

Collidify: Interactive Collider Decomposition for Convex Meshes and Primitives

Bitwise (pseudonym)

Independent researcher and developer of Collidify

1. Abstract

Automatic collider generation must balance decomposition quality, collider count, and runtime across both convex meshes and geometric primitives. Existing decomposition pipelines typically target only one representation, while downstream tools often rely on convex decomposition followed by post-hoc conversion to primitive colliders. We present **Collidify**, a greedy, GPU-batched method that unifies convex and primitive decomposition within a single framework. In paired median comparisons at fixed collider budgets, Collidify approaches CoACD's quality with roughly 100× lower end-to-end runtime on the tested workstation. At each step, Collidify evaluates representation-specific prospective child regions before committing to a cut: convex-mesh mode ranks cuts using an immediate-progress reward, whereas primitive modes rank them by anticipated fit to the requested collider family. Approximate batched scoring and progressive candidate filtering enable broad candidate evaluation without expensive look-ahead tree search.

On an external 461-model benchmark, Collidify approached CoACD's fixed-budget quality. At maximum budgets of 8, 16, and 32 colliders, its paired median relative neutral-concavity gaps against CoACD were +2.14% [0.70%, 4.46%], +1.63% [0.33%, 4.50%], and +0.84% [-1.60%, 3.70%], where brackets give pointwise 95% model-bootstrap confidence intervals. Median speedups were 110× [95.5×, 124.5×], 127× [121.7×, 136.3×], and 111× [106.7×, 117.8×]. At the same budgets, the paired median relative neutral-concavity gaps for VisACD against Collidify were +4.56%, +15.62%, and +29.85%, with Collidify speedups of 21.5×, 31.0×, and 35.5×. The corresponding V-HACD gaps were +15.49%, +39.56%, and +52.38%, with speedups of 8.31×, 7.96×, and 6.53×.

Controlled studies isolated the effects of Collidify's main design choices. Transplanting the immediate-progress reward into a CoACD-derived greedy shell reduced neutral concavity by 17.5%–18.1% at fixed budgets of 16 and 32 hulls. For decompositions requested at 16 parts, GPU-scored decompositions had a paired median relative union-volume change of +0.047% against materialized exact CPU scoring and ran 30.3× faster. Discrete orientation search improved box and capsule quality–complexity frontiers relative to principal-component-analysis-only fitting, while representation-conditioned sphere partitioning reduced final sphere-union volume by 13%–16% relative to the generic partition baseline. Collidify therefore targets competitive median fixed-budget quality at latency suited to iterative workflows. CoACD retains the stronger aggregate quality–part-count frontier when quality takes priority over latency.

2. Introduction

Real-time physics systems use simplified collision proxies to reduce simulation cost. Convex meshes support flexible geometric approximation. Boxes, capsules, and spheres impose stronger shape constraints but can reduce processing cost and simplify manual editing when they fit the asset. Creating collision proxies by hand is time-consuming, especially for large asset libraries and iterative workflows. Automatic generation must therefore balance decomposition quality, collider count, runtime, and the requested output representation.

Top-down decomposition methods repeatedly select an active region, evaluate candidate cuts, and commit to a partition until a quality threshold or part-count limit is reached. Broader candidate sets can expose better geometric separations but increase evaluation cost. For primitive outputs, partitioning and orientation fitting are coupled: a cut suited to convex hulls may yield poor boxes or capsules. Final quality therefore depends on candidate diversity, evaluation accuracy, and the representation used to score prospective parts.

Prior methods span greedy, tree-search, learned, visibility-based, and application-conditioned formulations. They also differ in whether they optimize general convexity, navigable free space, or direct primitive fit. Section 3 positions Collidify within these lines of work.

Collidify addresses both output families within a common top-down framework designed for iterative authoring. It replaces look-ahead tree search with parallel evaluation of immediate, representation-specific decisions. Each candidate plane is scored from prospective children in the requested collider family. A CPU-controlled greedy loop commits the resulting cut. Fixed-budget quality therefore emerges from a sequence of locally selected cuts. The method has no explicit budget-allocation objective.

This work makes three contributions: (1) an immediate-progress convex-mesh objective that ranks cuts by their current geometric benefit without look-ahead tree search; (2) a GPU-batched approximate scorer with progressive candidate filtering that makes broad candidate evaluation practical at low latency on the tested hardware; and (3) representation-conditioned cut selection that evaluates prospective child fit in the requested collider family before committing to a partition. Figure 1 illustrates the external convex-decomposition comparison under a 16-part budget.

Bunny decomposition example at a maximum of 16 parts

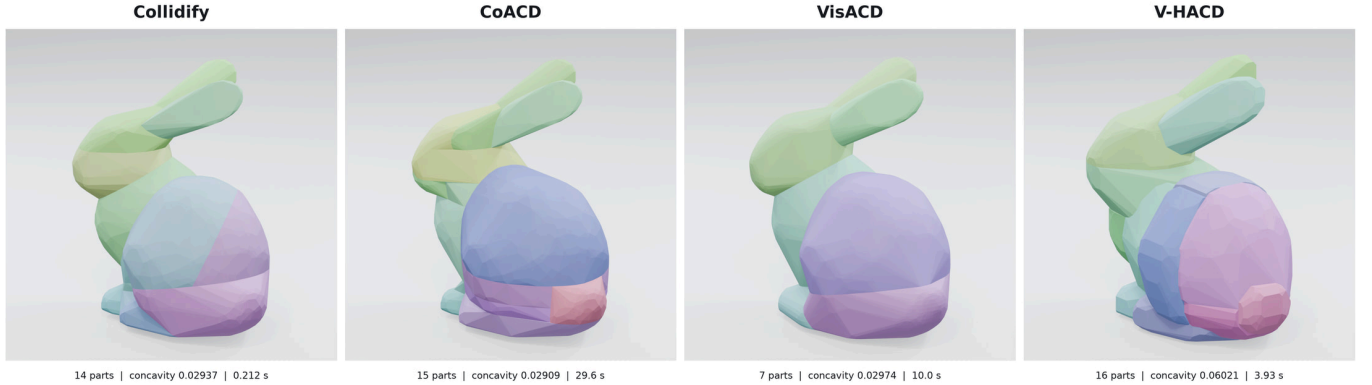


Figure 1. Bunny decomposition at a maximum of 16 convex parts. From left to right, each panel shows the feasible operating point with the lowest neutral concavity under a 16-part budget for Collidify, CoACD, VisACD, and V-HACD. The methods produced 14, 15, 7, and 16 parts, respectively. The corresponding neutral concavities are 0.029368, 0.029091, 0.029735, and 0.060215, and the observed end-to-end runtimes are 0.212 s, 29.59 s, 10.00 s, and 3.93 s.

3. Related work

3.1 Convex-mesh decomposition

Approximate convex decomposition relaxes exact convexity to obtain fewer collision proxies. Lien and Amato established recursive surface partitioning with a geometric concavity threshold [1]. V-HACD instead voxelizes the source volume and recursively chooses axis-aligned splits using a weighted objective that includes volume error, balance, and symmetry [2]. Its public implementation provides a practical classical baseline, but voxelization, reward formulation and axis-aligned search can limit geometric fidelity.

Several greedy methods evaluate the immediate result of one cut. Thul et al. select the active part with the largest absolute convex-hull approximation error and choose the crease-derived plane that minimizes the sum of the two child errors [13]. Because the source volumes of valid children sum to the parent volume, their plane objective is equivalent, up to sign, to maximizing the immediate reduction in total convex-envelope volume. RL-ACD includes the same form of hull-volume reduction as an immediate reward term, then learns a discounted action-value policy and adds a terminal reward for reaching the concavity target with fewer cuts [5]. Collidify does not claim hull-volume reduction itself as new. Its mesh objective adds a source-volume-weighted residual-work term, evaluates the combined quantities in normalized coordinates, expresses the score relative to the active parent hull, and repeatedly applies that score in a greedy leaf-selection policy. Section 4.2 defines this distinction precisely.

CoACD replaces one-step plane selection with Monte Carlo tree search over future cuts and introduces collision-aware concavity based on complementary boundary and interior errors [3]. Its tree search evaluates future cuts through sequential repeated scoring; in the external comparison, CoACD retains the stronger aggregate quality-part-count frontier. Collidify uses the same broad top-down problem structure and reports CoACD-style neutral concavity externally, but replaces future-tree evaluation with batched prospective-child scoring and a greedy decision objective.

VisACD avoids materializing every candidate cut during ranking. It computes visibility edges through GPU ray-mesh queries and scores a plane by the total length of visibility edges that it intersects [4]. This permits thousands of candidates and is rotation-equivariant with its visibility-derived plane family. Collidify also evaluates many candidates without constructing explicit child meshes, but its virtual children approximate the requested output representation and are ranked by representation-specific geometric fit rather than visibility-edge removal.

3.2 Learned and application-conditioned decomposition

RL-ACD formulates ACD as a partially observed Markov decision process and learns long-term plane values from local component observations [5]. Its deployment pipeline combines learned proposals with navigation-derived candidates and a geometric cut check. Collidify has no learned policy or training corpus; its decisions are computed directly from the current geometry, removing dependence on a learned model and its training distribution but giving up learned look-ahead.

Navigation-driven ACD defines free space that an application-specific object or character must traverse, then prevents the decomposition from blocking that space [12]. This changes the target from general surface or volume fidelity to task-dependent accessibility. Collidify is application-oriented in latency and collider representation, but its reported objective remains geometric. It does not model agent size, reachability, contact distributions, or other downstream task costs. Navigation-driven and Collidify decompositions can therefore prefer different approximations of the same object.

3.3 Primitive decomposition

Convex Primitive Decomposition directly targets boxes, capsules, spheres, cylinders, and other convex primitives [6]. It initializes one primitive per face, greedily merges adjacent regions by excess primitive volume, and derives each merged primitive's orientation from the eigenvectors of the accumulated area-weighted outer-product matrix of face normals. This bottom-up topology can combine local surface evidence without selecting cut planes. Collidify instead retains a top-down plane-splitting topology and evaluates fitted primitives for both prospective children before each cut, allowing the same partition framework to emit convex meshes or geometric primitives.

No public reference implementation of Convex Primitive Decomposition was available when the reported experiments were conducted. A new implementation would confound algorithmic differences with unverified implementation choices. The primitive experiments are therefore controlled mechanism studies against internal baselines, not claims that Collidify achieves state-of-the-art primitive-decomposition quality.

The individual ingredients used by Collidify, including greedy selection, convex-hull reduction, primitive fitting, sampling, and GPU execution, have prior precedent. The claimed distinction is their placement within one local decision: Collidify constructs representation-conditioned prospective children, scores a broad candidate set in batches, and commits one greedy cut without tree search.

4. Method

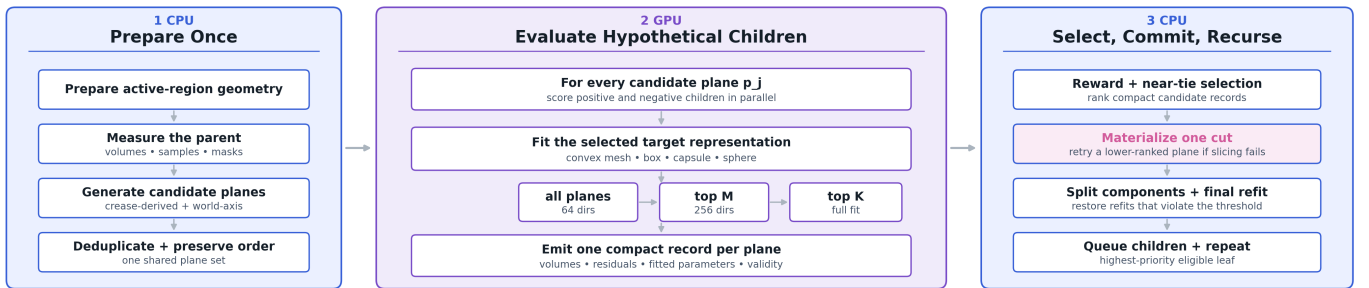


Figure 2. One Collidify decomposition step. The CPU prepares the active region, computes shared geometric data, and generates and deduplicates crease-derived and world-axis-aligned planes. The GPU evaluates both prospective children for every plane in the requested representation, optionally using progressive scoring, and returns one compact record per plane. The CPU applies the reward and near-tie policy, materializes the highest-ranked valid cut, handles retries and disconnected components, performs final primitive refitting or rollback, and queues the next eligible leaf.

4.1 Problem formulation

Let a source region M_i be split by a candidate plane p into prospective children $M_l(p)$ and $M_r(p)$. A region is eligible when its operational residual, the maximum of the volume residual and any enabled surface residuals, exceeds the requested threshold. Decomposition stops when no eligible region remains or the collider-count limit is reached. Because the limit is checked between steps, disconnected components can make the achieved count exceed the request. Eligible leaves are ranked by the volume residual; the operational residual breaks priority ties.

For each candidate, Collidify estimates the representations of both prospective children before committing to the cut: approximate support hulls in convex-mesh mode and fitted boxes, capsules, or spheres in primitive modes. A representation-specific reward ranks the candidates. Only the selected plane is materialized; lower-ranked candidates serve as fallbacks when a higher-ranked plane fails to produce two nonempty child meshes.

4.2 Greedy reward formulation

Convex-mesh reward

All quantities in this subsection are evaluated in centered working coordinates uniformly scaled to a unit bounding-box diagonal. Let H_i denote the exact convex-hull volume of the active parent, and let $\hat{H}_l$ and $\hat{H}_r$ denote the estimated support-hull volumes of its prospective children. Let V_i, V_l, V_r denote the corresponding source-mesh volumes, and let C_i, C_l, C_r denote their residual concavity estimates.

The immediate reduction in convex-envelope volume is

$$\Delta H = H_i - \hat{H}_l - \hat{H}_r.$$

The reduction in source-volume-weighted residual work is

$$\Delta W = V_i C_i - V_l C_l - V_r C_r.$$

The mesh candidate reward is

$$R_{\text{mesh}}(p) = \frac{\Delta H + \Delta W}{H_i} = \frac{H_i - \hat{H}_l - \hat{H}_r + V_i C_i - V_l C_l - V_r C_r}{H_i}.$$

ΔH is the established immediate convex-envelope reduction used, in equivalent form, by prior greedy and learned ACD objectives [5,13]. Collidify's added term is ΔW : it estimates how much source-volume-weighted residual work the cut removes and downweights residual-work improvements confined to tiny, nearly convex fragments. Two candidates can therefore receive similar ΔH values but different rankings when one leaves more residual work, or concentrates that work in a substantial child.

The volume-derived residual for child $\sigma \in \{l, r\}$ is

$$C_{\text{vol},\sigma} = 0.3 \left(\frac{3 |\hat{H}_\sigma - V_\sigma|}{4\pi} \right)^{1/3}.$$

When optional surface scoring is enabled, the operational residual is the maximum of this volume term and the two sampled directional surface discrepancies. Surface scoring was disabled in the reported benchmark.

The envelope-volume and volume-weighted residual terms scale differently without fixed normalization. Collidify first maps the active geometry into working coordinates with a unit bounding-box diagonal, making the radius-equivalent residual a fraction of the same fixed length scale, then divides the combined progress by the active parent's hull volume H_i . Because H_i is constant across candidates for one active leaf, the denominator sets the score scale without changing their order; unit-diagonal coordinate normalization controls the relative scale of ΔH and ΔW . The external benchmark likewise supplies every solver with a unit-diagonal input.

This objective differs from a worst-child objective such as

$$R_{\text{worst}}(p) = -\max(C_l, C_r),$$

which favors balanced child quality but does not directly reward total envelope reduction or weight remaining work by child size. Unlike RL-ACD's learned action value [5] or CoACD's tree-search value [3], R_{mesh} contains no estimate of future cuts. The decomposition policy selects an eligible leaf, ranks that leaf's candidates by the normalized immediate-progress score, commits the highest-ranked valid cut, and repeats. Fixed-budget behavior therefore emerges from the greedy sequence, without an explicit budget-allocation objective or global optimality guarantee. The controlled studies test whether the added residual-work term improves that sequence.

Primitive reward formulations. Primitive modes evaluate occupancy against the original source mesh M_0 , the root input before decomposition. For child $\sigma \in \{l, r\}$, let $\widehat{V}_\sigma$ denote source volume captured by the fitted primitive on its own side of the prospective plane. Let $\widehat{X}_\sigma$ denote captured source volume on the sibling side. If P_σ is the fitted primitive, its effective approximation volume is

$$\widetilde{A}_\sigma = \max\left(\text{Vol}(P_\sigma) - \widehat{X}_\sigma, \widehat{V}_\sigma\right).$$

This construction credits source-supported cross-plane occupancy, so extensions supported by geometry across the candidate plane are not counted as empty collider volume. The primitive volume residual uses the same radius-equivalent conversion,

$$C_{\text{vol},\sigma} = 0.3 \left(\frac{3 |\widetilde{A}_\sigma - \widehat{V}_\sigma|}{4\pi} \right)^{1/3},$$

with the same optional sampled surface terms described above.

For boxes, let V_0 and H_0 be the original source and source-convex-hull volumes. The box reward used for candidate ranking is

$$R_{\text{box}}(p) = \frac{1}{H_0} \left[H_0 - \widetilde{A}_l - \widetilde{A}_r + \frac{\widehat{V}_l^2}{\widetilde{A}_l} + \frac{\widehat{V}_r^2}{\widetilde{A}_r} - \frac{V_0^2}{H_0} \right].$$

Thus, the score favors fitted boxes that occupy little effective volume while capturing substantial source volume. Capsules and spheres instead share the reward

$$R_{\text{capsule}}(p) = R_{\text{sphere}}(p) = \frac{\widehat{V}_l + \widehat{V}_r}{\widehat{V}_l[C_l]_+ + \widehat{V}_r[C_r]_+}, \quad [x]_+ = \max(x, 0).$$

This quantity is the reciprocal of the captured-volume-weighted mean residual. Because total captured occupancy can vary among candidates, maximizing the reward is not generally equivalent to minimizing only the unnormalized residual numerator. For a valid candidate with positive captured volume, a zero denominator indicates zero measured residual and is assigned an infinite score. Capsule and sphere modes share this reward but fit different geometries, so they generally produce different $\widetilde{A}_\sigma$ and C_σ values for the same plane.

4.3 Candidate planes and local selection

Collidify combines geometry-adaptive candidates derived from concave crease edges with world-axis-aligned planes. Crease edges are sampled without replacement with probability proportional to squared crease angle, favoring sharper concavities. Each sampled edge anchors four planes near its midpoint. For adjacent unit face normals n_1 and n_2 , their normals are n_1 , n_2 , the normalized sum $n_1 + n_2$, and the normalized difference $n_1 - n_2$, providing two face-normal orientations and two angular bisectors.

The configured candidate budget counts sampled crease edges. Each sampled edge contributes four orientations, after which Collidify appends an axis-aligned grid configured with up to 40 positions per world axis and deduplicates the combined set. A root edge-sample budget of 500 therefore permits up to 2,120 raw planes. At this setting, the median deduplicated root workload was 1,689 planes in the exact CPU/GPU comparison and 1,699 in the separate cascade study. The realized count varies with the available crease edges, geometric duplicates, and the part-size scaling applied below the root. Evaluating this broader set efficiently motivates the GPU-batched scorer in Section 4.4.

World-axis candidates serve as a stable fallback when the crease-derived family fails to expose a useful separation and can supply cuts aligned with authored world axes. Section 6.5 compares the isolated crease-derived and axis-aligned families and reports the model-specific cases in which axis-aligned-only candidates produce smaller unions.

After evaluation, candidates are ranked by reward. In convex-mesh and capsule modes, candidates satisfying $R(p) \geq R_{\max} - 0.02|R_{\max}|$ are treated as near ties. Within this band, Collidify prefers the candidate with the smallest summed child approximation volume; mesh-mode differences below the estimator's resolution are treated as ties and retain reward order. The CPU then attempts candidates in rank order. If a plane fails to produce two usable child regions, it proceeds to the next candidate until a valid split is found or the retry limit is reached.

4.4 GPU-batched approximate evaluation

Collidify retains decomposition control on the CPU and uses the GPU for batched geometric evaluation. Convex-mesh mode rebuilds preprocessing for each active region. Primitive modes instead reuse preprocessing for the original source mesh M_0 while fitting the geometry of the current active region. The CPU prepares data that can be reused by every candidate: the region vertices and triangles, a surface point cloud, smoothed vertex normals, a triangle bounding-volume hierarchy (BVH), deterministic low-discrepancy volume samples, point-in-mesh masks, and the parent mesh and hull volumes. It then generates and deduplicates the candidate planes. These shared arrays and the plane array are uploaded in batches; candidate generation, tree traversal, and the eventual mesh cut remain on the CPU.

Explicitly slicing the parent mesh for every candidate would make broad plane evaluation too expensive. Collidify instead constructs two *virtual* children per plane. A classification kernel evaluates signed point-to-plane distances and stores the positive and negative assignments in packed bit masks. When a plane crosses a triangle, the kernel computes its edge-plane intersection points and adds them to both child support sets. These support sets include the boundary created by the cut without materializing child topology; the CPU slices the mesh only for the selected plane.

In convex-mesh mode, each plane is evaluated with a finite set of support directions. For each direction and side, parallel reductions retain the point with the largest projection, thereby defining an enclosing child polytope as a set of supporting half-spaces. The base directions form an octant-mirrored Fibonacci set; higher-fidelity evaluation can add directions derived from source-triangle normals and discard redundant supports. The method therefore constructs each child as an implicit support hull, without forming an explicit triangular convex hull.

The source volume on each side is evaluated without building a capped child mesh. Every parent triangle is analytically clipped against the candidate plane, and its signed-volume contribution is accumulated for the positive and negative sides. The excess volume of each support hull is then estimated with a deterministic sample cloud over the parent bounding box: samples that lie inside the implicit child hull but outside the source mesh contribute to hull overshoot. For child $\sigma \in \{l, r\}$, the operational estimate is

$$\hat{H}_\sigma = V_\sigma + |B| \frac{N_{\text{inside hull, outside mesh, } \sigma}}{N},$$

where B is the sampling domain, $|B|$ is its volume, V_σ is the analytically integrated clipped source volume, and N is the total deterministic sample count. The volume gap $\hat{H}_\sigma - V_\sigma$ is converted to the radius-equivalent residual used in Section 4.2. When optional surface scoring is enabled, sampled mesh-to-hull and hull-to-mesh distances are also reduced and the child cost is the maximum of the volume and surface terms.

Primitive modes use the same virtual-child principle but add a second batching dimension: box evaluation dispatches over plane-frame pairs, capsule evaluation over plane-axis pairs, and sphere evaluation only over planes. Each plane-orientation pair in box and capsule modes, and each plane in sphere mode, is evaluated for both prospective children. Projected extrema construct the candidate primitive fits, deterministic occupancy tests estimate captured source volume and residual volume, and GPU reductions retain one fit independently for each child before emitting one compact record per plane. For boxes and capsules, each child orientation is retained by minimizing its absolute effective-volume gap before the combined plane reward is evaluated. The procedure therefore permits different child orientations but does not jointly optimize the orientation pair against the complete plane reward. The readback contains captured volumes, residuals, fitted parameters, and validity information; it omits separate meshes and the full set of orientation trials for each candidate.

After the synchronous GPU readback, the CPU applies the representation-specific reward, stable ordering, near-tie policy, and any refinement or retry logic. Only the selected plane is passed to the production slicer and committed to the decomposition tree. Reusable compute-buffer pools reduce repeated allocation, while the progressive cascade described next reduces how many planes reach the most expensive scoring configuration.

This design is approximate because finite support-direction sets represent convex children and finite deterministic samples estimate their excess volumes. The controlled CPU reference instead materializes both children for every candidate and generates explicit convex hulls. Different scores and top-ranked planes are therefore expected. The relevant test is how well the cheaper approximate ranking preserves final decomposition quality despite its numerical differences from the exact evaluator.

4.5 Progressive scoring

Applying the highest-fidelity scorer to every candidate would erode the benefit of broad candidate evaluation. Collidify first ranks the complete set with an inexpensive pass. Only the best-scoring subset is rescored with a richer geometric representation, and a still smaller finalist set receives the full configuration. Candidate identities are preserved across stages, so the final records refer back to the original planes even though intermediate arrays have been compacted.

A large candidate set first uses 64 support directions and a 1,024-sample deterministic volume prefix. The best $M = \max(128, 4K_s)$ candidates then use 256 support directions and 4,096 samples, where the final plane-survivor count is $K_s = 32$. Those finalists are evaluated once more with additional nonredundant face-normal supports and an 8,192-sample prefix. Smaller candidate sets bypass stages that would not reduce work: a stage is skipped whenever it would retain the entire incoming candidate set, while intermediate-sized sets use the two-stage coarse-to-fine path. Box and capsule modes apply the same principle by first testing reduced frame or axis sets and then restoring the full orientation search for the surviving planes.

The stages reuse deterministic direction subsets and sample prefixes so that increased fidelity extends the coarse workload. Buffer pools are also retained across calls, reducing allocation and transfer overhead. The cascade is an algorithmic approximation: once a plane is removed, later stages cannot recover it. Progressive scoring is therefore a configurable speed–fidelity trade-off. Section 6.5 quantifies its acceleration and the largest model-specific quality losses relative to exhaustive fine scoring.

4.6 Representation-conditioned fitting

A partition that fits one collider family well may fit another poorly. Collidify therefore lets the target family participate directly in plane selection. Every candidate plane creates positive and negative support sets from the points assigned to each side plus the edge-plane intersections introduced by the prospective cut. Collidify fits the requested representation to both sets *before* deciding which plane to materialize. The four modes can therefore produce different partition trees from the same input geometry and candidate-generation family.

In this paper, *representation-conditioned* means that the requested collider family influences partition selection. *Representation-specific* describes a fit or reward defined for one collider family. These terms do not imply a learned conditional model.

For boxes, Collidify evaluates a discrete collection of orthonormal frames generated from sampled directions and roll angles, including the world-aligned identity frame. Support points are projected into each frame and reduced to per-axis minima and maxima, which define the enclosing oriented box. For capsules, a discrete hemisphere of axes is evaluated; each axis separates axial and radial coordinates, the largest radial distance defines the radius, and the remaining axial extent defines the cap-center interval. The GPU evaluates these discrete frames and axes in parallel. The positive and negative children select their orientations independently, so the two prospective colliders need not share an orientation. The controlled baseline uses a single principal-component frame [7]. Section 6.4 quantifies the resulting box and capsule quality improvement.

Because spheres are invariant to orientation, Collidify instead tightens their fit through center selection. For each child, the center is taken from its support-point axis-aligned bounding box, and the radius is set by the farthest support point, ensuring that all support points are enclosed. This construction is simple and deterministic but does not solve for the minimum enclosing sphere.

Primitive occupancy is evaluated against the original source volume. Sampled primitive volume that crosses the candidate plane is excluded from the error where it still covers source geometry. A fitted box, capsule, or sphere may therefore extend across the cut without being penalized for source-supported overlap, which can yield a tighter effective decomposition.

The retained candidate fits influence the selected cut and initialize the child colliders, but they are not immutable final colliders. After a plane is committed, disconnected-component handling can alter the leaves. Final refitting preserves enclosure while reducing each box or capsule’s sampled contribution to union volume and tightening sphere fits. Collidify then recomputes concavity and restores any update that violates the requested concavity threshold. Representation conditioning therefore means that fitted child quality participates in choosing the partition topology; the temporary fit scored during plane evaluation need not match the final emitted collider.

5. Experimental design

5.1 External benchmark comparison

CoACD [3], Collidify mesh mode, VisACD [4], and V-HACD [2] were evaluated through the same external orchestration and scoring layer. The common cohort comprised 461 models: 200 from Objaverse [8], 200 from ShapeNet [9], and all 61 models in the V-HACD dataset used in prior ACD evaluation [3], denoted VHACD61.

Stratified cohort construction. Objaverse and ShapeNet were sampled independently using the stratified procedure below. Eligible meshes were nonempty, had a nondegenerate bounding box, and could be evaluated by the generalized-winding-number solid model [11]. Watertightness was recorded but was not a selection requirement. Whole-shape difficulty was profiled using

$$C_{\text{profile}}(S) = 1 - \frac{\widehat{\text{Vol}}(S)}{\text{Vol}(\text{CH}(S))},$$

where source volume was estimated from 20,000 deterministic generalized-winding-number samples and convex-hull volume was computed explicitly. Profiles were divided into four concavity bands, $[0, 0.05)$, $[0.05, 0.15)$, $[0.15, 0.35)$, and $[0.35, 1.01]$, crossed with three genus bands, 0, 1–2, and 3+. Meshes with undefined non-manifold genus were grouped with the zero-genus stratum. The cohort was balanced across the nonempty strata, with closed and consistently wound meshes preferred within each stratum. Triangle count was excluded because it primarily measures runtime cost and is a poor proxy for decomposition difficulty. This construction reduces dataset-specific difficulty bias and balances geometric concavity and topology.

Common input repair and canonicalization. Each selected asset was converted once to a common input mesh before any solver ran. A method-independent PaMO-based preprocessing stage [10] repaired invalid or oversized inputs; well-formed manifold meshes were normally unchanged. Individual solvers could still apply their native preprocessing, and its cost remained in the measured end-to-end runtime.

The common mesh was centered at the center of its axis-aligned bounding box and uniformly scaled to a unit bounding-box diagonal, with no rotational normalization. Every method received this mesh as input, and the external scorer used it as the reference. For repaired inputs, fidelity is therefore defined against the common processed solid; fidelity to every triangle in a defective source mesh falls outside the comparison.

Native parameter sweeps. The benchmark varied one native granularity parameter per solver. This avoided imposing an artificial common threshold across unlike algorithms. The settings were:

Method	Swept parameter	Fixed algorithm settings
CoACD [3]	Concavity threshold {0.2, 0.1, 0.075, 0.05, 0.03}	Automatic preprocessing at resolution 50; sampling resolution 2,000; MCTS 150 iterations, maximum depth 3, and 20 nodes; merging disabled; at most 1,024 vertices per hull.
VisACD [4]	Concavity {0.2, 0.1, 0.05, 0.03, 0.02}	Concavity score mode; merging disabled; nonbinding 1,000-part safety cap.
Collidify	Concavity threshold {0.2, 0.1, 0.075, 0.05, 0.03, 0.02}	Convex-mesh output mode; merging disabled; nonbinding 1,000-part safety cap.
V-HACD v4 [2]	Maximum convex-hull count {2, 4, 8, 16, 32, 64, 128}	Voxel resolution 100,000; volume-error percentage 0.001; recursion depth 14; at most 256 vertices per hull; optimal splitting and flood fill enabled; shrink wrapping disabled.

The benchmark followed a single-parameter sweep: only the parameter used to trace each method’s Pareto frontier was varied. Other internal controls remained at settings supplied or recommended by the respective implementations, except for the explicit comparability and feasibility changes shown in the table. No setting was tuned separately by dataset or model, and the

external scorer was not used to select favorable method-specific configurations after results were observed.

Optional post-decomposition merging was disabled wherever the implementation exposed it, so no method could move along the part-count frontier through an unreported secondary merge policy. VisACD and Collidify used part caps of 1,000 as non-binding safety limits, CoACD retained its documented automatic preprocessing, and V-HACD shrink wrapping was disabled because projecting hulls inside the reference surface could violate the common 0.995 coverage requirement. V-HACD's 256-vertex hull limit and CoACD's 1,024-vertex limit avoid imposing the legacy 64-vertex bottleneck on methods otherwise able to emit higher-resolution convex pieces.

Each method returned its convex pieces and timing metadata, and the comparison used the observed quality–part-count points from the native parameter sweeps.

Execution hardware. All reported runs used a Windows 11 workstation with an Intel Core Ultra 9 285 processor, 32 GB of system memory, and an NVIDIA GeForce RTX 5080. The controlled studies used Unity 2022.3 through Direct3D 11.

Neutral external scoring. All submitted outputs were rescored by the same external implementation. To standardize the collider definition across face triangulations and any nonconvex adapter output, the scorer constructs a convex hull from every emitted piece before evaluation. For each emitted collider hull G_i , it evaluates the portion of the reference solid covered by that collider, $P_i = S \cap G_i$, and computes the collision-aware concavity introduced by CoACD [3]

$$C(P_i) = \max(H_b(P_i, \text{CH}(P_i)), 0.3R_v(P_i)),$$

$$R_v(P_i) = \left[\frac{3(\text{Vol}(\text{CH}(P_i)) - \text{Vol}(P_i))}{4\pi} \right]^{1/3}.$$

H_b is a sampled symmetric surface-to-surface Hausdorff term and R_v is the equivalent radius of the missing volume. The decomposition score is the maximum piecewise score,

$$C(S, \{G_i\}) = \max_i C(P_i),$$

so one poorly approximated region cannot be hidden by averaging it with many easy pieces. The executed scorer used 8,000 surface samples for the concavity term, 50,000 Monte Carlo points for containment and volume quantities, and 20,000 samples for the secondary reconstruction metrics.

When S is watertight, the scorer constructs the piece $S \cap G_i$ using an explicit manifold Boolean intersection, including the clipping faces, and evaluates its convex hull. For open, non-manifold, or self-intersecting references, the same scorer switches to a generalized-winding-number fallback [11]. A shared point cloud is classified inside S once, convex-collider membership is tested with exact half-spaces, and sampled piece volume and boundary points are used to estimate the same concavity terms. A nonempty unresolved piece is never silently assigned zero error. This fallback is identical for every method and is distinct from any concavity approximation used internally by Collidify, CoACD, VisACD, or V-HACD.

The external scorer also computes coverage, intersection over union (IoU), relative union-volume error, redundant overlap, Hausdorff distance, and Chamfer distance. Coverage is the fraction of sampled source volume contained in at least one submitted hull. An operating point is feasible only if coverage is finite and at least 0.995. The benchmark therefore does not reward a method for obtaining a low concavity score by omitting difficult source regions.

Pareto extraction and pairing. For each method and model, the feasible native-sweep outputs form an observed set of triples consisting of part count, neutral concavity, and end-to-end time. At a part budget $B \in \{8, 16, 32\}$, an eligible method–model output must be completed and successfully scored, have coverage of at least 0.995, finite nonnegative concavity, positive finite runtime, and a part count in $[1, B]$. Selection minimizes neutral concavity, then part count, runtime, and finally the swept parameter. No interpolation or extrapolation is performed between solver settings. A paired endpoint retains a model only when both methods have a selected output under the same budget.

For CoACD, the within-model relative neutral-concavity gap is $100(C_{\text{Collidify}}/C_{\text{CoACD}} - 1)\%$. For $M \in \{\text{VisACD}, \text{V-HACD}\}$, the reported relative neutral-concavity gap is $100(C_M/C_{\text{Collidify}} - 1)\%$. Speedup is $T_M/T_{\text{Collidify}}$. Each table entry is the median of these within-model quantities, preventing models with more operating points from receiving more weight.

Pointwise 95% confidence intervals resample the eligible paired models with replacement and recompute the median within-model effect. Each method–budget–cohort endpoint uses 20,000 bootstrap replicates. These intervals describe the stability of the observed paired cohort under model resampling; they are not population-generalization bounds.

Runtime is the end-to-end elapsed time, including method-specific preprocessing, decomposition, synchronization, and output conversion. Method-reported internal metrics do not enter the comparison.

5.2 Controlled mechanism studies

The controlled studies used 40 models: 35 drawn uniformly at random from VHACD61 after excluding meshes with fewer than 150 vertices, and five ShapeNet models used during development. The ShapeNet subset is not held out. Model selection was independent of the controlled-comparison outcomes. The Unity studies used three repeated runs per model, and the reward-transplant study used one. The matched-decomposition endpoint retained all 40 models; the root-plane CPU/GPU analysis retained 39 because one model failed structural validation. Statistical analyses treated the model as the independent unit; candidate planes, leaves, repeated runs, and timing repetitions were nested observations.

Continuous outcomes are reduced to one value per model using the rule specified for each endpoint. Unless otherwise stated, this reduction is the median across three runs; binary proportions are averaged. The internal reward ablation instead averages the three within-run log volume ratios and exponentiates the result. Confidence intervals are percentile intervals from model-level bootstraps. The analyses use 100,000 replicates except for the partition-conditioning study, which uses 20,000. Exact two-sided sign tests use model-level effects, omit exact ties, and are Holm-adjusted within the hypothesis family stated with each result. For the matched CPU/GPU decomposition endpoint, the comparison criterion requires the one-sided 95% upper confidence bound for the sample-median $\Delta_U(\text{GPU}, \text{CPU})$ to remain below +2%.

The controlled Unity experiments use high-resolution collider-union estimates. For collider union U and source solid S , source-normalized union excess is

$$E(U, S) = 100 \frac{\text{Vol}(U) - \text{Vol}(S)}{\text{Vol}(S)}.$$

For treatment A and reference R , relative union-volume change is

$$\Delta_U(A, R) = 100 \left(\frac{\text{Vol}(U_A)}{\text{Vol}(U_R)} - 1 \right).$$

These compactness metrics are distinct from the benchmark's neutral concavity. They quantify collider-union compactness relative to the source or another treatment. Surface contact fidelity, simulation stability, and other collision errors fall outside their scope.

The controlled studies isolate the following comparisons:

- Collidify's mesh reward and a CoACD-derived worst-child reward within a common restricted greedy shell
- Collidify's native reward and a worst-child baseline
- exact materialized CPU and approximate GPU mesh-plane scoring
- discrete orientation search and PCA-only fitting for boxes and capsules
- generic and representation-conditioned primitive partitioning
- exhaustive, two-stage, and progressive GPU scoring
- crease-derived, axis-aligned, and combined candidate families

For the cascade study, all arms used a fixed requested maximum of $K = 16$; no adaptive rematching was applied.

For the orientation-search comparison, both partition policies were rerun with a common maximum until they reached the same collider count. Each retained partition was then cross-fit with both terminal-fitting policies on common sample sets. Orientation cascades, capsule-specific plane filtering, and union-aware terminal refitting were disabled in both conditions, isolating orientation policy. The discrete-search box condition retained production regular-orientation tie-breaking, which was not tested separately. For each model, quality–complexity AUC was the trapezoidal mean of $E(U, S)$ over $\log_2 K$ for $K \in \{4, 8, 16, 32\}$. The contrast $\text{AUC}_{\text{search}} - \text{AUC}_{\text{PCA}}$ is negative when discrete search is better.

The primitive studies have no external algorithm baseline. No public reference implementation of Convex Primitive Decomposition [6] was available when the experiments were conducted, and the reported results do not compare against a re-implementation. The PCA-only, generic-partition, and cross-fitting controls isolate mechanisms within Collidify; they do not establish state-of-the-art primitive quality.

6. Results

6.1 External quality–runtime frontier

Table 1 summarizes the fixed-budget comparisons under the eligibility, selection, and pairing rules in Section 5.1. Because the two quality columns use different reference ratios, their signs have different meanings. A positive CoACD gap means that Collidify had higher neutral concavity and favors CoACD; a positive VisACD gap means that VisACD had higher neutral concavity and favors Collidify. Brackets contain pointwise 95% model-bootstrap confidence intervals.

Table 1. Paired fixed-budget comparisons with CoACD and VisACD.

Part budget B	Eligible CoACD pairs	Paired median relative neutral-concavity gap, Collidify vs. CoACD	Paired median speedup, Collidify over CoACD	Eligible VisACD pairs	Paired median relative neutral-concavity gap, VisACD vs. Collidify	Paired median speedup, Collidify over VisACD
8	356	+2.14% [0.70%, 4.46%]	109.9× [95.5×, 124.5×]	351	+4.56% [1.25%, 11.05%]	21.5× [17.1×, 26.0×]
16	413	+1.63% [0.33%, 4.50%]	127.4× [121.7×, 136.3×]	413	+15.62% [10.08%, 21.51%]	31.0× [27.6×, 35.2×]
32	443	+0.84% [−1.60%, 3.70%]	111.0× [106.7×, 117.8×]	443	+29.85% [22.22%, 36.28%]	35.5× [31.4×, 40.8×]

The paired median relative neutral-concavity gaps against CoACD were +0.84% to +2.14% across the three budgets, while Collidify was 110×–127× faster. The quality intervals exclude zero at budgets 8 and 16 but cross zero at budget 32. These medians also mask substantial model-level variation: Collidify had lower concavity on 131/356, 179/413, and 212/443 eligible models, and CoACD had lower concavity on the remaining pairs. The mean frontier curves, which are more affected by Collidify’s high-error tail, show a larger CoACD quality advantage.

Collidify had a clearer advantage over VisACD. At budgets of 8, 16, and 32 parts, Collidify had lower neutral concavity on 208/351, 287/413, and 331/443 eligible models and was 21.5×, 31.0×, and 35.5× faster, respectively.

Table 2 applies the same observed-operating-point and within-model pairing rule as Table 1 to the combined 461-model cohort. A positive relative gap means that V-HACD had higher neutral concavity than Collidify, and a speedup greater than one means that Collidify was faster. The final column reports lower-concavity wins as Collidify/V-HACD/ties.

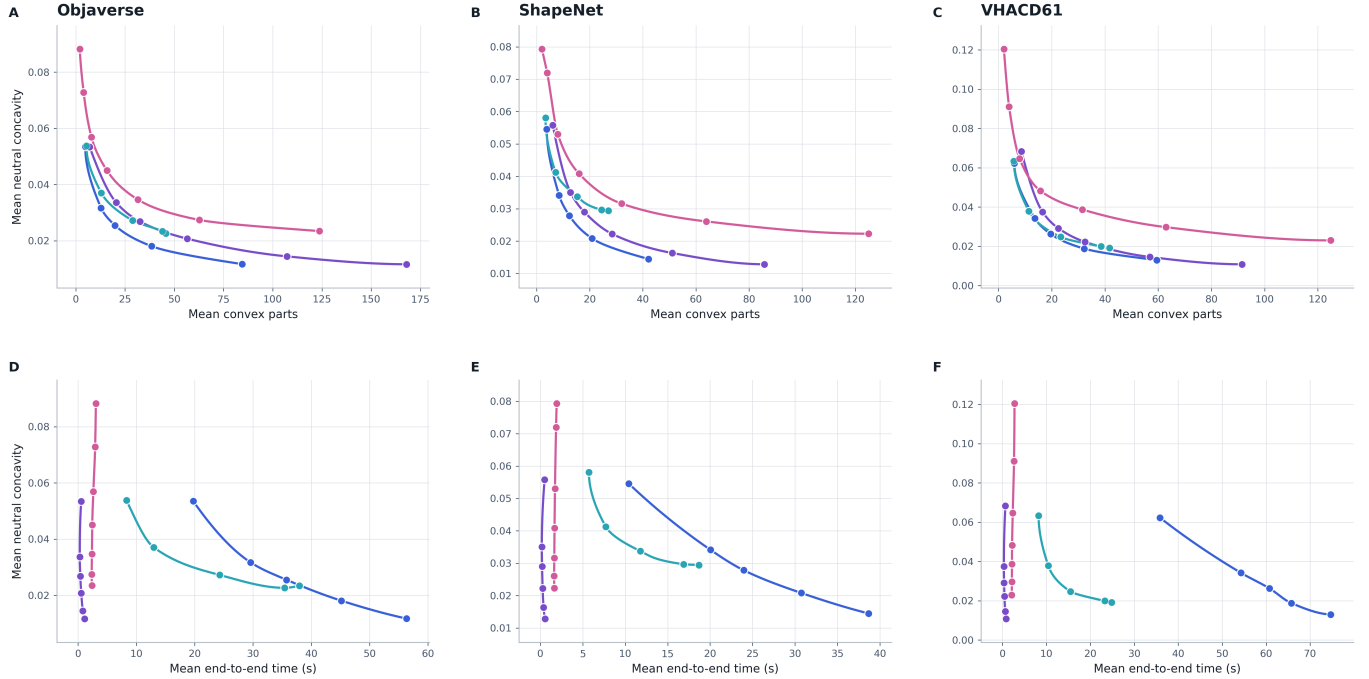
Table 2. Paired fixed-budget comparison with V-HACD by cohort. Brackets contain pointwise 95% model-bootstrap confidence intervals.

Cohort	Part budget B	Eligible pairs	Paired median relative neutral-concavity gap, V-HACD vs. Collidify	Paired median speedup, Collidify over V-HACD	Lower-concavity wins, Collidify/V-HACD/ties
Objaverse	8	155	+15.70% [6.93%, 29.63%]	8.42× [6.90×, 11.19×]	101/54/0
Objaverse	16	179	+37.02% [21.85%, 49.68%]	8.14× [6.79×, 10.15×]	133/46/0
Objaverse	32	190	+34.68% [24.72%, 53.30%]	6.87× [5.41×, 8.30×]	146/44/0
ShapeNet	8	168	+11.84% [2.77%, 22.62%]	8.42× [5.68×, 11.03×]	104/64/0
ShapeNet	16	183	+39.56% [27.89%, 53.20%]	8.29× [7.22×, 10.37×]	144/39/0
ShapeNet	32	193	+52.35% [40.51%, 63.83%]	6.53× [5.26×, 7.48×]	160/33/0
VHACD61	8	43	+16.54% [2.41%, 55.03%]	5.78× [2.98×, 9.84×]	30/13/0
VHACD61	16	55	+60.25% [39.18%, 80.33%]	7.51× [5.42×, 11.23×]	48/7/0
VHACD61	32	60	+84.97% [64.49%, 116.86%]	5.99× [4.19×, 7.70×]	54/6/0
Combined 461-model cohort	8	366	+15.49% [9.02%, 22.86%]	8.31× [7.11×, 9.66×]	235/131/0
Combined 461-model cohort	16	417	+39.56% [31.80%, 49.68%]	7.96× [7.21×, 9.21×]	325/92/0
Combined 461-model cohort	32	443	+52.38% [41.68%, 61.01%]	6.53× [5.61×, 7.28×]	360/83/0

External-benchmark quality, part-count, and latency frontiers

Observed native sweeps on 461 common inputs; lower-left is better and coverage must be at least 0.995.

— Collidify — CoACD — VisACD — V-HACD



Means are conditional on feasible operations at each native setting; exact fixed-budget comparisons are reported separately in Table 1.

Figure 3. External benchmark quality, part-count, and latency frontiers. Panels A–C plot mean neutral concavity against mean convex-part count for Objaverse, ShapeNet, and VHACD61. Panels D–F plot the same mean quality trajectories against observed mean end-to-end time on linear axes. Each trajectory includes feasible outputs from the solver's native sweep settings that meet the common 0.995 coverage requirement. The runtime axes retain each dataset's natural range. Lower-left is better; Collidify clusters near the left edge of the runtime panels. These aggregate trajectories complement the within-model, fixed-budget comparisons in Tables 1 and 2.

Supplementary Figures S2a–S2c provide qualitative decompositions from all four methods across VHACD61, ShapeNet, and Objaverse. These examples illustrate cases in which Collidify is competitive; quantitative conclusions rely on Tables 1 and 2.

6.2 Reward formulation

Two complementary experiments test whether the mesh reward contributes to the observed performance.

First, Collidify’s mesh reward was transplanted into a deliberately restricted CoACD-derived greedy shell. Both treatments used the same repaired inputs, axis-aligned candidates, explicit child hulls, neutral evaluator, eligible-leaf priority schedule, and exact fixed hull counts; MCTS and path refinement were disabled. Relative to the CoACD-derived worst-child reward, the paired median relative neutral-concavity changes were +3.57% at 4 hulls and −1.55% at 8 hulls. The low-budget endpoints showed no statistically detectable model-level improvement. At 16 and 32 hulls, the corresponding changes were −17.49% and −18.10%, with sign-test p -values of 0.00118 and 0.00320 after Holm adjustment across the four fixed-budget endpoints. Across $K \in \{4, 8, 16, 32\}$, the median per-model geometric mean of the transplant-to-baseline concavity ratio was 10.92% lower.

Second, the internal Collidify ablation replaced the Collidify mesh reward with the CoACD-derived worst-child score while retaining Collidify’s candidate generation, approximate prospective-child residuals, cascade, refinement, and postprocessing. The replacement increased final high-resolution collider-union volume by 1.92% at requested $K = 8$ and 2.13% at requested $K = 16$, favoring the Collidify reward in 31/40 and 34/40 models. Both contrasts remained significant after Holm adjustment across the eight prespecified collider-family-by-budget contrasts: four collider families at two budgets.

Because the studies use different systems and endpoints, their effect sizes are not directly comparable. Both indicate that the immediate-progress reward improves fixed-count outcomes after a moderate number of cuts despite the absence of explicit budget allocation. The low-budget endpoints show no consistent benefit.

Reward formulation examples at a fixed 16-part budget

Native CoACD worst-child reward (top) and Collidify immediate-progress reward (bottom); identical input, candidates, part count, camera, scale, and lighting within each model.

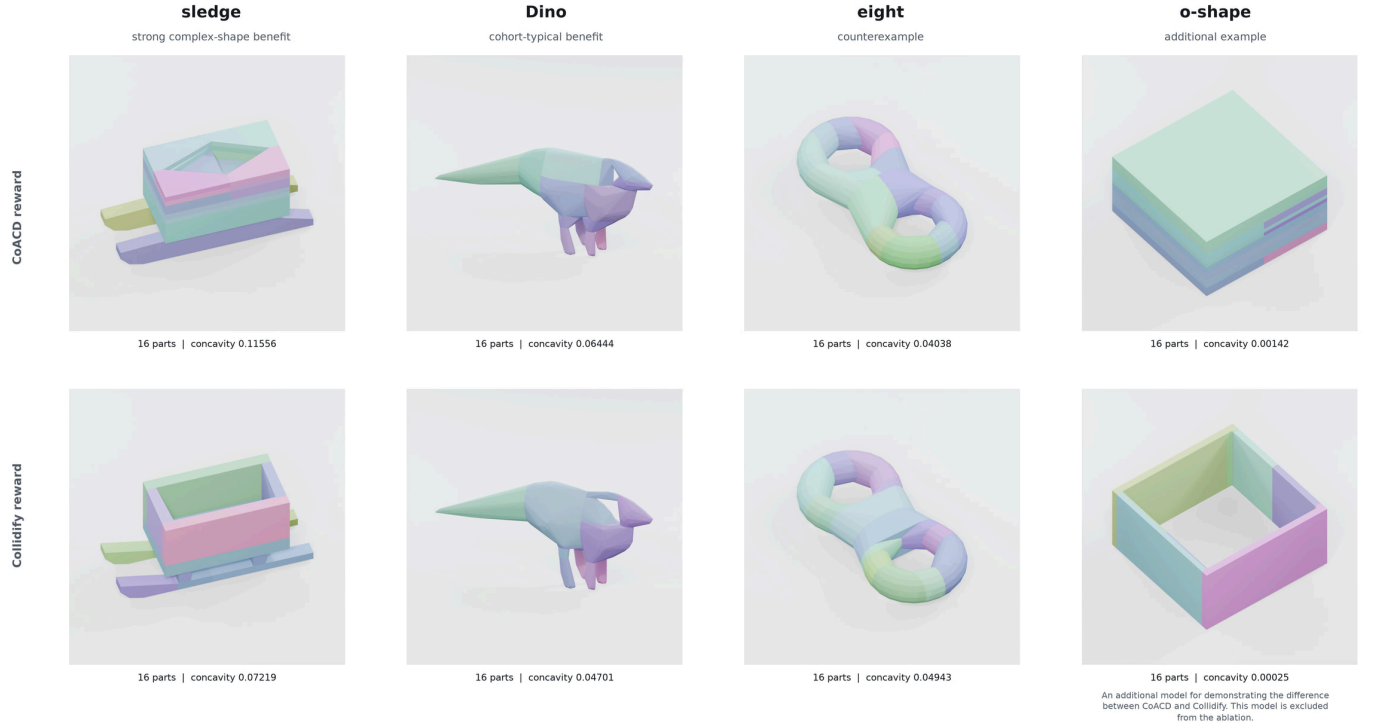


Figure 4. How the immediate-progress reward allocates a fixed hull budget. Each column compares the CoACD-derived worst-child reward (top) with Collidify’s immediate-progress reward transplanted into the same restricted greedy shell (bottom) at exactly 16 hulls. Lower neutral concavity is better. Sledge shows a strong benefit on a complex mechanical model, Dino is closest to the sample-median geometric-mean effect across the four budgets, and *eight* is the strongest counterexample under that summary. The *o-shape* is an additional qualitative example and is not included in the reported ablation statistics. Across the study, the paired median relative neutral-concavity changes were −17.49% at 16 hulls and −18.10% at 32 hulls; the counterexample demonstrates that the improvement is not universal.

Supplementary Figures S1a–S1j provide comparisons across 10 models and all four fixed hull budgets.

6.3 Approximate GPU scoring

At each workload, the materialized exact CPU reference and the full-fidelity approximate GPU scorer evaluated the same requested root candidate set for each of the 39 models that passed structural validation. Across increasing crease-edge sample budgets, the GPU scorer was 9.69 $\times$, 12.08 $\times$, 16.54 $\times$, and 18.49 $\times$ faster. At the largest workload, the scorers selected the same top-ranked plane in 54.7% of cases, but 89.7% of GPU selections ranked among the exact CPU scorer's top five. For CPU-optimal plane p_{CPU} and GPU-selected plane p_{GPU} , normalized CPU-reference regret is

$$100 \frac{r_{\text{CPU}}(p_{\text{CPU}}) - r_{\text{CPU}}(p_{\text{GPU}})}{\max_p r_{\text{CPU}}(p) - \min_p r_{\text{CPU}}(p)}.$$

When the observed CPU reward range is zero, normalized regret is defined as zero. Median normalized regret was zero, the 75th percentile was 0.674%, and the maximum observed regret was 8.63%; the two evaluators are therefore not uniformly interchangeable.

For the matched-decomposition endpoint, CPU- and GPU-scored runs were requested at $K = 16$ and paired by achieved part count across all 40 models. The paired achieved count had a median of 16 and ranged from 16 to 101. Under these matched counts, the GPU-scored system achieved a descriptive median speedup of 30.3 $\times$ in warm-start, synchronous, post-repair wall time. Its paired median relative high-resolution union-volume change against the exact materialized evaluator was +0.047%. Against the +2% sample-median noninferiority margin, the one-sided 95% upper confidence bound was +0.519%.

These findings support approximate batched evaluation in practice: although the GPU and CPU scorers often select different top-ranked planes, the selected candidates generally have similar exact scores, and the median change in final compactness remains small.

Approximate GPU scoring preserves near-optimal decisions at much lower cost

Identical root candidates compare materialized exact CPU child hulls with Collidify's full-fidelity batched GPU evaluator; the model is the inferential unit.

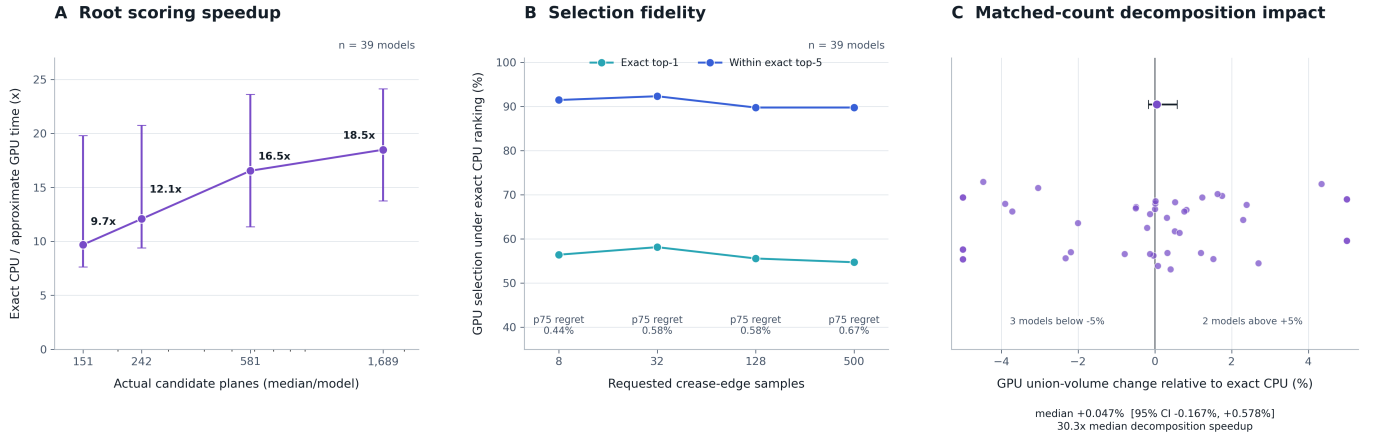


Figure 5. Approximate GPU plane-scoring trade-off. (A) Across 39 structurally validated models, GPU plane scoring was 9.69 $\times$ –18.49 $\times$ faster than the exact CPU reference at median root candidate-set sizes ranging from 151 to 1,689 planes. Whiskers are pointwise 95% model-bootstrap intervals. (B) Exact top-1 agreement remained 54.7%–58.1%, but 89.7%–92.3% of GPU selections ranked among the exact CPU top five, while the 75th-percentile CPU-reference regret remained below 0.7% of the observed exact-score range. (C) All 40 models were retained for decompositions requested at $K = 16$ and matched on achieved part count. The paired median relative high-resolution union-volume change was +0.047% (two-sided 95% CI: –0.167% to +0.578%; one-sided upper bound: +0.519%) at a descriptive median speedup of 30.3 $\times$. Local timers exclude candidate generation and preprocessing; decomposition timers begin from a common repaired mesh and exclude initial repair.

6.4 Representation-conditioned fitting and partitioning

The orientation-search study compared Collidify's discrete policy with PCA-only fitting. Across requested collider budgets of 4, 8, 16, and 32, discrete search reduced source-normalized union excess by 15.79–22.64 percentage points for boxes and 19.77–37.01 percentage points for capsules. All eight pointwise intervals excluded zero, and every family-by-budget sign test remained significant after Holm correction. The four-budget AUC reduction was 20.89 percentage points for boxes and 34.95 points for capsules. Cross-fitting, in which partitions and terminal fitting policies were exchanged between conditions, showed that terminal orientation selection produced the larger effect; representation-conditioned partition decisions contributed a smaller additional effect. The experiment did not isolate the incremental runtime cost of orientation search.

The orientation-search comparison and the narrower partition-conditioning ablation isolate different mechanisms. The orientation-search comparison changed the orientations available during prospective evaluation and controlled terminal fitting, with union-aware terminal refitting disabled. In the partition-conditioning ablation, family-specific candidate reductions were disabled, and both partitions passed through the same terminal search, union-aware refit, concavity refresh, and rollback procedure. This ablation therefore isolates the effect of target representation on cut selection after a shared terminal-fitting pipeline.

Under that narrower test, representation-conditioned sphere partitions reduced final controlled sphere-union volume by a median 13.2% at requested $K = 8$ and 15.7% at $K = 16$, favoring the conditioned policy in 30/40 and 32/40 models, respectively. The corresponding incremental box and capsule effects were smaller and uncertain: their family medians across the two budgets were -1.4% and -2.1% , and their pointwise intervals crossed zero. The contrast is representation-conditioned minus generic, so negative values favor representation-conditioned partitioning. Thus, Collidify's measured box and capsule advantage is supported primarily by discrete orientation search; the additional benefit of representation-conditioned partition selection is established most clearly for spheres.

Orientation search drives Box and Capsule gains; cut conditioning is a separate effect

A-B compare the complete discrete-search and PCA-only pipelines. C holds terminal fitting fixed and isolates only the additional value of representation-aware cuts.

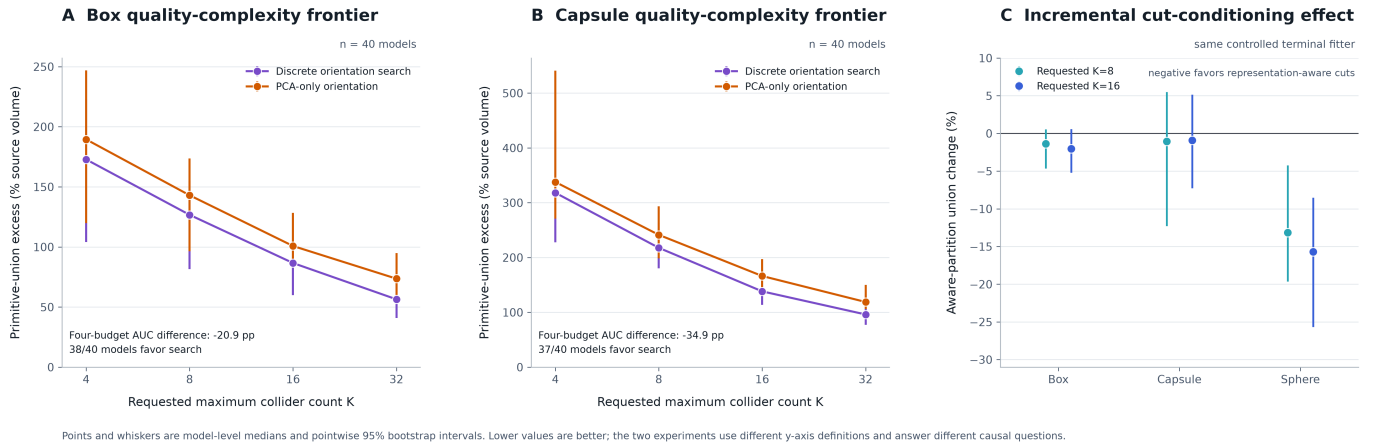


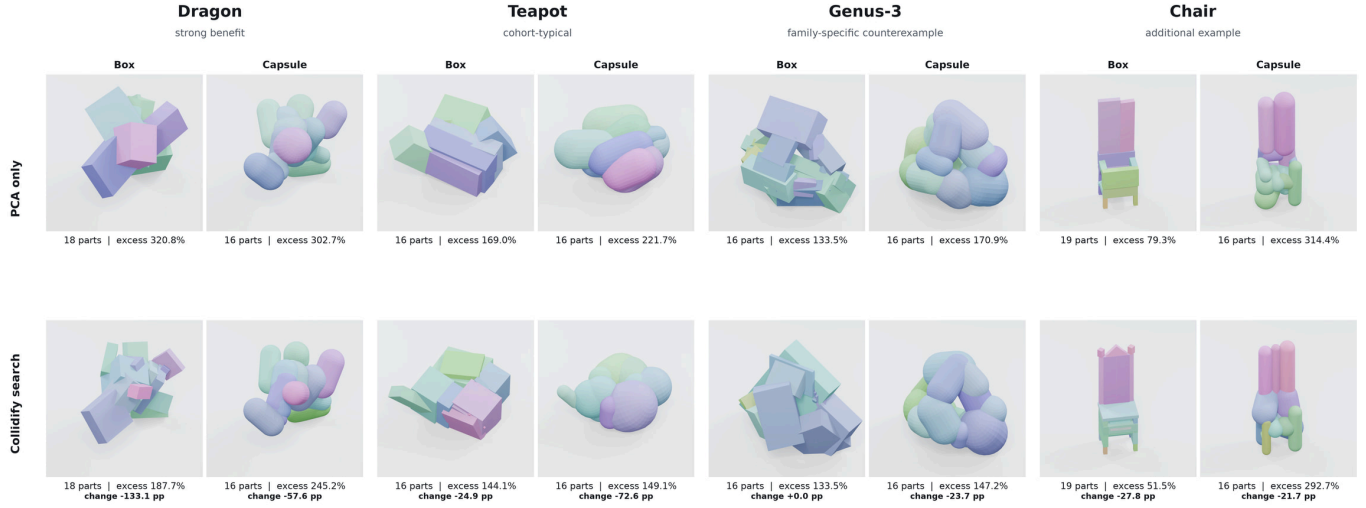
Figure 6. Discrete orientation search drives the box and capsule gains. (A–B) Matched-count decompositions compare Collidify's discrete-search fitting policy with the same overlap-aware partition objective restricted to one PCA orientation. Across the four requested collider budgets, discrete search reduced model-level quality–complexity AUC by 20.89 percentage points for boxes and 34.95 points for capsules; 38/40 and 37/40 models favored search, respectively. (C) A separate controlled experiment holds candidate generation and terminal fitting fixed and measures only the incremental effect of representation-conditioned partition selection. Box and capsule pointwise intervals cross zero, whereas representation-conditioned sphere partitions reduce final union volume by 13.2% at requested $K = 8$ and 15.7% at $K = 16$. Whiskers are pointwise 95% model-bootstrap intervals without end caps. Lower is better in every panel, but panels A–B report source-normalized union excess while panel C reports relative union-volume change; their vertical magnitudes should not be compared directly.

6.5 Candidate and cascade diagnostics

Crease-derived candidates were stronger in the paired median. Relative to crease-derived-only decompositions, axis-aligned-only decompositions had paired median relative union-volume changes of $+1.13\%$ at $K = 8$ and $+1.70\%$ at $K = 16$. Axis-aligned-only decompositions still produced the smaller union in 12/40 models at $K = 8$ and 9/40 at $K = 16$, showing that axis-aligned candidates help on a subset of models.

Discrete orientation search versus PCA-only fitting

Matched-count replay at requested $K=16$ (seed 42). Each pair shares the source mesh, overlap-aware objective, camera, lighting, and achieved collider count.



Locked 40-model result: four-budget AUC difference -20.89 pp for boxes (38/40 favor search) and -34.95 pp for capsules (37/40 favor search).

Negative changes mean less primitive-union volume beyond the source. Chair is a cohort member and is included as an additional example.

Figure 7. Discrete orientation search versus PCA-only fitting. Matched-count box and capsule decompositions are shown at requested $K = 16$. Within each pair, the source mesh, overlap-aware objective, camera, lighting, and achieved collider count are shared. Dragon is a strong-benefit case, Teapot is near the sample median, and Genus-3 is a family-specific counterexample. Chair is an additional qualitative example. Values report source-normalized union excess; lower is better.

Candidate diversity and cascaded scoring expose the speed-fidelity trade-off

Model-level summaries from controlled studies; lower union change is better, while higher speedup is better.

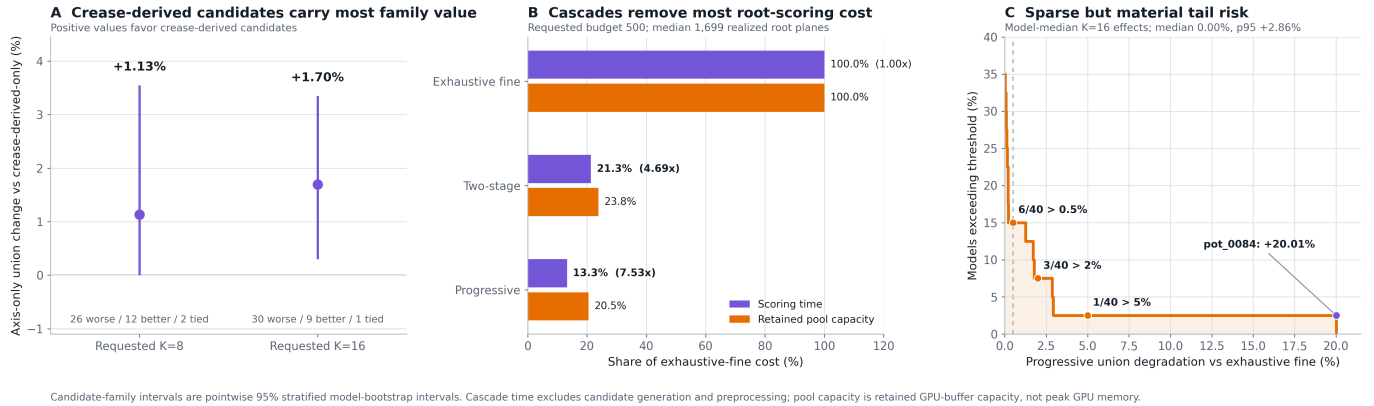


Figure 8. Candidate diversity and cascaded scoring expose the speed-fidelity trade-off. (A) Direct matched-count comparisons show paired median relative union-volume changes of +1.13% and +1.70% for axis-aligned-only against crease-derived-only decompositions at requested $K = 8$ and $K = 16$; positive values therefore favor the crease-derived family. Whiskers are pointwise 95% model-bootstrap intervals. (B) At the largest root workload (requested crease-edge sample budget of 500; median 1,699 realized planes), two-stage and progressive scoring required 21.3% and 13.3% as much time as exhaustive fine scoring, corresponding to speedups of 4.69 $\times$ and 7.53 $\times$, and retained 23.8% and 20.5% as much pooled-buffer capacity. (C) Progressive end-to-end compactness had a model median of 0.00% relative to exhaustive fine scoring, but 6/40 model medians exceeded +0.5%, 3/40 exceeded +2%, and the largest model-level degradation was +20.01%. The memory measure reports retained GPU buffer capacity; peak GPU memory, candidate generation, and preprocessing are outside its scope.

At the largest root-scoring workload, the progressive cascade was 7.53 $\times$ faster than exhaustive fine approximate-GPU scoring and retained 20.5% as much pooled-buffer capacity. In the fixed-request end-to-end decompositions, achieved counts agreed in 118/120 run-level workloads, and 106/120 produced exactly 16 parts in every arm. Retaining the two unequal-count workloads, the median relative union-volume change was 0.00%, but the distribution was heavy-tailed: 19 run-level workloads ex-

ceeded +0.5%, 10 exceeded +2%, and four exceeded +5%. After aggregation across runs, the largest model-level degradation was +20.01%. Comparisons with the two-stage variant indicate that the shared coarse top- K stage accounted for more of the systematic failure than the added progressive tier.

7. Discussion

7.1 Central interpretation

Collidify is a low-latency, GPU-batched, representation-conditioned collider decomposition framework. Convex meshes provide the broad external comparison, while the primitive modes test whether the same prospective-evaluation principle extends beyond convexity. CoACD's look-ahead search retains the stronger aggregate convex-mesh quality-part-count frontier. Collidify has no explicit fixed-budget planner, but evaluating a broad set of immediate choices yields paired median fixed-budget results close to CoACD at substantially lower latency. The CoACD quality-gap intervals exclude zero at budgets 8 and 16, while the interval at budget 32 includes zero. Within the eligible paired subsets, the paired median relative neutral-concavity gaps and latency favored Collidify over both VisACD and V-HACD. The primitive studies show two distinct effects: discrete orientation search improves boxes and capsules, while representation-conditioned partitioning is most clearly beneficial for spheres.

7.2 Practical significance

Many existing pipelines fit primitives only after choosing a partition, so the requested primitive family cannot influence the decomposition tree. Collidify evaluates prospective box, capsule, and sphere fits during cut selection, allowing authors to explore those outputs within the same workflow. Its roughly $100\times$ median convex-mesh speedup over CoACD also changes the iteration cycle: a decomposition that takes tens of seconds discourages experimentation, while one that takes a fraction of a second can be rerun after changes to thresholds, collider counts, repair settings, or output representations. In interactive authoring, this responsiveness may be more useful than a higher-quality but slower solution. Lower latency can also reduce offline processing cost for large asset collections.

7.3 Limitations

CoACD retains a quality advantage. Although Collidify achieves small paired median gaps under fixed part budgets, the aggregate mean trajectories show larger separation, consistent with substantial degradation on some models. At common neutral-concavity thresholds, Collidify generally requires more convex parts.

The experiments do not fully characterize rotation sensitivity. Although crease-angle sampling is rotation-invariant and crease-derived plane normals rotate equivariantly with the input, the current experiments do not establish rotation invariance for the complete method. World-axis-aligned planes can remain useful fallbacks for some models, although their incremental cohort-level effect was near zero.

Performance depends on surface topology. On highly uneven or irregular surfaces, the reward may favor cuts that isolate smoother regions while leaving protrusions connected to the remainder by thin or poorly supported geometry. Resolving those protrusions then requires additional cuts, which can produce less compact decompositions than more balanced evaluation strategies.

Box and capsule performance depends on geometry. For continuously curved surfaces, greedy primitive decomposition can produce poor cut sequences because locally attractive cuts may lead to poor full decomposition trees. Addressing these cases may require multi-step look-ahead, adaptive fallbacks, learned proposal mechanisms, or stronger search heuristics that preserve the efficiency of the batched evaluator.

The primitive experiments compare controlled Collidify variants rather than independent systems. Because no public Convex Primitive Decomposition implementation was available for external comparison [6], the results support only the tested internal mechanisms, not state-of-the-art primitive quality.

8. Conclusion

Collidify unifies convex-mesh and primitive decomposition in a greedy, representation-conditioned, top-down framework. It constructs approximate virtual children and evaluates many candidate cuts in parallel. This design avoids global tree search and achieves latency suited to interactive workflows.

Within the eligible paired subsets of the 461-model cohort, Collidify approached CoACD's typical fixed-budget convex-decomposition quality. At budgets of 8, 16, and 32 colliders, the paired median relative neutral-concavity gaps were +2.14% [0.70%, 4.46%], +1.63% [0.33%, 4.50%], and +0.84% [−1.60%, 3.70%], with median speedups of 110×, 127×, and 111×. The paired median relative neutral-concavity gaps and latency also favored Collidify over VisACD and V-HACD. Controlled experiments linked the observed behavior to the immediate-progress reward, approximate GPU evaluation, discrete primitive-orientation search, and representation-conditioned partitioning.

On the tested workstation, Collidify occupied a low-latency operating point with competitive median fixed-budget quality. Its common framework supports convex meshes, boxes, capsules, and spheres. Selective look-ahead or learned candidate selection could address difficult geometry while retaining low-latency evaluation.

Conflict of interest

The author develops and commercially sells Collidify and therefore has a financial interest in the software evaluated in this work. Collidify is proprietary, and the author designed the method, implemented the system, conducted the experiments, and prepared the paper. The external comparison uses the common orchestration, eligibility rules, and neutral evaluator described in Section 5.1, while the primitive results are controlled internal mechanism studies. These results should be interpreted in light of the author's commercial relationship to the product.

Software availability

Collidify is proprietary commercial software, and its source code is not publicly available. Product information and access to the commercial software are available through the [Collidify website](#).

Third-party material notice

Except where otherwise indicated, original material in this paper is licensed under [CC BY 4.0](#). Third-party benchmark models, source assets, and derived depictions, including material from the VHACD61 benchmark collection, ShapeNet, and Objaverse, are excluded and remain subject to the copyrights and license terms of their respective creators and source datasets. ShapeNet assets were used only as inputs to the reported experiments; no ShapeNet models or source assets are included in, distributed with, or required by the commercial Collidify software.

The five Objaverse models in Figure S2c are listed as CC BY 4.0 in Objaverse metadata. Their normalized geometry was rendered with each decomposition:

- *Alien mask*, Eva.Kraai, UID 94b779efb96e4825b00e1471bd60c61e
- *Bunny Cartoon Model*, xeratdragons, UID 24cad713d1f45bbada202d8fc9de4af
- *День 5. Смерть звезды.*, 3d0wl, UID 26312ddce65947c5b273e494185c4c30
- *Santiago Abascal*, Headsketch, UID 8ae3f9f5bda04309b1de45f69f6cffc6
- *Toon Ship test*, willphase1, UID 7cdee47bd9274ce2847a8a71fe042eef

References

- [1] J.-M. Lien and N. M. Amato. "Approximate Convex Decomposition of Polyhedra." *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, pp. 121–131, 2007. <https://doi.org/10.1145/1236246.1236265>
- [2] K. Mamou. "Volumetric Hierarchical Approximate Convex Decomposition." In *Game Engine Gems 3*, edited by E. Lengyel, A K Peters/CRC Press, pp. 141–158, 2016. <https://doi.org/10.1201/b21177-15>
- [3] X. Wei, M. Liu, Z. Ling, and H. Su. "Approximate Convex Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search." *ACM Transactions on Graphics*, 41(4), Article 42, 2022. <https://doi.org/10.1145/3528223.3530103>
- [4] E. Fokin and M. Savva. "VisACD: Visibility-Based GPU-Accelerated Approximate Convex Decomposition." *Eurographics 2026 Short Papers*, 2026. <https://arxiv.org/abs/2604.04244>
- [5] Y. Luo, Z. Pan, K. Wu, X. Du, Y. Zeng, X. Tang, Y. Wu, X. Jin, and X. Gao. "RL-ACD: Reinforcement Learning-based Approximate Convex Decomposition." *ACM Transactions on Graphics*, 44(6), Article 271, 2025. <https://doi.org/10.1145/3763270>
- [6] J. Knodt and X. Gao. "Convex Primitive Decomposition for Collision Detection." *Computer Graphics Forum*, 45(2), Article e70411, 2026. <https://doi.org/10.1111/cgf.70411>
- [7] K. Pearson. "On Lines and Planes of Closest Fit to Systems of Points in Space." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), pp. 559–572, 1901. <https://doi.org/10.1080/14786440109462720>
- [8] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. Vanderbilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. "Objaverse: A Universe of Annotated 3D Objects." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, 2023. <https://doi.org/10.1109/CVPR52729.2023.01264>
- [9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. "ShapeNet: An Information-Rich 3D Model Repository." arXiv:1512.03012, 2015. <https://arxiv.org/abs/1512.03012>
- [10] S. Oh, X. Yuan, X. Wei, R. Shi, F. Xiang, M. Liu, and H. Su. "PaMO: Parallel Mesh Optimization for Intersection-Free Low-Poly Modeling on the GPU." *Computer Graphics Forum*, 44(7), Article e70267, 2025. <https://doi.org/10.1111/cgf.70267>
- [11] A. Jacobson, L. Kavan, and O. Sorkine-Hornung. "Robust Inside-Outside Segmentation Using Generalized Winding Numbers." *ACM Transactions on Graphics*, 32(4), Article 33, 2013. <https://doi.org/10.1145/2461912.2461916>
- [12] J. Andrews. "Navigation-Driven Approximate Convex Decomposition." *ACM SIGGRAPH 2024 Conference Papers*, Article 56, 9 pages, 2024. <https://doi.org/10.1145/3641519.3657479>
- [13] D. Thul, L. Ladický, S. Jeong, and M. Pollefeys. "Approximate Convex Decomposition and Transfer for Animated Meshes." *ACM Transactions on Graphics*, 37(6), Article 226, 2018. <https://doi.org/10.1145/3272127.3275029>

Supplementary figures

sofa_0043: fixed-budget reward transplant

integrated frontier change -33.0% | strong improvement; upholstered shape

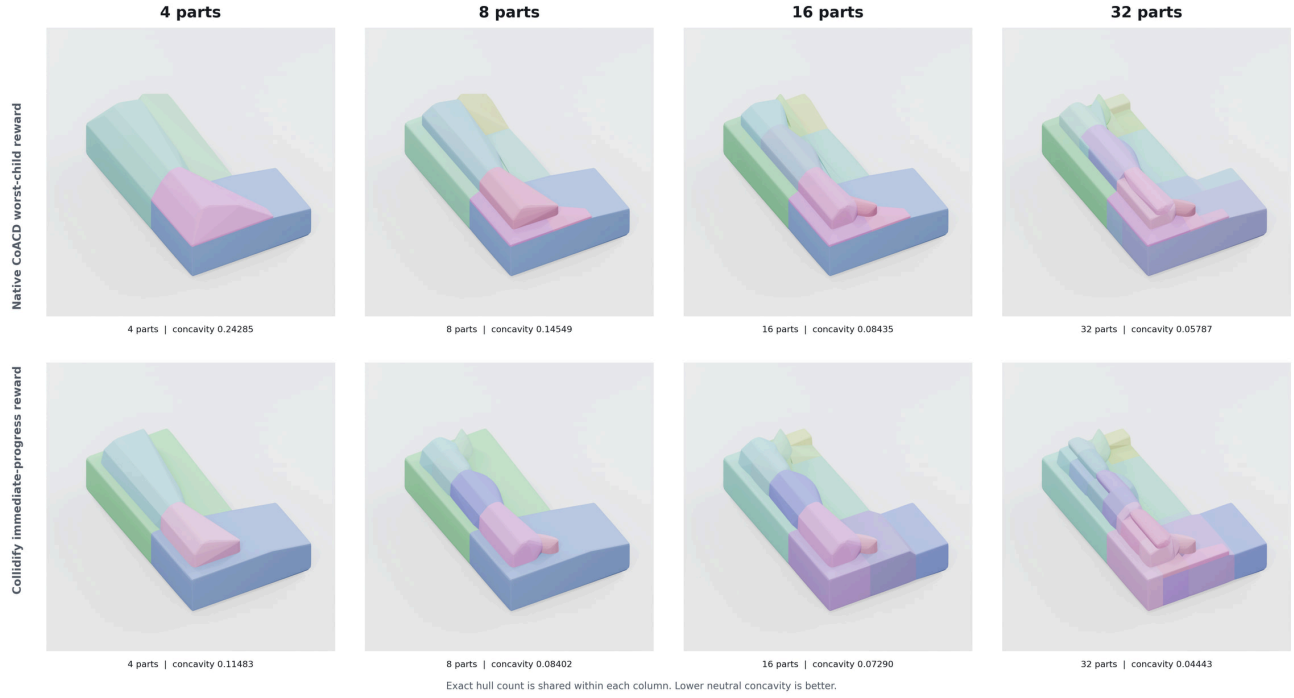


Figure S1a. Reward-transplant comparison: sofa_0043. The CoACD-derived worst-child reward is shown above and the Collidify immediate-progress transplant below at exact hull budgets of 4, 8, 16, and 32. Lower neutral concavity is better.

Chair: fixed-budget reward transplant

integrated frontier change -25.0% | strong improvement; articulated negative space

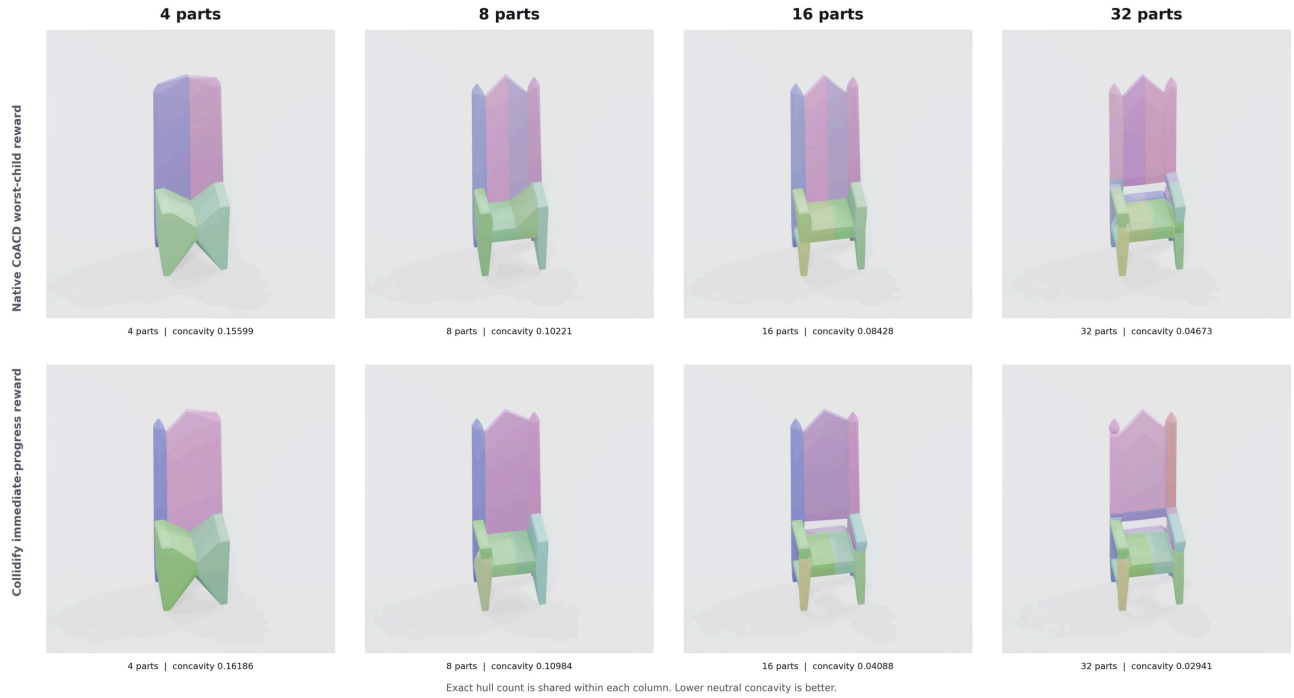


Figure S1b. Reward-transplant comparison: Chair. Exact fixed-budget comparison.

Bunny: fixed-budget reward transplant

integrated frontier change -24.7% | strong improvement; organic benchmark shape

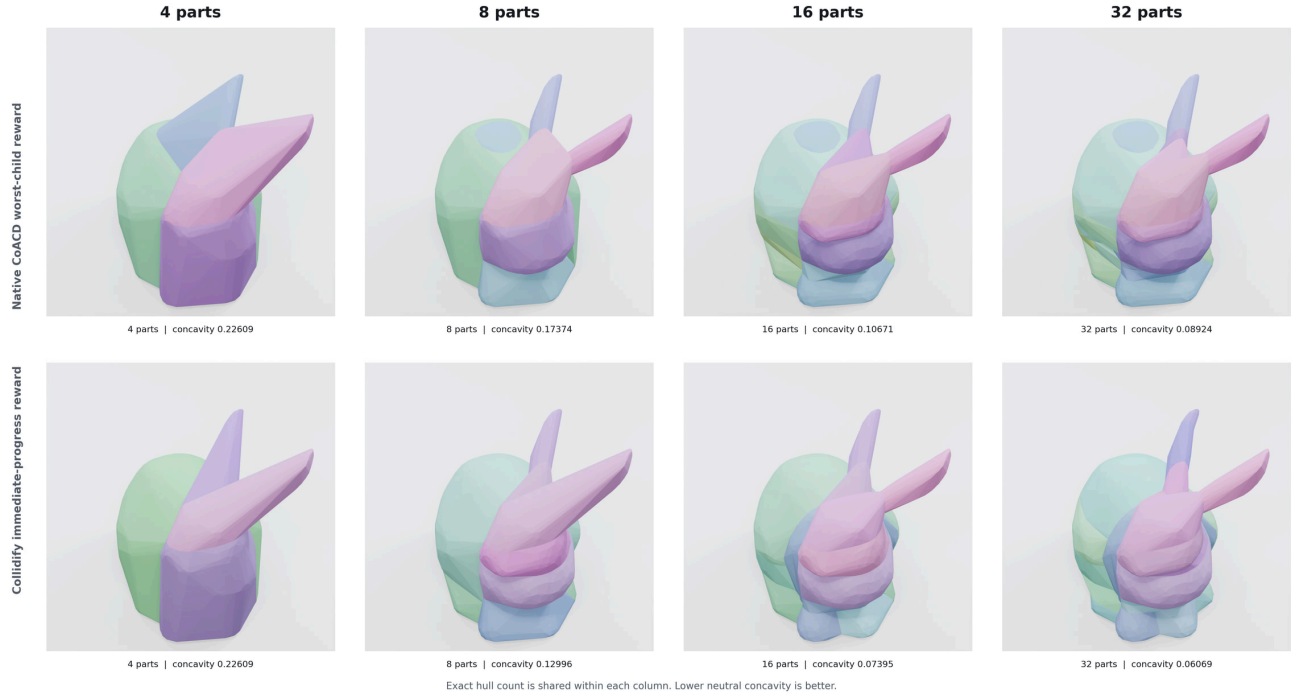


Figure S1c. Reward-transplant comparison: Bunny. Exact fixed-budget comparison.

pillow_0038: fixed-budget reward transplant

integrated frontier change +5.2% | moderate counterexample; smooth compact shape



Figure S1d. Reward-transplant comparison: pillow_0038. Exact fixed-budget comparison.

sledge: fixed-budget reward transplant

integrated frontier change -51.3% | second-largest improvement; severe native over-splitting

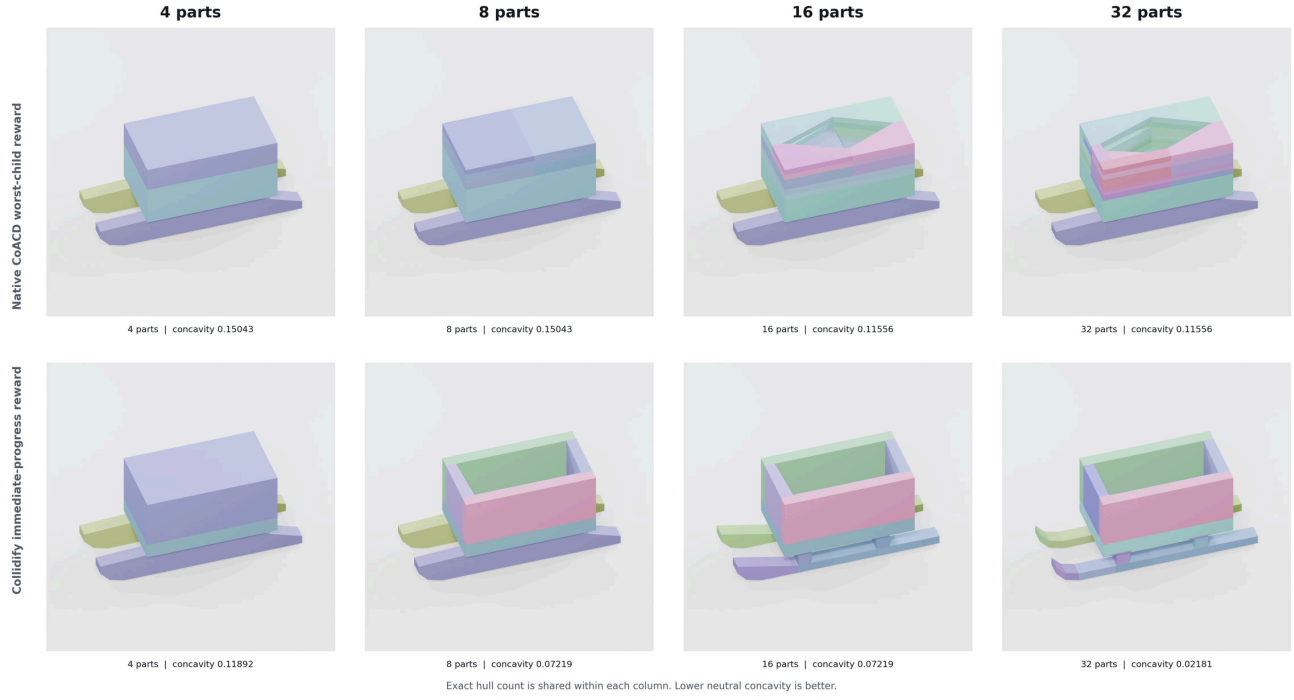


Figure S1e. Reward-transplant comparison: sledge. Sledge is a strong complex-shape benefit case in which the native reward spends the fixed hull budget over-splitting less informative regions.

hero: fixed-budget reward transplant

integrated frontier change -21.5% | strong improvement; complex disconnected geometry

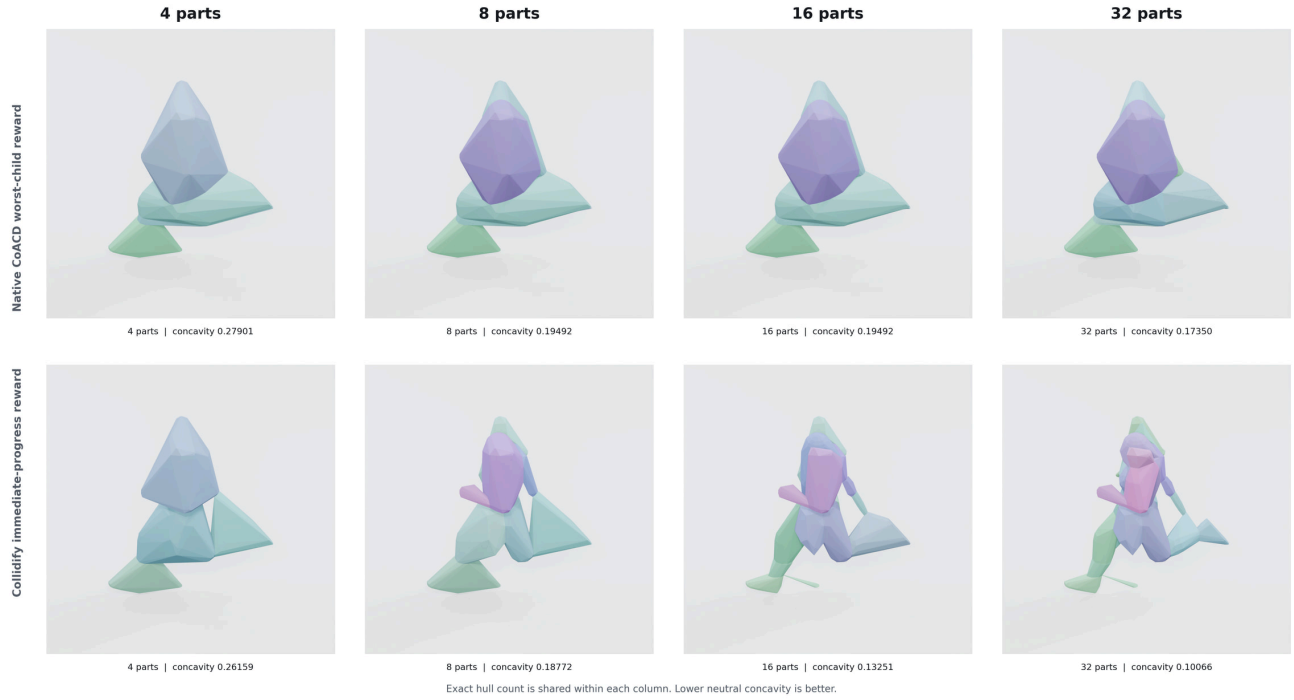


Figure S1f. Reward-transplant comparison: hero. Exact fixed-budget comparison.

shark: fixed-budget reward transplant

integrated frontier change -29.5% | strong improvement; elongated organic shape

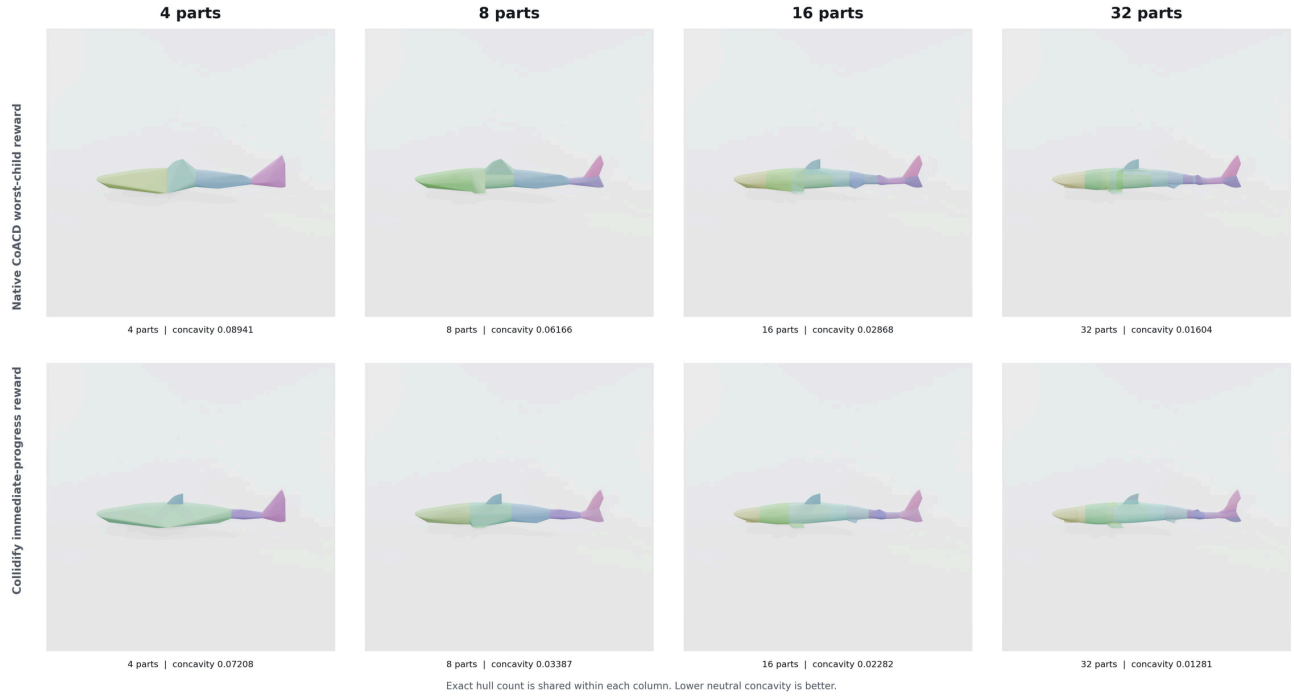


Figure S1g. Reward-transplant comparison: shark. Exact fixed-budget comparison.

screwdriver: fixed-budget reward transplant

integrated frontier change -29.3% | strong improvement; elongated mechanical shape

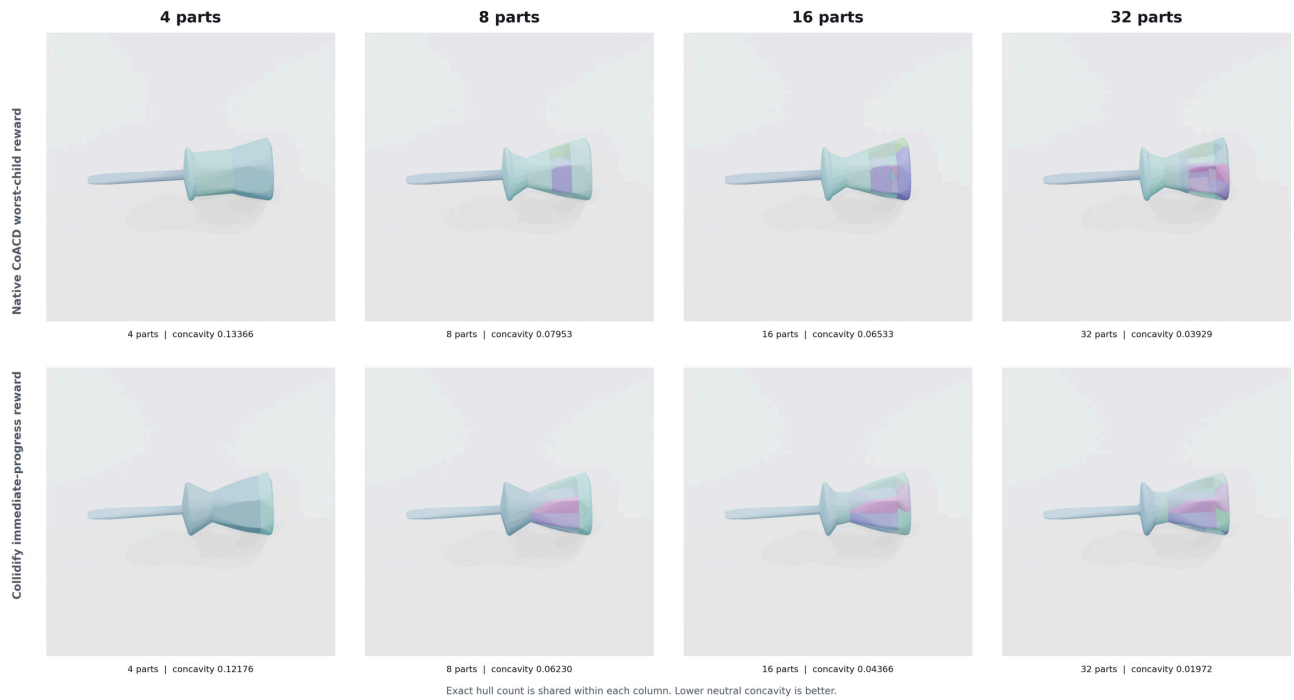


Figure S1h. Reward-transplant comparison: screwdriver. Exact fixed-budget comparison.

elk: fixed-budget reward transplant

integrated frontier change +5.6% | moderate counterexample; articulated organic shape

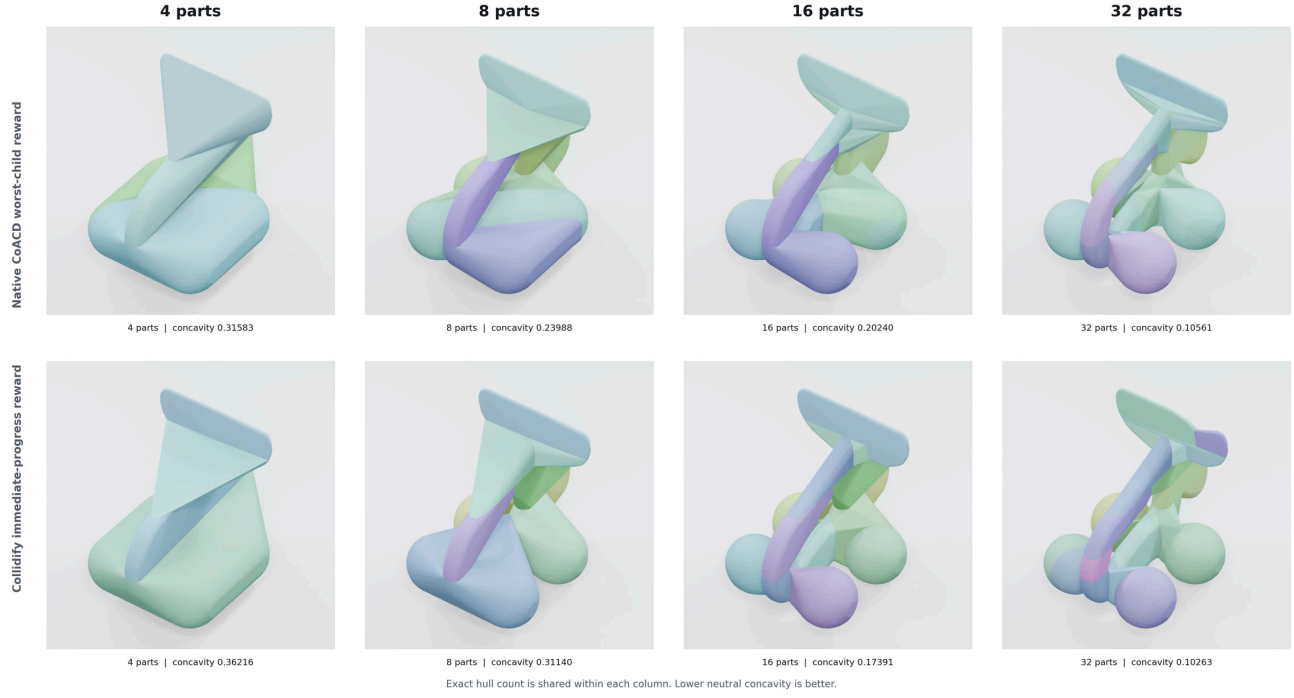


Figure S1i. Reward-transplant comparison: elk. Elk is a moderate counterexample.

homer: fixed-budget reward transplant

integrated frontier change -26.8% | strong improvement; complex articulated shape

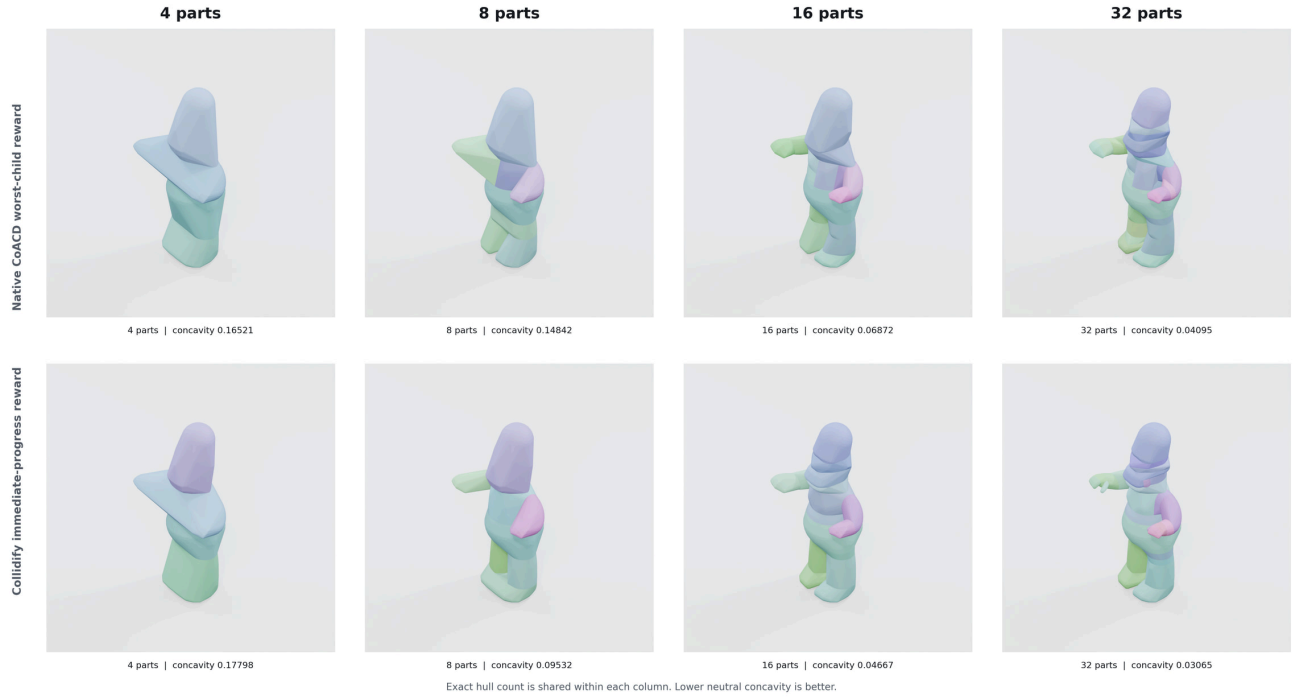


Figure S1j. Reward-transplant comparison: homer. Exact fixed-budget comparison.

VHACD61 decomposition examples at a maximum of 16 parts

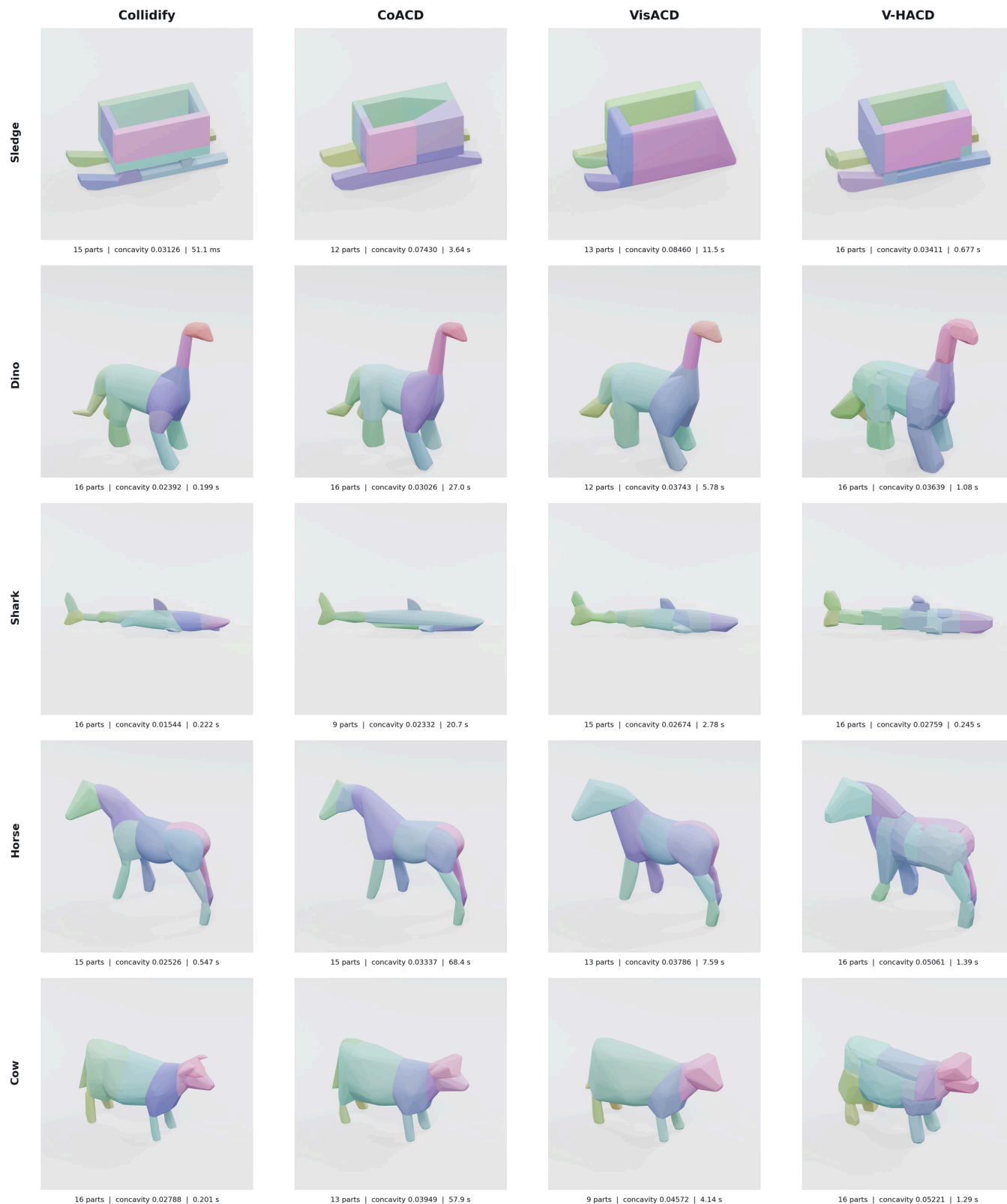


Figure S2a. Dataset-spanning decomposition examples across methods: VHACD61. Sledge, Dino, Shark, Horse, and Cow are shown for Collidify, CoACD, VisACD, and V-HACD. Each panel shows the feasible operating point with the lowest neutral concavity under a 16-part budget; actual part count, neutral concavity, and end-to-end runtime are printed below it.

ShapeNet decomposition examples at a maximum of 16 parts

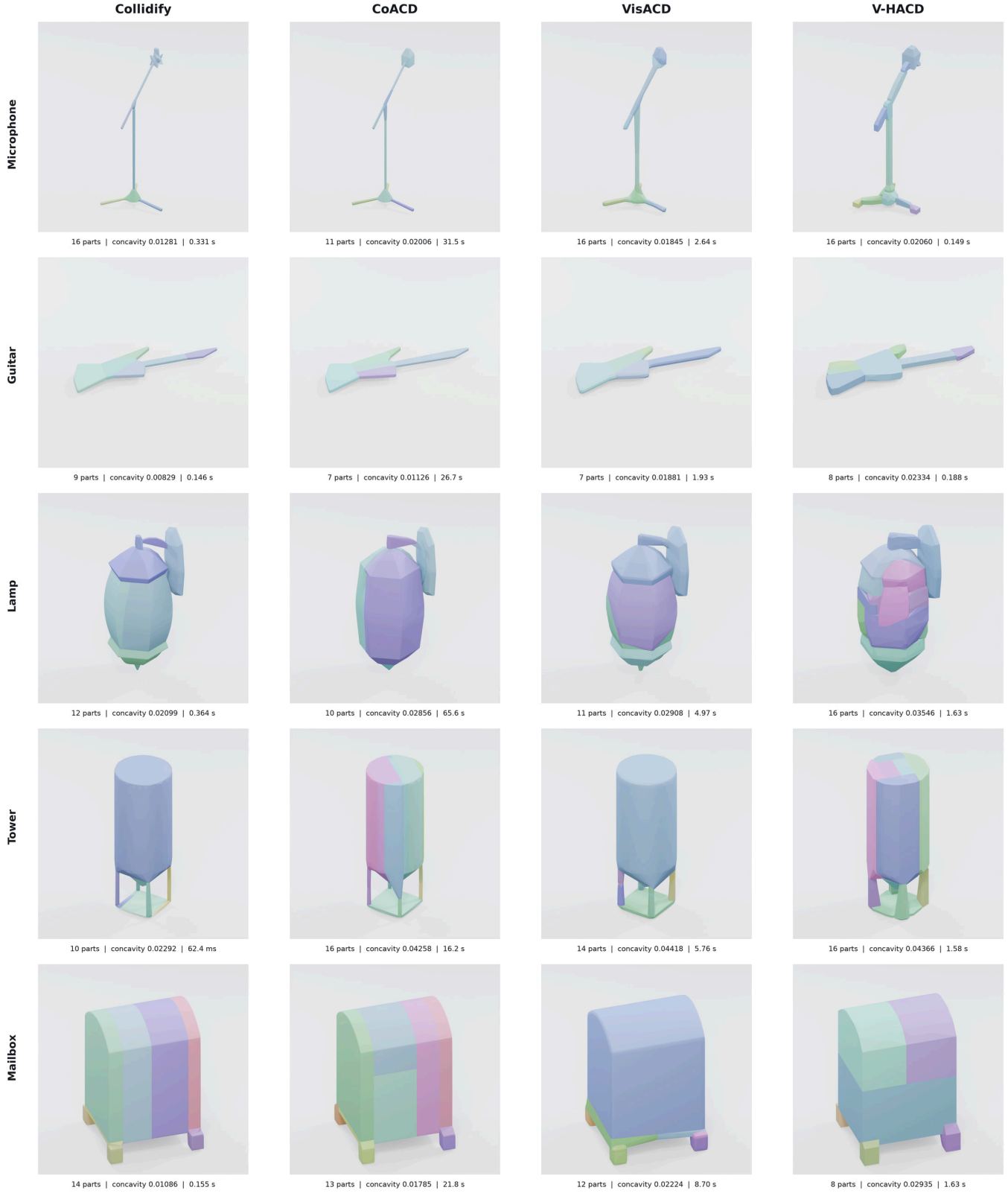


Figure S2b. Dataset-spanning decomposition examples across methods: ShapeNet. Microphone, Guitar, Lamp, Tower, and Mailbox are shown for Collidify, CoACD, VisACD, and V-HACD. Each panel shows the feasible operating point with the lowest neutral concavity under a 16-part budget; actual part count, neutral concavity, and end-to-end runtime are printed below it.

Objaverse decomposition examples at a maximum of 16 parts

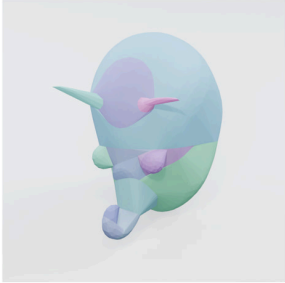
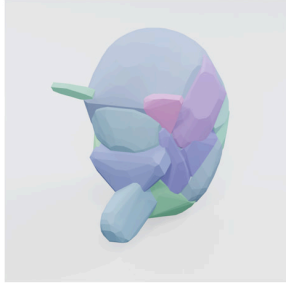
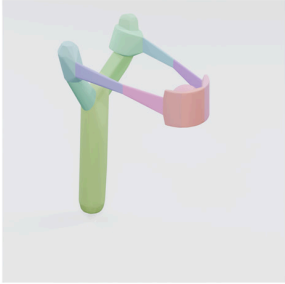
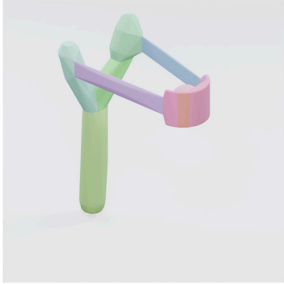
	Collidify	CoACD	VisACD	V-HACD
94b779ef	 16 parts concavity 0.01522 0.191 s	 13 parts concavity 0.02353 36.3 s	 15 parts concavity 0.02076 7.54 s	 16 parts concavity 0.06538 3.04 s
24cadc71	 15 parts concavity 0.03237 0.484 s	 15 parts concavity 0.03275 68.7 s	 13 parts concavity 0.03692 4.51 s	 16 parts concavity 0.05005 1.05 s
26312ddc	 14 parts concavity 0.01890 0.181 s	 11 parts concavity 0.02691 29.6 s	 12 parts concavity 0.03486 8.60 s	 16 parts concavity 0.03814 0.818 s
8ae3f9f5	 15 parts concavity 0.02018 0.480 s	 16 parts concavity 0.02782 83.3 s	 7 parts concavity 0.03823 8.28 s	 16 parts concavity 0.05111 2.69 s
7cdee47b	 15 parts concavity 0.01913 0.134 s	 14 parts concavity 0.02045 18.0 s	 15 parts concavity 0.02753 4.42 s	 16 parts concavity 0.03286 0.619 s

Figure S2c. Dataset-spanning decomposition examples across methods: Objaverse. Five examples identified by their Objaverse ID prefixes are shown for Collidify, CoACD, VisACD, and V-HACD. Each panel shows the feasible operating point with the lowest neutral concavity under a 16-part budget; actual part count, neutral concavity, and end-to-end runtime are printed below it.