VISCULPT: VISUAL-CENTRIC AGENTIC GEOMETRY EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D geometry editing is a critical yet labor-intensive part of the graphics pipeline, requiring artists to translate creative intent into precise operations in complex professional software. Large language models (LLMs) have shown promise for script-based 3D creation, but script generation is less suited to perception-driven editing of arbitrary existing meshes, where execution must remain visually grounded and untouched regions should be preserved. We present a *visual-centric*, training-free multi-agent system that edits existing 3D meshes directly in Blender by emulating the iterative workflow of human artists. Rather than generating scripts or regenerating geometry, our system operates through the Blender GUI: multimodal LLM agents observe the viewport, reason about the current mesh state, and execute localized edits through simulated user interactions. Experiments on a curated benchmark provide initial evidence that this agentic approach can follow natural language instructions, perform representative localized mesh edits, and preserve the overall identity of the input asset. Our results highlight a complementary regime for language-driven 3D editing: direct in-place modification of existing meshes within the native 3D editing workflow. We view this work as an exploratory step toward visual-centric agentic geometry editing in professional graphics software.

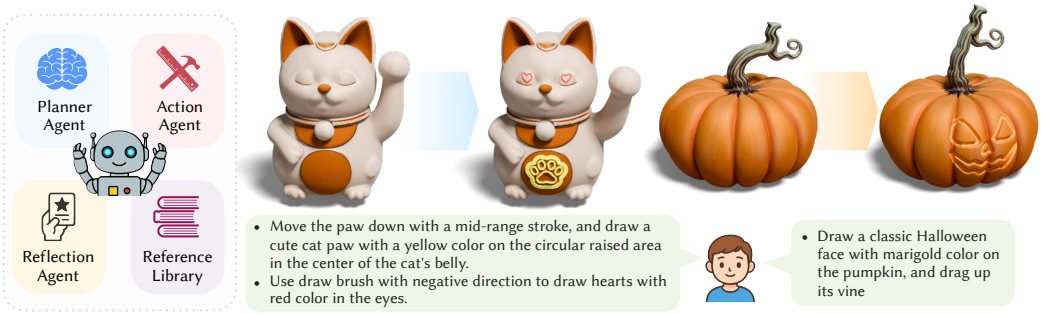


Figure 1: **Visual-Centric 3D Geometry Editing.** We present a training-free multi-agent system for editing existing 3D meshes from natural language. The system (left) plans editing steps, manipulates geometry directly through the Blender GUI, and visually evaluates intermediate results. The examples (right) show localized shape deformation and detailed surface sculpting on existing assets, while preserving the identity of the original mesh.

1 INTRODUCTION

Localized 3D geometry editing modifies an existing asset according to an instruction while preserving unrelated geometry. In professional software such as Blender, this requires translating high-level intent into tool choices, spatial targets, and precise operations. Automating this process could make such edits accessible to users without specialized modeling expertise.

Recent generative methods (Zhang et al., 2024; Barda et al., 2024; Gao et al., 2025; Barda et al., 2025) produce prompt-aligned 3D edits, but approaches that regenerate geometry can alter instance-specific details beyond the requested change. Script-based LLM agents (Lu et al., 2025; Du et al., 2024;

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.

Blender Foundation, 2026) provide programmatic control over professional 3D software, including edits to existing meshes. Our focus is how to translate perceptual editing intent into precise, localized actions while preserving unrelated geometry and limiting execution cost. In sculpting interfaces, this requires grounding instructions in visible mesh regions and controlling continuous brush strokes whose dense trajectories are difficult to specify directly.

We present *ViSculpt*, a *visual-centric* system for language-driven mesh editing through Blender's graphical user interface. Its core is a compact action abstraction: *Smear* for region coverage, *Drag* for local deformation, and *Draw* for surface detail. The agent predicts primitive actions and their parameters, while geometric procedures construct executable mouse trajectories. This separates semantic planning from dense trajectory generation, providing a structured interface between language instructions and sculpting operations. Visual feedback guides subsequent actions and supports iterative refinement of the edited mesh.

A *Planner Agent* decomposes instructions into primitive actions, an *Action Agent* executes them in Blender, and a *Reflection Agent* evaluates visual changes to determine whether to proceed or refine the edit. The pipeline uses pretrained models without task-specific training or fine-tuning.

In our initial evaluation, we assess ViSculpt on 20 editing tasks through a blinded user study with 48 participants and auxiliary VLM-based assessments, obtaining subjective ratings competitive with human-created references. In subsequent qualitative Blender MCP comparisons, ViSculpt with the lightweight Gemini 3.5 Flash Lite backbone produces more faithful local edits than the Claude Sonnet 4.6 baseline and results competitive with Codex using GPT-6 Astra at max reasoning effort. In the Codex comparison, **ViSculpt averages 1–2 minutes and $0.10 per edit, versus about 10 minutes and $4.00 for the baseline**. These comparisons highlight the potential for efficient localized editing with a lightweight backbone.

In summary, our main contributions are as follows:

- We introduce a visual-centric paradigm for language-driven geometry editing, formulating localized edits to existing meshes as interaction with professional 3D software guided by visual feedback, with the goal of fulfilling editing intent while preserving asset identity and untouched geometry.
- We present a training-free multi-agent system that combines compact action primitives with geometric trajectory generation to translate semantic plans into executable local edits in Blender and iteratively refine the results through visual feedback.
- We provide empirical evidence of edit quality and asset identity preservation through a user study with human-created references and qualitative agent comparisons. Measurements in the subsequent comparisons demonstrate runtime and cost advantages with a lightweight backbone.

2 Related Work

3D Geometry Editing Geometry processing provides the algorithmic foundation for manipulating and analyzing 3D data (Botsch et al., 2010), including mesh simplification (Garland & Heckbert, 1997; Hoppe, 1999), remeshing (Alliez et al., 2003; Yan et al., 2009), surface smoothing (Desbrun et al., 1999), shape deformation (Sorkine & Alexa, 2007; Sorkine et al., 2004), and correspondence (Ovsjanikov et al., 2012). Within this area, *geometry editing* refers to tasks that modify a shape to satisfy user-specified requirements. Classical editing methods are typically driven by geometric signals such as curvature, topology, or error metrics, and thus provide limited support for high-level semantic intent. Recent generative methods (Zhang et al., 2024; Barda et al., 2024; Gao et al., 2025; Barda et al., 2025; Li et al., 2024a) learn to produce edited shapes conditioned on an input shape and an edit specification. These methods are powerful for generation and representation-level transformation, but our setting instead requires direct, localized edits to an existing mesh while preserving unintended regions. We therefore view generative editing as a complementary direction. Our work targets the remaining gap between semantic editing intent and expert GUI operation by enabling visual-centric geometry editing in professional 3D software directly from natural language.

3D Agentic System Advances in LLMs (OpenAI, 2023; Seed, 2025; GLM-V Team, 2025; OpenAI, 2025; Google, 2025; xAI, 2025b; Qwen, 2025) have enabled agentic systems that interleave planning, perception, and tool use. In 3D content creation, prior agents commonly translate natural-language

instructions into executable programs, including Blender Python scripts, procedural programs, and parametric modeling code (Yuan et al., 2024; Lu et al., 2025; Alrashedy et al., 2024; Du et al., 2024; Raistrick et al., 2023; 2024; Yamada et al., 2025; Ahuja & Contributors, 2025; Hu et al., 2024; Lv et al., 2024; Sun et al., 2025; Huang et al., 2024). Related work further improves controllability through domain-specific languages that expose semantic 3D concepts as compositional primitives (Sharma et al., 2018; Kania et al., 2020; Du et al., 2018; Wu et al., 2021; Chen et al., 2025; Jones et al., 2020), or by coupling perception and planning with low-level geometric operations (Liu et al., 2025; Fu et al., 2024; Ling et al., 2025). Most of these systems follow a script-generation paradigm, targeting asset creation or editing through an intermediate program representation. Our work is complementary: we study direct GUI-based editing in professional 3D software, focusing on existing meshes, visual feedback, localized operations, and preservation of untouched regions.

Multimodal Large Models Recent multimodal foundation models can integrate visual and linguistic signals for complex tasks (Radford et al., 2021; Li et al., 2022; Lu et al., 2025; Jatavallabhula et al., 2023), providing a general interface for grounding language in visual observations and 3D scenes. In 3D generation, vision-language models (VLMs) often serve as semantic priors for optimizing implicit representations (Zhang et al., 2024; Chen et al., 2024); for example, CLIP-Mesh (Mohammad Khalid et al., 2022) uses VLM-based discrimination over rendered views to guide text-consistent geometry synthesis. Segmentation foundation models such as Segment Anything (SAM) (Kirillov et al., 2023; Ravi et al., 2024b; Wei et al., 2024; Shen et al., 2023) are likewise used in 3D reconstruction and to transfer strong 2D priors to 3D settings (Cen et al., 2023; Yang et al., 2023). Most prior work, however, applies these models to generation or reconstruction rather than interactive editing. We instead use VLMs and SAM-like segmentation models to ground visual-centric geometry editing inside professional 3D software, aligning localized operations with user intent.

3 METHOD

Given an input 3D mesh and a natural language description of the desired changes, our goal is to automatically modify the mesh such that the output aligns with the user's intent. Our system is training-free and fully automatic.To this end, we introduce a multi-agent system built on primitive editing actions, with three components: a Planner Agent, an Action Agent, and a Reflection Agent. An overview of the system is shown in Figure 2. We first describe the primitive mouse actions that form the basis of our editing operations in Section 3.1, then detail how the agents compose a whole pipeline in Section 3.2.

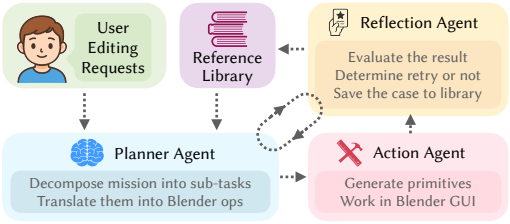


Figure 2: Overview of ViSculpt. The Planner, Action, and Reflection Agents respectively plan editing steps, execute them in Blender, and evaluate results. Visual feedback guides iterative refinement, while reference libraries support planning and execution.

3.1 PRIMITIVE ACTIONS

In Blender sculpting, brush editing operations often depend on precise mouse trajectories. Directly generating such trajectories with LLM agents is difficult because the action space is continuous and high-dimensional. We observe that many brush strokes can be represented by three primitive mouse trajectories: *Smear*, *Drag*, and *Draw*. This abstraction constrains generation to a compact action vocabulary while remaining expressive enough for the representative brush operations considered in this work. Figure 4 summarizes the three primitives.

Smear The *Smear* trajectory implements area filling for brushes that require near-uniform coverage over a surface region, such as smoothing or texture painting. We first use a VLM to coarsely localize the target region in the current Blender viewport and produce a bounding box. This box prompts the Segment Anything Model (SAM) (Kirillov et al., 2023) to generate a binary mask, from which we extract the exterior boundary using OpenCV. For *Smear*, the LLM agent only specifies the brush radius; the trajectory is then constructed deterministically from the mask. Given the radius, we iteratively erode the mask to obtain nested contours, sample points along them, and connect the samples into a continuous outside-in filling stroke that maintains consistent coverage.

Drag The *Drag* trajectory supports local deformation by pulling or pushing a selected part of the model along a specified displacement. It is central to manipulations such as limb elongation, pose adjustment, and protrusion creation. We parameterize *Drag* by an anchor point and a displacement vector. The anchor requires accurate localization of a mesh feature, but directly querying even advanced VLMs (Google, 2025) for image-space coordinates is unreliable and often yields inconsistent or hallucinated positions. We therefore introduce *QuadLoc*, a coarse-to-fine quadrant localization scheme that casts localization as a sequence of visual multiple-choice decisions. Given the current viewport image, we overlay a 2×2 grid with four uniquely colored quadrants (Red, Blue, Green, Yellow) and ask the VLM to select the quadrant containing the target anchor, instead of predicting exact coordinates, as shown in the left part of Figure 3. We crop to the selected quadrant and repeat the query recursively until the quadrant size falls below a preset threshold. The center of the final quadrant is used as the anchor point. The displacement vector encodes direction and magnitude and is inferred by prompting a VLM to interpret the requested deformation; in practice, we find this prediction reliable.



Figure 3: QuadLoc procedure. To mitigate VLM spatial imprecision, QuadLoc recursively queries the model for the quadrant containing the target (red outline), progressively narrowing the search space from left to right.

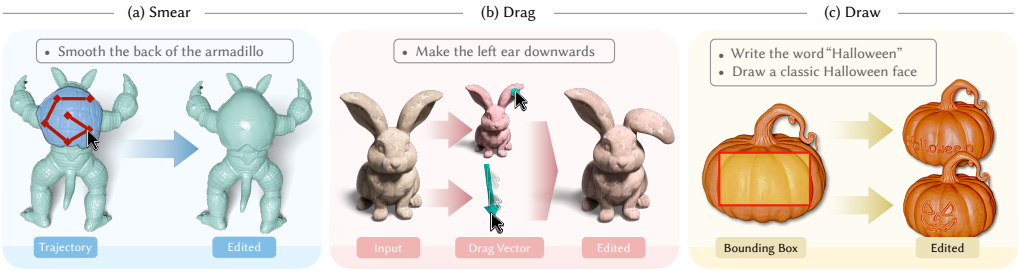


Figure 4: Three primitive mouse trajectories. (a) Smear, generated with segmentation and morphological operations; (b) Drag, defined by an anchor point from QuadLoc and a VLM-predicted displacement vector; and (c) Draw, created through heuristic stroke extraction for text and contour extraction for arbitrary shapes.

Draw The *Draw* trajectory handles high-frequency surface details, such as inscribed text or intricate patterns. Unlike *Smear* and *Drag*, *Draw* requires stroke-level precision. We use different strategies for *text* and *arbitrary shapes*. For text, we synthesize handwriting-like paths by rasterizing the target string into a binary mask using the handwriting-style font "Patrick Hand". We compute the medial axis of the glyph mask with `scikit-image`, convert the skeleton into a topological graph, and traverse graph edges in an order that approximates handwriting dynamics. For arbitrary shapes, such as symbols, emojis, or logos, we use a 2D image generation model to synthesize a high-contrast binary reference from the user's description. We then extract salient contours from the generated image and map them to viewport coordinates as drawing strokes.

3.2 Agent System for Geometry Editing

We next describe the three specialized agents in our editing system. For clarity, we use a 3D fox mesh as a running example: the user asks to raise its right arm beside its face in a friendly wave, smooth its front torso, and draw a heart emoji on its chest, as shown in Figure 5.

3.2.1 Planner Agent

The Planner Agent translates high-level, potentially ambiguous user intent into an ordered list of executable commands. It has two components: a *Decomposer* and a *Translator*.

Decomposer The Decomposer performs semantic parsing and task planning from the current mesh state, the user instruction, and retrieved reference information. Let $\mathcal{G}$ denote the current mesh, $\mathcal{D}_0$ the

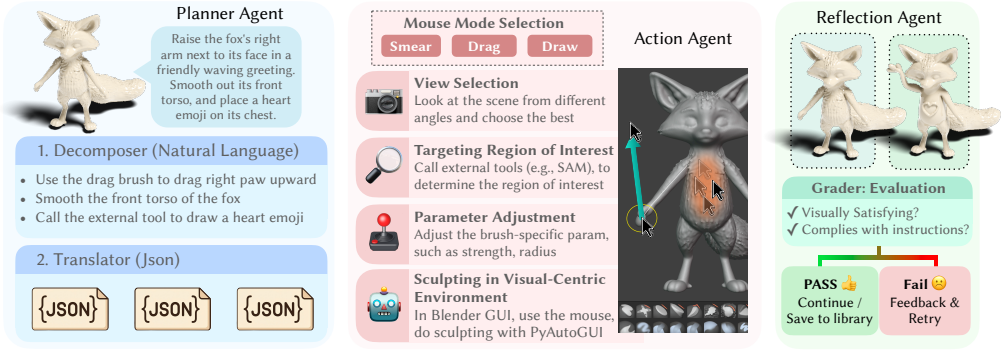


Figure 5: Example of our agentic editing pipeline. The Planner Agent decomposes a natural language request into sub-tasks and structured JSON commands. The Action Agent executes each command in Blender by selecting a primitive action, choosing a view, localizing the target region, and simulating GUI interactions. The Reflection Agent evaluates the result against the instruction, triggering refinement when needed and archiving successful trajectories for future reuse.

initial user instruction, and $\mathcal{R}$ the information retrieved from the Reference Library (Appendix C). The Decomposer $\mathbf{P_d}$ produces a structured plan $\mathcal{D}_1 = \mathbf{P_d}(\mathcal{G}, \mathcal{D}_0, \mathcal{R})$. The plan $\mathcal{D}_1$ is an ordered list of sub-tasks that disambiguate the intended operations and their dependencies. In our running example, the Decomposer splits the request into distinct actions shown on the left of Figure 5.

Translator While the Decomposer determines *what* to do, the Translator determines *how* to execute it. The Translator $\mathbf{P_t}$ converts $\mathcal{D}_1$ into machine-readable specifications for the Action Agent. It uses the Blender API documentation from the Reference Library $\mathcal{R}$ and optional iterative feedback $\mathcal{F}$ from the Reflection Agent (Section 3.2.3) to generate JSON commands: $\mathbf{J} = \mathbf{P_t}(\mathcal{D}_1, \mathcal{F}, \mathcal{R})$. The resulting specification $\mathbf{J}$ contains the attributes needed to execute the operation and can be refined at execution time using visual observations. In our implementation, most refinements occur at the execution level, such as adjusting localization or parameters, or retrying the same sub-task, rather than re-planning the entire task.

3.2.2 Action Agent

The Action Agent executes the Planner's JSON commands $\mathbf{J}$ by interacting with the Blender GUI. Rather than relying on script-based API calls, it is *visual-centric*: it perceives Blender's viewport and synthesizes mouse events in a human-like interaction loop. The execution proceeds in three stages: *View Selection*, *Target Segmentation*, and *GUI Control*. Refer to the middle of Figure 5 for an illustration of these stages.

View Selection Before editing, artists adjust the viewpoint to expose the target area. Similarly, the agent selects a view that maximizes target visibility and reduces occlusion. It renders six canonical views ($+x, -x, +y, -y, +z, -z$) and let the VLM to choose the view with the clearest visibility of the region referenced by the Planner. For most object-centric assets without severe self-occlusion, selecting among six canonical views provides a reliable and efficient proxy for full viewport navigation.

Target Segmentation The agent localizes the region of interest with a coarse-to-fine pipeline. It captures the selected viewport, queries the VLM for a bounding box around the instructed target, and uses this box to prompt a Grounded Segment Anything Model (Ravi et al., 2024b) to produce a binary mask. The mask constrains subsequent operations to the intended region, reducing unintended edits to unrelated geometry.

GUI Control Given the selected view and mask, the agent executes the operation through screen-space manipulation of the Blender GUI. Using `PyAutoGUI`, it configures brush parameters from the JSON command, maps primitive trajectories (Smear, Drag, Draw) from image space to viewport

coordinates, and simulates mouse button states (press/hold/release). As the interaction uses Blender's native stroke processing, it inherits built-in stabilization and pressure simulation, yielding edits consistent with human workflows.

3.2.3 REFLECTION AGENT

Multi-step editing without verification can accumulate errors. We therefore introduce a *Reflection Agent* that evaluates each executed sub-task and decides whether to accept the result or request refinement, as shown in the right part of Figure 5.

To assess edit quality, we employ a VLM-based Visual Grader. The Grader takes a pre-edit screenshot $\mathcal{I}_{pre}$, a post-edit screenshot $\mathcal{I}_{post}$, and the Planner-generated sub-goal $\mathcal{D}_{sub}$ as input. It compares the visual change from $\mathcal{I}_{pre}$ to $\mathcal{I}_{post}$ with the semantic intent of $\mathcal{D}_{sub}$ using a visual question-answering formulation. The Grader outputs a scalar score $S \in [0, 15]$ and a textual critique. Given a threshold τ, we accept the sub-task if $S \geq \tau$ and advance to the next sub-task; accepted instances are archived in the Previous Success Cases library to inform future decisions. If $S < \tau$, the agent enters a *Refinement Loop*: the critique is provided as feedback context $\mathcal{F}$ to the Planner and Action Agent, which revise parameters (e.g., increasing brush strength) and re-execute the sub-task. We emphasize that this grader is used as a practical visual critic for iterative refinement, rather than as a definitive objective measure of geometric correctness.

4 RESULTS AND COMPARISONS

Environment Setup Our experiments were conducted in two phases. The initial evaluation used Gemini 3 Flash for the three agents and Gemini 3 Pro for target localization in the Action Agent. Subsequent comparisons with Claude Sonnet 4.6 and GPT-6 Astra use Gemini 3.5 Flash Lite for all three agents, including target localization. Across our recent tests, including multi-step edits, this current configuration averages approximately 1–2 minutes per editing task. We use Grounded SAM2 (Ravi et al., 2024a) for mask generation and Z-Image (Cai et al., 2025) for free-form Draw reference images. We access Gemini models through the official API. To demonstrate extensibility, we also include custom procedural brushes, including star-pattern and Perlin-noise brushes. All experiments are conducted on Windows 11 with Blender 4.5 LTS.

Benchmark and User Study Direct 3D geometry editing remains a challenging task, with few automatic systems supporting localized, instruction-driven changes to existing meshes. We therefore evaluate the initial configuration against human artists using a blinded user study. We curate 20 geometry-editing tasks and produce two results for each task: one generated by our system and one manually created in Blender by human artists, yielding 40 edited meshes in total. Participants viewed randomized pre-/post-edit mesh pairs together with the corresponding instruction, and rated each result from 0 to 10 for instruction adherence, visual quality, and geometric plausibility. The study included 48 participants: 39 non-experts and 9 experts, with experts defined as participants with prior publication experience in computer graphics venues. Since visual-centric editing of existing meshes is a relatively new setting, large-scale standardized comparisons against multiple automatic baselines are currently difficult to establish fairly across the same task scope, software environment, and evaluation protocol. We therefore use human-created edits as the primary reference, providing a direct and stable comparison for the in-place editing tasks studied here.

The results of the user study are summarized in Table 1 and Figure 10. Our method achieves an average score of 7.53, comparable to the 7.20 average score of human-created edits. This result indicates that our visual-centric agent pipeline can produce perceptually plausible geometry edits across the evaluated task set. We further conduct an auxiliary VLM-based evaluation using the same questionnaire, repeating each evaluation five times to reduce sampling variance.

Figure 9 presents qualitative editing results, showing the input mesh, the editing instruction, and our output. Comparisons with human-created edits are shown in Figure 11; these examples complement the quantitative user-study results.

Comparison with Human Artists Compared with manual editing by human artists, our method substantially reduces the interaction burden while achieving competitive perceptual quality. For

Table 1: Human and VLM evaluation of the initial ViSculpt configuration against human-created edits. Scores are mean and standard deviation (SD) across participants and tasks on a 0–10 scale; VLM rows identify the evaluators.

		Mean Score $\pm$ SD	
Group	**N**	**Human Reference**	**Ours**
All Users	**48**	$\mathbf{7.20 \pm 2.19}$	$\mathbf{7.53 \pm 2.13}$
Experts	9	7.12 ± 2.23	7.36 ± 2.17
Non-Experts	39	7.22 ± 2.19	7.57 ± 2.12
All VLMs	**55**	$\mathbf{6.54 \pm 1.01}$	$\mathbf{6.51 \pm 0.87}$
Claude 4.5 Opus, Sonnet (2025)	10	5.51 ± 0.54	5.68 ± 0.18
Doubao Seed 1.8 (2025)	5	7.84 ± 0.16	7.17 ± 0.08
GLM 4.6V (2025)	5	6.77 ± 0.17	6.56 ± 0.20
GPT 5.2, 5.2 Pro (2025)	10	5.93 ± 0.09	5.90 ± 0.06
Gemini 3 Pro, Flash (2025)	10	7.28 ± 0.91	7.26 ± 0.83
Grok 4 (2025a), 4.1-fast (2025b)	10	7.01 ± 1.20	7.17 ± 0.98
Qwen3 VL Plus (2025)	5	5.85 ± 0.31	5.81 ± 0.30

instance, painting a smiley face on a mesh may require an artist to prepare a texture in an external image editor and configure Blender's stencil-mapping workflow. With our system, the user can express the same intent directly in natural language, e.g., "paint a smiley face on the rabbit". For edits that imply symmetry, such as "droop both ears," the agent also infers the corresponding brush configuration instead of requiring manual tool setup. Our system demonstrates a path toward accessible geometry editing in which non-expert users can perform localized mesh edits through high-level visual intent.

5 Discussion and Ablation Studies

We compare our visual-centric editing paradigm with script-centric and generative alternatives, examining their operating regimes and trade-offs. We then ablate our primitive abstraction to evaluate its role in stabilizing execution and grounding semantic instructions into precise GUI actions.

Discussion: Script-Centric Approaches Script-centric systems (Lu et al., 2025; Du et al., 2024) connect language models to professional 3D software and are well suited to tasks with explicit procedural structure, construction history, or programmatic decomposition. We focus on localized, perception-driven edits to *existing* meshes, where preserving untouched geometry is essential. ViSculpt performs these edits directly through the Blender GUI using visual feedback, without task-specific training or fine-tuning.

For the subsequent comparisons, we evaluate two configurations of Blender MCP Server (Blender Foundation, 2026), an official Blender Lab connector that exposes Blender's Python API to LLMs: Claude Desktop with Claude Sonnet 4.6 (Anthropic, 2026), and Codex with GPT-6 Astra at medium and max reasoning effort. Both comparisons use ViSculpt's current configuration with the lightweight Gemini 3.5 Flash Lite backbone. We omit LL3M (Lu et al., 2025) from our comparisons as the system is currently inaccessible to the public.

With this lightweight backbone, ViSculpt produces markedly better qualitative edits than the Claude-based baseline (Figure 6) and competitive results with GPT-6 Astra at max reasoning effort (Figure 7). In the Claude comparison, ViSculpt more faithfully realizes the requested local changes while preserving the overall asset structure.

The armadillo task in Figure 7 further illustrates the contrast: GPT-6 Astra (medium) severely distorts both arms, whereas the max configuration produces a plausible pose but slightly overstretches the upper arms. ViSculpt yields a more natural pose without visible distortion.



Figure 6: Qualitative comparison with the Claude-based script-centric baseline. Each pair shows Claude Desktop + Blender MCP with Claude Sonnet 4.6 (left) and ViSculpt with Gemini 3.5 Flash Lite (right). ViSculpt more faithfully realizes local edits while preserving asset structure.



Figure 7: Armadillo editing for "Wave the arms downwards." From left to right: input, Codex + Blender MCP with GPT-6 Astra at medium and max reasoning effort, and ViSculpt with Gemini 3.5 Flash Lite.

ViSculpt combines this editing quality with substantially lower runtime and cost in the Codex comparison: edits take 1–2 minutes on average at approximately $0.10 each, versus about 10 minutes and $4.00 for Codex with GPT-6 Astra (max). Together, these results highlight a training-free route to high-quality localized editing with a lightweight backbone.

Comparison with Generative Approaches Recent 3D generative models (Zhang et al., 2024; Li et al., 2024b; Yang et al., 2025; Team, 2025a; Xiang et al., 2025) have achieved impressive results in synthesizing novel 3D assets. Our setting differs in its objective: the edited mesh should remain identical to the input except for the localized change specified by the instruction. To probe this distinction, we construct a generative baseline using a "Render, Edit, Reconstruct" pipeline: render the input mesh into an image, edit the image with a text-guided 2D model, and reconstruct a 3D shape from the edited view using a large 3D generative model. We use Nano Banana Pro (Google DeepMind, 2025) for 2D editing and Hunyuan 3D 2.0 (Team, 2025b) for 3D generation.

As shown in the right part of Figure 12, this pipeline can produce a plausible object, but it often drifts from the original geometry. The bottleneck is the detour through a single image: projection discards hidden geometry, topology, and fine surface detail, while reconstruction must hallucinate the missing information. The result is therefore a regenerated mesh conditioned on an edited rendering, rather than an in-place edit of the original asset. This causes visible changes in regions that should remain untouched.

This comparison highlights the complementary roles of the two paradigms. Generative models are well suited to creating new shapes or large semantic redesigns, whereas our visual-centric agent directly manipulates the current mesh state through GUI feedback and is therefore better aligned with localized, preservation-critical edits. Attention-injection techniques (Shen et al., 2025; Hertz et al., 2022) may reduce appearance drift in image editing, but they do not remove the geometric information loss introduced once the editing process leaves the native 3D mesh.

Ablation: Without Primitive Abstraction One of the central design choices of our system is to restrict low-level execution to three primitive mouse actions: *Smear*, *Drag*, and *Draw*. This abstraction makes the action space tractable for a language-driven agent operating through a GUI. Without such primitives, the Action Agent would need to directly generate long, high-dimensional free-form mouse trajectories together with brush configurations, localization decisions, and execution timing.

In our formulation, the primitives decouple *what* to do from *how* to trace the stroke, letting the planner express compact intentions while deterministic geometric procedures realize trajectories. We view these three primitives as a key inductive bias that stabilizes execution and makes our visual-centric pipeline feasible. To validate this design, we compare our method against a baseline where the language model directly outputs the dense sequence of mouse coordinates needed for the geometric operation. As shown in Figure 12, removing the primitives sharply degrades performance: raw-coordinate outputs are often jagged and misaligned with the underlying 3D structure.

Ablation: QuadLoc We find that our QuadLoc strategy significantly outperforms naive VLM querying. As illustrated in Figure 8, directly prompting the VLM to output the coordinates of a specific part (e.g., the mouse's paw) results in inaccurate localization (left). In contrast, with QuadLoc, our method achieves precise localization of the target point (right).

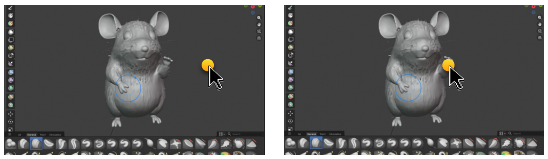
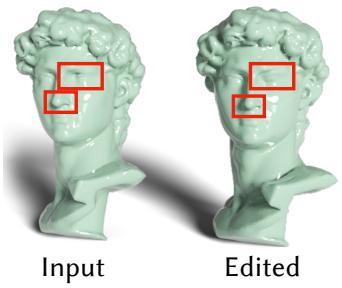


Figure 8: Ablation of QuadLoc. Directly prompting the VLM for coordinates yields inaccurate localization (left), while QuadLoc localizes the target more precisely (right).

6 Conclusion

We present a training-free multi-agent framework for 3D mesh editing that emulates the visual-centric workflow of human artists. Unlike script-centric methods, our system interacts directly with the Blender canvas, leveraging vision-language models to interpret user intent and assess progress via visual feedback. We bridge natural language and low-level actions by abstracting sculpting operations into three primitive mouse trajectories: *Smear*, *Drag*, and *Draw*. Our experiments demonstrate promising, intent-aligned localized edits that preserve the original mesh identity. Rather than claiming universal superiority over script-centric or generative approaches, it represents an alternative paradigm with distinct strengths and trade-offs. It highlights direct GUI-based interaction as a viable and distinct direction for editing existing 3D assets, particularly when precise in-place editing and region preservation are paramount.

Limitations and Future Work Several limitations remain.



First, the Planner Agent struggles to translate highly abstract instructions into concrete, multi-step actions. For instance, an ambiguous prompt like "make David more handsome" confuses the agent, leading to unpredictable deformations limited to basic *Clay Strip* brush operations, as illustrated in the figure. Second, despite our multi-view strategies, severe self-occlusions and complex internal geometries challenge our 2D screenshot-based perception. Equipping the agent with autonomous, free-viewpoint camera manipulation will substantially broaden its operational scope. Third, current mouse trajectory primitives are restricted to localized surface deformations. Incorporating *topological primitives* would enable fundamental structural edits, such as boolean operations, drilling holes, or merging objects. Finally, to address geometric artifacts imperceptible via 2D renderings (e.g., non-manifold edges or self-intersections), future systems could integrate a geometric analysis module into the Reflection Agent, thereby strengthening the alignment between visual plausibility and geometric validity.

AI USE STATEMENT

Generative models used within ViSculpt, in comparison baselines, and for auxiliary VLM evaluation are documented in the method and experimental sections. Generative AI also assisted with implementing, modifying, and debugging the agent system, Blender interaction code, and experimental scripts. We used Codex and Cursor with GPT and Claude models for manuscript drafting, revision, language polishing, translation, literature search and summarization, and reference formatting. Codex additionally assisted with LaTeX formatting, compilation, and layout checks.

Beyond the model uses documented in the method and experiments, we did not use generative AI for research ideation, conceptual or theoretical development, hypothesis formulation, experimental design, data or asset generation and curation, data processing, statistical analysis, result interpretation, or the creation or modification of graphical content in scientific figures and charts. Human reference edits were produced without AI assistance.

The authors reviewed and ran the code, checked experimental records and statistical results, verified citations against original sources, and inspected images and manuscript text. We take responsibility for the final content, claims, and artifacts of this work.

REFERENCES

Siddharth Ahuja and BlenderMCP Contributors. BlenderMCP - blender model context protocol integration, 2025.

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. In *SIGGRAPH*. 2003.

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.

Anthropic. System card: Claude sonnet 4.5. Technical report, Anthropic, 2025.

Anthropic. Introducing Claude Sonnet 4.6. `https://www.anthropic.com/news/claude-sonnet-4-6`, 2026.

Amir Barda, Vladimir Kim, Noam Aigerman, Amit Haim Bermano, and Thibault Groueix. Magicclay: Sculpting meshes with generative neural fields. In *ACM Trans. Graph. (SIGGRAPH Asia)*, 2024.

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 *CVPR*, 2025.

Blender Foundation. Blender MCP Server. `https://projects.blender.org/lab/blender_mcp`, 2026.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. CRC press, 2010.

Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Xin Jin, Liangchen Li, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. Segment anything in 3d with nerfs. *Advances in Neural Information Processing Systems*, 36, 2023.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024.

Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, et al. Img2cad: Conditioned 3-d cad model generation from single image with structured visual geometry. *IEEE Transactions on Industrial Informatics*, 2025.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*, 1999.

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Trans. Graph. (SIGGRAPH Asia)*, 37(6), 2018.

Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. Blenderllm: Training large language models for computer-aided design with self-improvement. *arXiv preprint arXiv:2412.14203*, 2024.

Rao Fu, Jingyu Liu, Xilun Chen, Yixin Nie, and Wenhan Xiong. Scene-llm: Extending language model for 3d visual understanding and reasoning. *arXiv preprint arXiv:2403.11401*, 2024.

Will Gao, Dilin Wang, Yuchen Fan, Aljaz Bozic, Tuur Stuyck, Zhengqin Li, Zhao Dong, Rakesh Ranjan, and Nikolaos Sarafianos. 3d mesh editing using masked lrms. In *ICCV*, 2025.

Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *ACM Trans. Graph. (SIGGRAPH)*, 1997.

GLM-V Team. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006v6*, 2025. URL https://arxiv.org/abs/2507.01006v6. Version 6, revised January 1, 2026.

Google. Gemini 3 pro model card. Technical report, Google, 2025.

Google DeepMind. Gemini 3 Pro Image model card. Technical report, Google DeepMind, November 2025. URL https://deepmind.google/models/model-cards/gemini-3-pro-image/.

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.

Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Visualization*, 1999.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3D scenes as blender code. In *ICML*, 2024.

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *ECCV*, 2024.

Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Trans. Graph. (SIGGRAPH Asia)*, 39(6), 2020.

Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. *NeurIPS*, 33, 2020.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023.

Hongbo Li, Haikuan Zhu, Sikai Zhong, Ningna Wang, Cheng Lin, Xiaohu Guo, Shiqing Xin, Wenping Wang, Jing Hua, and Zichun Zhong. Nasm: Neural anisotropic surface meshing. In *ACM Trans. Graph. (SIGGRAPH Asia)*, 2024a.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. *arXiv preprint arXiv:2405.14979*, 2024b.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A language and vision agentic framework for 3d scene generation. *arXiv preprint arXiv:2505.02836*, 2025.

Xinhang Liu, Yu-Wing Tai, and Chi-Keung Tang. Agentic 3d scene generation with spatially contextualized vlms. *arXiv preprint arXiv:2505.20129*, 2025.

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.

Jiaxi Lv, Yi Huang, Mingfu Yan, Jiancheng Huang, Jianzhuang Liu, Yifan Liu, Yafei Wen, Xiaoxin Chen, and Shifeng Chen. Gpt4motion: Scripting physical motions in text-to-video generation via blender-oriented gpt planning. In *CVPR*, 2024.

Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia*, 2022.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Update to gpt-5 system card: Gpt-5.2. Technical report, OpenAI, 2025.

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph. (SIGGRAPH)*, 31(4), 2012.

Qwen. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, et al. Infinite photorealistic worlds using procedural generation. In *CVPR*, 2023.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *CVPR*, 2024.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024a.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024b.

ByteDance Seed. Seed1.8 model card: Towards generalized real-world agency. Technical report, ByteDance, 2025.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural shape parser for constructive solid geometry. In *CVPR*, 2018.

Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Anything-3d: Towards single-view anything reconstruction in the wild. *arXiv preprint arXiv:2310.14261*, 2023.

Tiancheng Shen, Zilong Huang, Xiangtai Li, Zhijie Lin, Jiyang Liu, Yitong Wang, Jiashi Feng, Ming-Hsuan Yang, and Jun Hao Liew. Qk-edit: Revisiting attention-based injection in mm-dit for image and video editing. In *ICCV*, 2025.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Symp. Geom. Proc.*, 2004.

Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symp. Geom. Proc.*, 2007.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. In *3DV*, 2025.

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

Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. 2025b.

Xiaobao Wei, Renrui Zhang, Jiarui Wu, Jiaming Liu, Ming Lu, Yandong Guo, and Shanghang Zhang. Nto3d: Neural target object 3d reconstruction with segment anything. In *CVPR*, 2024.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *CVPR*, 2021.

xAI. Grok 4 model card. Technical report, xAI, 2025a.

xAI. Grok 4.1 model card. Technical report, xAI, 2025b.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *Tech report*, 2025.

Yutaro Yamada, Khyathi Chandu, Bill Yuchen Lin, Jack Hessel, Ilker Yildirim, and Yejin Choi. L3GO: Language agents with chain-of-3D-thoughts for generating unconventional objects. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, 2025.

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum*, volume 28, 2009.

Yunhan Yang, Xiaoyang Wu, Tong He, Hengshuang Zhao, and Xihui Liu. Sam3d: Segment anything in 3d scenes. *arXiv preprint arXiv:2306.03908*, 2023.

Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, Zi-Xin Zou, Yukun Huang, Ying-Tian Liu, Hao Xu, Ding Liang, Yan-Pei Cao, and Xihui Liu. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. *arXiv preprint arXiv:2507.06165*, 2025.

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.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. *ACM Trans. Graph. (SIGGRAPH)*, 43(4), 2024.

A Additional Qualitative Editing Results



Figure 9: Results of our method on various 3D mesh editing tasks. For each example, we show the input mesh on the left, the natural language description of the desired changes in the bottom, and the edited mesh generated by our method on the right.

B Additional Evaluation and Comparisons

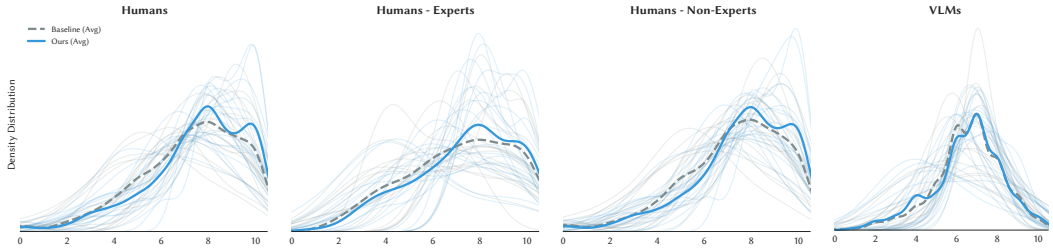


Figure 10: Visualization of score distributions across all 20 test examples. The plots display the Kernel Density Estimation (KDE) of user scores (0-10 scale) for our method (blue) and the baseline (gray). Thin semi-transparent lines represent the score distribution for each individual example, illustrating the performance consistency across diverse inputs. Thick lines indicate the aggregated average distribution of each group.



Figure 11: The comparison between our method and human editing. In each example, we show the input mesh on the left, the edited mesh generated by our method on the right, and the edited mesh created by human artists in the middle.

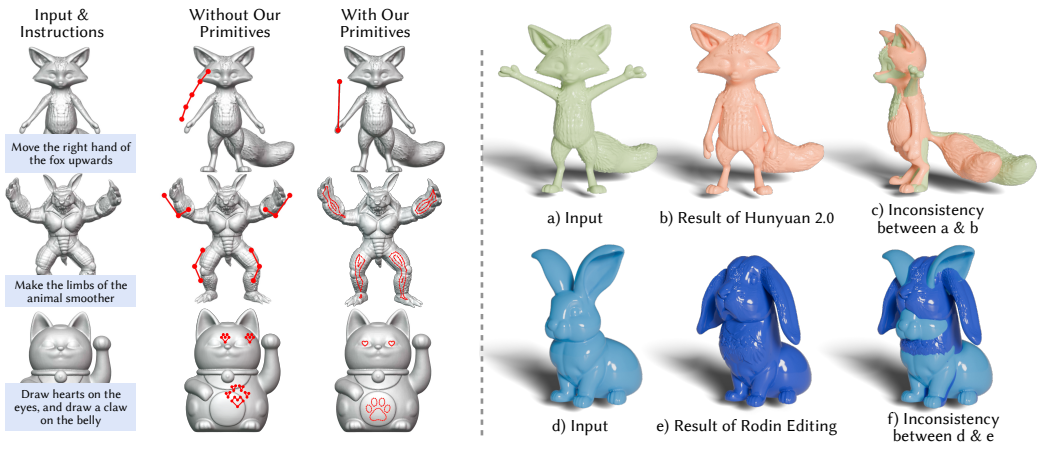


Figure 12: **Left: Ablation study on the effect of our Primitive Abstraction**. Compared to a baseline where the LLM directly outputs raw mouse coordinates, our primitive-based approach effectively grounds semantic instructions into precise, geometrically consistent GUI actions. **Right: Results of generative methods**. We test Hunyuan 2.0 and Rodin on our editing task. For Rodin, we use its editing mode. Given the instruction "Let the fox put its hand down" and "Put down the ears of the rabbit", the generated result exhibits significant geometric deviations from the original input. The overlay view (c) and (f) visualizes the structural misalignment between the input and output.

C Reference Library

To equip the agents with external knowledge, we maintain a Reference Library $\mathcal{R}$ queried via retrieval-augmented generation (RAG), as shown in Figure 13, which comprises three sub-libraries:

- *Blender Knowledge Library* stores official documentation for sculpting tools and Python APIs of Blender. The Translator queries this library to ensure that generated compatible JSON commands.
- *User-Provided References* store user-specific assets (e.g., custom brushes), user preferences, and style guidelines.
- *Previous Success Cases* store successful past operations. For a new task, the Planner retrieves the top five most similar cases to guide planning and translation. Please note that this part of the reference library is optional and is enabled only according to the user's specific needs. To prevent cross-task interference, we leave Previous Success Cases disabled in our main experiments, so each editing task is handled independently by a fixed, training-free configuration.

The Blender Knowledge Library and User-Provided References are populated before deployment, whereas Previous Success Cases are updated online during execution. When the Grader assigns a high score to a completed task, we archive the corresponding JSON command and execution context, enabling the agent to reuse effective strategies in subsequent tasks. Thus, the agent improves over time by accumulating high-quality exemplars.

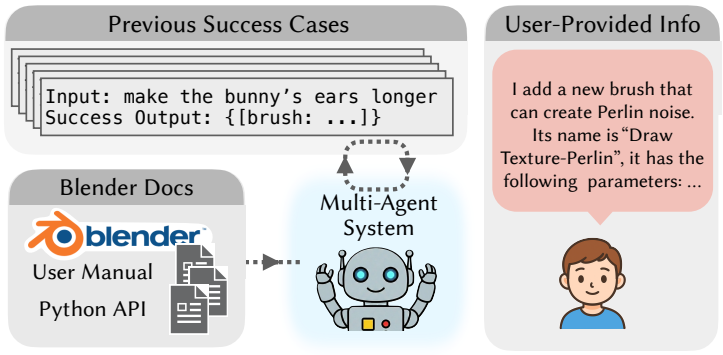


Figure 13: The Reference Library augments the multi-agent system via Retrieval-Augmented Generation (RAG). It integrates knowledge from three sources: Previous Success Cases (accumulated runtime experience), Blender Docs (official software manuals), and User-Provided Info (custom tool definitions).