MESHQUERY: AGENTIC SEAM PLANNING FOR UV PARAMETERIZATION

Anonymous authors
Paper under double-blind review

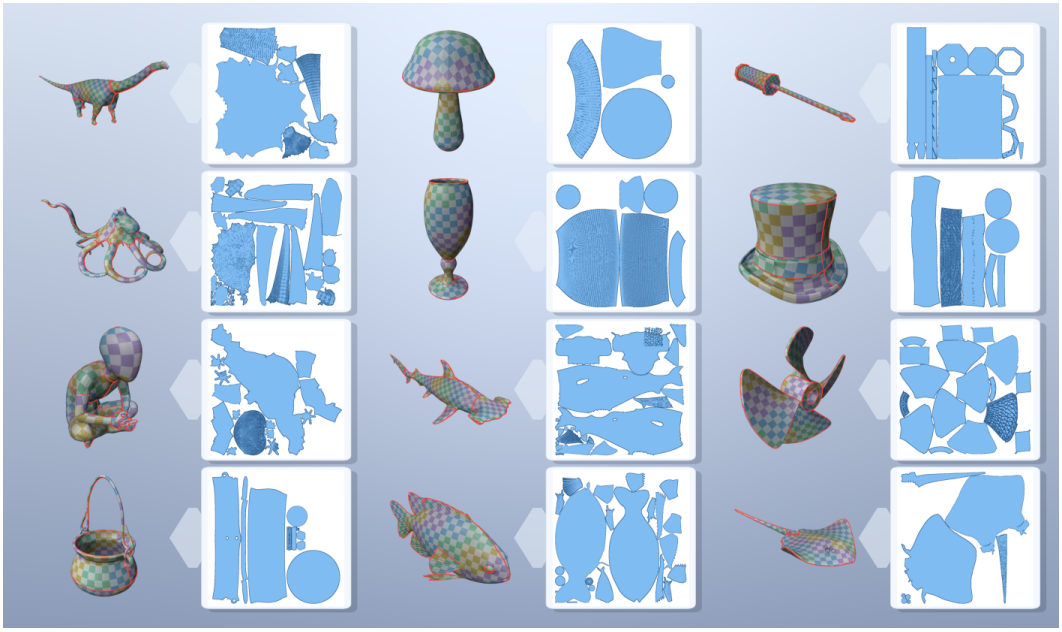

Figure 1: **MESHQUERY generates artist-aligned UV seams.** Conversely to hand-crafted geometric optimization, MESHQUERY enables a VLM agent to plan seams the way an artist does: it iteratively refines seams expressed as programs in a Domain-Specific Language (DSL), based on unwrapping feedback and on the mesh's geometric, topological, semantic attributes, and downstream needs.

ABSTRACT

We present MESHQUERY, a training-free agentic approach to automatic UV unwrapping of production-grade quad meshes. A Vision-Language Model (VLM) plans artist-aligned seams using a set of edge-selection tools, conditioned on domain-specific UV-unwrapping knowledge expressed in natural language and refined with a feedback loop. We design a queryable mesh representation together with a domain-specific language (DSL) that enables the agent to retrieve mesh information on demand, express a seam plan as a compact program of edge-selection operators over topological, geometric, and semantic mesh attributes, and iteratively refine it from UV quality feedback. On Adobe Substance 3D and Toys4K meshes, MESHQUERY produces $2.9\times/4.29\times$ fewer charts and $1.63\times/1.7\times$ shorter seams than the strongest baseline, and professional artists prefer its results in 80.9% of comparisons. Ultimately, decoupling high-level intent planning from low-level edge selection and compact mesh representation let MESHQUERY run on different backend VLM and scale to meshes an order of magnitude larger than autoregressive seam prediction.

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.

1 INTRODUCTION

Surface parameterization plays a key role in 3D modeling pipelines, for example, for texture mapping, mesh editing, decaling, and fabrication Sheffer et al. (2007). In most cases, the surface comes in the form of a 2-manifold polygon mesh, potentially open and/or exhibiting multiple connected components, and is first cut along seams to form components with disk topology, reduce distortion, and ensure a bijective mapping with the 2D domain, e.g., texture space. In this work, we focus on *seam planning*, i.e., the selection of edges to mark as seams. Seam planning is difficult to automate because it combines high-level considerations (semantics, downstream usage) and low-level geometric considerations (distortion, number of islands), which are hard to satisfy simultaneously. Moreover, high-level considerations are hard to quantify (semantics, symmetry) and often depend on the use case and type of shape. In practice, none of the existing automatic seam placement solutions Wang et al. (2025); Young (2018); Blender Foundation (2026); Zhang et al. (2024); Li et al. (2018) handle all cases well and recent data-driven approaches Li et al. (2025); Xu et al. (2025; 2026); Ma et al. (2026) struggle to scale to larger meshes and require training on a dataset of high-quality seams, which is difficult to acquire. Therefore, UV unwrapping remains a manual, time-consuming process.

In this paper, we introduce MESHQUERY, an agentic framework for planning seams on quad meshes built around a Vision-Language Model (VLM) equipped with domain specific tools. As we focus on the agentic loop, we limit our scope to quad-meshes, where multi-edge selection tools are readily available and of high quality. A core challenge is how the VLM interfaces with the input mesh. Interfacing directly with mesh vertices and edges is costly, especially for large meshes. An alternative is to encode the mesh into the VLM's latent space. We avoid this route, as it requires an additional training stage for the mesh encoder and is dependent on a specific VLM. Our solution works directly with the existing modalities of the VLM, namely text and images. Building on the VLM's ability to write programs, we represent the seam plan as a seam-selection program, for which we design a Domain Specific Language (DSL). The DSL expresses selection of edges by combining geometry (e.g., edge visibility, length) and connectivity (e.g., edge loops). The DSL additionally includes an active perception mechanism which selectively requests information from the mesh, allowing the method to scale to large meshes. The DSL lets us take advantage of the VLM's ability to reason about the task at a high level, while grounding the seam plan in precise, mesh-specific edge selection.

Contributions. We propose a novel agentic UV unwrapping framework able to generate high-quality 2D parameterizations on quad meshes (both organic and hard-surfaces) that can follow production-specific constraints, adapt to asset-specific requirements, and scale to meshes of more than $300k$ faces. We introduce a representation of the input mesh that is better suited for active and UV-seam centric perception by a VLM agent, and a Domain Specific Language to express seam selections in a compact and scalable way. We also report how we refined the system prompt (skill file) of our agentic system to improve surface parameterization quality, as an example of how individual artists may adapt the system to their needs. We demonstrate that MESHQUERYachieves better results than state-of-the-art, both qualitatively through a user study and quantitatively using common metrics such as distortion, island count, and UV occupancy.

2 RELATED WORK

Optimization-driven seam placement aims to find seams that minimize a user-specified objective, typically minimal seam length and surface distortion. Some include additional measures, such as seam visibility Sheffer & Hart (2002). Early works first compute a set of seams using heuristics (e.g., cutting through high-curvature regions or clustering normals) and unwrap separately Sander et al. (2001); Lévy et al. (2002); Gu et al. (2002); Zhou et al. (2004); Julius et al. (2005); Lucquin et al. (2017). Later works optimize seam length and distortion simultaneously, which they achieve by formulating a differentiable energy function Poranne et al. (2017); Sharp & Crane (2018), representing the cutting and mapping functions with neural networks for differentiable optimization Zhang et al. (2024); Srinivasan et al. (2024), or using specialized optimizers Sorkine et al. (2002); Li et al. (2018); Pietroni et al. (2022). The optimized objectives typically do not account for artistic considerations such as semantics. In our approach, we expect the VLM to reproduce these considerations from its training data (e.g., UV-unwrapping tutorials) and prompting (skill files) and make decisions based on these considerations through its vision- and language understanding. At a low level, our approach uses heuristic tools to mark seams.

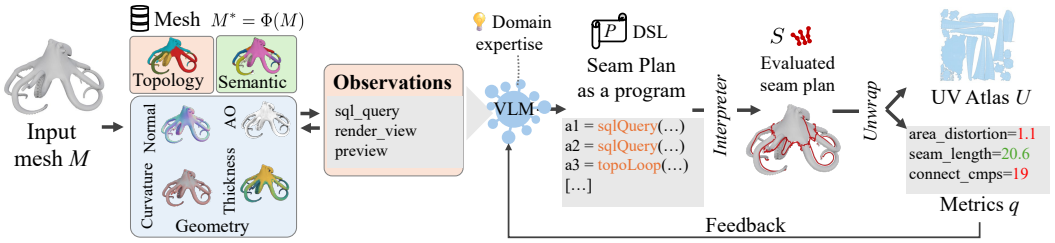

Figure 2: **Agentic UV seam planning.** The input mesh M is first converted into a queryable representation M^* that stores geometric, topological, and semantic attributes. Then a VLM agent, conditioned on seam-preference knowledge, iterates a three-stage loop: it *observes* M^* through SQL queries and seam-selection statement previews, then generates a cohesive program P, evaluates the plan through its UV quality metrics q, and iteratively refines it until all quality bounds are met.

Data-driven methods place seams based on examples of high-quality seams. Graph neural networks Teimury et al. (2020); Xu et al. (2026) and autoregressive methods Li et al. (2025); Xu et al. (2025); Chen et al. (2025); Ma et al. (2026) struggle with large meshes and are bound to the quality of the dataset they are trained on Deitke et al. (2023); Zhang et al. (2025). Because our agent operates on a high-level planning stage and uses tools for the low-level seam selection, we can scale to meshes with $> 300K$ vertices. PartUV Wang et al. (2025) circumvents the need for seam-specific training data using part segmentation Liu et al. (2025). We exploit implicit knowledge about seam placement in VLM rather than explicit seam-placement data. Like PartUV, we perform low-level seam placement with existing tools, but use a wider, different set of tools in an agentic feedback loop.

3D representations for LLMs encode geometry into tokens that LLMs can process. Prior work serializes meshes into language-compatible tokens Wang et al. (2024b); Fang et al. (2025) or projects 3D features to internal LLM representations Xu et al. (2024); Hong et al. (2023); Deng et al. (2025). However, encoding high-resolution input meshes is costly, and projections to VLM's latents are tied to the corresponding VLM. Instead, we query task-relevant mesh information *on demand* for efficient reasoning and refinement. Additionally, our framework supports any backend VLM. This principle is grounded in retrieval-augmented generation (RAG) Lewis et al. (2020); Asai et al. (2024), which selectively retrieves task-relevant information rather than placing all available evidence in context.

Agentic interface design determines how agents perceive and interact with an environment. Yang et al. (2024a) and Wang et al. (2024a) show that the representation of *observations* and *actions* can substantially affect performance. Low-level interactions are usually abstracted through structured skills, programs, tools, and planners, enabling reasoning over higher-level task Ahn et al. (2022); Huang et al. (2022); Singh et al. (2022); Liu et al. (2023); Yao et al. (2022); Schick et al. (2023); Shinn et al. (2023); Wang et al. (2023); Wu et al. (2023). Agents for computer use often involve specific point-and-click interfaces Zhang et al. (2026); Agashe et al. (2025). However, placing seams in 3D requires complex camera placement to enable point-and-click selection. We circumvent this problem by designing an efficient relational database interface for queries over mesh properties.

3 Method

We propose an agentic framework to mark seams on a mesh for UV unwrapping. Our framework (illustrated in Fig. 2) is an iterative loop where a VLM invokes fixed functions called *tools* and processes their responses. First, we build an efficient representation of the input mesh $M^* = \Phi(M)$ that a VLM can perceive using observation tools that query mesh semantics, geometry, and topology (Sec. 3.1). Second, we define a Domain-Specific Language (DSL) that enables a VLM to mark seams via aggregate seam-selection statements that operate on M^* (Sec. 3.2). The VLM previews each statement on the geometry through a DSL interpreter and composes a joint *seam selection program* P, which the interpreter instantiates as a seam plan S. Third, a fixed parameterization and packing operator Ψ processes S to produce the UV atlas, and we evaluate quality metrics q that we return to the agent as feedback (Sec. 3.3). The agent iteratively proposes new programs P until it meets the stopping criterion. Lastly, we specify a *seam preferences skill* (domain-specific system prompt) to

condition higher-level direction and strategy for UV seam planning (Sec. 3.4). This skill contains optional artist's knowledge that we distilled to help the agent create a higher-quality UV atlas.

3.1 QUERYABLE MESH REPRESENTATION

We define a preprocessor Φ which, before the agentic loop, computes attributes of the input mesh M and stores them in a relational database. This constitutes a queryable representation $M^* = \Phi(M)$ that enables SQL-based queries to retrieve compact mesh information using few LLM tokens, instead of serializing the whole mesh into the VLM's context.

Mesh Attributes fall into three categories. *Topological attributes* come from connectivity. A mesh is partitioned into topologically connected components, and each face and edge belongs to exactly one component *Geometric attributes* are either global (e.g., symmetry planes) or local to an edge or face (position, curvature, normal, length, dihedral angle, thickness, and ambient occlusion). We use ambient occlusion as a proxy for edge visibility, enabling the VLM to query hidden seams. *Semantic attributes* are text labels associated with each face and edge; edges between two labels are flagged as semantic boundaries. We obtain them by prompting a VLM to assign labels to mesh partitions defined by a segmentation algorithm (Liu et al., 2025). See Appendix A.3 for the full list of attributes.

3.2 SEAM-SELECTION DSL

A seam plan is a subset of mesh edges, $S \subseteq E$, selected as seams. The agent specifies this plan through a *seam selection program* P, in a domain-specific language (DSL) whose execution by the interpreter produces S. A program consists of a jointly generated collection of statements $P = \{p_1, \ldots, p_N\}$. When P is evaluated, a topological sort ensures that no statement is executed before its inputs are available. A program is thus rejected if it does not constitute a directed acyclic graph. We also statically check that each input has the expected type; otherwise, we reject the program. Lastly, a program can be rejected at runtime, as some operators may raise exceptions when their result is likely to lead to bad seams. Fig. 3 shows our program-based representation of edge sets.

Statements correspond to calls to our built-in edge-selection operators. Each statement p_i specifies an operator τ, a list of input arguments provided to this operator, and an output name. Each input argument is either a constant literal (e.g., an SQL query) or the output of a previous statement, referenced by its name. To assist the agent in exploring the space of possible programs, static typing ensures that the input arguments match the data type operator τ expects. Since many 3D shapes exhibit a symmetry along an axis, each statement may carry a `mirror` tag, in which case its inputs are reflected across the detected symmetry plane. In order to ease agent iterations, our DSL also allows the insertion of comments about the intent of a statement.

Operators τ are the core routines that operate on the mesh representation M^* and return sets of edges. Crucially, they enable the VLM to work at a higher level of abstraction than individual edge selection. For example, `topological_loop` and `geodesic` are operators that return aggregate sets of edges. Operator arguments may refer to geometric quantities that are difficult for the VLM to specify exactly; therefore, each operator can be evaluated in *preview mode* τ^*. The VLM previews operators to validate that they produce a non-empty set of edges. To support this further, the DSL also includes *search* variants $\hat{\tau}^*$. Rather than requiring an exact argument θ, a search statement specifies a small candidate region $\theta + \epsilon$, and returns ranked candidates as executable DSL statements (e.g., `find_loops`(θ) returns ranked candidates `topological_loop`$(\theta + \epsilon)$). See Appendix 4 for implementation details of the full operators collection.

Interpreter evaluates the submitted program P into a seam plan S expressed as an explicit list of edge indices. It parses all statements p_i and orders them with respect to their dependencies (a statement can only be executed once all its arguments that reference other statements have been evaluated). The interpreter may reject DSL statements and provide informative error messages for cyclic dependencies, static typing issues, or runtime errors.

3.3 UV QUALITY FEEDBACK LOOP

At iteration n, the agent evaluates its seam plan S_n by calling the parameterization algorithm Ψ, which flattens the 3D mesh into a 2D UV atlas based on S_n. The agent can then assess its quality

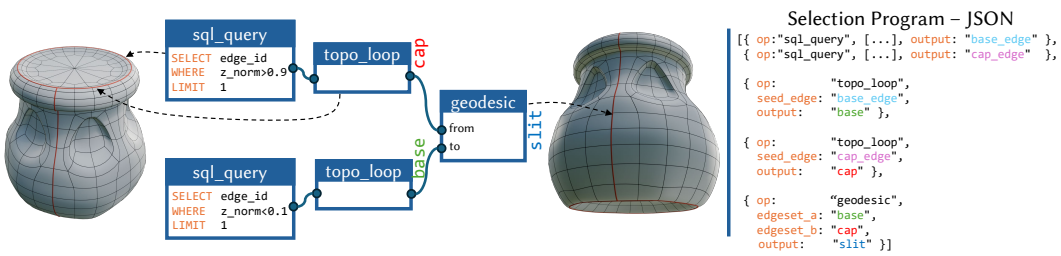

Figure 3: A **domain-specific language (DSL)** allows expressing edge selection and plan manipulation via a high-level seam selection program (JSON) based on aggregate edge selection operators.

using feedback metrics q_n, assign an ID to S_n for future reference (e.g., comparison, rollback), stop iterating once it satisfies the stopping criterion, and return its best plan $S_n^\star$.

Feedback metrics q help the agent assess the quality of the UV atlas. resulting from its proposed seam plan S_n and refine it for the next iteration S_{n+1}. Feedback is typically provided as compact text that consumes few VLM tokens. It features *global UV atlas metrics* such as seam length, number of UV charts, distortion, and seam network connectivity. It also includes *local UV chart metrics* such as chart convexity and flags seams associated with high-distortion UV charts. In complex situations, the VLM may also request distortion heatmap visualizations rendered on the 3D model. The full implementation of the metrics is listed in Appendix A.6.

Stopping criterion. The agent loop terminates when the quality criteria are satisfied or when the maximum time budget is exceeded. The quality criteria require that no face collapses to zero UV area, and that the maximum per-face area distortion and mean conformal distortion are below user-specified thresholds: $D_{\text{area}}^{\max} < c_1 \wedge D_{\text{conf}} < c_2$.

Plan selection. The agent returns the best-scored plan $S_n^\star$ based on a lexicographic ranking. Plans are first ranked by feasibility (area and conformal distortion under predefined target thresholds), then by fewer sliver charts (charts below 0.5% of the surface), then by lower overall and conformal distortion, then by higher convexity and UV space utilization. If no plan is feasible, it returns the one with the smallest relative violation of the distortion thresholds. Note that the preferences skill described below can be used to steer this design towards other specific goals.

3.4 Seam Preferences Skill

To guide the VLM in solving the seam-planning task, we provide API documentation for the agent tools and the seam-selection DSL grammar, with examples. In addition to conditioning the VLM planning strategy, we manually distill seam placement handbooks from publicly available online resources such as Blender Foundation (2026) and Maya Autodesk (2023). For example, we include recommendations for handling common shapes, as well as situational cases such as head shapes, which, for instance, require a single seam line from the neck to the top of the head.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on two 50-asset datasets. We randomly sample 50 assets from 18 available classes of Adobe Substance 3D Assets library Adobe (2026) to form Substance-50. It contains production-grade quad meshes with artist UVs and spans 1,344–53,634 polygons (median 9,392). In addition, we randomly sample 50 quad meshes from the Toys4K dataset Stojanov et al. (2021) to form Toys4K-50, spanning 61–330,640 polygons (median 4,752). See Appendix A.1.

Baselines. We compare against multiple baselines on the produced seams and related UV atlas on both datasets. UV atlas receives the same rigid average island scaling and packing via Blender at a 0.001 margin. We compare against FlattenAnything (Zhang et al., 2024), OptCuts (Li et al., 2018),

XATLAS (Young, 2018), Blender Smart UV Blender Foundation (2026), and PARTUV (Wang et al., 2025). On SUBSTANCE-50, we also compare with artists' UVs. See Appendix A.4.

Evaluation protocol. We quantify seam and UV atlas quality with several metrics. We measure *UV chart count*, *UV seam length* as the total chart-boundary length in UV space, and normalized *3D seam length*. To assess seam visibility, we measure mean normalized *seam ambient occlusion (AO)*. We also include *Non-Disk* as the fraction of UV charts classified as non-disks, *Compactness* as a chart's similarity to a perfect circle, chart *Convexity*, and seam *Jaggedness* of UV charts' boundary regularity. Lastly, we compute standard distortion metrics such as *area*, *conformal*, and *texel spread*, and report the metric implementations in the Appendix A.6.

Implementation details. We set the VLM to Claude Opus 5 with high reasoning effort, default sampling configuration, and a 1M token context. It operates our tools through the Claude Code Harness. We evaluate the parameterization and DSL operators in Blender 5.1.2 on CPU (see Appendix A.5). We use PartField as semantic edge- and face- priors. We set runtime timeout to 40 min, and the stopping criterion to distortion constants to $\big[D^{\max}\text{area} < 2.0\big] \wedge \big[D\text{conf} < 1.5\big]$.

4.2 RESULTS

Quantitative results. We report the comprehensive benchmark in Table 1. Our method outperforms all competitor baselines on seam-quality metrics and produces a valid seam set and UV atlas for all mesh sizes. Compared to the most competitive baseline, our method achieves 2.90×/4.29× fewer UV charts, 1.63×/1.69× shorter UV seams, and 1.44×/1.66× shorter 3D seams than PARTUV on SUBSTANCE-50/TOYS4K-50 with minimal distortion cost. Furthermore, our metric statistics closely align with the artist reference; e.g., 19.6 for ours vs. 19.5 in the number of UV charts. Our method consistently produces high-quality seams that yield clean cuts on the 3D surface, especially in more occluded areas whenever possible. However, seam placement is constrained by limitations of the UV parameterization. In production, even charts cut along artist-authored seams are typically refined manually by relaxing and pinning UV coordinates before texturing. This refinement step is orthogonal to seam generation and therefore outside the scope of this work.

Human evaluation results. We conduct an A/B study comparing Ours against PARTUV, the strongest baseline in Tab. 1. From a fixed randomly sampled pool of 80 meshes (40 from each evaluation set), every annotator rates a random subset of 40 (20 per set), choosing the better seams or marking a tie. Across 15 annotators (10 professional artists) and 600 ratings, annotators prefer Ours on 79.7% of decisive ratings (ties excluded); the preference holds on both SUBSTANCE-50 75.7% and TOYS4K50 83.6%. Restricted to the 10 professional artists, the overall preference (ties excluded) is 80.9%. Beyond the pairwise comparisons, annotators also flag whether the preferred seams are usable as a hot-start for production work. Ours-preferred seams are marked at 53.0%. Our approach therefore aligns more closely with professional artist expectations than the strongest available baseline.

Qualitative results. Fig. 4 shows comparisons to baselines on the TOYS-4K dataset. MESHQUERY solves the UV unwrapping through a minimal set of seams. Generally, our method produces seams in occluded areas, e.g., under the Panda's arms, or around high-dihedral-angle edges, e.g., Candy, and follows the natural structure of typical objects when mesh attributes such as ambient occlusion and curvature are at parity, e.g., the Ball.

Mesh size performance scaling. In Fig. 5(left), we show dot plots of tool calls and wall-clock runtime across increasing polygon sizes. Notably, the number of tool calls (top left), which reflects the agent's high-level reasoning, correlates negligibly with mesh size (Spearman $\rho = +0.11$ over all meshes). For meshes with similar semantics, geometry, and topology, it interpolates toward a constant value. We argue that this level of abstraction enables reasoning to depend on shape complexity rather than polygon count. For example, mugs - despite having a high vertex count range of $1,608 - 250,296$ (157x span)- and Plate at $2,786 - 33,794$ (12x span) - retain similar high-level plans and tool-call counts that lead to high-quality UV parameterizations. While the operators are bounded by $O(n \log n)$ time complexity as the polygon count n increases, the comprehensive solution runtime (bottom left), however, is bounded by the feedback loop parameterizations that require sparse linear solves over $O(n)$ mesh vertices, yielding superlinear time $O(kn^{3/2})$ for manifold meshes under sparse direct factorization. Our experiments achieve a 11.8-minute runtime, of which 71% is

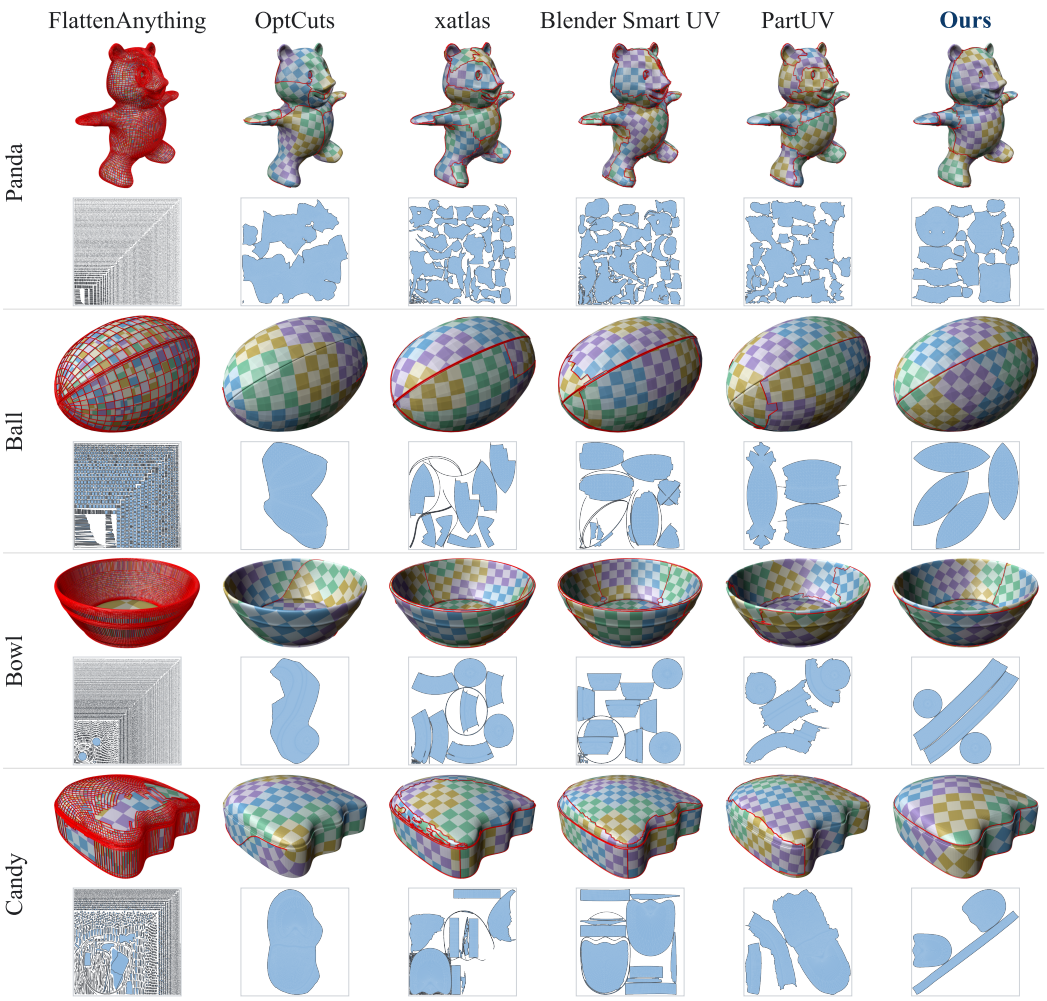

Figure 4: **Qualitative comparison on TOYS4K-50.** For each asset, the top row shows a UV checkerboard and generated seams in red, while the bottom row shows the parametrized UV atlas.

model reasoning and about 29% is tool server time. MESHQUERY scales up to $300k$ polygons, one order of magnitude higher than autoregressive methods Li et al. (2025); Xu et al. (2026).

Mesh Preprocessing Time. Building the mesh representation M^* is a one-off cost which takes median 2.6 minutes (range 1.7–15.0), of which the relational representation is 2.8 seconds (at most 21.3 seconds on the largest asset). The majority of the cost comes from PartField segmentation over a fixed number of partitions (17.4%, median 36.8 seconds) and a one-time VLM labeling cost per partition (totaling 77.9%, median 105.8 seconds). See Appendix Fig. 9.

VLM Token cost. The model reads a near-constant amount of text tokens about a mesh regardless of its size (Fig. 5(right)). Over 100 runs, the tool output read by the agent has a median of $35,934$ tokens per run (Spearman $\rho = +0.15$ against log polygon count), and the outer face quartiles differ by only $1.12\times$; the budget is similar on both datasets (SUBSTANCE-50 median 35,924, TOYS4K-50 36,159). Most of this text is feedback from the parametrized UV atlas (67%) rather than perception (25%). Conversely, serializing the mesh instead would scale linearly with resolution: the same asset's OBJ file is a median of 229,604 tokens ($\rho = +0.99$), exceeding the observed text by up to $465\times$ on the largest meshes. This compact representation results in a median context size of 157k tokens (max 503k, $\rho = +0.23$). Model output is dominated by thinking (median 38,922 of 62,940 output tokens per run) and tracks asset complexity rather than polygon count.

Table 1: **UV parameterization benchmark**. Here, (n) is the number of successful runs. We highlight in **bold** and underlined values that indicate the best and second-best results, respectively. Methods producing failure cases (†) or excessively high mean distortion (‡) are excluded from the ranking.

	Seam quality								Distortion		
Method (n)	UV Charts ↓	UV Seam length ↓	3D Seam length ↓	AO ↑	Non-disk ↓	Compact. ↑	Convex. ↑	Jagged. ↓	Texel spread ↓	Area dist. ↓	Conformal dist. ↓
Substance-50											
FlattenAnything Zhang et al. (2024)‡ (50)	(19.7)	(17.0)	(5.9)	(0.545)	(0.298)	(3.042)	(0.902)	(0.58)	(1.703)	(4.067)	(43.716)
OptCuts Li et al. (2018)† (38)	(3.8)	(6.3)	(3.4)	(0.504)	(0.051)	(0.473)	(0.819)	(0.84)	(0.197)	(1.172)	(1.133)
XATLAS (LSCM) Lévy et al. (2002) (50)	355.5	44.2	26.6	0.503	0.065	0.428	0.853	1.21	0.163	**1.095**	1.111
Blender Smart UV Sheffer et al. (2007) (50)	162.6	37.9	21.5	0.521	0.071	0.408	0.807	1.14	**0.159**	1.102	1.162
PARTUV Wang et al. (2025) (50)	56.8	20.6	9.8	0.493	0.055	0.444	0.792	**0.74**	0.171	1.117	**1.030**
Ours (50)	**19.6**	**12.6**	**6.8**	**0.527**	**0.014**	**0.611**	**0.891**	0.78	0.179	1.178	1.101
Artist reference (50)	19.5	13.8	7.0	0.547	0.059	0.568	0.886	0.50	0.166	1.117	1.086
Toys4K-50											
FlattenAnything Zhang et al. (2024)† (49)	(13974.4)	(108.1)	(159.2)	(0.499)	(0.002)	(0.509)	(0.984)	(0.91)	(0.761)	(1.596)	(17.187)
OptCuts Li et al. (2018)† (37)	(1.5)	(4.6)	(3.7)	(0.490)	(0.198)	(0.413)	(0.786)	(0.53)	(0.190)	(1.183)	(1.140)
XATLAS (LSCM) Lévy et al. (2002) (50)	885.0	33.6	24.6	0.519	**0.040**	0.418	0.841	0.68	**0.138**	**1.094**	1.386
Blender Smart UV Sheffer et al. (2007) (50)	151.6	23.9	17.3	0.522	0.067	0.394	0.812	0.70	0.156	1.097	1.156
PARTUV Wang et al. (2025) (50)	39.9	15.0	10.1	0.501	0.108	0.425	0.804	0.82	0.192	1.143	**1.054**
Ours (50)	**9.3**	**8.9**	**6.1**	**0.533**	0.063	**0.610**	**0.904**	**0.47**	0.215	1.201	1.172

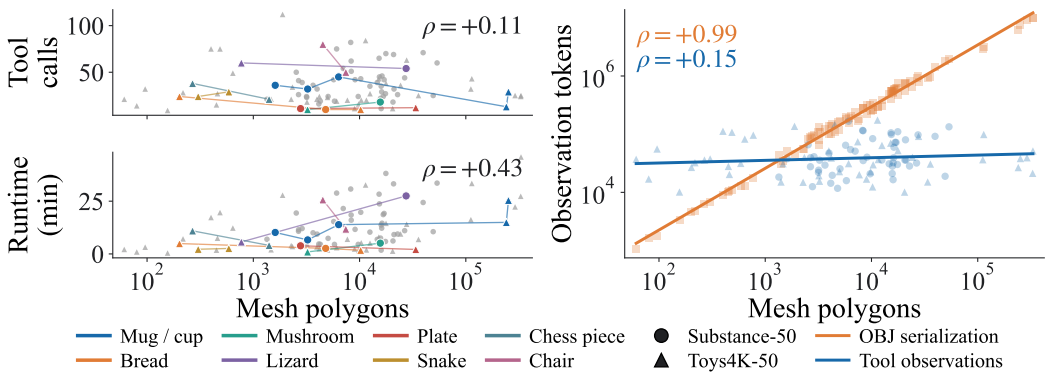

Figure 5: **Tool calls, runtime, and token consumption across varying mesh polygon sizes.** *Left*: Colored paths connect meshes from the same semantic class, but with different geometry and topology. Tool call exhibits high-level agent reasoning, has negligible correlation with mesh size, while runtime scales super-linearly due to parameterization feedback cost per iteration. *Right*: VLM token cost across varying mesh polygon sizes is near-constant, while direct mesh serialization grows linearly.

4.3 Ablation Studies

Feedback Loop. In Tab 2 (a), we observe that the plans with feedback loops are better than the first attempt on every distortion metric. The UV feedback raises the share of meshes inside the acceptable distortion gate from 66% on the first attempt to 97%. The agent applied 342 plan versions over the 100 meshes, a median of 3 per mesh (mean 3.42, max 9). Most gains occurred within 3 iterations, reaching 91 accepted plans, and after 9 revisions, reaching 97% acceptance. Therefore, we consider the closed-loop feedback as an essential component of our solution. See Fig. 6 for an example.

LLM model comparison. In Tab 2 (b) we compare benchmark evaluations between Claude Opus 5 and Qwen3.8-27B. We see a significant improvement when using frontier models with stronger reasoning and higher token capacity (200K vs 1M context tokens). Nonetheless, the small open-source model completed the task, though at higher distortion costs, validating that the proposed method is portable to smaller models.

Table 2: **Ablations.** We ablate the feedback loop (a) and the LLM model (b) and report distortion metrics (red indicates regression).

Variant		Area dist. ↓	Angular dist. ↑	Conformal dist. ↓	Texel spread ↓
(a) Closed-loop feedback					
Substance-50	**With feedback (ref.)**	1.151	0.928	1.100	0.178
	No closed loop	1.341 +16.5%	0.887 −4.4%	1.419 +29.0%	0.377 +111.6%
Toys4K-50	**With feedback (ref.)**	1.201	0.903	1.173	0.215
	No closed loop	1.525 +27.0%	0.822 −8.9%	1.359 +15.9%	0.512 +137.8%
(b) LLM model					
Substance-50	**Claude Opus 5 (ref.)**	1.151	0.928	1.101	0.178
	Qwen3.8-27B	1.362 +18.3%	0.854 −7.9%	1.370 +24.4%	0.348 +95.4%
Toys4K-50	**Claude Opus 5 (ref.)**	1.201	0.903	1.172	0.215
	Qwen3.8-27B	1.450 +20.7%	0.838 −7.2%	1.750 +49.4%	0.390 +81.1%

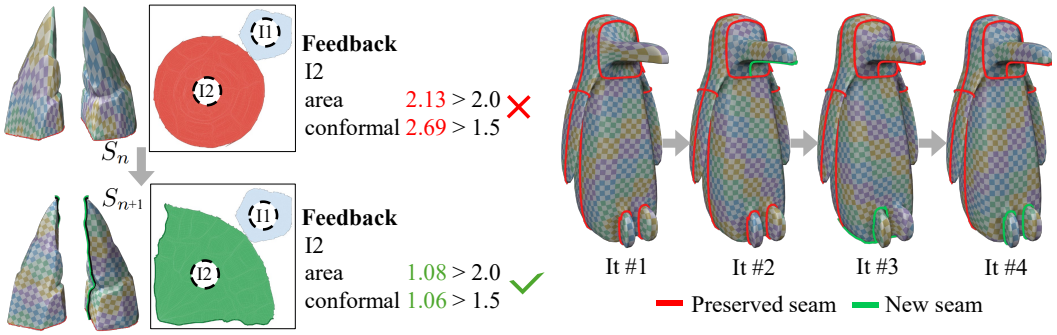

Figure 6: **Closed-loop feedback.** ***(Left)*** The cone of the rock chart fails distortion bounds. After refinement, the agent adds one apex-to-base slit, opening the cone into a fan while leaving the base chart untouched. *Right*: a penguin model with the generated seams at each iteration.

5 Conclusion

We introduced MeshQuery, an agentic framework for UV unwrapping in which a VLM perceives a mesh through compact queries, acts via a seam-selection DSL, and refines its plan based on parameterization feedback. We let the agent operate at a higher level of abstraction than individual edge selection, via aggregate operators, enabling the VLM to focus on high-level intent. Our experimental results show strong human alignment, achieving over 80.9% professional artist preference over SOTA baselines across a benchmark of small to large meshes and hard and organic surfaces. In addition, artists rate 53% of our meshes as usable as hot-starts for production use. Lastly, our mesh and DSL representation is highly compact and token-efficient, enabling a VLM to efficiently reason on large and complex meshes with 10^5 faces, an order of magnitude higher than autoregressive seam placement methods.

Challenges and future work. We focus on quad meshes, as they are well suited for existing multi-edge selection tools, like edge loops and are the main class used in production. Our main contributions (DSL, mesh queries, skill files) theoretically generalize to triangular meshes and early explorations with SeamCut Lucquin et al. (2017) are encouraging. Regarding runtime, there are two bottlenecks. First, PartField Liu et al. (2025) can be costly for larger meshes. We expect that improvements in 3D segmentation will transfer directly to speed and accuracy improvements for MeshQuery. Second, the iterative agentic loop is expensive. We expect that efficiency gains from general agentic frameworks and VLMs (e.g., Qwen in Table 2) as well as more informative evaluation signals would translate in fewer cycles. Finally, we focus on the placement of seams. In future work, we want to explore the interaction of parameterization jointly with UV-placement, e.g., for refining UV-maps or selecting the right parameterization approach for a given topological disk.

AI USE STATEMENT

In this work, we used generative AI tools for **[tasks with required disclosure]** *implement the method "Seamcut" Lucquin et al. (2017) from its paper description (no reference code is publicly available), which we include in the agent toolset; the implementation was verified against the paper's described behavior via automated tests and reviewed by 2 authors.* We have not used generative AI tools for **[other tasks with required disclosure]** *help develop theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, support qualitative and thematic data analysis, interpret results*, and **[the rest of the required disclosure tasks]** *generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation, clean and reformat dataset,* are not applicable to this work.

Additionally, we used generative AI tools for **[tasks with recommended disclosure]** *create or modify scientific figures or images, create or edit software code*. We have reviewed all AI-assisted work. For example, 2 authors verified and tested LLM-generated code for correctness. Figures were verified and tested by 1 author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Adobe. Substance 3D Assets. `https://substance3d.adobe.com/assets/`, 2026. Accessed: 2026-09-18.

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Wang. Agent s: An open agentic framework that uses computers like a human. In *International Conference on Learning Representations*, volume 2025, pp. 22924–22946, 2025.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Autodesk. Uv mapping tips. `https://help.autodesk.com/view/MAYAUL/2023/ENU/?guid=GUID-1CEF8A5B-1051-40F1-ACFE-F29AD066F56F`, 2023. Autodesk Maya 2023 User Guide, accessed September 21, 2026.

Blender Foundation. Smart uv project. Blender Manual, 2026. URL `https://docs.blender.org/manual/en/latest/modeling/meshes/editing/uv.html`. Accessed: 2026-09-18.

Yuguang Chen, Xinhai Liu, Yang Li, Victor Cheung, Zhuo Chen, Dongyu Zhang, and Chunchao Guo. Artuv: Artist-style uv unwrapping. *arXiv preprint arXiv:2509.20710*, 2025.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13142–13153, 2023.

Jiajun Deng, Tianyu He, Li Jiang, Tianyu Wang, Feras Dayoub, and Ian Reid. 3d-llava: Towards generalist 3d lmms with omni superpoint transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3772–3782. IEEE, 2025.

Shuangkang Fang, I Shen, Yufeng Wang, Yi-Hsuan Tsai, Yi Yang, Shuchang Zhou, Wenrui Ding, Takeo Igarashi, Ming-Hsuan Yang, et al. Meshllm: Empowering large language models to progressively understand and generate 3d mesh. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 14061–14072, 2025.

Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM Trans. Graph.*, 21(3): 355–361, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566589.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Alec Jacobson and contributors. common-3d-test-models: Repository containing common 3d test models in original format with original source if known and obj mesh. `https://github.com/alecjacobson/common-3d-test-models`, 2013. Accessed: 2026-09-22.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum*, 2005.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *2019 IEEE/CVF conference on computer vision and pattern Recognition (CVPR)*, pp. 9593–9603. Ieee, 2019.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566590. URL `https://doi.org/10.1145/566654.566590`.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Minchen Li, Danny M Kaufman, Vladimir G Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM transactions on graphics (TOG)*, 37(6): 1–13, 2018.

Yang Li, Victor Cheung, Xinhai Liu, Yuguang Chen, Zhongjin Luo, Biwen Lei, Haohan Weng, Zibo Zhao, Jingwei Huang, Zhuo Chen, et al. Auto-regressive surface cutting. *arXiv preprint arXiv:2506.18017*, 2025.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9704–9715. IEEE, 2025.

Victor Lucquin, Sebastien Deguy, and Tamy Boubekeur. Seamcut: Interactive mesh segmentation for parameterization. In *ACM SIGGRAPH 2017 Technical Briefs*, 2017.

Xueqi Ma, Xingguang Yan, Congyue Zhang, and Hui Huang. Meshtailor: Cutting seams via generative mesh traversal. *arXiv preprint arXiv:2603.27309*, 2026.

Nico Pietroni, Corentin Dumery, Raphael Falque, Mark Liu, Teresa Vidal-Calleja, and Olga Sorkine-Hornung. Computational pattern making from 3D garment models. *ACM Transactions on Graphics*, 41(4):1–14, July 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530145. URL `https://dl.acm.org/doi/10.1145/3528223.3530145`.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. Autocuts: simultaneous distortion and cut optimization for uv mapping. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 409–416, 2001.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Nicholas Sharp and Keenan Crane. Variational surface cutting. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

Alla Sheffer and John C Hart. Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Visualization, 2002. VIS 2002.*, pp. 291–298. IEEE, 2002.

Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundations and trends in computer graphics and vision*, 2(2):105–171, 2007.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *IEEE Visualization, 2002. VIS 2002.*, pp. 355–362. IEEE, 2002.

Pratul P Srinivasan, Stephan J Garbin, Dor Verbin, Jonathan T Barron, and Ben Mildenhall. Nuvo: Neural uv mapping for unruly 3d representations. In *European conference on computer vision*, pp. 18–34. Springer, 2024.

Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1798–1808. IEEE, 2021.

Fatemeh Teimury, Bruno Roy, Juan Sebastián Casallas, David MacDonald, and Mark Coates. Graphseam: Supervised graph learning framework for semantic uv mapping. *arXiv preprint arXiv:2011.13748*, 2020.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *ICML*, 2024a.

Zhaoning Wang, Xinyue Wei, Ruoxi Shi, Xiaoshuai Zhang, Hao Su, and Minghua Liu. Partuv: Part-based uv unwrapping of 3d meshes. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480. IEEE, 2025.

Duoteng Xu, Yuguang Chen, Jing Li, Xinhai Liu, Xueqi Ma, Zhuo Chen, Dongyu Zhang, and Chunchao Guo. Seamcrafter: Enhancing mesh seam generation for artist uv unwrapping via reinforcement learning. *arXiv preprint arXiv:2509.20725*, 2025.

Hao Xu, Yuqing Zhang, Yiqian Wu, Xueqi Ma, Ding Liang, Yan-Pei Cao, Ying-Tian Liu, and Xiaogang Jin. Seamgen: Artist-aligned uv seam generation via graph flow matching. *arXiv preprint arXiv:2607.12379*, 2026.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *European Conference on Computer Vision*, pp. 131–147. Springer, 2024.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024a.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Jonathan Young. xatlas: Mesh parameterization / UV unwrapping library. `https://github.com/jpcy/xatlas`, 2018. Accessed: 2026-09-19.

Qijian Zhang, Junhui Hou, Wenping Wang, and Ying He. Flatten anything: unsupervised neural surface parameterization. *Advances in Neural Information Processing Systems*, 37:2830–2850, 2024.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025.

Yinuo Zhang, Zian Cheng, Ziya Zhao, Zongyu Li, Bingshuo Liu, Qingbin Liu, Junxian Cai, Xi Chen, Zhiying Tu, Dianhui Chu, Xiaoyan Yu, and Dianbo Sui. PSBench: Editing image via GUI agents in photoshop. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=O93cZGxYB1`.

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 45–54, 2004.

Appendix

A Implementation Details

A.1 Datasets

Table 3 extends the dataset description in Sec. 4.1. SUBSTANCE-50 spans 18 distinct categories of the Adobe Asset Library, has 28 assets with multiple connected components; 27 are bilaterally symmetric, and five have an estimated genus of at least two. TOYS4K-50 covers 44 categories and contains single-component, 2-manifold meshes; 29 are bilaterally symmetric, and one has estimated genus at least two. The distribution of mesh sizes is comparable to commonly used 3D mesh datasets, as shown in Fig. 7. In contrast, SUBSTANCE-50 includes a selection of production-grade and manually artist-composed quad meshes. Fig. 13–16 show all evaluation assets.

A.2 DSL operators

We illustrate the set of tools that compose the agent interface at Tab 4. For every tool, we report its signature, description, and time complexity. We include a tool for each DSL operator, plus additional tools for mesh observation and plan management.

A.3 Mesh Representation

Database model The eight tables in `mesh.db` (Fig. 8) follow the Blender hierarchy: `collection` stores asset paths, scene renders, and textual asset descriptions; `mesh` stores global information; and `component` stores connected-component topology, vertical extent, and localized genus cuts. The `face` and `edge` tables contain geometric signals and semantic labels, with endpoint, direction, length, and semantic-boundary fields for edges and area rank for faces. `face_edge` and `edge_neighbor` encode incidence and adjacency. Precomputed candidates reside in `feature`, with their type, seed edge, member edges, extent, score, and type-specific attributes. Coordinates use the Blender Z-up frame and isotropic normalization $(p_a - \min_a)/s$, where $s = \max(\text{span}_x, \text{span}_y, \text{span}_z)$. Suffix `_pct` denotes a within-mesh rank in $[0, 1]$, and `_raw` retains world-unit or radian values.

Segmentation and semantic labels. (Fig. 9) A single PartField (Liu et al., 2025) inference pass (10 samples per face, batches of 100k) yields an agglomerative face tree, which is cut into $K = 8$ clusters by undoing the last $K - 1$ merges. Face-level labels are transferred to the input polygons by nearest normalized centroid, and islets created by the transfer are merged back so that every cluster is one connected region. Claude Sonnet 5 first names the object and its surface type, then labels up to eight clusters in descending face count. Each call receives the original and highlighted six-view renders and the existing label vocabulary. Remaining clusters are `unlabelled`. Faces inherit their cluster label, edges inherit the label of their left face, and an edge is flagged as a semantic boundary when its two faces carry different labels.

Mesh features. Ambient occlusion casts 8 Fibonacci-lattice hemisphere rays of length $0.3\,d$ (d the bounding-box diagonal) from each edge midpoint, offset by 0.002 of the ray length along the averaged normal; the hit fraction is rank-normalized. Thickness casts 4 inward rays in an $8°$ cone and stores the median hit distance (NULL for open sheets). Curvature is the signed vertex angle deficit $2\pi - \sum_i \alpha_i$ (positive on domes, negative on saddles), averaged over an edge's two endpoints. Dihedral is Blender's signed face angle for two-manifold edges and zero otherwise. Normals are the unweighted mean of adjacent face normals; the radial normal is the dot product with the in-plane radial direction. Rank normalization is $\text{rank}/(N - 1)$ with min-rank ties; edge length is normalized by the maximum length. All ray queries share one world-space BVH.

Precomputed geometric features. *Ring candidates* are connected runs of transition edges (semantic boundary, or curvature or dihedral rank ≥ 0.9) of at least 6 edges. Each run is fitted with its own PCA plane and accepted as a ring when thickness/radius ≤ 0.15 and in-plane angular coverage ≥ 0.9; the most tangent edge is stored as the seed. *Curvature ridges* are connected runs of dihedral rank ≥ 0.92 with at least 6 edges. *Genus necks* come from the per-component Euler characteristic and a localized handle or tunnel cut. *Symmetry planes* test the three axis-aligned planes through the bounding-box center: up to 1500 edge midpoints are reflected and matched to their nearest neighbor in a grid of cell

Table 3: **Dataset statistics.** Entries are minimum / median / maximum over their respective set.

Property	SUBSTANCE-50	TOYS4K-50
Vertices	1,344 / 9,233 / 53,638	62 / 4,751 / 330,654
Object parts	1 / 1 / 6	1 / 1 / 1
Connected components	1 / 2 / 31	1 / 1 / 1
Triangle fraction	0 / 0.003 / 0.095	0 / 0 / 0.050

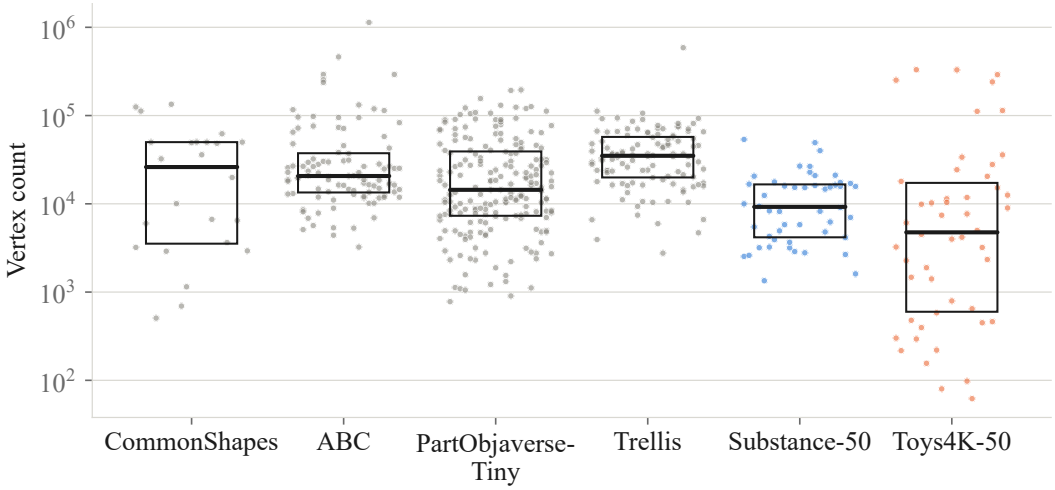

Figure 7: **Dataset mesh size comparison.** We show the overall distribution of mesh sizes across common 3D mesh datasetsJacobson & contributors (2013); Koch et al. (2019); Deitke et al. (2023); Yang et al. (2024b); Xiang et al. (2025). Substance-50 and Toys4k-50 have comparable mesh size distribution with respect to the others, measured by the number of vertices.

size $0.02\,d$; a plane is accepted when the median reflection error is below $0.06\,d$ and at least 40% of points match within two cells, and the plane with the lowest error is stored.

A.4 BASELINES

Blender Smart UV uses angle limit $66°$ and island margin 0.03. XATLAS uses default chart options. OptCuts runs headless with $\lambda = 0.999$ and distortion bound 4.1, seed 0. PARTUV uses all default values at seed 42, distortion threshold 1.25, and the same PartField sampling as our preprocessing. FlattenAnything uses 10,000 points and 10,000 iterations, reseeded per object. Every baseline is given a 40-minute budget; samples that time out are recorded as failed.

A.5 PARAMETERIZATION ALGORITHM

Parameterization algorithm's quality is mesh-dependent: for instance conformal performs best on hard-surface objects, while ABF++ performs better on organic ones. We choose the parameterization algorithm based on the distribution of dihedral-angle edges: the framework selects LSCM (hard-surface specific) if the percentage of such edges is above a given threshold; otherwise, Min. Stretch is used. In addition, the framework may fallback from Min. Stretch to ABF++ if the current seams do not reach the user-specified distortion thresholds.

A.6 METRICS

Chart count. Two faces are in the same chart when they share a full UV edge; the number of connected components in the UV atlas is the chart count.

Seam Length. UV seam length is the total UV length of chart boundary edges. 3D seam length is the 3D length of the cut edges, each physical cut counted once, divided by the total 3D area. The

Table 4: **Agent Tool interface.** "?" marks an optional argument; "→" gives the return type. A `Base` statement is evaluated from SQL selections alone; a `Dependent` statement takes the output of a `Base` statement as an argument, so a program is a two-layer dependency graph with in-degree at most one. Time complexity is given in n, the face count of the mesh; m is the number of statements in the submitted program and r the number of iterations of the flattener.

Tool	Details	Time
1. Perception		
Query		
query(SQL) → `SQL-edge | SQL-face, Stats`	Is the main tool of the agent for collecting mesh attribute observations. Returns valid SQL-edges or SQL-faces fitting the query, as well as their aggregate distribution statistics.	$O(n)$
DSL statements `Base`: evaluated from SQL selections		
topological_loop(SQL-edge) → `Base`	from an input SQL-Edge, it proceeds through a topological walk over the mesh and returns the set of connected edges. the chain that goes straight through every valence-4 vertex via the opposite edge, ending at poles or the boundary	$O(n)$
boundary_loop(SQL-face) → `Base`	returns Mesh boundary loop as the edges with exactly one incident face, chained into cycles	$O(n)$
region_boundary(SQL-face) → `Base`	Returns the boundary loop of the input face set: the edges that have exactly one incident face inside the set, chained into a closed or open loop.	$O(n)$
thin_loop(SQL-edge) → `Base`	**Searches** for a thin, two-sided part into its front and back faces and returns the loop of edges where the two sides meet. The two sides are found by 33 breadth-first splits, each separating the faces at a different fold-sharpness threshold, plus one vote by the sign of each face normal against the sheet's PCA plane. The division with a single loop and the most balanced sides is returned.	$O(n)$
semantic_interface(Label, Label) → `Base`	Returns the set of edges of connected edges whose incident faces carry different semantic labels. The sets are chained into connected groups, sorted by size.	$O(n)$
seamcut_seam(SQL-edge, SQL-edge, SQL-edge …, β?) → `Base`	Returns a crease-following path through 2–8 ordered waypoints, one segment per consecutive pair. Each segment is an anisotropic geodesic (SeamCut): a short-term vector Dijkstra distance field with look-back 3 under a curvature-aligned metric of strength β, then a streamline traced from the target back to the source and snapped onto mesh edges. Unlike `geodesic`, it hugs a curved crease instead of cutting straight across.	$O(n \log n)$
crease_select(Angle, SQL-edge?) → `Base`	Returns an unordered set of edges whose dihedral angle is at least `Angle`, in one scan of the edge table. Without an angle, it keeps the sharpest decile, dihedral rank ≥ 0.9.	$O(n)$
find_loop(SQL-edge?) → `Base[]`	**Searches** an area for every distinct `topological_loop(`θ`)` testing up to 24 θ-edges by farthest-point sampling. Loops are ranked by coverage, then flatness, and duplicates on the same plane are dropped.	$O(n)$
find_extremities(SQL-edge) → `Base[]`	**Searches** each a component for and returns a `region_boundary(`θ`)` end. Protrusions are the local maxima of surface distance from the component center, found by the Dijkstra Algorithm.	$O(n \log n)$
DSL statements `Dependent`: attached to `Base` outputs		
geodesic(Base | SQL-edge, Base | SQL-edge, SQL-edge …) → `Dependent`	Returns the shortest chain of edges from a start edge to an end edge, Via A* search over the edge-adjacency graph, guided by straight-line distance and penalizing turns. A `Ref` points to the realized edges of the statement and snaps to one of those edges. Both anchors must lie on the same component.	$O(n \log n)$
2. Validation		
submit_plan(program) → `PlanVersion`	Interprets a program P. `Base` statements are realized first, `Dependent` afterward, mirror twins last. A refinement stage snaps jagged loops, drops duplicate seams, and merges rings one face apart or thin parallel bands. A cut-topology certificate then checks that every chart is a disc, genus 0 with at least one boundary.	$O(m\, n \log n)$
apply_seams() → `Atlas, Feedback`	First parametrizes the seams S (options SLIM; LSCM, ABF++), then packs UV charts, and evaluates	$O(r\, n^{3/2})$
inspect_cut_topology() → `CutTopology`	Computes the cut topology of a plan: the face regions the seams separate, the connected seam networks, and free seam ends that do not join another seam.	$O(n \log n)$

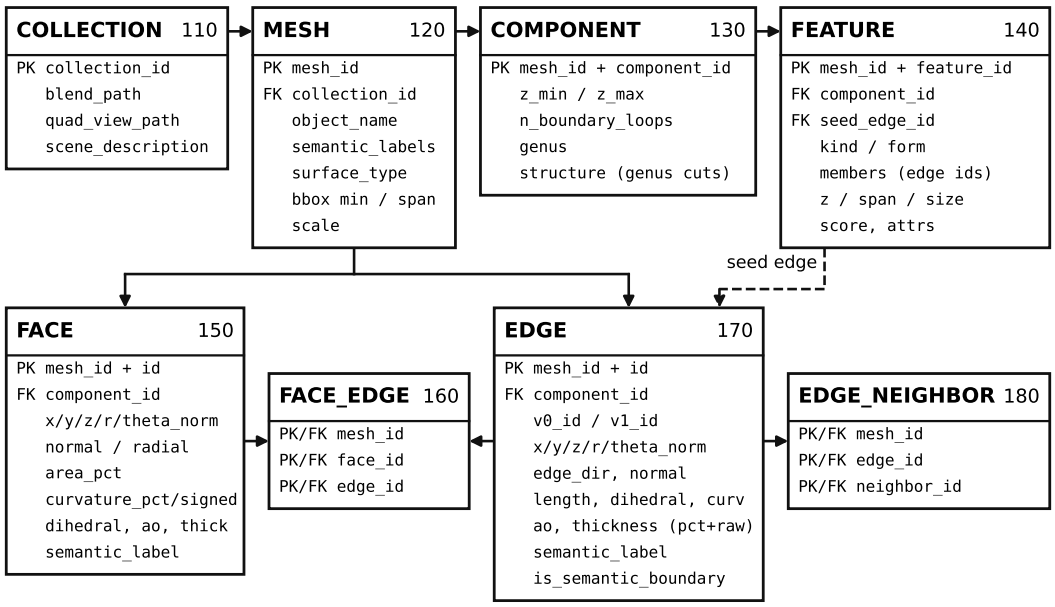

Figure 8: **Relational mesh representation (`mesh.db`).** PK and FK denote primary and foreign keys. Solid arrows indicate one-to-many relations; the dashed arrow links a candidate to its seed edge. Geometric signals are stored per face and per edge.

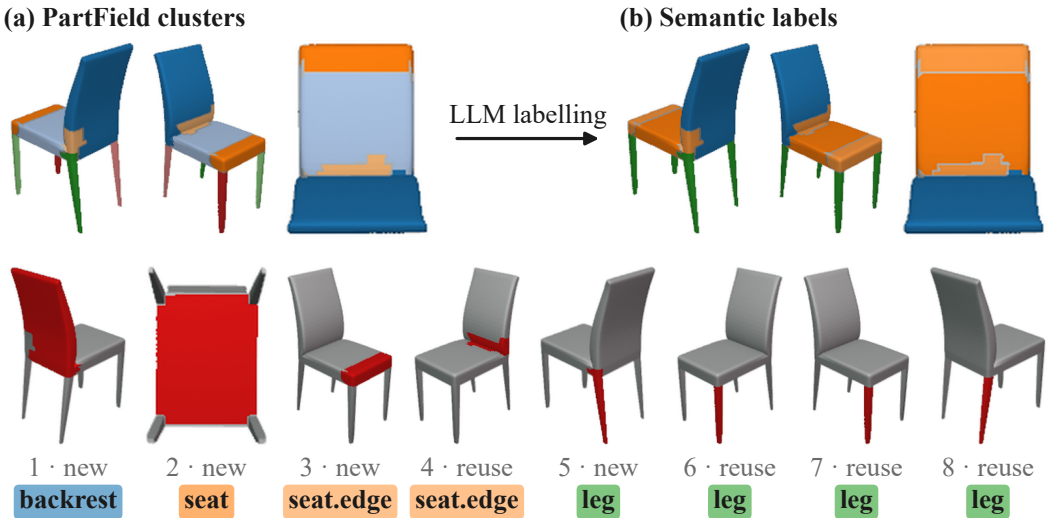

Figure 9: **VLM semantic labeling**. (a) *Input*: PartField splits the mesh into $K = 8$ clusters based on the mesh semantics and geometry. (b) *Output*: the VLM produces a final set of labels for each face. The Bottom row illustrates the labeling process. For each k-th cluster, we render the mesh with the part colored red. Then we prompt the VLM to inspect the image and provide a text label for the selected area. We set the VLM to coin new terms only when strictly necessary, or reuse previously used ones to compose a label vocabulary.

non-disk fraction is the share of charts whose Euler characteristic differs from one (annuli, handles, closed components).

Chart shape. Following Ma et al. (2026), for a chart with UV area a, perimeter p, and convex-hull area h, compactness is $4\pi a/p^2$ and convexity is $\min(1, a/h)$; both are averaged over charts, skipping charts with zero area or perimeter. For jaggedness, each closed UV boundary loop is resampled

at spacing 0.005 (8 to 4096 points) and we average the second difference $\|q_{i-1} - 2q_i + q_{i+1}\|$ of consecutive samples q_i.

Seam visibility. Occlusion at a surface point is measured by casting 8 cosine-weighted hemisphere rays and taking the fraction that hit the mesh again: 0 on an open surface, 1 deep inside a crevice. We evaluate it at $N = 1000$ points placed along the seams, spaced evenly by arc length so that each sample stands for the same amount of seam, giving values $s_1, \ldots, s_N$. The same measurement on N points spread over *all* edges of that mesh gives $e_1, \ldots, e_N$, which describe how occluded that surface is in general.

Reporting the mean of the s_i would mostly report the shape: a convex object occludes nothing, so the mean is near zero wherever the seams run. We therefore score the seams against their own mesh, by comparing every seam value with every reference value and taking the fraction of comparisons in which the seam point is the more occluded one, ties counted as half:

$$\mathrm{AO} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \Big[\mathbf{1}\{s_i > e_j\} + \tfrac{1}{2}\, \mathbf{1}\{s_i = e_j\} \Big].$$

Being a fraction of comparisons, AO is in $[0, 1]$ by construction and needs no further normalization: 1 means every seam point is more hidden than every point of the surface, 0.5 that seams are indistinguishable from the surface at large, and below 0.5 that they run along exposed regions. It is therefore comparable across meshes, and stays defined on a convex shape, where all $2N$ samples are zero, and AO is exactly 0.5.

Area distortion. We follow the same implementation as (Wang et al., 2025). A face f is kept if $\mathrm{area}_{3D}(f)$ is at least 10^{-4} of the mean 3D face area and $\mathrm{area}_{2D}(f) > 0$. Its area ratio is normalized by the atlas-wide ratio, then folded and capped at 10:

$$s_f = \frac{\mathrm{area}_{2D}(f) \sum_g \mathrm{area}_{3D}(g)}{\mathrm{area}_{3D}(f) \sum_g \mathrm{area}_{2D}(g)}, \qquad \mathrm{area_dist}(f) = \min\left(10, \max\left(s_f, \frac{1}{s_f}\right)\right).$$

Thus $d(f) \geq 1$, equal to one for an isometric face, and stretching or shrinking by the same factor scores the same. Overall area distortion is the plain mean of $d(f)$ over kept faces; the cap at 10 follows the original code and keeps a few degenerate faces from dominating the mean. Worst-chart area distortion recomputes the same mean per chart, with the sums and kept faces restricted to that chart, and takes the largest.

Angular distortion. We follow the same implementation as (Wang et al., 2025). Per face, let e_u and e_v be the 3D directions along which u and v increase, and θ the angle between them. Then

$$cos\theta(f) = \frac{e_u \cdot e_v}{|e_u|, |e_v|}, \qquad Q_{\mathrm{ang}} = 1 - \frac{1}{F} \sum_f \big|\cos\theta(f)\big|$$

over faces with a non-degenerate UV triangle. It is one when every face is mapped with perpendicular axes; it tests orthogonality only, not equal stretch along the two axes.

Conformal distortion. Per face, take the 2×2 Jacobian from the tangent plane to UV with singular values $\sigma_1 \geq \sigma_2$. Conformal distortion is the mean of

$$K = \sigma_1 / \sigma_2$$

over

$$\mathcal{V}_K = \{f \in \mathcal{V}_A : \det[u_{f,1} - u_{f,0} \;\; u_{f,2} - u_{f,0}] > 10^{-12}\},$$

the nondegenerate, orientation-preserving faces. Thus flipped and collapsed UV triangles are omitted while every atlas still contributes one aggregate. One is ideal, and lower is better.

Texel spread. With $t(f) = \log_2\big(\mathrm{area}_{2D}(f)/\mathrm{area}_{3D}(f)\big)$ the per-face log texel density, texel spread is the standard deviation of $t(f)$ weighted by $\mathrm{area}_{3D}(f)$, in octaves. It is zero when every face gets the same texture resolution and does not depend on the global UV scale.

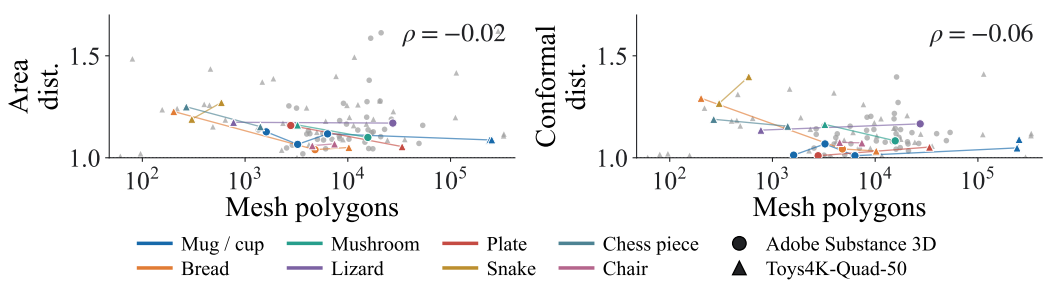

Figure 10: **Distortion performance scaling over mesh size.** MESHQUERYdistortion performance does not degrade over larger meshes.

A.7 MEASURING TOKEN COUNT

The observation tokens in Fig. 5(right) are the total text returned by tool calls during one run, while *OBJ serialization* is the byte size of the mesh's OBJ files. Both are converted to tokens at four characters per token. Overall, for 100 meshes, the median run reads 36k observation tokens (min 10k, max 175k), against 230k tokens for the serialized mesh, and the two scale differently with resolution: the serialized size grows linearly with polygon count (Spearman $\rho = +0.99$, fitted slope 1.06 per decade in log–log), whereas observation tokens are nearly flat ($\rho = +0.15$, slope 0.05 per decade). The largest per-turn context, including cached prompt and all prior tool results, has a median of 157k tokens and a maximum of 503k.

B EXTENDED RESULTS

We extend Fig. 5 of the main paper with pointwise distortion plots in Fig. 10. Our method scales well to larger meshes, and distortion performance is uncorrelated with mesh polygon size.

Tool usage statistics. Fig. 12 counts the tool calls of the selected plans across the evaluation benchmark. We observe over $3,049$ tool calls, of which 30.9% are queries, 40.2% are statement previews, and 28.9% are validations. In addition, we also report 55.8% base DSL statements and 44.2% dependent types.

Qualitative results of main paper Fig. 4 are extended in Fig. 11.

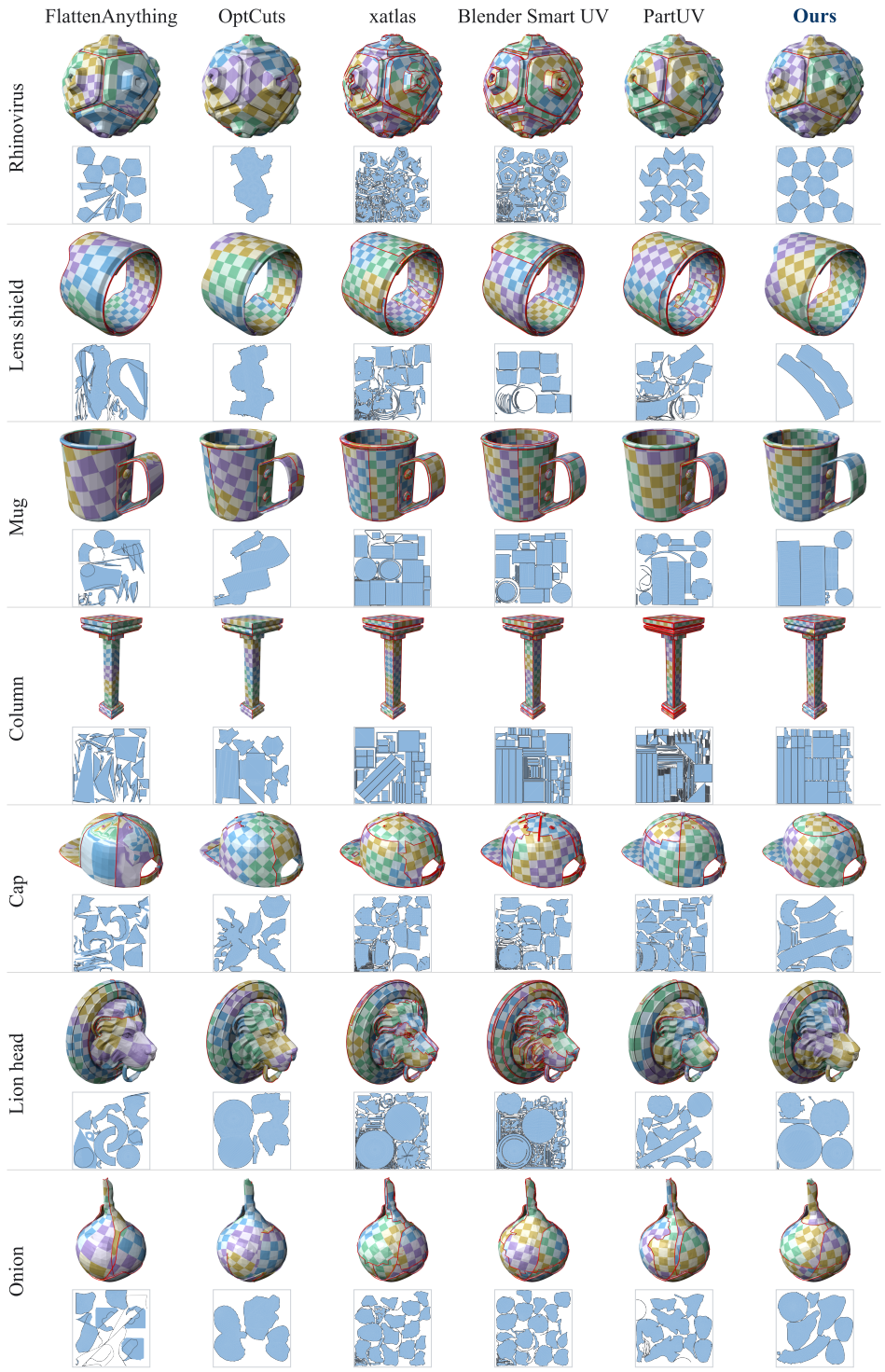

Figure 11: **Qualitative comparison on SUBSTANCE-50.** For each asset, the top row shows a UV checkerboard and generated seams in red, while the bottom row shows the parametrized UV atlas.

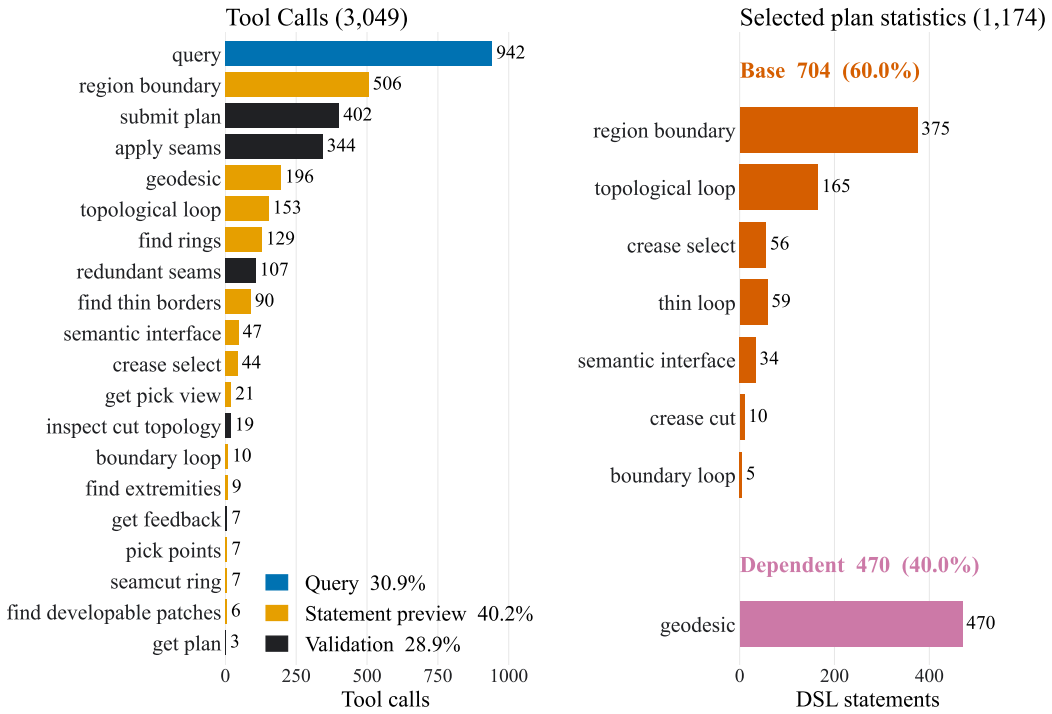

Figure 12: **Tool use across both evaluation sets (100 runs).** *Left:* Distribution of the 3,049 tool calls by tool and call kind. *Right:* Distribution of the 1,174 DSL statements in the 100 selected programs, grouped by the type of edge selection returned.

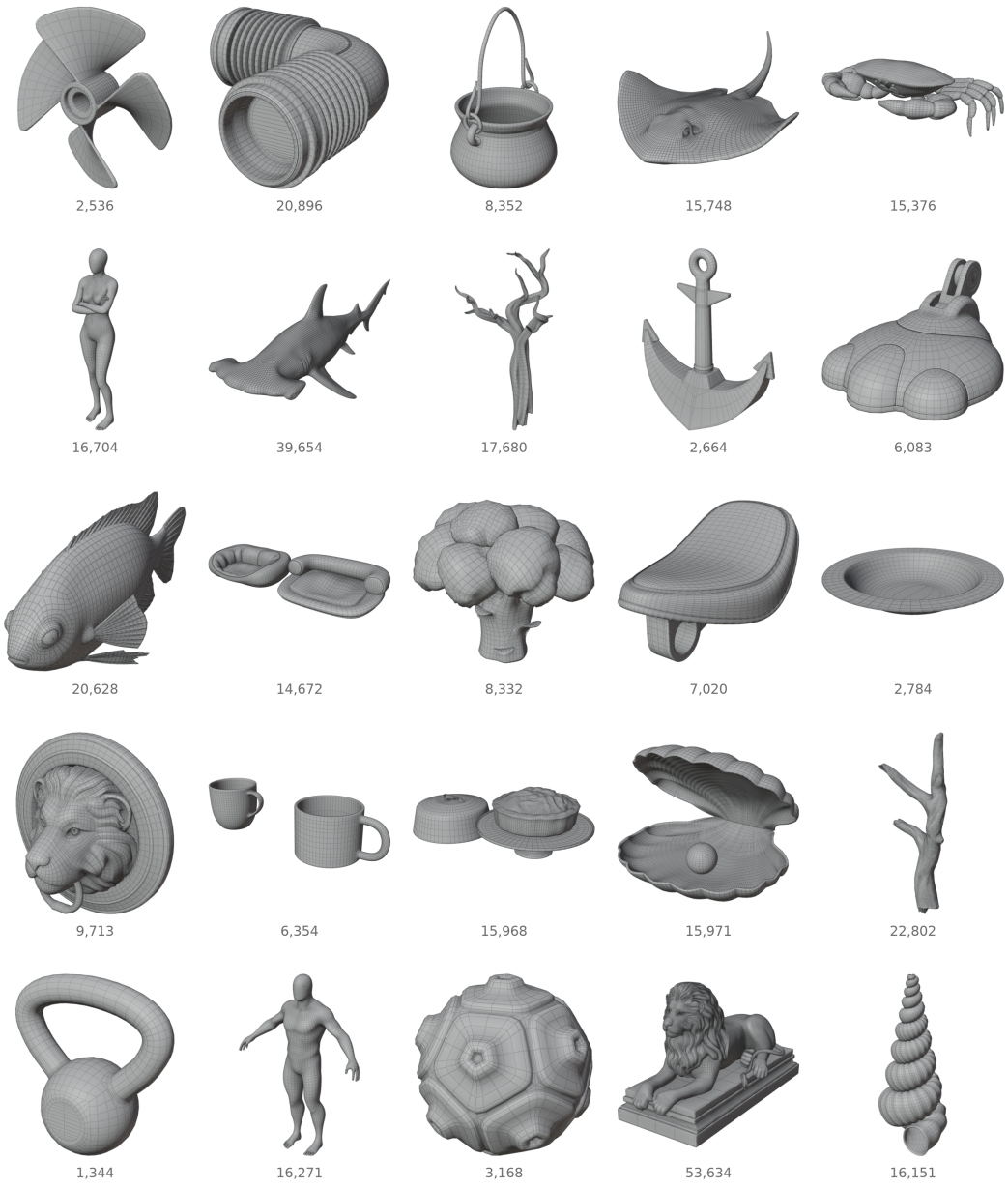

Figure 13: SUBSTANCE-50 evaluation assets.

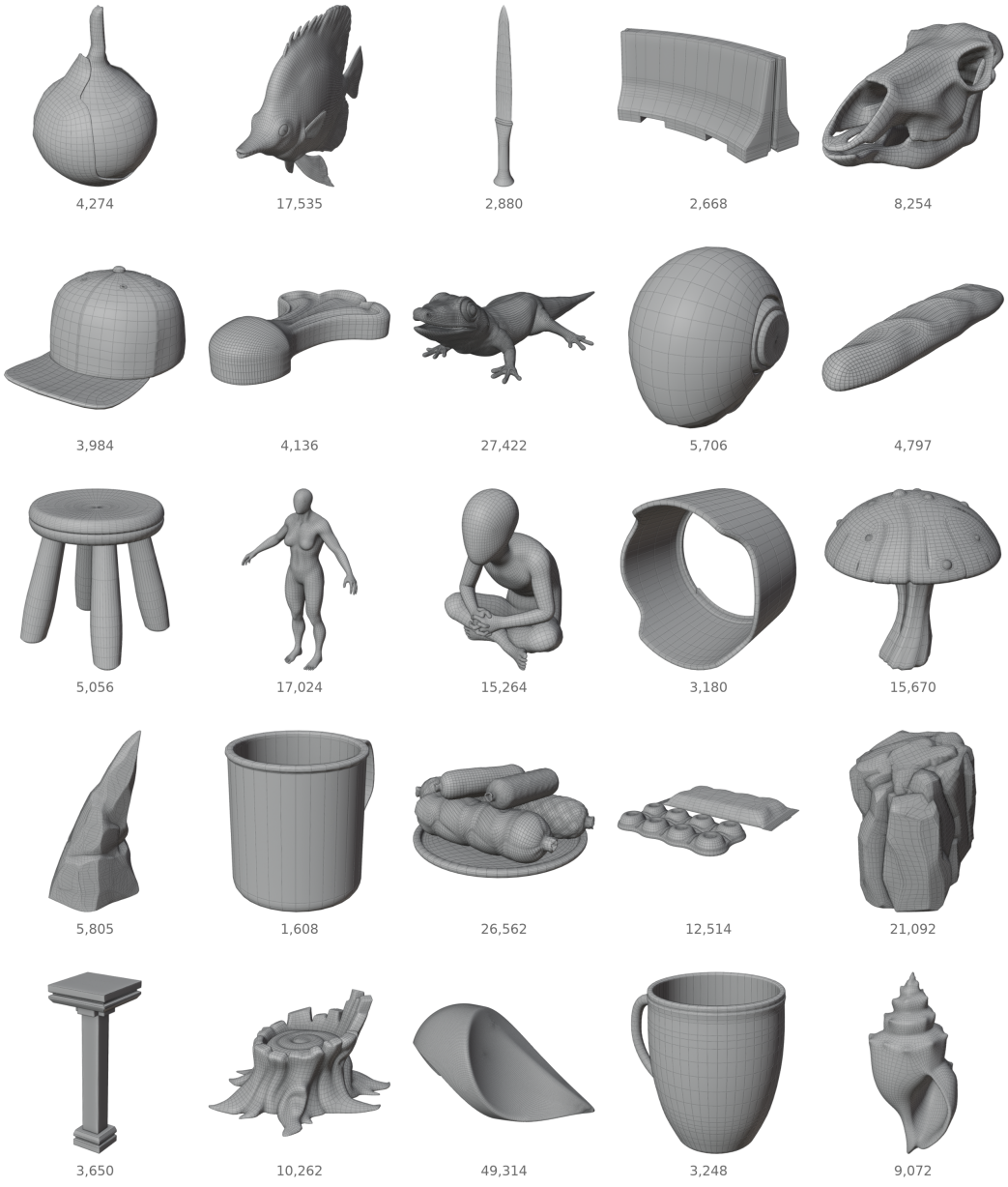

Figure 14: SUBSTANCE-50 evaluation assets (continued).

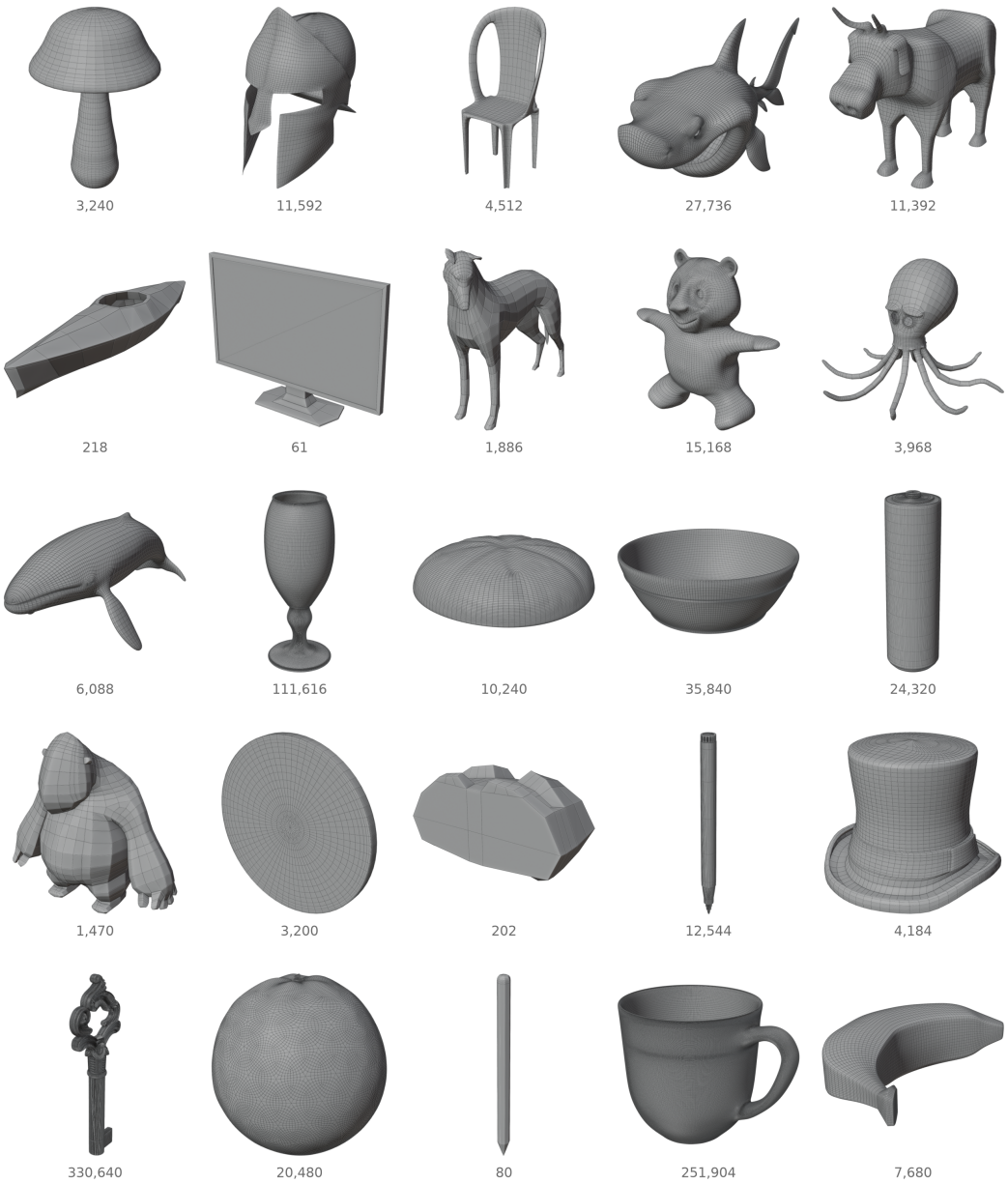

Figure 15: TOYS4K-50 evaluation assets.

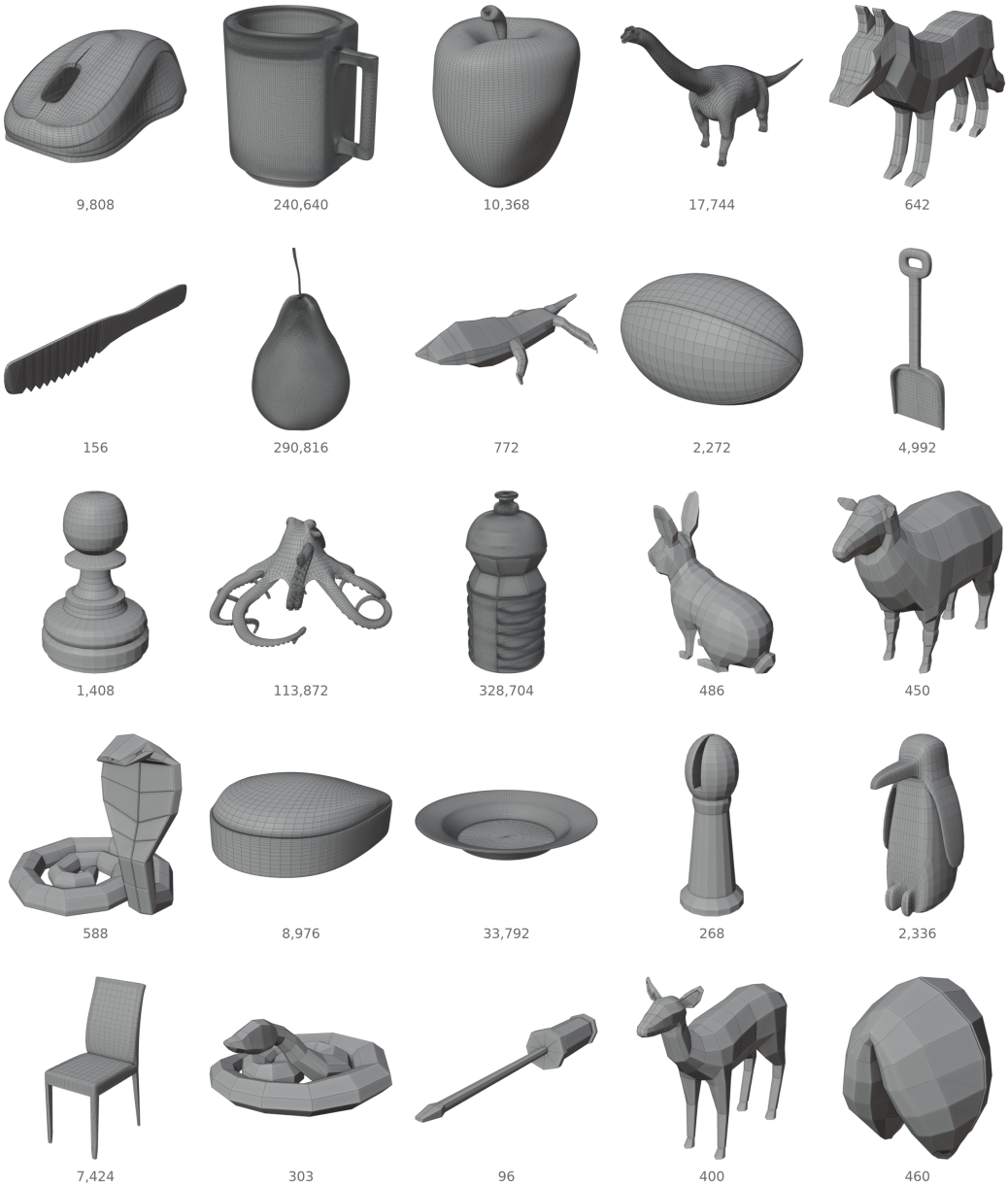

Figure 16: TOYS4K-50 evaluation assets (continued).