Jiuxiang Gu, et al. Lrm: Large reconstruction model for single image to 3d. In *ICLR*, 2024. 23

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. 2, 24

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. 1, 30

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. 25

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. *arXiv preprint arXiv:2404.17672*, 2024a. 3, 25

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*, 2026a. 25

Xin Huang, Tengfei Wang, Ziwei Liu, and Qing Wang. Material anything: Generating materials for any 3d object via diffusion. *arXiv preprint arXiv:2411.15138*, 2024b. 24

Yinming Huang, Shuyuan Tu, Xi Yan, et al. Agentic visual generation: From generative models to agentic control. *arXiv preprint arXiv:2609.06758*, 2026b. 25

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. 25

Zijian Huang, Jay Zhangjie Wu, Zian Wang, et al. Ape: Agentic prompt enhancer for image generation and editing. *arXiv preprint arXiv:2606.00204*, 2026c. 25

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. 24

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. 36

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH*, pp. 409–416, 1999. 24

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. 35

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. 4, 25, 26

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. 25

Bahjat Kawar, Shiran Zada, Oran Lang, et al. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276*, 2022. 24

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. 24

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. 3, 25

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. 24

Jeonghwan Kim, Yushi Lan, Armando Fortes, Yongwei Chen, and Xingang Pan. Fastmesh: Efficient artistic mesh generation via component decoupling. In *3DV*, 2026. 23

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. 36

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. 24

Yuki Koyama, Issei Sato, and Masataka Goto. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics*, 2020. 24

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *ICCV*, 2025. 2, 24

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. 24

Long Le, Jason Xie, William Liang, et al. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *ICLR*, 2025. 25

Seunggwan 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. 24

Bo Li, Jiahao Kang, Yubo Ma, et al. Sketchfacegs: Real-time sketch-driven face editing and generation with gaussian splatting. In *CVPR*, 2026a. 24

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. 2, 24

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. 24

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*, 2025a. 24

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)*, 2025b. 2, 24

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)*, 2026b. 1, 2, 5, 24, 25, 26, 29

Peng Li, Suizhi Ma, Jialiang Chen, et al. Cmd: Controllable multiview diffusion for 3d editing and progressive generation. In *ACM SIGGRAPH Conference Papers*, 2025c. 24

Sikuang Li, Chen Yang, Jiemin Fang, et al. Sculpt: Subtractive composition for 3d part generation. *arXiv preprint arXiv:2608.13541*, 2026c. 23

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*, 2025d. 23

Weiyu Li, Antoine Toisoul, Tom Monnier, et al. Meshflow: Efficient artistic mesh generation via meshvae and flow-based diffusion transformer. In *CVPR*, 2026d. 23

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*, 2025e. 23

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. In *NeurIPS*, 2025f. 23

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. 25

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. 23

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*, 2025a. 25

Chen-Hsuan Lin, Jun Gao, Luming Tang, et al. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 23

Youtian Lin, Yikang Yang, Zhanpeng Hu, et al. Procedura: Agentic 3d modeling with procedural control. *arXiv preprint arXiv:2608.26238*, 2026. 3, 25

Yuchen Lin, Chenguo Lin, Panwang Pan, et al. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. In *NeurIPS*, 2025b. 23

Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *CVPR*, 2025. 23

Chenxi Liu, Selena Ling, and Alec Jacobson. Gimmbo: Interactive generative image model merging via bayesian optimization. In *SIGGRAPH*, 2026a. 24

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*, 2026b. 25

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. 24

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*, 2025a. 2, 25

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, et al. PartField: Learning 3D feature fields for part segmentation and beyond. In *ICCV*, 2025b. 23

Ruoshi Liu, Rundi 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. 23

Shiyu Liu, Yucheng Han, Peng Xing, et al. Step1X-Edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025c. 24

Zichen Liu, Yue Yu, Hao Ouyang, et al. Magicquill: An intelligent interactive image editing system. In *CVPR*, 2025d. 25

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *CVPR*, 2024. 23

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. 3, 25

Changfeng Ma, Yang Li, Xinhao Yan, et al. P3-SAM: Native 3D part segmentation. *arXiv preprint arXiv:2509.06784*, 2025a. 23

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*, 2025b. 25

Yiwei Ma, Jiayi Ji, Ke Ye, et al. I2EBench: A comprehensive benchmark for instruction-based image editing. In *NeurIPS*, 2024. 24

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. 25

Ziqi Ma, Hongqiao Chen, Yisong Yue, and Georgia Gkioxari. Feedforward 3D editing via text-steerable image-to-3D. *arXiv preprint arXiv:2512.13678*, 2025c. 2, 25, 26

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. 25

Chenlin Meng, Yutong He, Yang Song, et al. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 24

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. 36

Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 24

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. 25

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. 24

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020. 23

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. 24

Rui Nie, Chuang Wang, Haitao Zhou, et al. EditFlow3D: Automated local editing of 3D assets with trajectory preservation. *arXiv preprint arXiv:2608.03179*, 2026. 2, 24, 25, 26

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *ICLR*, 2024. 25

Yansong Ning, Jingwen Ye, Zhongkai Wu, et al. Vibeworlding: Can multimodal agents construct 3d open worlds end-to-end? *arXiv preprint arXiv:2608.15265*, 2026. 3, 25

Nimra Noor, Muhammad Bilal, Abdullah Hussain, and Hassan Baig. Nova3d: Code-native generation of programmable 3d assets. *arXiv preprint arXiv:2607.22738*, 2026. 25

Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 6, 29, 36

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. 25

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. 24

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. 24

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. 2, 25

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 23

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. 2, 25, 26

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. 24

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. 24

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. 24

Qwen Team. Qwen3-VL-32B-Instruct. https://huggingface.co/Qwen/Qwen3-VL-32B-Instruct, 2025. Official model card. 26

Alexander Raistrick, Lahav Lipson, Zeyu Ma, et al. Infinite photorealistic worlds using procedural generation. In *CVPR*, 2023. 25, 26

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. 24

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019. 26

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. 23

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. 24

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. 3, 25

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. 24

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. 2, 24

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. 24

Shelly Sheynin, Adam Polyak, Uriel Singer, et al. Emu Edit: Precise image editing via recognition and generation tasks. In *CVPR*, 2024. 24

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024. 23

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. 24

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, et al. Meshgpt: Generating triangle meshes with decoder-only transformers. In *CVPR*, 2024a. 23

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*, 2024b. 24

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. 24

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. 23

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. 3, 25

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. 36

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. 25

Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, et al. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. In *ICLR*, 2025a. 23

Jiaxiang Tang, Ruijie Lu, Zhaoshuo Li, et al. Efficient part-level 3d object generation via dual volume packing. In *NeurIPS*, 2025b. 23

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. 25

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. 36

Tencent Hunyuan3D Team. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025. 23

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. 24

Chunshi Wang, Haohan Weng, Junliang Ye, et al. Polyflow: Continuous topology embedding flow matching for artist-style mesh generation. *arXiv preprint arXiv:2606.30673*, 2026a. 23

Chunshi Wang, Junliang Ye, Yunhan Yang, et al. Part-x-mllm: Part-aware 3d multimodal large language model. In *ICLR*, 2026b. 24

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*, 2026c. 3, 25, 26

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)*, 2024a. 24

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*, 2025a. 25

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*, 2026d. 25

Su Wang, Chitwan Saharia, Ceslee Montgomery, et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *CVPR*, 2023a. 24

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. 26

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*, 2026e. 1, 3, 25

Yuxuan Wang, Xuanyu Yi, Haohan Weng, et al. Nautilus: Locality-aware autoencoder for scalable mesh generation. In *ICCV*, 2025b. 23

Zhengyi Wang, Cheng Lu, Yikai Wang, et al. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2023b. 23

Zhengyi Wang, Jonathan Lorraine, Yikai Wang, et al. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024b. 23

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. Genartist: Multimodal llm as an agent for unified image generation and editing. In *NeurIPS*, 2024c. 25

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. 6, 29

Cong Wei, Zheyang Xiong, Weiming Ren, Xinrun Du, Ge Zhang, and Wenhu Chen. Omniedit: Building image editing generalist models through specialist supervision. *arXiv preprint arXiv:2411.07199*, 2024. 24

Haohan Weng, Yikai Wang, Tong Zhang, C. L. Philip Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. In *ICLR*, 2025a. 23

Haohan Weng, Zibo Zhao, Biwen Lei, et al. Scaling mesh generation via compressive tokenization. In *CVPR*, 2025b. 23

Jiawei Weng, Saining Zhang, Zhenxin Diao, et al. Feedforward 3d editing learns from semantic-part transformation. *arXiv preprint arXiv:2605.27351*, 2026. 1, 2, 5, 24, 25, 26, 29

Chenfei Wu, Jiahao Li, Jingren Zhou, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a. 4, 10, 24, 26

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)*, 2024a. 24

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. 25

Shuang Wu, Youtian Lin, Feihu Zhang, et al. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. In *NeurIPS*, 2024b. 23

Yongliang Wu, Zonghui Li, Xinting Hu, et al. Kris-bench: Benchmarking next-level intelligent image editing models. *arXiv preprint arXiv:2505.16707*, 2025b. 24

Ruihao Xia, Yang Tang, and Pan Zhou. Towards scalable and consistent 3d editing. *arXiv preprint arXiv:2510.02994*, 2025. 1, 2, 3, 5, 24, 25, 26, 29

Jianfeng Xiang, Zelong Lv, Sicheng Xu, et al. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, 2025. 2, 6, 23, 30

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, et al. Native and compact structured latents for 3d generation. In *CVPR*, 2026. 4, 10, 23, 26

Shitao Xiao, Yueze Wang, Junjie Zhou, et al. Omnigen: Unified image generation. *arXiv preprint arXiv:2409.11340*, 2024. 24

Bojun Xiong, Jialun Liu, Jiakui Hu, et al. Texgaussian: Generating high-quality pbr material via octree-based 3d gaussian splatting. In *CVPR*, 2025. 24

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. 25

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. 23

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. 25

Xinhao Yan, Jiachen Xu, Yang Li, et al. X-Part: High fidelity and structure coherent shape decomposition. *arXiv preprint arXiv:2509.08643*, 2025. 23

Yue Yang, Fan-Yun Sun, Luca Weihs, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *CVPR*, 2024a. 3, 25

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. 25

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, et al. SAMPart3D: Segment any part in 3D objects. *arXiv preprint arXiv:2411.07184*, 2024b. 23

Yunhan Yang, Yuan-Chen Guo, Yukun Huang, et al. HoloPart: Generative 3D part amodal segmentation. *arXiv preprint arXiv:2504.07943*, 2025a. 23

Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, et al. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. In *SIGGRAPH Asia*, 2025b. 23

Chongjie Ye, Yushuang Wu, Ziteng Lu, et al. Hi3dgen: High-fidelity 3d geometry generation from images via normal bridging. *arXiv preprint arXiv:2503.22236*, 2025a. 23

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)*, 2025b. 1, 25, 26

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*, 2026a. 24

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)*, 2026b. 1, 2, 4, 5, 24, 25, 26, 29

Ruijie Ye, Jiayi Zhang, Zhuoxin Liu, et al. Agent banana: High-fidelity image editing with agentic thinking and tooling. *arXiv preprint arXiv:2602.09084*, 2026c. 25

Yang Ye, Xianyi He, Zongjian Li, et al. ImgEdit: A unified image editing dataset and benchmark. *arXiv preprint arXiv:2505.20275*, 2025c. 24

Yuxiao Ye, Haoran He, Fangyuan Kong, et al. Edit-r2: Context-aware reinforcement learning for multi-turn image editing. *arXiv preprint arXiv:2606.05950*, 2026d. 25

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. 24

Kyeongmin Yeo, Yunhong Min, Jaihoon Kim, and Minhyuk Sung. Matlat: Material latent space for pbr texture generation. *arXiv preprint arXiv:2512.17302*, 2025. 24

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. 24

Qifan Yu, Wei Chow, Zhongqi Yue, et al. Anyedit: Mastering unified high-quality image editing for any idea. In *CVPR*, 2025. 24

Ruihan Yu, Lian Fu, Muyao Niu, et al. Kaininja: Extending native 3d generators to the part level. *arXiv preprint arXiv:2609.15659*, 2026. 23

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. 25

Xianfang Zeng, Xin Chen, Zhongqi Qi, et al. Paint3d: Paint anything 3d with lighting-less texture diffusion models. *arXiv preprint arXiv:2312.13913*, 2023. 23

Zilai Zeng, Mingdeng Cao, Zijie Li, et al. Towards robust sequential decomposition for complex image editing. In *CVPR*, 2026. 25

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)*, 2023a. 23

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*, 2026a. 23

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*, 2026b. 25

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023b. 24, 25, 26

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. 23

Longwen Zhang, Qixuan Zhang, Haoran Jiang, et al. Bang: Dividing 3d assets via generative exploded dynamics. *ACM Transactions on Graphics (SIGGRAPH)*, 2025a. 23

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023c. 2, 24

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. 6, 29, 36

Xuying Zhang, Yutong Liu, Yangguang Li, et al. Tar3d: Creating high-quality 3d assets via next-part prediction. In *ICCV*, 2025b. 23

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*, 2026c. 25

Yudi Zhang, Yeming Geng, and Lei Zhang. Scribblesense: Generative scribble-based texture editing with intent prediction. *IEEE Transactions on Visualization and Computer Graphics*, 2026d. 24

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*, 2025c. 24

Haozhe Zhao, Xiaojian Ma, Liang Chen, et al. Ultraedit: Instruction-based fine-grained image editing at scale. In *NeurIPS*, 2024. 24

Ruowen Zhao, Junliang Ye, Zhengyi Wang, et al. Deepmesh: Auto-regressive artist-mesh creation with reinforcement learning. In *ICCV*, 2025a. 23

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*, 2025b. 23

Xiangyu Zhao, Peiyuan Zhang, Kexian Tang, et al. Envisioning beyond the pixels: Benchmarking reasoning-informed visual editing. *arXiv preprint arXiv:2504.02826*, 2025c. 24

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*, 2026a. 25

Yuming Zhao, Zangyueyang Xian, Qijian Zhang, et al. Seamflow: Structure-aware flow matching on edge probabilities for artist-like uv unwrapping. In *SIGGRAPH Asia*, 2026b. 23

Zibo Zhao, Wen Liu, Xin Chen, et al. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *NeurIPS*, 2023. 23

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*, 2025d. 23

Wentao Zheng and Ancong Wu. Pwm-artgen: Part world model for articulated object generation. *arXiv preprint arXiv:2607.02045*, 2026. 23

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. 24

Matt Zhou, Ruining Li, Xiaoyang Lyu, et al. Articraft: An agentic system for scalable articulated 3D asset generation. *arXiv preprint arXiv:2605.15187*, 2026a. 25

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)*, 2026b. 2, 24, 25, 26

Zijun Zhou, Yingying Deng, Xiangyu He, Weiming Dong, and Fan Tang. Multi-turn consistent image editing. *arXiv preprint arXiv:2505.04320*, 2025. 24

Xinnan Zhu, Ruijie Xu, Jiayu Ying, et al. Jointedit3d: Feed-forward 3d scene editing in a unified latent space. *arXiv preprint arXiv:2606.13345*, 2026a. 2, 25

Yiheng Zhu, Kangle Deng, Jean-Philippe Fauconnier, et al. Cubepart: An open-vocabulary part-controllable 3d generator. In *SIGGRAPH*, 2026b. 23

Zixin Zhu, Haoxiang Li, Xuelu Feng, He Wu, Chunming Qiao, and Junsong Yuan. Georemover: Removing objects and their causal visual artifacts. In *NeurIPS*, 2025. 24

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. 24

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. 24

A Extended related work

3D generation. An influential line of text-to-3D generation optimizes a 3D representation against a 2D diffusion prior (Poole et al., 2023; Lin et al., 2023; Wang et al., 2023b), with related approaches using multi-view diffusion (Liu et al., 2023; Shi et al., 2024; Long et al., 2024) and feed-forward reconstruction (Hong et al., 2024; Xu et al., 2024). Current systems train a diffusion or flow model directly on a native 3D latent, which may be a vector set (Zhang et al., 2023a; Zhao et al., 2023; Zhang et al., 2024; Li et al., 2025e; Zhao et al., 2025d; Tencent Hunyuan3D Team, 2025; Li et al., 2025d) or a triplane (Wu et al., 2024b), or a sparse or structured voxel grid (Xiang et al., 2025; 2026; Li et al., 2025f; Ye et al., 2025a). All four conventional methods we evaluate build on the structured latent of TRELLIS (Xiang et al., 2025). A separate line generates low-polygon, artist-style meshes autoregressively, one face or one token at a time (Nash et al., 2020; Siddiqui et al., 2024a; Chen et al., 2024d; 2025f; 2024f; Weng et al., 2025a; Wang et al., 2024b; Tang et al., 2025a; Weng et al., 2025b; Zhao et al., 2025a; Lionar et al., 2025; Wang et al., 2025b; Song et al., 2026; Kim et al., 2026; Hao et al., 2024) or by flow matching in a mesh latent space (Li et al., 2026d; Wang et al., 2026a). These methods target compact mesh structure rather than instruction-guided changes to an existing asset. TAR3D (Zhang et al., 2025b) autoregressively predicts quantized triplane tokens, while SeamFlow (Zhao et al., 2026b) predicts cuts for UV unwrapping on an existing mesh. Part-aware modeling includes segmentation (Yang et al., 2024b; Liu et al., 2025b; Ma et al., 2025a), generative decomposition and completion (Yang et al., 2025a; Zhang et al., 2025a; Yan et al., 2025; Li et al., 2026c), and generation of objects as sets of parts (Chen et al., 2025c; Lin et al., 2025b; Yang et al., 2025b; Tang et al., 2025b; Chen et al., 2025d; Yu et al., 2026; Zhu et al., 2026b; Liao et al., 2026). Assembler (Zhao et al., 2025b) places input parts into a complete object, while SIMART and PWM-ArtGen address articulated assets (Zhang et al., 2026a; Zheng & Wu, 2026). Texture synthesis is a line of work of its own (Richardson et al., 2023; Chen et al., 2023; Zeng et al., 2023;

Yeh et al., 2024; Shao et al., 2025), including PBR material generation (Siddiqui et al., 2024b; Huang et al., 2024b; He et al., 2025; Chen et al., 2025g; Fei et al., 2026; Yeo et al., 2025; Xiong et al., 2025). The retexture turns of EditHero ask for this operation on one part while the rest is kept. CMD (Li et al., 2025c) 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 part structure of an object as the unit of editing and as the source of exact ground truth.

3D editing, complete list. The four conventional methods we evaluate represent the two main families of editors on native 3D latents. Trained editors learn from paired edits. Much as ControlNet (Zhang et al., 2023c) conditions an image model, they feed the source latent to the generator through an added control branch. PartFlow (Weng et al., 2026) does this with a source-control branch, and 3DEditFormer (Xia et al., 2025) does it with dual-guidance attention (see also Cai et al., 2025; Gat et al., 2026; Yin et al., 2026). Training-free editors reuse pretrained generators without edit-specific training. Inversion-based editors such as VoxHammer (Li et al., 2026b) invert the source along the sampling trajectory and reuse its cached latents and attention keys and values in the preserved region. Inversion-free editors follow the difference between the flow conditioned on the source and the flow conditioned on the target, which is what Nano3D (Ye et al., 2026b) does after FlowEdit (Kulikov et al., 2025), and related variants follow the same idea (Zhou et al., 2026b; Nie et al., 2026). 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. The full list follows. Early editing methods modify a neural field or a Gaussian splat with a 2D diffusion prior (Haque et al., 2023; Sella et al., 2023; Zhuang et al., 2023; Chen et al., 2024e; Wang et al., 2024a; Khalid et al., 2024; Zhuang et al., 2024; Wu et al., 2024a; Chen et al., 2024b), and some of them take sketches (Mikaeili et al., 2023; Liu et al., 2024) or drag handles (Qu et al., 2025) as conditions instead of text. Recent methods also use mesh reconstruction or native 3D latents (Chen et al., 2024c;a; Qi et al., 2024; Kim et al., 2025; Barda et al., 2025; Parelli et al., 2026; Zhou et al., 2026b; Gat et al., 2026; Hsiao et al., 2026; Chi et al., 2026; Chen et al., 2026; Yin et al., 2026; Ye et al., 2026a; Ai et al., 2026; Park et al., 2026). Other systems edit Gaussian scenes or scene atlases (Shu et al., 2025; Fang et al., 2026); object insertion in 3D scenes and 3D object inpainting are also studied separately (Li et al., 2025a; Feng et al., 2025). Part-level and interactive editors let a user act on one part at a time (Dong et al., 2024a;b; Wang et al., 2026b; Sella et al., 2026; Slim et al., 2026), and sketch-based systems have supported iterative refinement since Teddy (Igarashi et al., 1999; Nealen et al., 2007; Li et al., 2020; 2022; Guillard et al., 2021; Binninger et al., 2024; Zhou et al., 2023; Bandyopadhyay et al., 2024; Li et al., 2025b; Lee et al., 2025; Li et al., 2026a; Zhang et al., 2026d). Sketch2CAD (Li et al., 2020) and MeshPad (Li et al., 2025b) 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 (Meng et al., 2021; Avrahami et al., 2022; Hertz et al., 2022; Gal et al., 2022; Ruiz et al., 2023; Mokady et al., 2022; Couairon et al., 2022; Kawar et al., 2022; Tumanyan et al., 2022; Cao et al., 2023; Zhang et al., 2023c; Brack et al., 2024) to models trained on edit pairs (Brooks et al., 2022; Sheynin et al., 2024; Geng et al., 2023; Fu et al., 2024; Zhao et al., 2024; Hui et al., 2024; Yu et al., 2025; Wei et al., 2024; Qian et al., 2025) and to unified generators that edit in context (Xiao et al., 2024; Zhang et al., 2025c; Liu et al., 2025c; Labs et al., 2025; Wu et al., 2025a; Deng et al., 2025). Geometry-aware image editing uses 3D cues for spatial transformations or object removal (Bhat et al., 2026; He et al., 2026; Zhu et al., 2025), and BlenderFusion (Chen et al., 2025b) 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 (Brochu et al., 2010; Gonzalez et al., 2017; Koyama et al., 2017; 2020). Recent work applies the same idea to generative image search or adapter merging (Rajagopalan et al., 2026; Liu et al., 2026a). That work treats editing as a sequence from the start, and what it studies is how the state converges over many rounds. Benchmarks for image editing (Wang et al., 2023a; Zhang et al., 2023b; Basu et al., 2023; Ma et al., 2024; Ye et al., 2025c; Zhao et al., 2025c; Wu et al., 2025b; Ge et al., 2024) 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 (Zhang et al., 2023b; Zhou et al., 2025;

Ma et al., 2026; Huang et al., 2026a; Zeng et al., 2026; Yang et al., 2026; Chen et al., 2025e). DiffSeg30k (Ci et al., 2025) instead uses multi-turn edits to benchmark localized detection of generated content. MagicBrush provides intermediate target images reviewed by human annotators (Zhang et al., 2023b). EditHero instead supplies complete 3D reference states that can be rebuilt deterministically, with the edited parts known at every turn. Multi-turn editors respond with memory of earlier turns or reinforcement learning over the sequence (Zhang et al., 2026b; Xu et al., 2026; Ye et al., 2026d). Repeated image replication can accumulate artifacts that no-reference quality metrics fail to capture (Tang et al., 2026). Related work studies different sources of accumulated error: repeated training on synthetic outputs (Alemohammad et al., 2024; Zhang et al., 2026c) and exposure bias within diffusion sampling (Ning et al., 2024). We study successive 3D edits at inference time, where holes, loose pieces and displacement can be computed directly.

Agentic generation and editing by code, complete list. LLMs can build 3D content by writing procedural code or calling tools. They do this for scenes and layouts (Sun et al., 2023; De La Torre et al., 2024; Hu et al., 2024; Yang et al., 2024a; Feng et al., 2023; Tam et al., 2025) and for editing an existing scene in Blender from renders (Huang et al., 2024a; Lu et al., 2025; Wang et al., 2026c; Pang et al., 2026). Related code-based systems generate objects and CAD programs (Yuan et al., 2024; Jones et al., 2025; Dai et al., 2025; Khan et al., 2024; Rukhovich et al., 2024; Alrashedy et al., 2024; Mallis et al., 2024; Le et al., 2025). Recent systems treat the program as the asset itself, with an agent that plans, writes and revises procedural code under visual feedback (Lin et al., 2026; Wang et al., 2026d; Noor et al., 2026; Liu et al., 2026b; Ning et al., 2026; Ataei et al., 2026). EmbodiedGen V2 (Wang et al., 2026e) describes this as vibe coding of 3D worlds through a dialogue of instructions, in which every edit is local and keeps the rest of the state. This is the workflow that EditHero measures. Agentic pipelines exist for images as well (Wang et al., 2024c; Liu et al., 2025d; Chen et al., 2025a; Xue et al., 2024; Ma et al., 2025b; Ye et al., 2026c; Zhao et al., 2026a; Liang et al., 2025; Wu et al., 2026; Huang et al., 2026c;b). Related MLLM-guided editors and unified generators use learned multimodal reasoning (Huang et al., 2023; Fang et al., 2025; Lin et al., 2025a). Benchmarks for these agents (Gu et al., 2025; Gao et al., 2026) score the final state against one goal.

Ground truth by construction, complete list. Part-annotated collections (Chang et al., 2015; Mo et al., 2019; Wang et al., 2025a; Ding et al., 2025) and procedural asset generators (Zhou et al., 2026a; Raistrick et al., 2023) provide objects and parts for assembly. EditHero's chains are closer in spirit to synthetic benchmarks whose targets are produced by a program (Johnson et al., 2017; Greff et al., 2022; Deitke et al., 2022) than to annotated datasets. The assembly engine that produces each state is deterministic, and rebuilding a turn from its logged operations reproduces the state exactly. What is not automatic is quality, so every chain went through manual review (Appendix B).

Comparison with existing benchmarks. 3DEditVerse (Xia et al., 2025) combines pose changes with image-guided appearance edits, while Nano3D-Edit-100k (Ye et al., 2026b) uses a native 3D editor guided by edited images. Uni3DEdit-Bench (Weng et al., 2026) uses semantic-part transformations, and Delta3D (Preintner et al., 2026) removes parts from CAD assemblies. Steer3D's EDIT3D-BENCH (Ma et al., 2025c) and the editing subset of 3D-Alpaca (Ye et al., 2025b) reconstruct edited images into target assets. Edit3D-Bench (Li et al., 2026b), Eval3DEdit (Zhou et al., 2026b), BenchUp (Gat et al., 2026) and EditFlow-Bench (Nie et al., 2026) have no target asset, so they score region preservation and alignment to an edited image instead. Scene-level benchmarks follow the same pattern (Patro et al., 2026; Liu et al., 2025a; Zhu et al., 2026a). Each sample specifies a single editing goal, represented by an asset or an edited image. BlenderGym (Gu et al., 2025) and 3DCodeBench (Gao et al., 2026) let an agent iterate toward one fixed goal, and EZBlender (Wang et al., 2026c) bundles several edits into one prompt. Table 4 places EditHero next to these benchmarks. The columns follow the properties that the evaluation relies on, which are whether a target state exists and how it is produced, whether instructions form a sequence, whether the correct state is known after every turn, and whether a method is scored on its own previous output.

B Benchmark construction details

Why a forward construction. Section 3 explains why the targets are assembled forward instead of lifted from edited images. Assembling forward raises two problems of its own. The chosen part

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 3D editor 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 (Zhang et al., 2023b)	2D image	annotated image	sequence (few)	✓	✓
3DEditVerse (Xia et al., 2025)	3D	pose edits / 2D lift	1	–	–
Nano3D-Edit-100k (Ye et al., 2026b)	3D	native edit	1	–	–
Uni3DEdit-Bench (Weng et al., 2026)	3D	semantic-part edits	1	–	–
EDIT3D-BENCH (Steer3D) (Ma et al., 2025c)	3D	2D lift	1	–	–
Delta3D (Preintner et al., 2026)	3D	part removal	1	–	–
3D-Alpaca (Ye et al., 2025b)	3D	2D lift	1	–	–
Edit3D-Bench (Li et al., 2026b), Eval3DEdit (Zhou et al., 2026b)	3D	none	1	–	–
BenchUp (Gat et al., 2026), EditFlow-Bench (Nie et al., 2026)	3D	none	1	–	–
BlenderGym (Gu et al., 2025), 3DCodeBench (Gao et al., 2026)	3D scene, code	goal state	goal	–	✓
EZBlender (Wang et al., 2026c)	3D scene	–	bundle	–	–
EditHero (ours)	3D, part-level	program-built state	sequence (2–30)	✓	✓

has to make sense on the object (a hat on a head or a barrel on a porch, but not a wheel on a teapot), and it has to be attached where and how a person would attach it. For reference construction, we need accepted placements to satisfy explicit geometric constraints and remain 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 (Ding et al., 2025), whose underlying objects come from Objaverse-XL (Deitke et al., 2023). We use its textured part meshes and supplied part decomposition. For retrieval, we generate part descriptions with Qwen3-VL-32B-Instruct (Qwen Team, 2025); these descriptions are distinct from the editing instructions subsequently reviewed by people. Each chain starts from a textured, part-segmented object, which we call the *host*. Its parts are grouped into a small number of *slots*, which are subassemblies such as a head, an arm or a porch canopy that an instruction can name and that are 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. In our experience automatic grouping places parts correctly but names them badly, so naming is done by hand.

Operations. *Add* places a new part from the library on a free face of an existing part, *remove* deletes a named slot or a recorded set of slots, *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 (Wu et al., 2025a) and is applied to the unchanged slot geometry by the texturing module of TRELLIS.2 (Xiang et al., 2026). When a support part is edited, the parts attached to it are detached in separate turns, so every turn stays one operation on one named part. Instructions describe parts in natural words and never contain internal slot identifiers or library metadata.

Placement. Placement is deterministic, in the spirit of procedural datasets whose labels come from the generating program (Johnson et al., 2017; Greff et al., 2022; Raistrick et al., 2023). For a replacement or an add-on part, the engine retrieves candidates by the similarity of their caption embeddings to the requested part (Wang et al., 2020; Reimers & Gurevych, 2019), 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 (within 0.75 to 2 times its original size), snaps it into contact, and applies four checks during automatic candidate screening. 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 $S_k = o_k(\cdots o_1(S_0))$ 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). Flags were fixed by editing the logged operations and rebuilding the affected turns, never by editing meshes directly. 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* focuses on material edits, with two chains also containing additions and removals. Family *G* mixes geometry operations, and family *X* mixes geometry and material operations. The released set contains 144 A, 81 M, 50 R, 57 P, 89 G and 64 X chains.

C Evaluation protocol details

Self-rollout. A method receives the initial mesh and, at turn k, the instruction together with any condition it needs (a target image, a mask). Its input state at turn k is its own output at turn $k-1$. The ground truth is never re-encoded and there is no reset. If a method fails a turn, for instance with an empty or unloadable output, the later turns of that chain are recorded as blocked and the chain counts as a failure. We never restart from ground truth to rescue a chain. Infrastructure faults are handled separately (Appendix F).

Conditioning. Some methods need a target image or a target mask. To give every method the same target images, we render every ground-truth state once with a single camera rig per chain (below) and give each image-conditioned method the target render of the current turn. Mask-conditioned methods receive the ground-truth edit region under the same rig. This is a strong condition, because the target image shows what the edited object should look like. We therefore treat all methods as working with target-image access and report it as a condition of the experiment.

One frame per chain, one camera that sees every edit. All states of a chain, ground truth and predictions alike, are measured in one normalization taken from the union bounding box of all ground-truth states of that chain, and they are rendered with one camera rig taken from the bounding sphere of all ground-truth vertices. The rig has four views 90° apart at one elevation, a fixed focal length and fixed lighting. Re-centering per turn or per part would hide displacement, and a per-turn camera would make image metrics incomparable across turns. A single default direction, however, does not guarantee that the edited part is visible. With one fixed azimuth and elevation for every chain, many small additions on the far side of the object change almost nothing 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. Surface samples of that region are ray-cast toward candidate directions against the full state to estimate the visible projected area, and the direction that makes the worst turn most visible is fixed

for the chain (Appendix E). After rendering, chains are selected by the measured pixel change. A chain enters the evaluation only if every turn changes at least 0.5% of the frame in the conditioning view. Three chains just below the threshold were kept after a visual check of their ground-truth renders showed that every edit is visible; this check used no method output. This selects 457 chains and 2750 edits. 16 chains have no such direction and 12 still fall below the threshold, and all of them stay in the released data. The rig uses the extent of future states and is therefore a spatial prior. It is not passed to any method as geometry, and each method still sees the object in its own native frame, fixed at the initial state.

Why these definitions. Each choice in Section 4 removes a way in which a method can score well without editing, and Figure 6 shows one real turn for each. Whole-object similarity is dominated by content that no instruction touches, so returning the initial object can score higher than completing the requested local edit on an already drifted state (Figure 6b). Splitting every metric into E_k, U_k and the whole object separates doing the edit from leaving the rest alone. The regions come from the named parts rather than from voxel or pixel differences, because a retexture has no voxel difference at all even when a method regenerates the whole object (Figure 6d). The union of both states lets one definition cover all four operations, and the one-cell dilation absorbs a part that lands one cell off, so the same misplacement lowers IF but is not counted again as damage in CC. IF and CC are anchored to the model's own previous state because under self-rollout that state differs from the ground truth from the first turn. Against the ground truth, a part lost at an earlier turn counts as a successful removal now (Figure 6a), and a replacement that only deletes comes within 0.02 of the F-score of the no-op baseline (Figure 6c). The ground-truth-anchored $m_{E_k}(\hat{S}_k, S_k)$ is still reported, since prior work uses it. Together with IF, it separates resemblance to the target from what the model changed at the current turn.

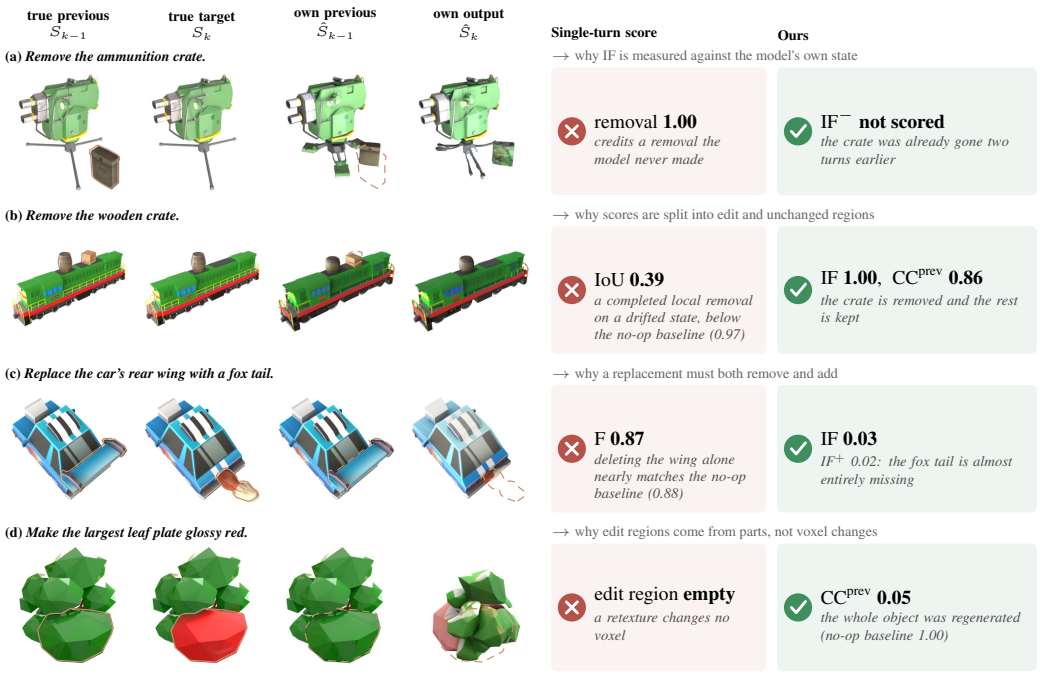

Figure 6: **Why the metrics are defined on regions and on the model's own state.** Four turns from the evaluation, in the conditioning view. Outlines mark the edited parts before and after the turn, dashed on the model's own states. For each turn, the red card gives the score a single-turn benchmark would report and why it misleads, and the green card gives our score (Eqs. (1)–(5)); the line above the cards 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.

$$\begin{aligned} \mathrm{IoU}_R(X, Y) &= \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|(\mathcal{V}(X) \cup \mathcal{V}(Y)) \cap R|}, \\ \mathrm{Recall}_R(X, Y) &= \frac{|\mathcal{V}(X) \cap \mathcal{V}(Y) \cap R|}{|\mathcal{V}(Y) \cap R|}. \end{aligned} \tag{6}$$

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

$$\begin{aligned} \mathrm{F}^{\tau}_R(X, Y) &= \frac{2\,\mathrm{Pr}\,\mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}}, \\ \mathrm{Pr} &= \operatorname*{mean}_{p \in \mathcal{Q}(X) \cap R} \mathbb{I}\big[d(p, Y) < \tau\big], \\ \mathrm{Re} &= \operatorname*{mean}_{q \in \mathcal{Q}(Y) \cap R} \mathbb{I}\big[d(q, X) < \tau\big]. \end{aligned} \tag{8}$$

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

$$\begin{aligned} \mathrm{MAE}_R(I, J) &= \operatorname*{mean}_{p \in R} \frac{1}{3} \|I(p) - J(p)\|_1, \\ \mathrm{PSNR}_R(I, J) &= -10 \log_{10} \left(\operatorname*{mean}_{p \in R} \frac{1}{3} \|I(p) - J(p)\|_2^2 \right), \end{aligned} \tag{9}$$

and $\mathrm{SSIM}_R(I, J)$, $\mathrm{LPIPS}_R(I, J)$ are the SSIM (Wang et al., 2004) and LPIPS (Zhang et al., 2018) maps of (I, J) averaged over the pixels of R. $\mathrm{DINO}_R(I, J)$ 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 two main comparisons are in Table 6. Image metrics are computed in the four fixed views against the ground-truth render, composited on black, with the LPIPS map of AlexNet (Zhang et al., 2018), the SSIM map (Wang et al., 2004) and the patch features of DINOv2 ViT-B/14 (Oquab et al., 2023) (16×16 patches), each averaged over the region.

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 (Weng et al., 2026), Nano3D (Ye et al., 2026b), 3DEditFormer (Xia et al., 2025), and VoxHammer (Li et al., 2026b) with their released code and weights. Each method is

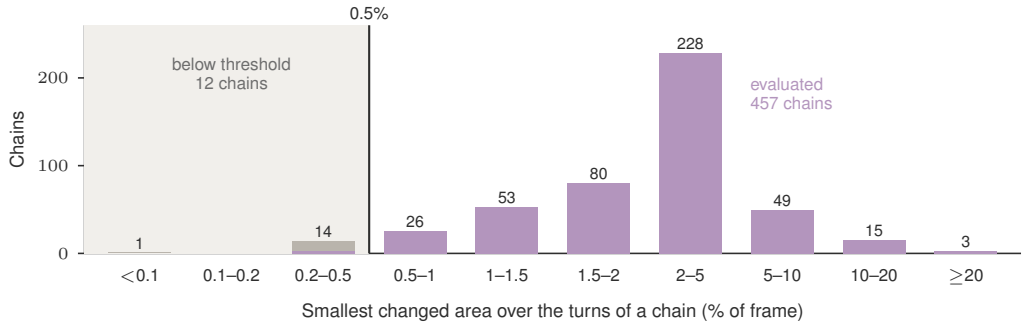

Figure 7: **Visibility-based selection of evaluation chains.** For each of the 469 released chains that have a camera direction, the smallest fraction of the frame that any of its turns changes in the conditioning view (ground-truth state k against state $k-1$, fixed rig). A chain is evaluated when every turn changes at least 0.5% of the frame; three chains below the line were included after visual review. 457 chains are evaluated, 12 other chains fall below the threshold, and 16 further chains have no direction under which every edit is visible; all excluded chains stay in the released data.

wrapped in an adapter that only supplies inputs and records outputs, and the sampling code, seeds and hyperparameters are those of the authors. PartFlow, Nano3D and 3DEditFormer take the current state and the target render. VoxHammer also takes the ground-truth edit mask. Easy3E (Hu et al., 2026) is not included because its released code conditions its guidance on one of 16 fixed orthographic views at zero elevation, so it cannot receive the target render in the fixed perspective view of a chain that the other methods receive. Nano3D's official interface has add, remove and replace modes but no material mode. Under the official routing a material instruction is rejected and the rest of the chain is blocked, which happens on the 132 evaluation 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 four methods, exported through TRELLIS, carry no metallic value and would render as dark metal, so we render them as non-metallic. The LLM outputs of Section 5.4 set their own materials and are rendered as exported.

The TRELLIS regeneration reference runs the released TRELLIS-image-large pipeline (Xiang et al., 2025) on the target render of every turn, in the same fixed view that the conventional methods receive. It sees neither the instruction nor its previous output, so its turns are independent. Its output lies in the normalized frame of TRELLIS, which from one view does not fix the orientation, and on some turns of a chain 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 2750 turns of the 457 chains.

Table 5 reports, per method, the chains completed, the turns that succeeded, the turns blocked after a failure, 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 failures that remain are properties of the methods. 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 under its official routing, but with the replace routing above it completes all chains except those two. We keep these as failures rather than drop the chains, because a method that cannot run a turn has not completed the chain.

G WHOLE-OBJECT METRICS, PER-TURN CURVES AND END STATE

Table 7 reports the whole-object and preservation metrics under the part-based protocol of Section 4 on the chains that every method completes, so that every method is scored on the same executed turns (excluding undefined metric values), and over all successful turns for the methods that complete almost every chain. The metric means in the two groups agree to within 0.015. The no-op baseline

Table 5: Execution coverage on the 457 evaluation 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 successful turns.

Method	Cond.	Chains	Turns	Blocked	Failure category
PartFlow	image	457/457	2750/2750	0	–
Nano3D	image	455/457	2744/2750	6	model no edited voxels (2)
3DEditFormer	image	457/457	2750/2750	0	–
VoxHammer	image + mask	418/457	2564/2750	186	model empty sparse output (39)

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
Main evaluation (457 chains)				
PartFlow	1875	960	411	504
Nano3D	1925	961	434	530
3DEditFormer	1891	966	410	515
VoxHammer	1746	894	371	481
No-op baseline	1810	946	381	483
TRELLIS regen.	1793	967	347	479
Code-editing comparison (55 chains)				
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
Opus 5.5	300	125	71	104
Fable 5.1	286	125	60	101

has a higher surface IoU to the target than every method (0.71 against at most 0.58 on these chains), and its LPIPS to the target render is lower than every method's. Nano3D has the highest IoU and the highest preservation of the four, and its IF is close to that of VoxHammer. VoxHammer reaches its IF with an untouched-region IoU to the initial object of only 0.45 (CC^0). The TRELLIS regeneration reference, which builds every turn from the target render alone, sits at the other extreme. It has the highest IF on these chains (0.51) together with the lowest IoU (0.30) and the lowest preservation (CC^{prev} 0.39), because it reconstructs the whole object from the target image.

Figure 8 and Table 8 follow the metrics turn by turn on the 416 chains completed by every method, with 95% intervals from a bootstrap (Efron, 1979) over hosts (chains built on the same host are not independent). Similarity to the target goes down with the turn index, and so does the no-op baseline's, because the target moves further from the initial object, and the gap between the two does not close. The TRELLIS regeneration reference, whose turns are independent of one another, stays almost flat (IoU 0.31 at turn 1 and 0.27 at turn 7), but this alone does not separate target difficulty from inherited errors. 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 say where a chain ends up. Table 9 therefore scores the last turn of every chain of this set, grouped by chain length, next to the first turn of the same chains. All four conventional methods lose similarity and preservation from the first to the last turn in every length group, with larger losses on the longest chains than on the shortest. The IoU of PartFlow falls by 0.10 on chains of 2–3 turns

Table 7: Whole-object metrics of the self-rollout runs. Upper block: the 416 chains on which every method succeeded on every turn, so all rows share the same executed turns; IF excludes undefined cases. Lower block: all successful turns of the 457 chains, for the methods with full or near-full coverage. IF: instruction following, anchored to the model's own previous state; CC^{prev}, CC^0: preservation of the untouched regions against the method's own previous state and the initial state; drift: mean change in the unchanged image region against the method's own previous render. The no-op baseline and the TRELLIS regeneration reference are evaluated on the same turns and are not bolded. Best value per column in bold over the four conventional methods of the upper block.

Method	n	IoU↑	IF↑	CC^{prev} ↑	CC^0 ↑	LPIPS↓	drift↓
416 chains completed by every conventional method							
PartFlow	2467	0.515	0.125	0.862	0.673	0.177	0.014
Nano3D	2467	**0.583**	0.316	**0.887**	**0.730**	**0.160**	**0.009**
3DEditFormer	2467	0.414	0.201	0.774	0.490	0.198	0.018
VoxHammer	2467	0.380	**0.321**	0.715	0.452	0.220	0.022
No-op baseline	2467	0.712	0.000	1.000	1.000	0.144	0.000
TRELLIS regen.	2467	0.296	0.507	0.385	0.282	0.203	0.041
All successful turns (methods with full or near-full coverage)							
PartFlow	2750	0.507	0.122	0.862	0.666	0.181	0.014
Nano3D	2744	0.575	0.311	0.887	0.718	0.163	0.009
3DEditFormer	2750	0.408	0.196	0.773	0.481	0.203	0.018
No-op baseline	2750	0.700	0.000	1.000	1.000	0.151	0.000

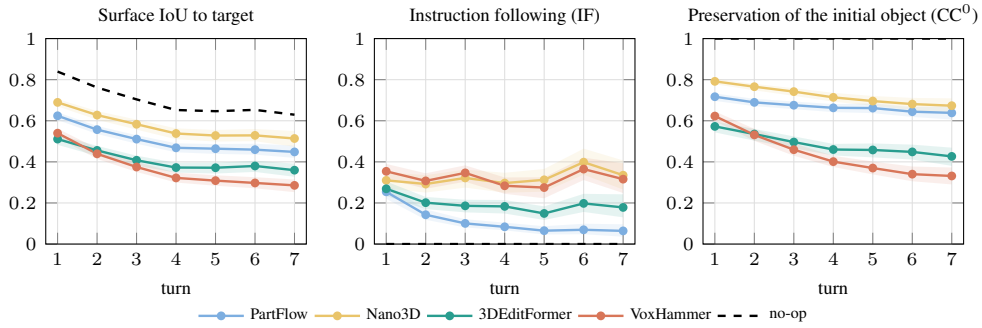

Figure 8: Per-turn metrics on the 416 chains that every method completes (228 hosts). Shaded: 95% host-clustered bootstrap intervals. The dashed line is the no-op baseline, which has IF $= 0$ and $CC^0 = 1$ by definition; its IoU compares the unchanged initial object with the current target.

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 0.82–0.88 at the first turn to 0.61–0.77 at the end, because the target moves away from the initial object. End-state IoU therefore has to be read against this baseline, and no method ends closer to the target than the object it started from. The TRELLIS regeneration reference, 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 can have two sources. The later instructions may be harder, or the state that the model hands to itself may be worse. To tell them apart we rerun Nano3D and 3DEditFormer on 200 chains with one change, which is that 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 two conditions per turn on the same chains, and Figure 4(c) follows only the 147 chains with at least five turns, so the turn-1 values of the two differ slightly.

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 reference, whose turns are independent.

Turn	1	2	3	4	5	6	7
n chains	416	416	414	373	301	233	135
IoU							
PartFlow	0.624	0.557	0.511	0.468	0.464	0.459	0.448
Nano3D	0.690	0.628	0.583	0.538	0.528	0.529	0.514
3DEditFormer	0.511	0.456	0.408	0.372	0.371	0.380	0.360
VoxHammer	0.540	0.439	0.375	0.322	0.309	0.297	0.286
No-op baseline	0.839	0.762	0.704	0.652	0.647	0.653	0.630
TRELLIS regen.	0.305	0.300	0.300	0.295	0.293	0.285	0.274
IF							
PartFlow	0.254	0.143	0.100	0.084	0.065	0.070	0.064
Nano3D	0.310	0.292	0.322	0.297	0.313	0.398	0.335
3DEditFormer	0.269	0.201	0.186	0.183	0.149	0.198	0.178
VoxHammer	0.354	0.307	0.346	0.284	0.275	0.365	0.316
No-op baseline	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TRELLIS regen.	0.535	0.472	0.517	0.492	0.483	0.534	0.535
CC^0							
PartFlow	0.717	0.689	0.676	0.663	0.662	0.644	0.638
Nano3D	0.792	0.766	0.742	0.714	0.696	0.681	0.673
3DEditFormer	0.573	0.535	0.496	0.460	0.458	0.448	0.427
VoxHammer	0.623	0.531	0.459	0.401	0.370	0.340	0.331
No-op baseline	1.000	1.000	1.000	1.000	1.000	1.000	1.000
TRELLIS regen.	0.302	0.293	0.292	0.277	0.276	0.257	0.248

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	2–3 turns (43 chains)		4–6 turns (238 chains)		7–10 turns (124 chains)		11–30 turns (11 chains)	
Method	IoU	CC^0	IoU	CC^0	IoU	CC^0	IoU	CC^0
PartFlow	0.66 → 0.56	0.70 → 0.67	0.61 → 0.43	0.72 → 0.63	0.63 → 0.43	0.72 → 0.62	0.67 → 0.47	0.72 → 0.57
Nano3D	0.73 → 0.67	0.78 → 0.75	0.68 → 0.50	0.80 → 0.68	0.69 → 0.50	0.78 → 0.64	0.75 → 0.50	0.81 → 0.58
3DEditFormer	0.55 → 0.45	0.59 → 0.53	0.49 → 0.33	0.55 → 0.40	0.54 → 0.36	0.60 → 0.42	0.58 → 0.24	0.65 → 0.24
VoxHammer	0.57 → 0.46	0.61 → 0.50	0.53 → 0.28	0.63 → 0.35	0.54 → 0.29	0.61 → 0.33	0.57 → 0.22	0.61 → 0.22
No-op baseline	0.88 → 0.77	1.00 → 1.00	0.82 → 0.61	1.00 → 1.00	0.85 → 0.62	1.00 → 1.00	0.86 → 0.64	1.00 → 1.00
TRELLIS regen.	0.28 → 0.30	0.28 → 0.29	0.31 → 0.31	0.30 → 0.28	0.31 → 0.28	0.32 → 0.25	0.28 → 0.28	0.28 → 0.21

The inherited state explains much of the decline in this diagnostic. 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, $CC_{\Delta}^{\mathrm{prev}}$ measures preservation outside the one-cell-dilated voxel difference between the two ground-truth states, rather than the part-based region of Eq. equation 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. A model that starts from a state it did not produce changes more of the untouched region, as it does at turn 1, while continuing from its own output reproduces that output closely. Preserving one's own state and preserving the correct state are different things, and the two should not be conflated.

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, which excludes the parts of the

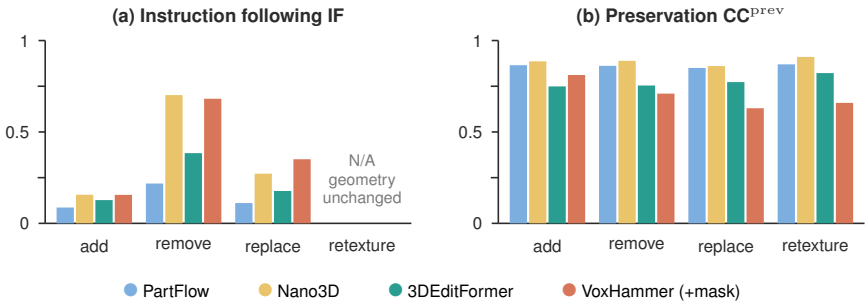

Figure 9: **Metrics by operation.** Means over successful, defined turns. Retexture IF is undefined (N/A, not zero). Coverage and conditioning differ by method; the methods are not compared on the same turns here.

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
Nano3D								
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^{prev}_{Δ}	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
3DEditFormer								
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^{prev}_{Δ}	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

named slot whose color stays the same. 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 change the region by 0.68–0.98 of the ground-truth amount 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 about that of not editing at all. Every one of its material chains completed, and on these chains its whole-object IoU (0.67) is the highest of the four methods, because these ground-truth edits do not require geometry changes. Figure 10 shows what the other methods do instead. Their outputs also change the appearance of untouched parts, so the errors spread over the whole object rather than stay on the named part. PartFlow and Nano3D produce slightly brighter renders than the target on average and 3DEditFormer slightly darker, while VoxHammer grows darker along the chain and is darker than the target on 90% of the retexture turns. Nano3D completes every material chain and scores well on similarity, yet it changes the named material less than any other method.

J EDITING BY CODE: 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. The frameworks differ, so each comparison is between a model together with its framework. Every chain is edited by its own independent agent, which starts a fresh session and shares no memory with other chains. Each model receives the source mesh, the chain's instructions and target renders, and the camera

Table 11: Change inside the ground-truth changed-pixel region on retexture turns (view rows are turn × view; a view in which the ground-truth change covers fewer than 50 pixels is skipped, 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 the unedited state (*no edit*).

		Own change	Residual to target	
Method	Views	/ GT change	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

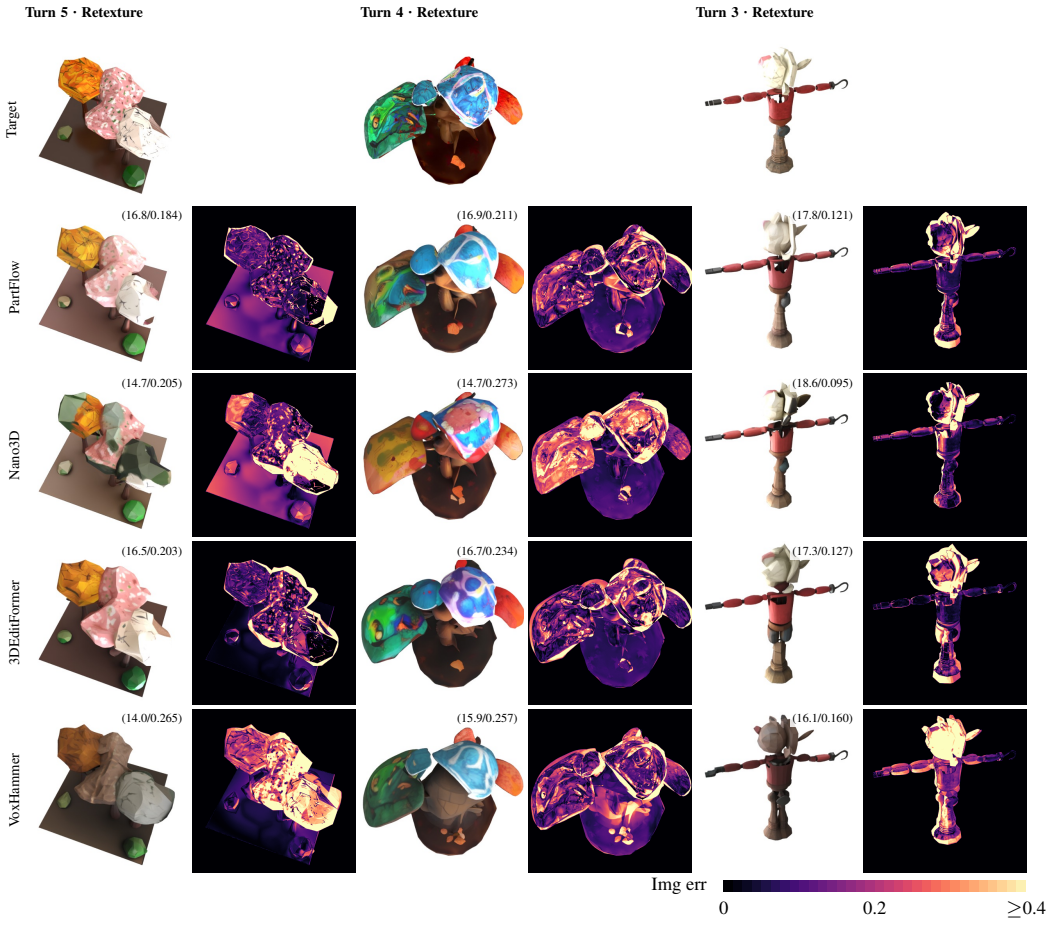

Figure 10: Retexture turns from three chains in the conditioning view. For each method the left tile is its render and the right tile its per-pixel error to the target render (mean absolute RGB difference; color bar at the bottom right). The numbers in the top-right corner are (PSNR in dB / LPIPS) of the render against the target view. The ground truth changes only the color of one part, while the errors of the conventional methods spread over the whole object.

parameters. It edits its own previous output and saves a full mesh after every turn. All models use the same base editing prompt (Appendix O) and our adaptation of img2threejs v2.0.0 (img2threejs contributors, 2026), which 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 three hours. GLM 5.3 Flash and DeepSeek V4.1 Flash had 200 model responses and three 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 conventional methods.

Cost. Table 12 reports the amortized cost per edit for all six 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.

Table 12: Cost per delivered edit. Conventional methods: median time per successful turn on one local H100 (90th percentile in parentheses). All LLM 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)	–	–
Sol 6 (medium)	87 (183)	848k / 2.67k	7.8
Astra (medium)	95 (143)	325k / 2.36k	5.8
Opus 5.5	149 (291)	1216k / 9.20k	10.9
Fable 5.1	258 (557)	1555k / 18.26k	9.3
GLM 5.3 Flash	358 (619)	680k / 4.66k	12.8
DeepSeek V4.1 Flash	358 (1036)	1697k / 14.13k	15.9

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 add the symmetric Chamfer distance (Fan et al., 2016) and the 95th percentile of the two-sided nearest-neighbor distance (HD95 (Huttenlocher et al., 1993; Taha & Hanbury, 2015)), which shows floating fragments and spikes that a mean hides. We also add normal consistency (NC (Mescheder et al., 2018)) and the F-score at two thresholds (Knapitsch et al., 2017; Tatarchenko et al., 2019). The same HD95 and NC are also restricted to an untouched region (subscript u), which here lies outside the dilated voxel difference $S_0 \triangle S_k$ between the initial and the target state. This region can include parts that were edited and later restored, whereas the cumulative part-based region of Eq. equation 5 excludes them. The last geometry column (Δfrag.) 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 indicate fewer such components, which need not mean a better reconstruction. For images we report PSNR, LPIPS (Zhang et al., 2018), silhouette IoU from the alpha channel and DINOv2 ViT-B/14 cosine similarity (Oquab et al., 2023) to the target render. They are given separately for the conditioning view (subscript 0), which every image-conditioned method sees, and for the three held-out views 90° apart (subscript 1-3), which no method sees. $\text{DINO}_{\text{retex}}$ is the similarity in the conditioning view on retexture turns only. All numbers are means over successful turns, except Δfrag., which is a median.

Three things are worth noting. We observe little average gap between conditioning and held-out views for the four conventional methods (PSNR within 0.5 dB, LPIPS within 0.01). In the untouched region the no-op baseline has an HD95 of 0.009, and every conventional method is three to eight

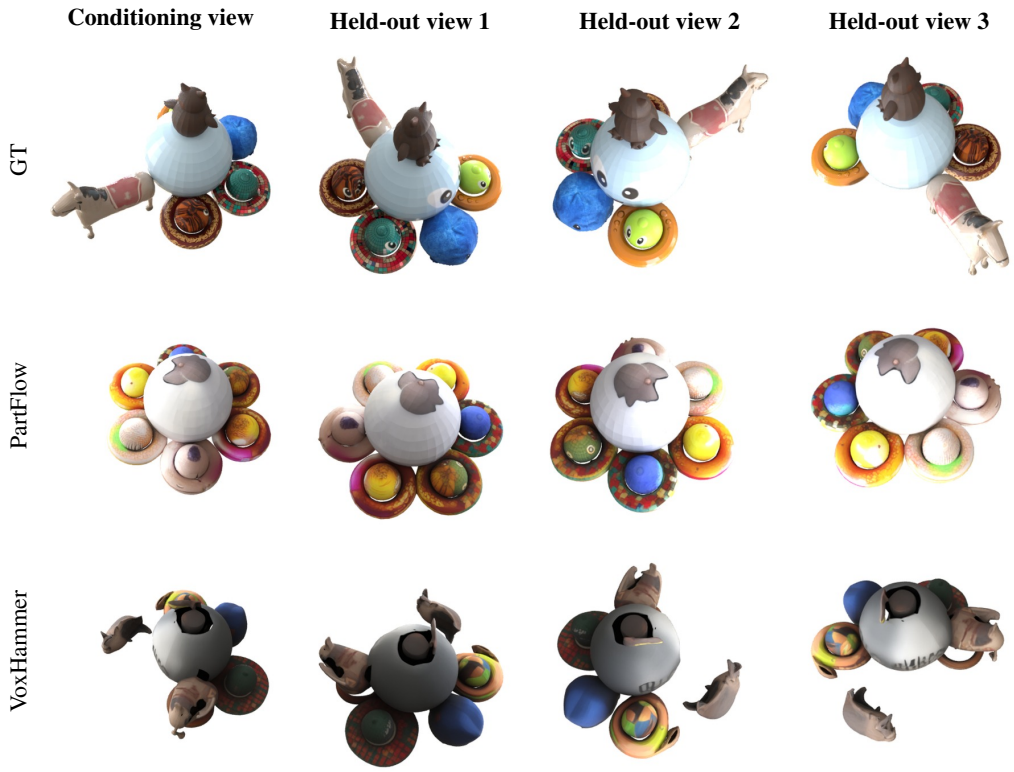

Figure 11: **Conditioning and held-out views.** GT, PartFlow and VoxHammer at turn 6 of one chain, in the conditioning view and the three held-out views. No method sees the held-out views, although the edit mask that VoxHammer receives carries 3D information about the edit.

times higher. 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 13: Extended geometry metrics over successful turns on the 457 evaluation chains. Subscript u: untouched region; Δfrag.: median fragment-count difference relative to ground truth.

Method	CD↓	HD95↓	NC↑	$F_{.01}$ ↑	$F_{.02}$ ↑	$HD95_u$ ↓	NC_u ↑	Δfrag.
PartFlow	0.022	0.115	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

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

Method	$PSNR_0$	$PSNR_{1\text{-}3}$	$LPIPS_0$ ↓	$LPIPS_{1\text{-}3}$ ↓	Sil_0 ↑	$Sil_{1\text{-}3}$ ↑	$DINO_{1\text{-}3}$ ↑	$DINO_{retex}$ ↑
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	15.2	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

L MORE QUALITATIVE COMPARISONS

Figures 12 and 13 follow two more evaluation chains. Rows are methods and columns are the first five turns; under each method's output is its per-pixel error to the ground truth of that turn (mean absolute RGB difference). One is a table setting whose edits are mostly material changes, and the other is a house whose porch gains and loses parts.

M LLMS NEXT TO CONVENTIONAL METHODS

Figures 14–16 show three chains of the LLM set with every method side by side. The four conventional methods come first, followed by the six LLMs that edit by code in the order of Table 3. Columns are the first five 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 conventional methods drift in color and shape even on turns that ask for a small change, while most LLMs change the named part and leave the rest of the object as it was.

N QUALITATIVE EXAMPLES OF CHAINS

Figures 17–26 show ground-truth chains from EditHero, one block per chain and one card per state from the source to the last turn, with the operation and the instruction printed on each card. Geometry-only chains are shown with one flat color per slot, so that added, removed and replaced parts can be told apart, and chains with material edits are shown with the assets' own textures.

O PROMPTS

The following templates illustrate material restyling and replacement-instruction drafting; reviewers revised the generated instructions (Section 3). The prompt given to the LLMs that edit by code 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¿.

Task prompt for editing by code (excerpt). The prompt given to every LLM that edits by code for every chain starts with the task below. It continues with the code-modelling skill and the rendering and resource rules, which are 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
```

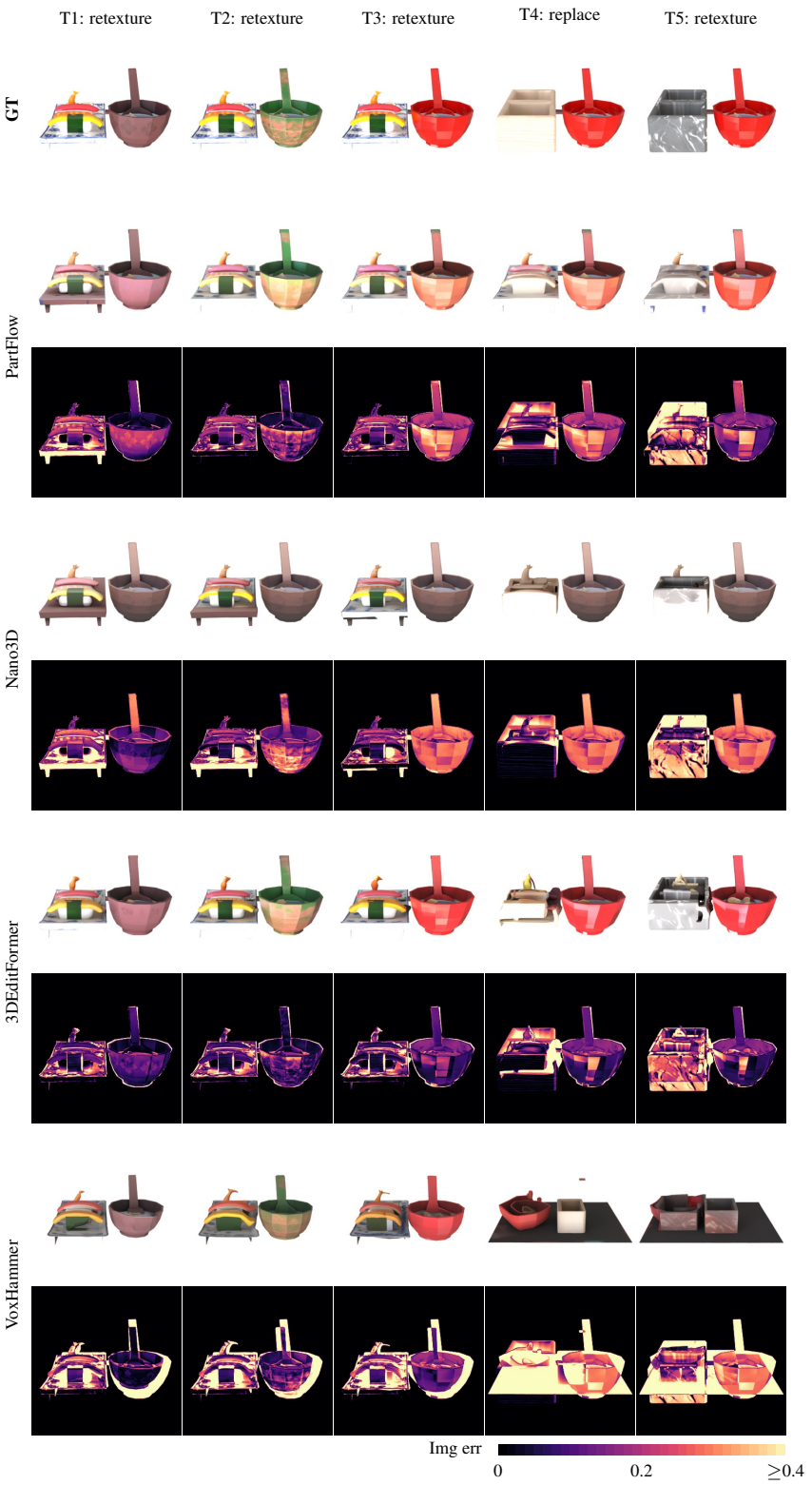

Figure 12: **A table setting** (turns 1–5): turns 1–3 and 5 change the material of the sushi tray, the soup bowl or the compartment tray, and turn 4 replaces the sushi tray and the sushi with a wooden compartment tray. Shown in the conditioning view.

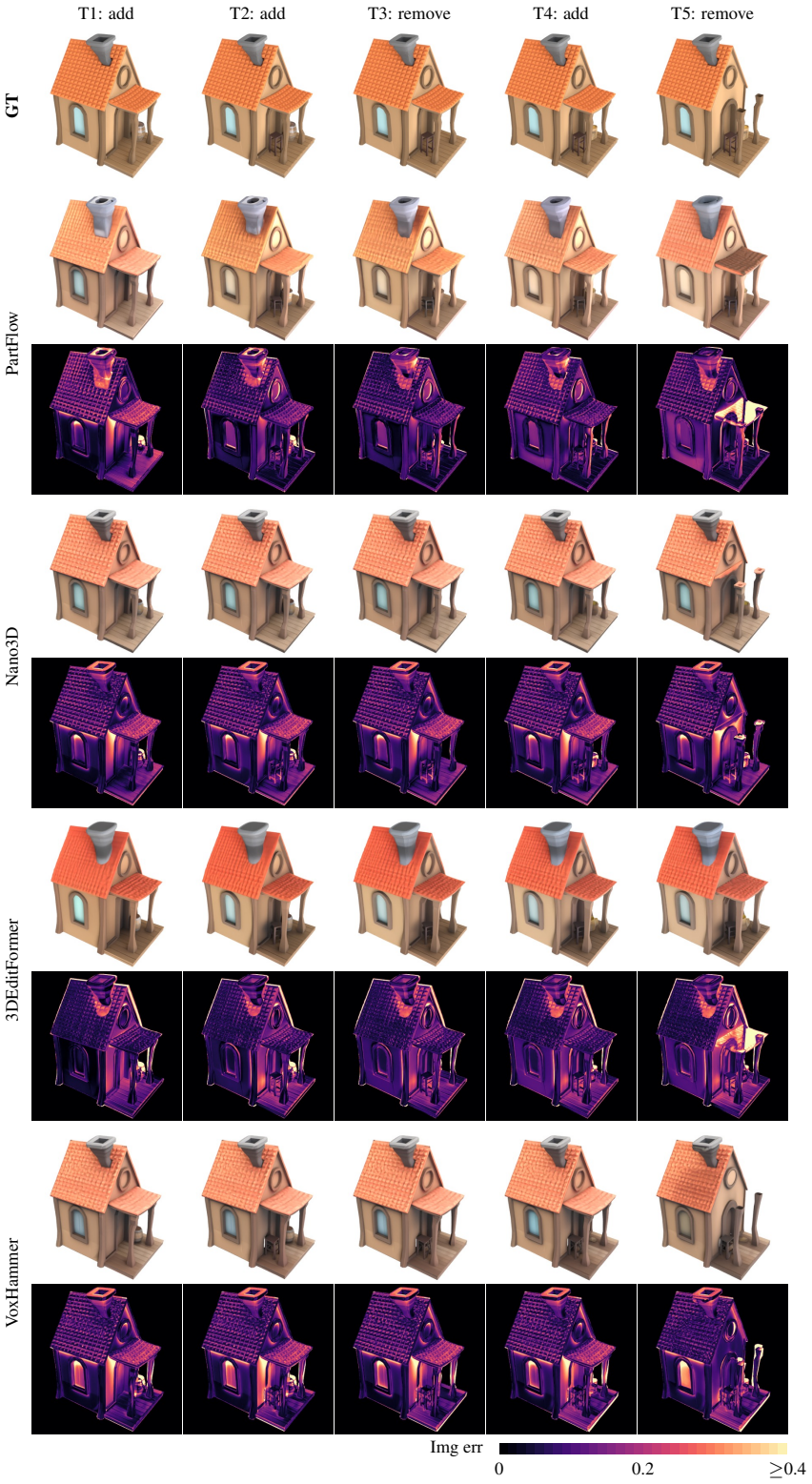

Figure 13: **A house** (turns 1–5): three additions and two removals on the porch. Shown from a view other than the conditioning view.



Figure 14: **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.

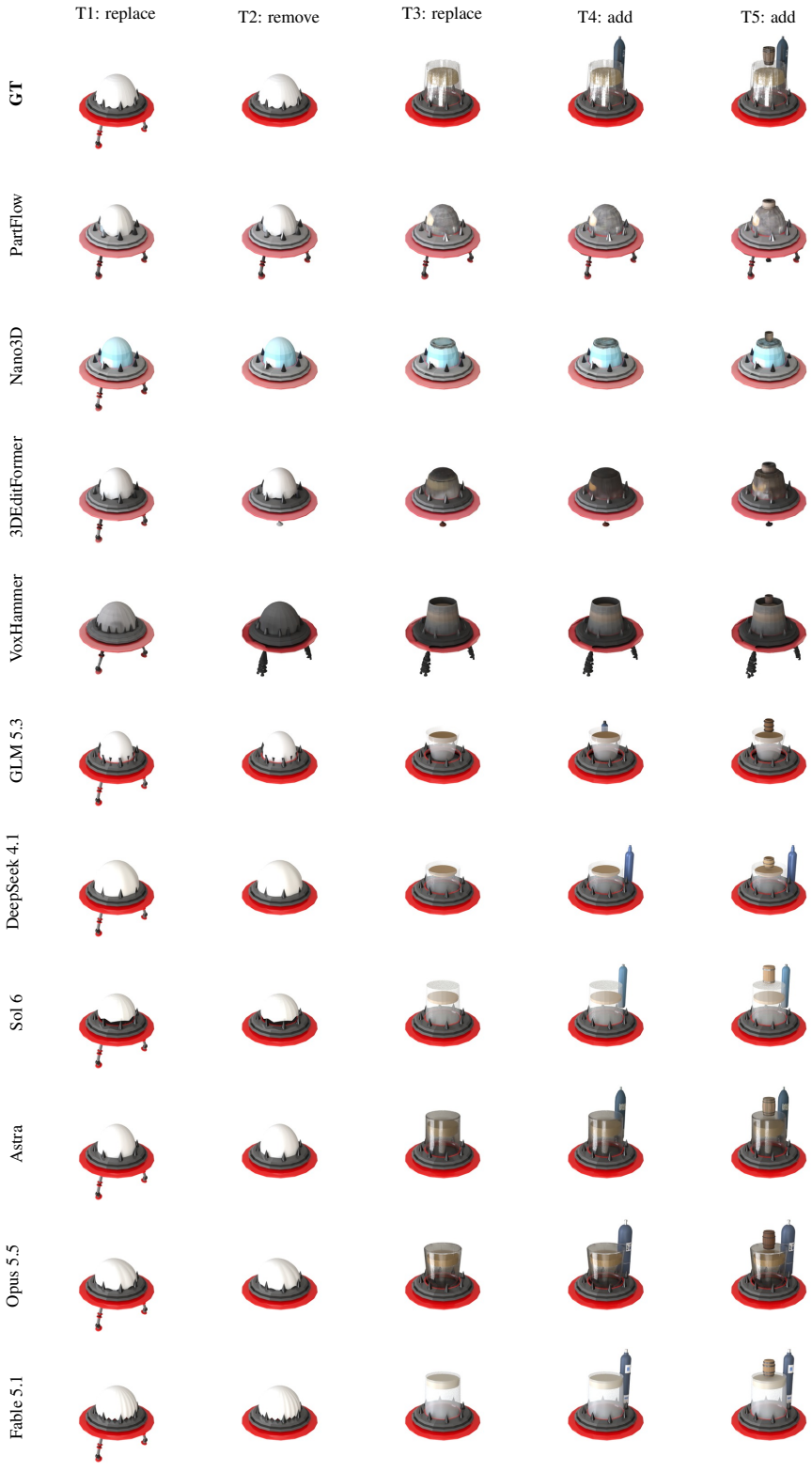

Figure 15: **A flying saucer** (turns 1–5): the dome is replaced by a parachute canopy, the landing legs are removed, the canopy is replaced by a transparent cylinder, and a gas cylinder and a barrel are added. Shown from a view other than the conditioning view.



Figure 16: **A character** (turns 1–5, all four 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 view other than the conditioning view.

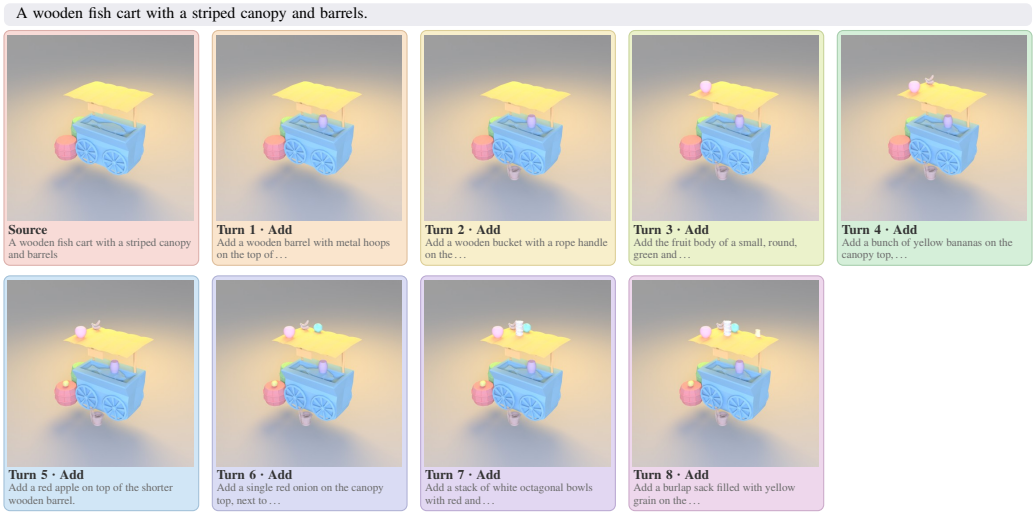

Figure 17: Add-only chain.

```
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.
```

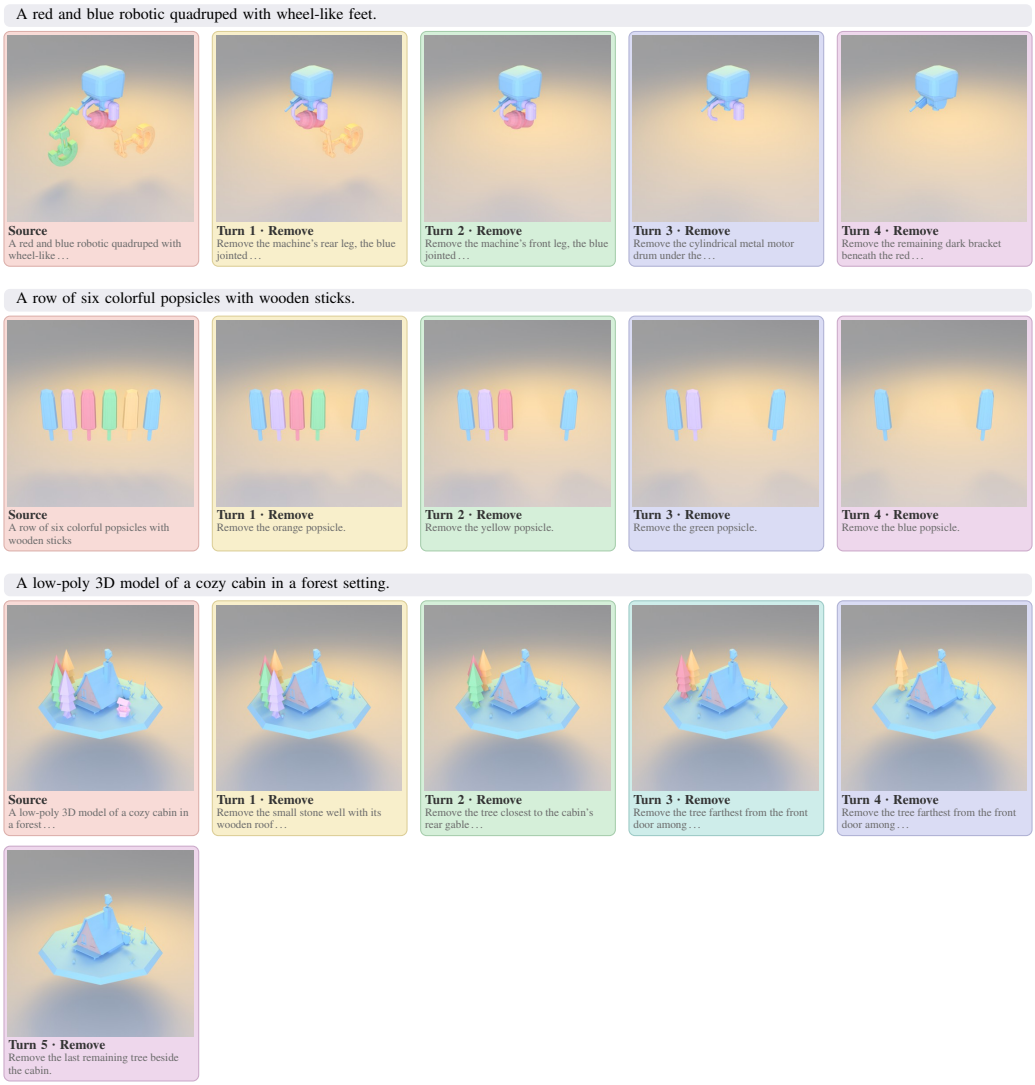

Figure 18: Remove-only chains.

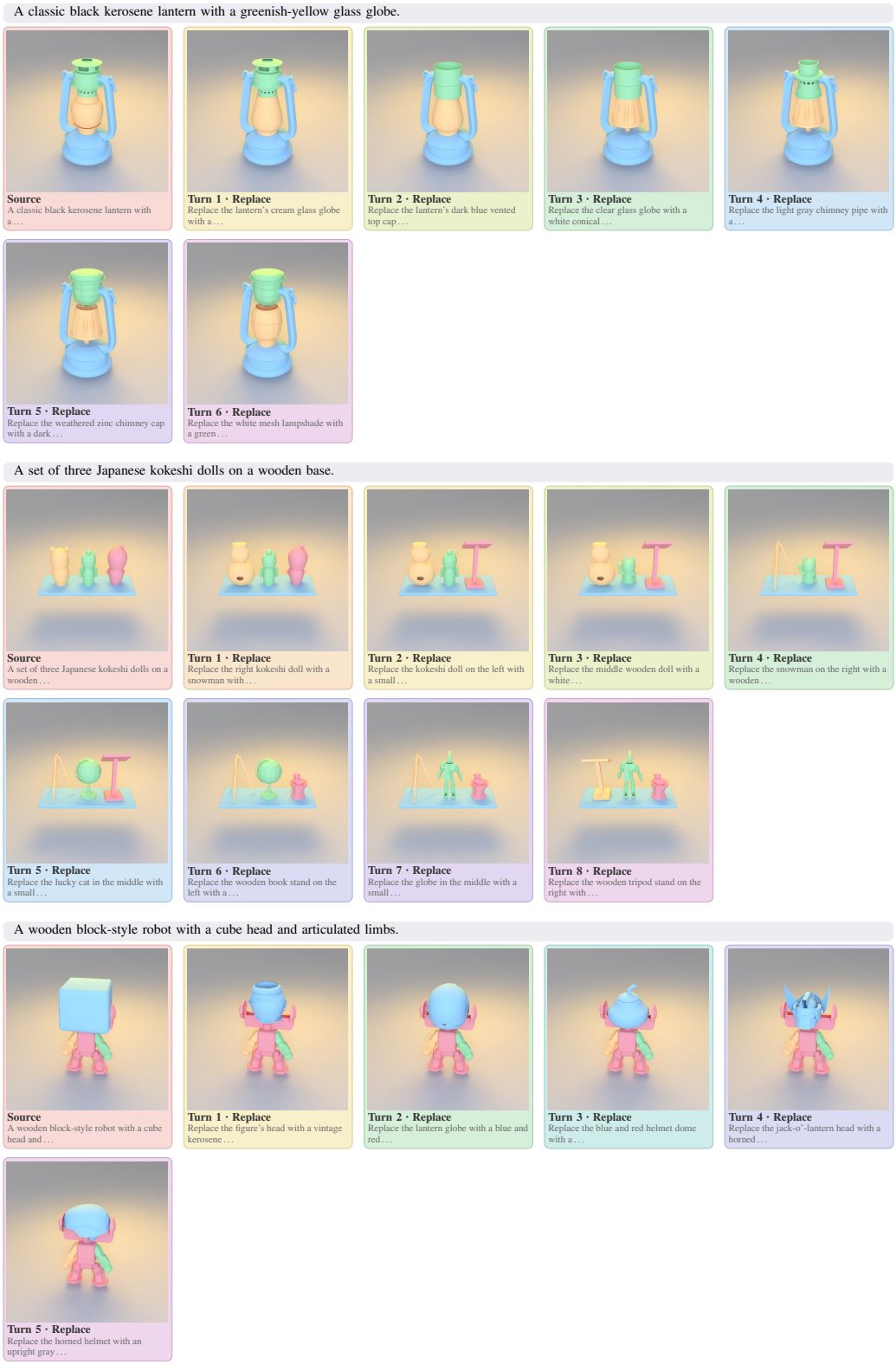

Figure 19: Replace-only chains.

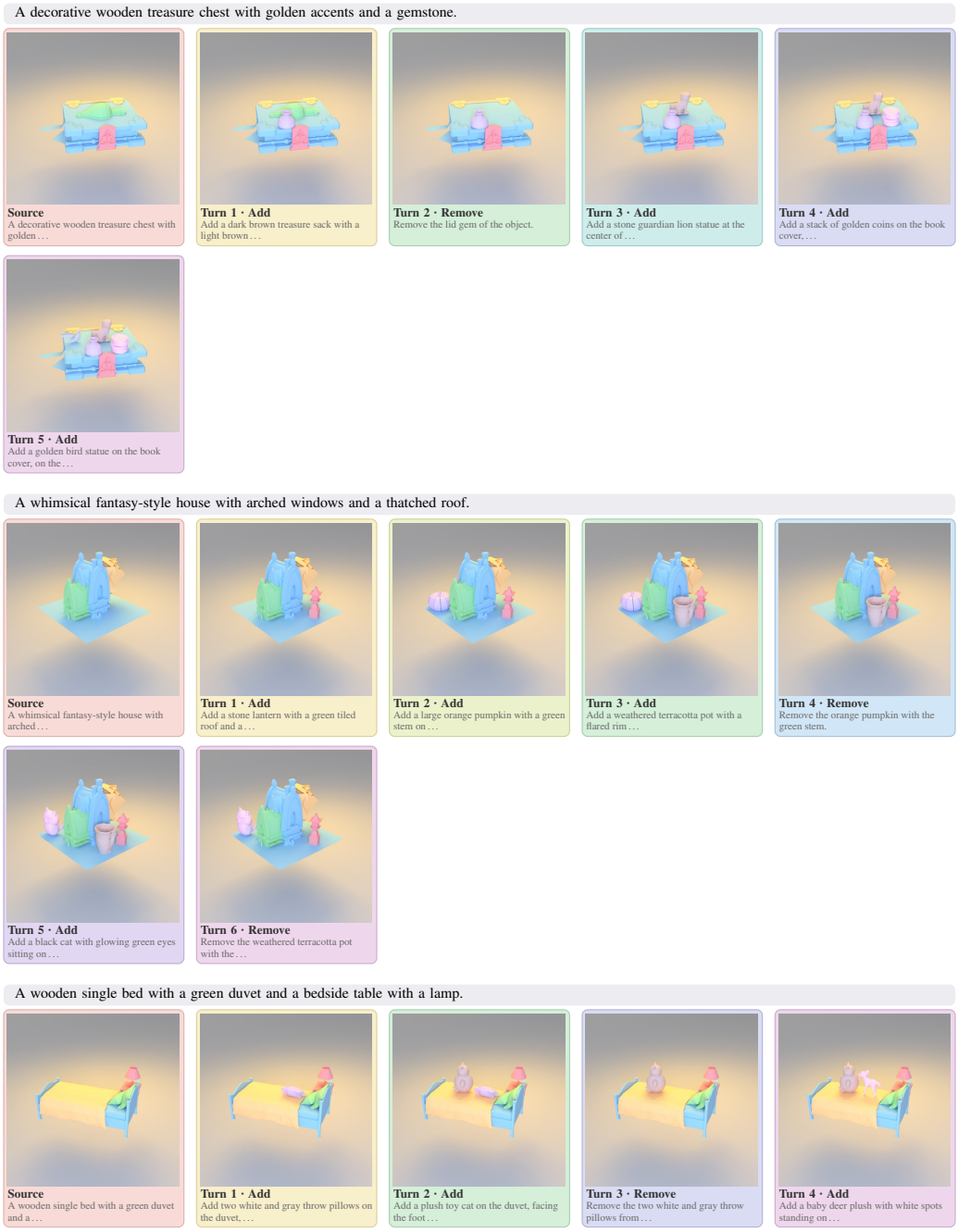

Figure 20: Add-and-remove chains.

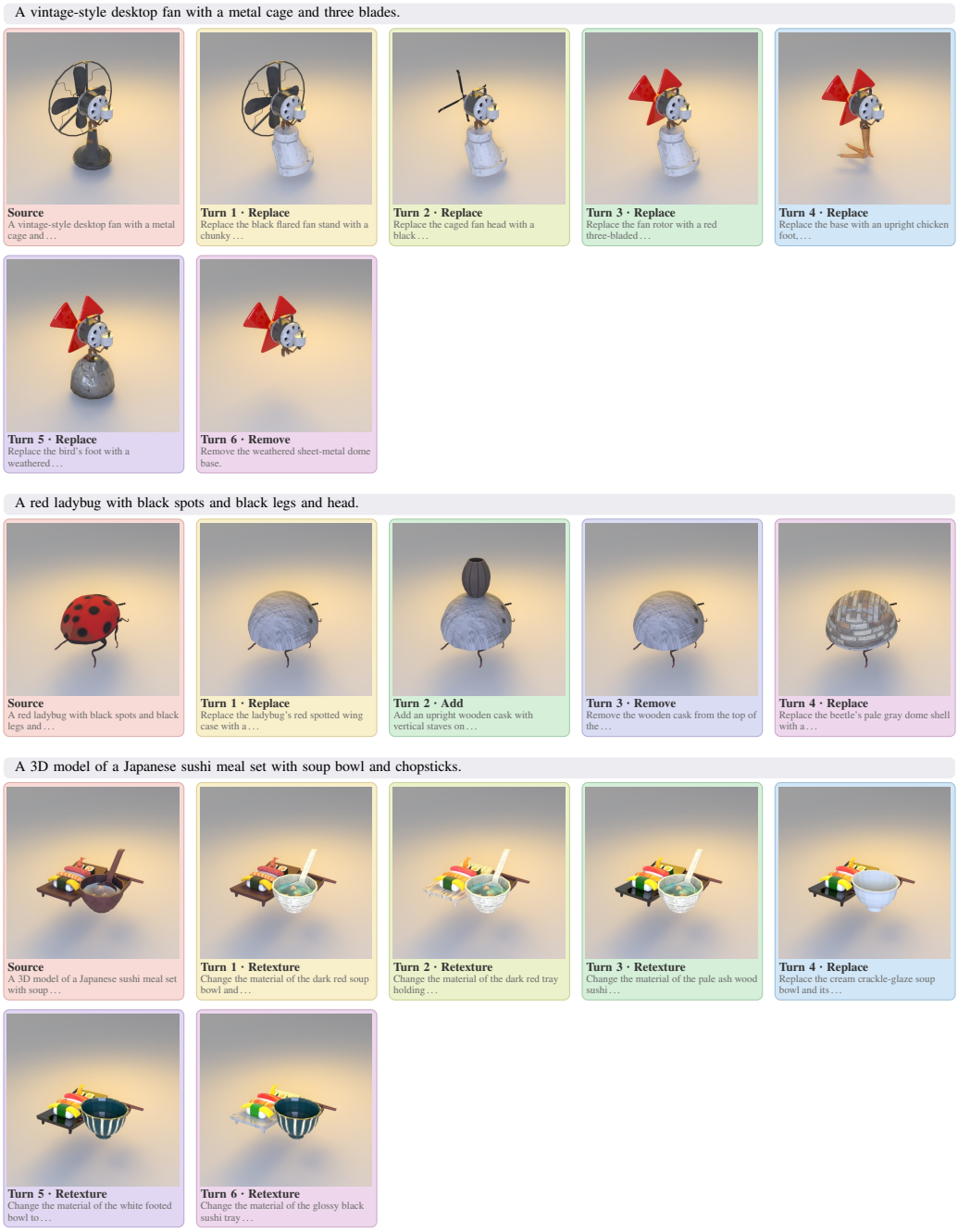

Figure 21: Mixed chains with geometry and material edits.

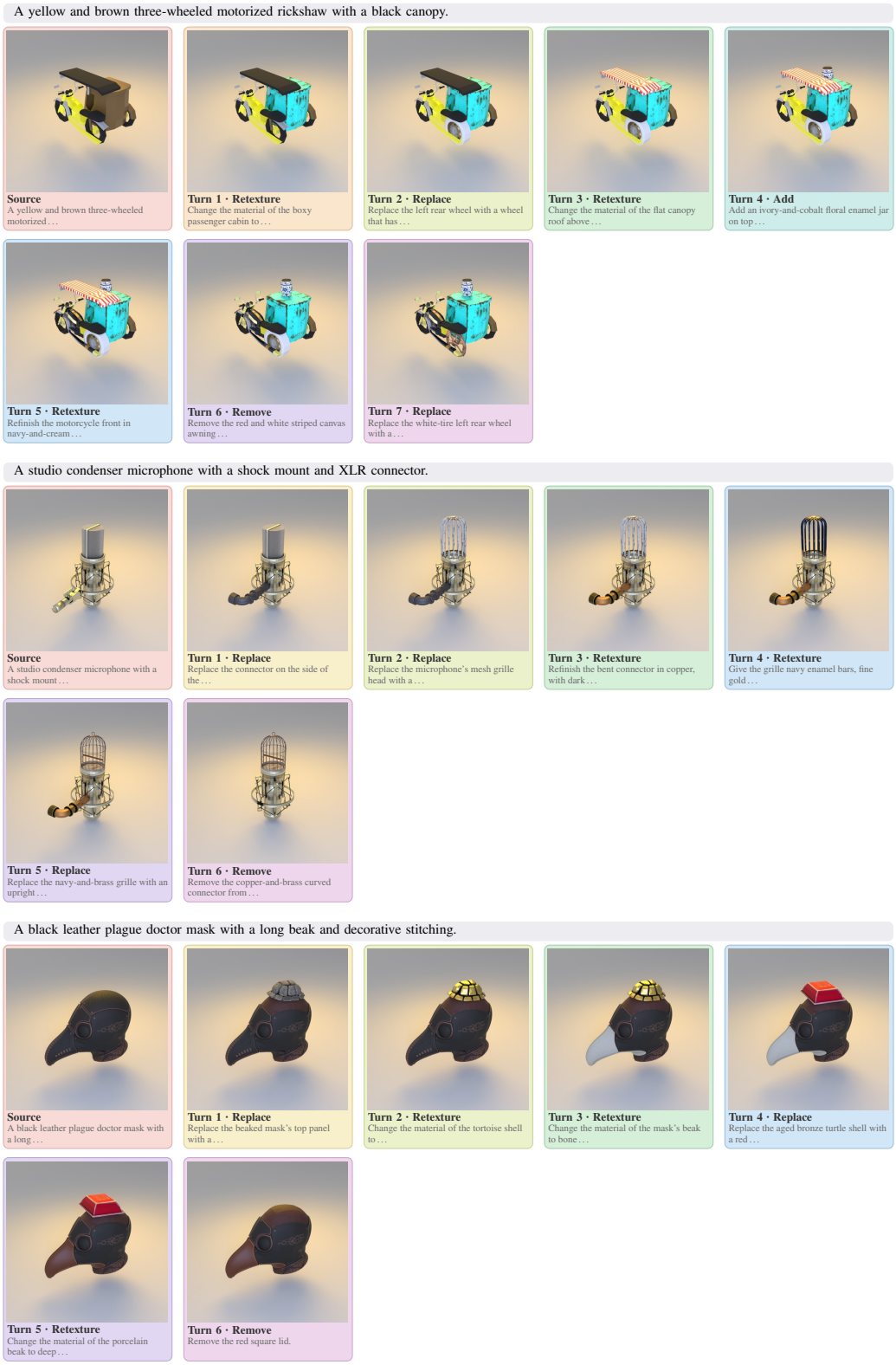

Figure 22: Chains with all four operations.

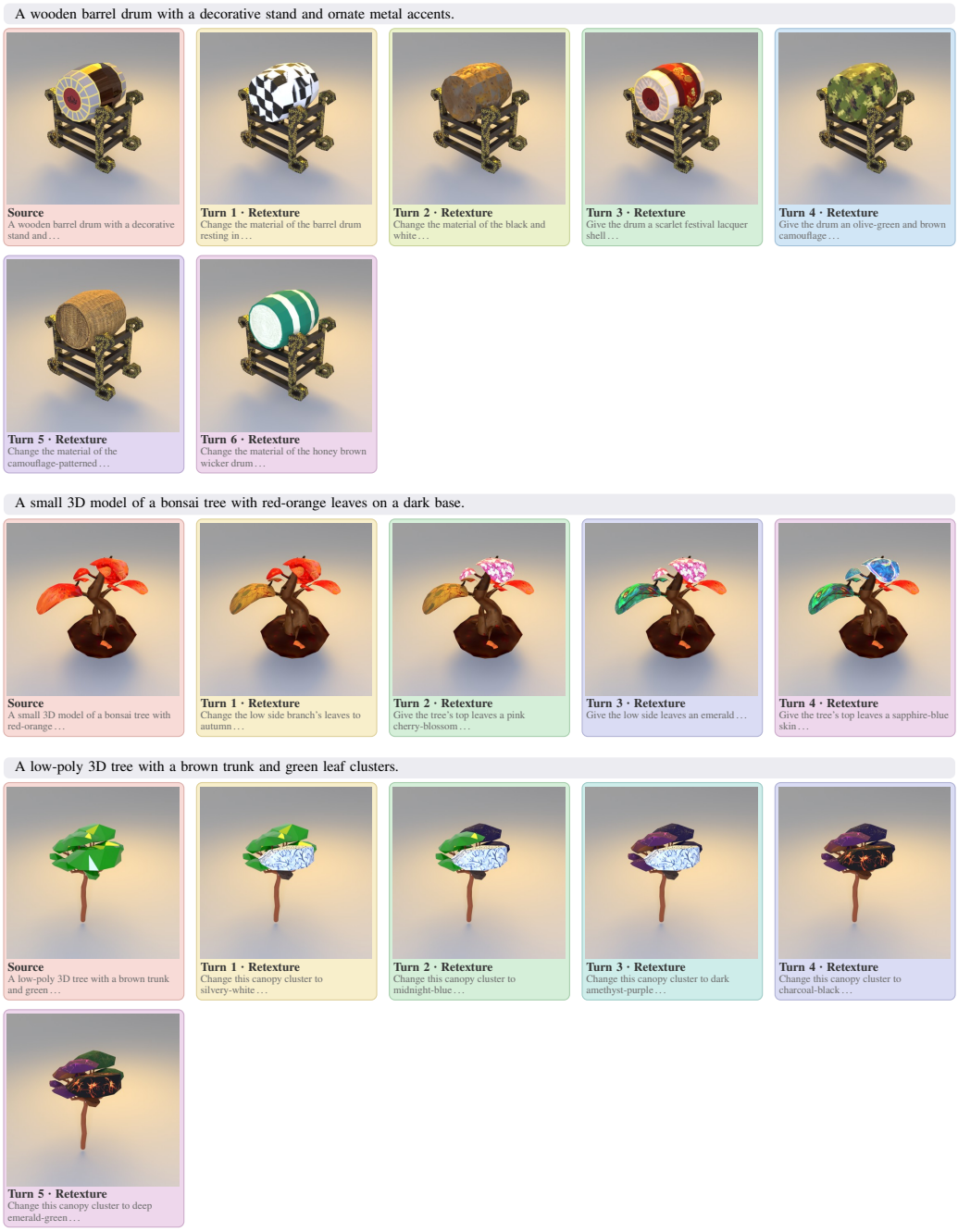

Figure 23: Retexture-only chains.

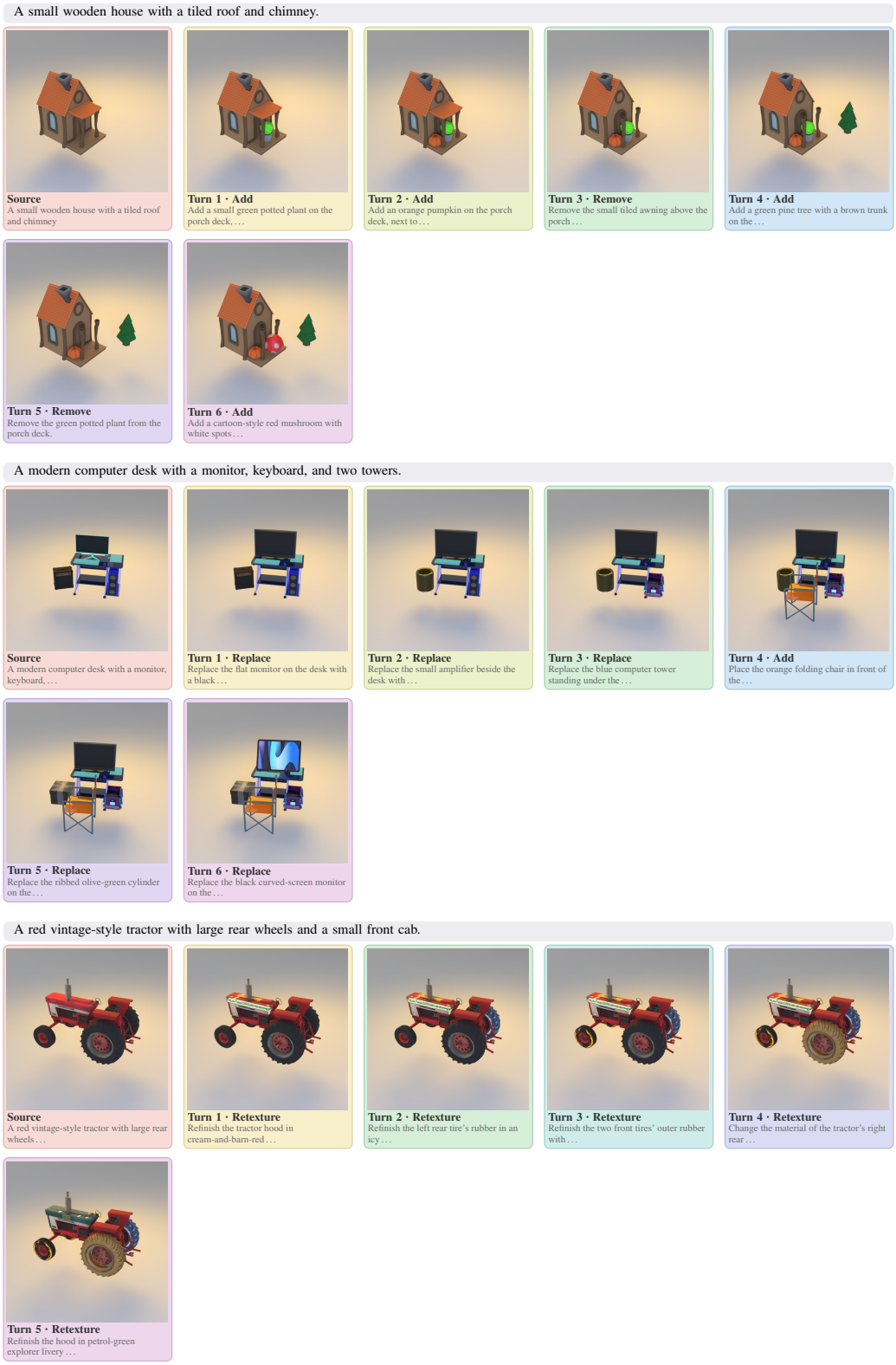

Figure 24: Scenes.

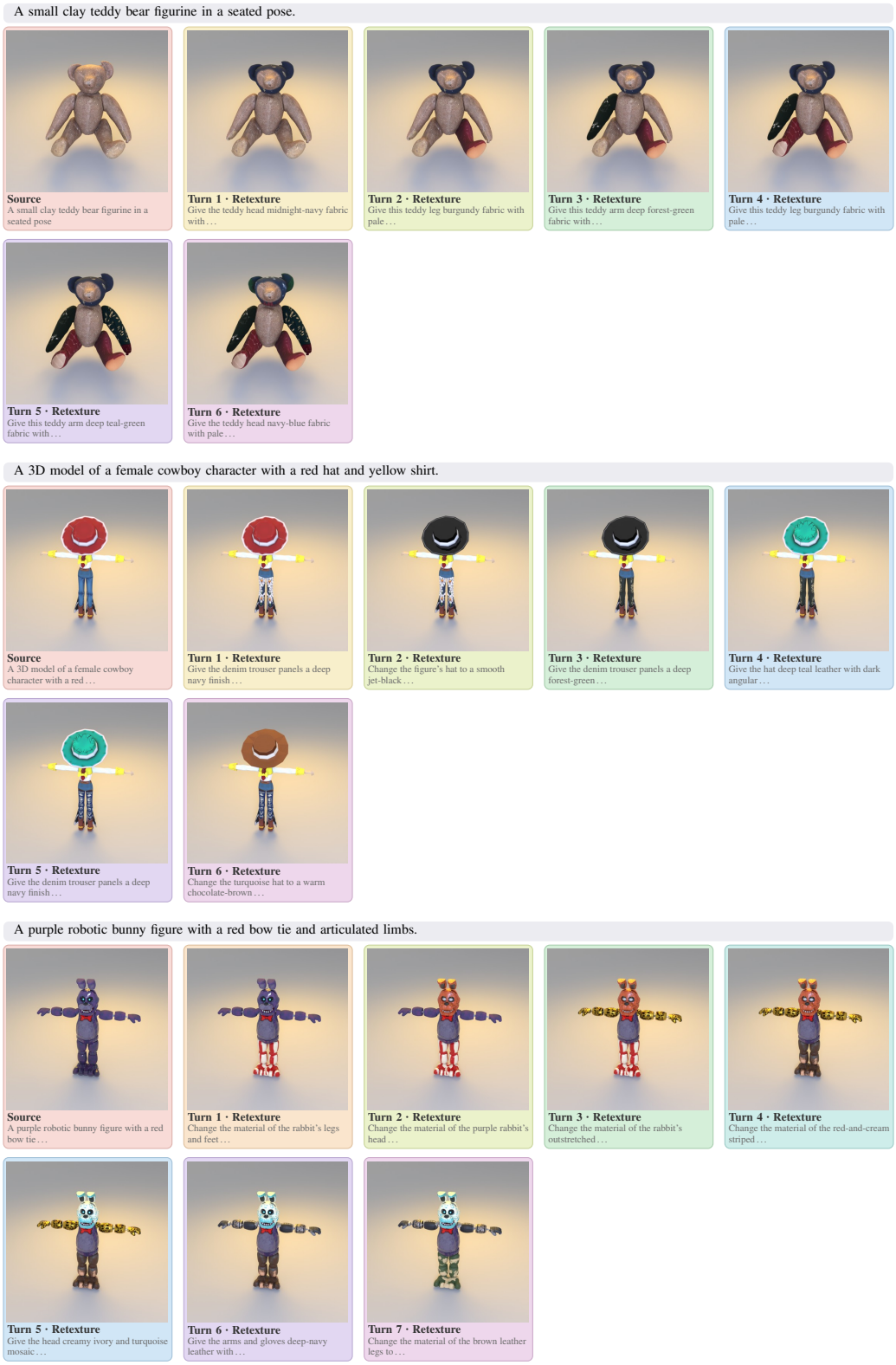

Figure 25: Characters.

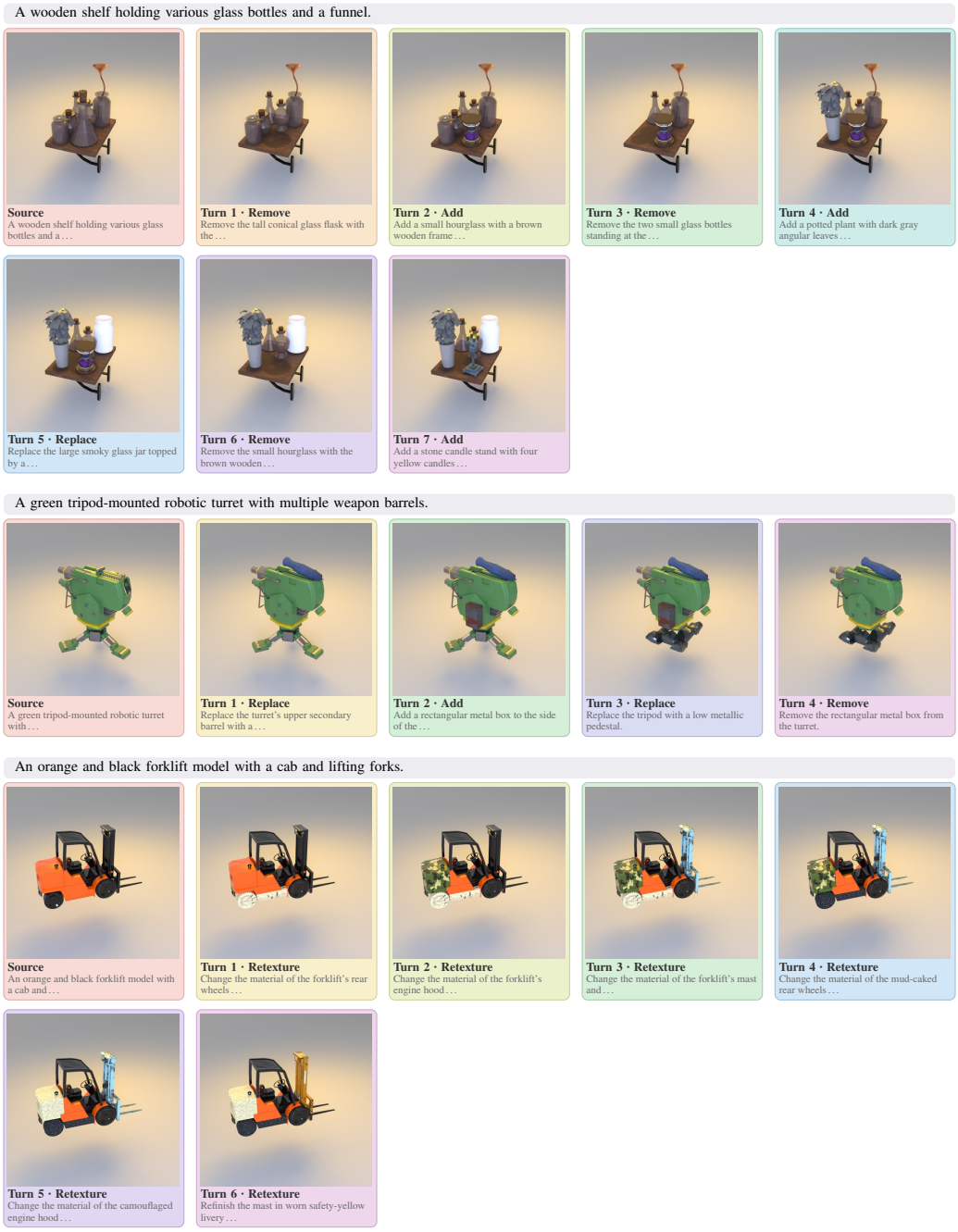

Figure 26: Objects.