GROWBREP: GROWING CLOSED B-REP SHELLS ONE FACE AT A TIME UNDER GRAMMAR CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Boundary representation (B-Rep) is the native format of computer-aided design (CAD), and generating it means producing continuous geometry and discrete topology at the same time, with topology as a hard constraint. A B-Rep is usable downstream only if it closes into a solid, yet existing generators can only check closure after generation. We present GrowBrep, an autoregressive B-Rep generator that makes closure a property of its sequence grammar. GrowBrep serializes a solid as a shell that grows one face at a time along a frontier of open edges. Each edge is introduced exactly once and closed exactly once by a second half-edge, and the sequence cannot end while any edge remains open, so every sequence the model is allowed to finish describes a closed, consistently oriented shell. The grammar does not constrain geometry, which the model still has to learn. Geometry is tokenized by per-primitive autoencoders with a finite scalar quantization bottleneck. A decoder-only transformer generates the sequence, conditioned on a complexity token and optionally on an input point cloud, and illegal tokens are masked out at every decoding step. On ABC-1M, 81.7% of GrowBrep's generation attempts yield valid watertight solids, versus 49.0% for the strongest baseline, under the same strict validity check on every attempt of every method without rejection sampling. On the distribution-level metrics (COV, MMD, and JSD) it is comparable to the strongest baselines. Conditioned on point clouds, it produces a valid solid for 80.0% of the inputs, compared with 51.6% for the closest prior method on its own published test set, at $65\times$ lower cost per cloud. Drawing five independent samples per cloud and keeping the first valid solid raises validity to 95.8%.

1 INTRODUCTION

Mechanical CAD is authored and exchanged in boundary representation (B-Rep). Surfaces and curves are given in analytic parametric form, and faces are trimmed by loops and joined along shared edges into a shell. Unlike a mesh, which approximates a surface with planar facets, a B-Rep records the exact surfaces and their trimming boundaries. Machining plans, finite-element meshes, and assembly constraints can be computed directly from the faces and their adjacency, and a designer can edit the part face by face (Guo et al., 2022; Jayaraman et al., 2023). Converting a B-Rep to a mesh, a voxel grid, or a signed distance field discards both: the type and parameters of each surface, and the loops that trim each face. Errors in the two are not symmetric: a slightly worse surface fit gives a slightly worse part, whereas a topological error is fatal. A shell is closed and consistently oriented only if every edge is visited by exactly two half-edges (oriented uses of the edge by a loop) of opposite orientation. If a single edge is visited only once, the CAD kernel returns no solid at all, and no downstream tool can use the output, however well the surfaces are fitted.

Existing methods take one of two routes. Diffusion-based methods generate the geometry of faces, edges, and vertices at once and obtain the relations between primitives either by geometric matching afterwards or from a separate topology model (Xu et al., 2024; Li et al., 2025a). Autoregressive methods generate quantized primitives one after another and represent the connections between them with pointers or indices (Xu et al., 2025; Li et al., 2026a). On either route, topology is an output the model must predict, and closure can be judged only after all primitives have been decoded and assembled. A few wrong connections then invalidate the solid.

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.

Our key idea, sketched in Figure 2, is to make the sequence describe how a shell is built. The model starts from a seed face and grows the shell face by face along a frontier, the set of currently open edges. The frontier is the topological state, so closure and orientation consistency become tests of token legality applied during sampling; Section 3.4 states exactly what the grammar does and does not guarantee.

Our contributions are as follows.

- We propose a growth sequence for B-Rep solids whose unit is the face. Generation starts from a seed face and expands along the frontier of open edges, recording the shell as a sequence of new and reused edges.
- Using the frontier state carried by the sequence, we enforce closure at decoding time: an edge that has already been closed cannot be used again, a non-existent edge cannot be referenced, and generation cannot end while the frontier is non-empty. Because the decoder chooses only among legal tokens, closure of the shell becomes a property of the sequence itself.
- We evaluate unconditional and point-cloud-conditioned generation under the same protocol and show that GrowBrep raises validity: 81.7% of unconditional attempts and 80.0% of single point-cloud-conditioned samples yield valid solids without loss of geometric fidelity.
- We exploit the low cost of a single point-cloud-conditioned sample. Taking up to five independent samples per cloud and keeping the first that passes the validity check raises validity from 80.0% to 95.8%. Even with all five samples counted, the total cost stays about $13\times$ below that of DualBrep (Liu et al., 2026b).

2 Related Work

CAD construction sequences. Sketch-and-extrude and other program-synthesis approaches write CAD as executable commands or constraint programs and let a kernel build the solid (Wu et al., 2021; Xu et al., 2022; Seff et al., 2022; Khan et al., 2024; Rukhovich et al., 2025). Such solids are valid by construction, in a stronger sense than any boundary-level guarantee, but the operator set and the available construction histories limit what they can express, and free-form surfaces often lie beyond both. GrowBrep generates the B-Rep directly.

Diffusion-based B-Rep generation. Diffusion-based methods generate the geometry of all faces and edges at once and obtain their connections separately, either from geometric proximity after decoding or from a dedicated topology model (Xu et al., 2024; Lee et al., 2025; Liu et al., 2025; Lu et al., 2026; Liu et al., 2026b; Li et al., 2025a; 2026b; Liu et al., 2026a; Sun et al., 2026). Some of them predict or constrain the topology before the geometry, but in all of them validity is checked only after the geometry has been generated and assembled.

Autoregressive B-Rep generation. Closest to our work are methods that quantize a B-Rep into tokens and predict them one by one. SolidGen (Jayaraman et al., 2023) synthesizes a solid in the hierarchical order of vertices, edges, and faces with three separate models and, following PolyGen (Nash et al., 2020), masks invalid logits at sampling time, including the local constraint that an edge is referenced by at most two faces. Stitch-A-Shape (Li et al., 2025b) follows the same bottom-up order but samples only the vertices autoregressively, predicting edges, loops, and surfaces with feed-forward networks. AutoBrep (Xu et al., 2025) orders faces by a breadth-first traversal of face adjacency and represents connections with pointers into a local window of already generated faces, while BrepARG (Li et al., 2026a) splits faces and edges into two sub-sequences, with adjacency given by a pair of face indices per edge. BrepGPT (Li et al., 2025c) unifies geometry and topology through a half-patch decomposition, and ParaCAD (Qin et al., 2026) generates parametric surfaces autoregressively from a point cloud and recovers edges and vertices by intersecting the surfaces globally. The constraints these methods enforce are local or syntactic. SolidGen's mask rules out a third use of an edge but not an edge that is used only once, and a pointer or index layout keeps the sequence self-consistent without guaranteeing that the shell closes. Our grammar instead makes closure an invariant of the whole sequence: every edge is used exactly twice, with opposite orientations, and the sequence cannot end before all edges are closed (Section 3.4).

3 METHOD

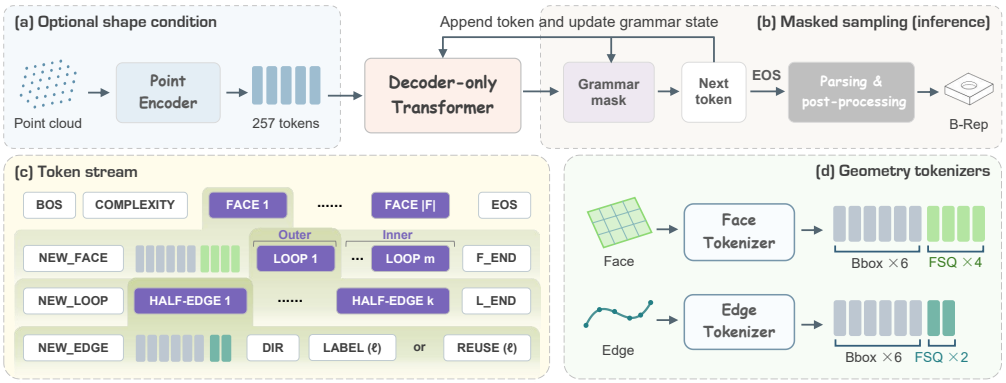

Figure 1: Overview of GrowBrep. A decoder-only transformer generates the token stream one token at a time. (a) Optional point-cloud prefix tokens condition the generation. (b) At inference, a grammar mask restricts the next token, and the completed sequence is parsed and post-processed into a B-Rep. (c) The stream nests faces, loops, and half-edges. (d) Per-primitive tokenizers encode each face and edge as bounding-box tokens and FSQ codes.

3.1 B-REP PRELIMINARIES

A B-Rep solid consists of a combinatorial shell and the geometry attached to it. Let F denote its set of faces, E its edges, and V its vertices. Each face $f \in F$ carries an ordered list of loops; the first loop is the outer boundary and the remaining loops are holes. Each loop is a cyclic sequence of half-edges, where a half-edge is a pair (e, o) of an edge $e \in E$ and an orientation $o \in \{+1, -1\}$ recording whether the loop traverses e along or against its intrinsic direction. Geometrically, each face carries a 32×32 grid of points sampled from its trimmed parametric surface, each edge 32 points along its curve, and each vertex a position; coordinates are normalized per solid into $[-1, 1]^3$.

3.2 GEOMETRY REPRESENTATION

We adopt AutoBrep's per-primitive geometry tokenizer (Xu et al., 2025) (Figure 1d): two small residual convolutional autoencoders (Chen et al., 2025) with a finite scalar quantization (FSQ) bottleneck (Mentzer et al., 2024), one for faces and one for edges. The face codec maps the $32 \times 32 \times 3$ UV grid to a 16-dimensional latent and decodes it back to the same shape; the edge codec maps the 32×3 curve to an 8-dimensional latent. Quantization uses a 4-dimensional lattice with per-dimension levels $[8, 5, 5, 5]$, so a face latent is split into 4 tokens and an edge latent into 2. Each primitive also carries a bounding box given by two opposite corners, whose six coordinates are quantized uniformly to 1,024 levels over the normalized range $[-1, 1]$. In total, a face is encoded by 10 geometry tokens and an edge by 8.

3.3 FRONTIER-GROWTH SERIALIZATION

Frontier. We serialize a solid by growing its shell one face at a time (Figure 2a). The traversal maintains a frontier, the set of edges that have been introduced but used only once and still need a second face. Generation starts from a seed face, whose entire boundary enters the frontier. Every later face must attach to the frontier: its outer loop begins by consuming an open edge, and each of its edges not yet seen is introduced and joins the frontier. The traversal ends when the frontier is empty, at which point every edge has been used exactly twice.

Token stream. The sequence is a list of faces (Figure 1c). Each face opens with a `NEW_FACE` token, followed by its geometry codes and then its loops, and closes with `F_END`. Each loop opens with `NEW_LOOP` and closes with `L_END`. The first loop of a face is its outer loop and every later loop is a hole, so "exactly one outer loop per face" holds by position and needs no extra constraint.

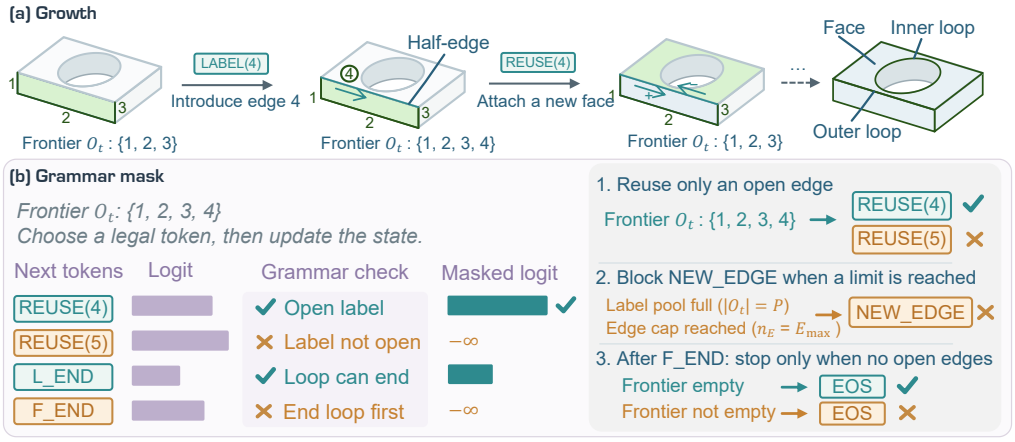

Figure 2: Frontier growth and the grammar mask. (a) Opening an edge adds its label to the frontier O_t; a reuse from a neighboring face closes it with the opposite orientation. (b) The mask admits $\texttt{REUSE}(\ell)$ only for $\ell \in O_t$, removes NEW_EDGE at the caps, and admits EOS only when O_t is empty.

Each loop is a sequence of half-edges, and a half-edge takes one of two forms, opening a new edge or closing one that is already open; the syntax of the stream is

$$\begin{aligned}
\text{sequence} &::= \texttt{BOS COMPLEXITY face}^{+}\ \texttt{EOS} \\
\text{face} &::= \texttt{NEW_FACE}\ b^{\mathrm{f}}_{1:6}\ q^{\mathrm{f}}_{1:4}\ \text{loop}^{+}\ \texttt{F_END} \\
\text{loop} &::= \texttt{NEW_LOOP half-edge}^{+}\ \texttt{L_END} \\
\text{half-edge} &::= \texttt{NEW_EDGE}\ b^{\mathrm{e}}_{1:6}\ q^{\mathrm{e}}_{1:2}\ \texttt{DIR LABEL}(\ell)\ \mid\ \texttt{REUSE}(\ell),
\end{aligned}$$

where b are the bounding-box tokens and q the FSQ geometry codes of Section 3.2, ℓ is a label, and COMPLEXITY is the target-complexity token, one of EASY, MID, HARD, or UNCOND. The sequence contains no vertex tokens; vertices are recovered from curve endpoints at parse time (Section 3.5).

The two half-edge forms are deliberately asymmetric. Opening an edge emits NEW_EDGE, its bounding-box and geometry tokens, a direction token, and a label; closing an edge takes a single token. Sequence length scales with the number of edges, which is half the number of half-edges. The intrinsic direction of an edge is the direction in which its opening half-edge traverses it, so the opening half-edge has $o = +1$ and the reusing half-edge $o = -1$; the direction token DIR records whether the decoded curve runs along or against that direction. A label names an edge while it stays open. Labels are drawn from a recycled pool of $P = 128$ labels: an opened edge takes any label that is free at that moment, a choice left to the model (training sequences use uniformly random free labels), and the label is returned when the edge closes; $\texttt{REUSE}(\ell)$ closes the edge that currently carries label ℓ. Identifying an open edge with its label, the frontier after t tokens is

$$O_t = \{\, \ell : \text{the last } \texttt{LABEL}(\ell) \text{ is not followed by } \texttt{REUSE}(\ell) \,\}, \tag{1}$$

and $|O_t| \le P$ at all times. An edge is addressable as long as it is open, regardless of how long ago it was opened. We train a decoder-only transformer on these sequences with a next-token cross-entropy loss that excludes the randomly assigned LABEL tokens.

Growth order. The grammar admits many growth orders, and an order whose peak frontier exceeds P cannot be serialized, because each open edge occupies one label. We therefore grow greedily to keep the frontier small: we seed from the face with the fewest edges and always place the adjacent face that leaves the smallest frontier, with a breadth-first fallback (Appendix B). A small frontier also keeps each label close to its reuse.

3.4 Grammar-constrained sampling

The syntax fixes what a sequence looks like; the grammar adds rules that tie the next token to the current frontier. Let $\mathcal{A}_t$ denote the set of tokens the grammar permits after the prefix $x_{1:t}$ (Figure 2b).

Grammar rules. Beyond the syntax, the grammar has four rules. The outer loop of every face after the seed must begin with a REUSE, which attaches the face to the frontier and keeps the shell connected. The other three hold at the positions where the syntax admits the token at all (a half-edge slot for REUSE and NEW_EDGE, the face boundary after F_END for EOS), except the attach slot and full loops:

$$\texttt{REUSE}(\ell) \in \mathcal{A}_t \iff \ell \in O_t, \tag{2}$$

$$\texttt{NEW_EDGE} \in \mathcal{A}_t \iff n_E(t) < E_{\max} \wedge |O_t| < P, \tag{3}$$

$$\texttt{EOS} \in \mathcal{A}_t \iff O_t = \emptyset, \tag{4}$$

where $n_E(t)$ is the number of edges opened so far and $E_{\max} = 255$ is the largest edge count for which every closed shell without inner loops fits within the parser's primitive budget of $N = 512$; the second condition keeps a free label for the new edge. Rule (3) bounds the representation. Without it, a model that keeps opening edges would generate until it exhausts the token budget, because EOS stays illegal while any edge is open; once a cap is reached, NEW_EDGE disappears, the model can only reuse open edges, and O_t drains until EOS becomes legal. Skipping any removal that would empty the permitted set rules out deadlock.

Closed by construction. Combinatorially, a shell is closed and consistently oriented when every edge appears in exactly two half-edges and the two carry opposite orientations. The first condition rules out open edges and non-manifold edges; the second means that the two faces meeting at an edge traverse it in opposite directions, so their orientations agree. Both are necessary for a generated B-Rep to become a solid, and the rules above enforce both.

Closure guarantee. Every token sequence that ends with EOS and satisfies $x_{t+1} \in \mathcal{A}_t$ at every step describes a connected shell in which every edge appears in exactly two half-edges, with opposite orientations.

Indeed, NEW_EDGE gives an edge its first use and puts its label in O_t. By (2) a second use must name a label in O_t, and the reuse removes it, so no sequence can express a third use. A reuse is the reversed twin of the opening half-edge, so the two uses have opposite orientations. By (4) the sequence cannot end while an edge has a single use, and the attach rule makes every face share an edge with an earlier one.

The guarantee holds regardless of what the model has learned, but it is combinatorial; whether the kernel accepts the result as a watertight solid depends on how accurately the surfaces and curves are fitted and on the post-processing (Section 3.5).

Masked sampling. This guarantee is a statement about sequences, and the training sequences satisfy it by construction; masking makes it hold for every sample as well. At each step we sample from the model distribution restricted to $\mathcal{A}_t$ (Figure 1b),

$$\tilde{p}_\theta(x_{t+1} \mid x_{1:t}) = \frac{p_\theta(x_{t+1} \mid x_{1:t})\,\mathbf{1}[x_{t+1} \in \mathcal{A}_t]}{\sum_{x \in \mathcal{A}_t} p_\theta(x \mid x_{1:t})}, \tag{5}$$

so every sampled token lies in $\mathcal{A}_t$; we set the logits outside $\mathcal{A}_t$ to $-\infty$ before temperature and nucleus sampling. An incremental state machine produces $\mathcal{A}_t$; it tracks O_t, the number of half-edges in the current loop and of loops in the current face, and the number of edges opened so far, and it updates in constant time per token. If $\mathcal{A}_t$ is ever empty we raise an error rather than lift the mask, since silently lifting it would void the guarantee unnoticed.

3.5 Parsing and post-processing

Parsing a completed sequence yields exact per-type primitive counts, the full half-edge topology, and the quantized codes; one batched pass of the codec decoder produces the face grids and edge curves. Prior methods merge edge endpoints into vertices by geometric proximity (Xu et al., 2024; Li et al., 2026a; Liu et al., 2025) or derive them from curve intersections (Liu et al., 2026b), and a single wrong split leaves a loop open while a wrong merge changes the topology. We instead recover vertices without distance thresholds, by a union-find over the $2|E|$ curve endpoints driven by the decoded loops. Write h^{in} and h^{out} for the entry and exit endpoints of half-edge h, that

is, the start and end of its edge in its intrinsic direction, swapped when $o = -1$. For every loop $(h_1, \ldots, h_k)$ we identify

$$h_i^{\text{out}} \sim h_{i+1}^{\text{in}}, \qquad i = 1, \ldots, k, \qquad h_{k+1} := h_1; \tag{6}$$

the equivalence classes of the endpoints under $\sim$ are the vertices V, positioned at the decoded curve endpoints, so their number and membership are fixed by the topology. The `DIR` tokens tell the parser which physical endpoint of each edge meets which vertex, and hence which curves to flip.

Post-processing turns the decoded grids and polylines into a parametric solid: it fits each face to the lowest-residual plane, cylinder, sphere, cone, or torus, falling back to a B-spline, fits each edge to a line, circular arc, ellipse, or cubic curve, partitions the half-edge topology into kernel-level wires, and sews the faces. Because no single tolerance suits all shapes, the export retries in a fixed order, the validity ladder: analytic primitives, then surface reconciliation, then a B-spline refit, returning the first result that passes the strict validity check, else a final boundary snap (Appendix B). Tolerances scale with the extent of the decoded shape.

4 Experiments

We evaluate GrowBrep on unconditional and point-cloud-conditioned generation with a GPT-style decoder-only transformer (Brown et al., 2020) of 406 M parameters (453 M with the codec decoder; Appendix B). Unless stated otherwise, all numbers use one untuned sampling configuration ($T = 1.0$, top-$p = 0.9$, no classifier-free guidance).

Table 1: Unconditional B-Rep generation; distribution metrics use the ABC-1M test set as reference.

Method	#params	ms/face↓		Build%↑	Valid%↑	COV%↑	MMD↓	JSD↓
	(M)	sampling	post-proc.				($\times 10^2$)	($\times 10^2$)
BrepGen (Xu et al., 2024)	292	681.2	**53**	34.57	32.49	65.56	**1.438**	1.820
DTGBrepGen (Li et al., 2025a)	**178**	**64.9**	417	37.60	30.02	56.06	1.629	1.525
BrepDiff† (Lee et al., 2025)	260	141.1	1711	44.06	31.55	63.89	1.522	1.481
AutoBrep† (Xu et al., 2025)	1034	175.2	93	74.37	48.98	**69.77**	1.473	1.546
GrowBrep	453	78.2	329	**97.25**	**81.65**	69.60	1.447	**1.392**

† We use the authors' released weights.

4.1 Setup

Dataset and baselines. We use the ABC-1M data split (Koch et al., 2019; Xu et al., 2025) (Appendix C) and evaluate all methods on its test set. We train BrepGen (Xu et al., 2024) and DTGBrepGen (Li et al., 2025a) on the same split ourselves; for BrepDiff (Lee et al., 2025) and AutoBrep we use the released weights.

Unconditional generation. Each method generates about 10,000 shapes. We report two rates over generation attempts: **Build%**, the fraction of attempts that write a STEP file, and **Valid%**, the fraction that also pass the watertight-solid check of the Open CASCADE Technology (OCCT) kernel (Open Cascade SAS, 2026) (Appendix D). The gap between them separates failures to parse from failures to close. For distribution-level quality we report Coverage (COV), Minimum Matching Distance (MMD), and Jensen–Shannon Divergence (JSD) with the standard implementation (Xu et al., 2024). Uniqueness (Uniq%) is the fraction of valid solids that do not duplicate another generated solid (Table 5). Each evaluation uses 1,000 reference shapes from the test set and 3,000 generated shapes; we report the mean of 10 evaluations.

Point-cloud-conditioned generation. The input cloud consists of 4,096 points sampled uniformly by area from the surface of the ground-truth solid. A frozen Michelangelo encoder (Zhao et al., 2023b) maps the cloud to 257 prefix tokens (Figure 1a). **Valid%** is defined as above. Fidelity is measured by **CD**, the bidirectional Chamfer distance between points sampled on the generated and the ground-truth surfaces, and by **F-score**, the bidirectional F1 score averaged over five absolute distance thresholds.

4.2 UNCONDITIONAL GENERATION

Table 1 shows that GrowBrep turns 81.65% of its generation attempts into valid watertight solids, versus 48.98% for the strongest baseline, and writes a STEP file for 97.25% of attempts versus 74.37%. On the three distribution metrics it is comparable to the strongest baselines, with the best JSD and the second-best MMD and coverage. GrowBrep reaches these numbers with 2.3× fewer parameters than AutoBrep and 2.2× faster sampling per face. End to end it is slower, 407 versus 268 ms per face, because four-fifths of our per-face cost is the post-processing ladder, which spends CPU time to recover solids.

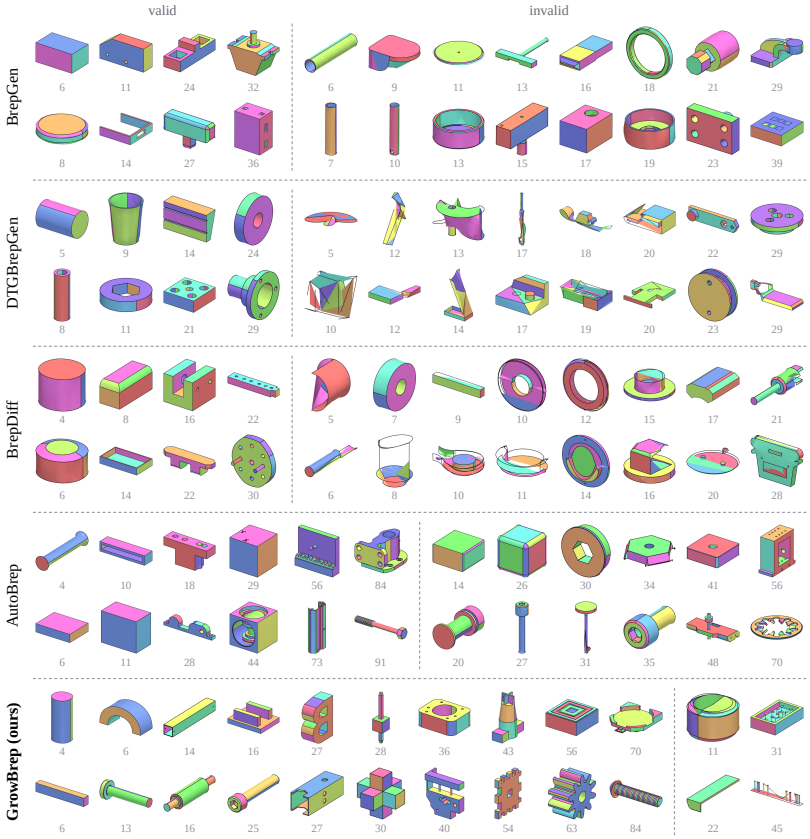

Figure 3: Unconditional samples, 24 per method in two rows: valid solids left of the dashed line and built-but-invalid samples (a STEP file is written but the validity check fails) right of it, split according to each method's validity rate in Table 1. Each side is sorted by ascending face count (shown below each sample).

Figure 3 shows valid and built-but-invalid samples of each method, sorted by face count. Across all samples, no valid solid of the three diffusion baselines has more than 50 faces, although BrepGen and DTGBrepGen were trained on our training set, which contains parts of up to 100 faces; AutoBrep and GrowBrep both produce valid parts with more than 80 faces. The valid GrowBrep samples include analytic and free-form surfaces, holes, inner loops, and tangent faces.

4.3 POINT-CLOUD-CONDITIONED GENERATION

Conditioned on point clouds, GrowBrep generates a valid solid for 80.0% of the 1,000 input clouds (Table 2), versus 51.6% for DualBrep, measured under the OCCT version it specifies on clouds from its published test list (Appendix D.1). On the clouds for which both methods produce geometry, GrowBrep's Chamfer distance is lower by 0.0023 on average, so the higher validity does not come at the expense of fidelity. Figure 4 shows generations next to the ground truth and to DualBrep, with

Table 2: Point-cloud-conditioned B-Rep generation (1,000 test clouds; CD and F-score are computed on the 501 clouds for which both methods produce geometry).

Method	Valid%↑	s/cloud↓ sampling	s/cloud↓ post-proc.	Fidelity CD↓	Fidelity F-score↑
DualBrep (Liu et al., 2026b)	51.6	41.5	794.4	0.0365	0.423
GrowBrep (ours), $K = 1$	80.0	**2.2**	**10.6**	0.0342	0.439
GrowBrep (ours), $K = 5$	**95.8**	11.1	53.0	**0.0335**	**0.443**

columns ordered by ground-truth complexity; on these shapes our reconstructions are more faithful, and the gap is widest on complex shapes.

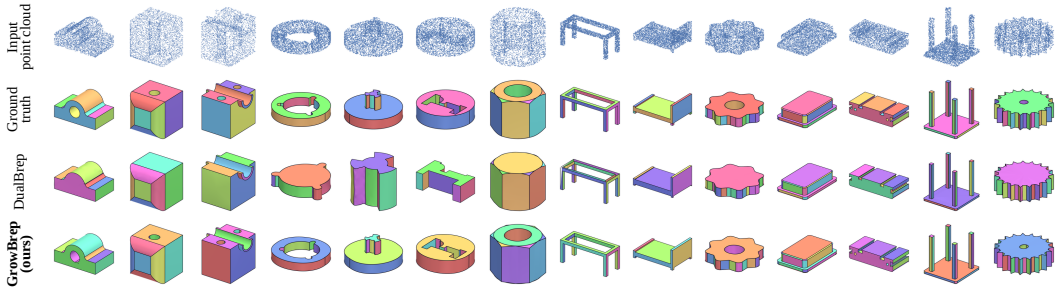

Figure 4: Point-cloud-conditioned generation. From top to bottom: input point cloud, ground truth, DualBrep, and GrowBrep; columns are sorted by ascending ground-truth face count.

Table 3: Best-of-K for GrowBrep, point-cloud-conditioned.

K	Build%↑	Valid%↑	CD↓	F-score↑
Test set (1,000 clouds)				
1	96.80	80.00	0.03589	0.4330
2	98.20	88.80	0.03558	0.4352
3	99.00	93.00	0.03529	0.4371
4	**99.10**	94.90	0.03492	0.4380
5	**99.10**	**95.80**	**0.03490**	**0.4382**
Hard set ($\geq$ 50 faces)				
1	82.63	32.34	0.06259	0.3776
2	86.23	48.50	0.05581	0.3888
3	87.13	56.59	0.05316	0.3962
4	88.02	60.18	0.05336	0.3955
5	**88.92**	**63.47**	**0.05308**	**0.3964**

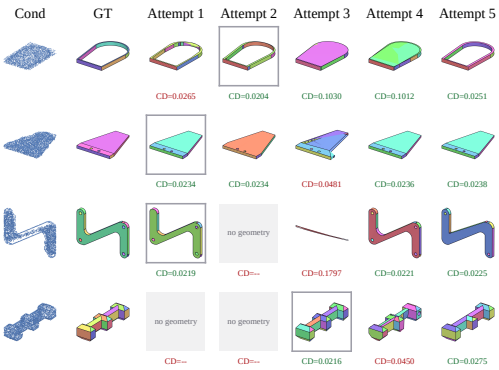

Figure 5: Five independent samples (columns Attempt 1–5) for each of four conditioning clouds (Cond) with their ground truth (GT). Below each sample is its CD, green for a valid solid and red for an invalid or missing one; the frame marks the first valid sample kept by best-of-K.

The difference in cost is even larger: a single sample costs 19× less to generate and 75× less to post-process than DualBrep, whose post-processing refits the faces under 24 axis-aligned rotations of the shape. Because a sample is cheap, validity can be raised by taking several. Keeping the first valid solid among K independent samples, a rule that needs no ground truth, raises validity from 80.0% to 95.8% at $K = 5$ on the full test set and from 32.3% to 63.5% on a separate hard set of shapes with at least 50 faces (Table 3), without loss of fidelity and still at 13× lower total cost than DualBrep. The five samples in Figure 5 agree on the overall shape and differ in local topology; on the hard set at least one is valid far more often than a single one, which indicates that failures are largely independent across samples.

4.4 Ablations

Table 4: Ablations of grammar-constrained sampling and the validity ladder.

(a) Grammar-constrained sampling

Sampling constraints	Build%↑	Valid%↑	Mean tokens
Syntactic mask	91.53	85.80	1,316
+ **construction constraints (ours)**	**98.30**$^{+6.77}$	**89.70**$^{+3.90}$	898

(b) Post-processing validity ladder

Configuration	Build%↑	Valid%↑	Post-proc. (s)
Analytic primitives	99.41	84.17	**3.95**
+ surface reconciliation	99.16$^{-0.25}$	85.68$^{+1.51}$	5.65
+ B-spline refit	98.74$^{-0.42}$	86.26$^{+0.58}$	8.38
+ **boundary snap (ours)**	98.41$^{-0.33}$	**87.69**$^{+1.43}$	9.30

Grammar-constrained sampling. Table 4(a) compares two levels of masking. The syntactic mask, standard practice for token-by-token generation of structured sequences (Nash et al., 2020; Jayaraman et al., 2023), guarantees only that the stream parses into faces, loops, and half-edges; the construction constraints add the grammar rules of Section 3.4. They raise the build rate from 91.53% to 98.30% and the validity rate from 85.80% to 89.70%, and shorten the mean sequence from 1,316 to 898 tokens, so more sequences terminate within the token budget and more of those that terminate close into solids. Without any mask almost no sequence parses: a small probability mass falls on illegal tokens at every step, and over roughly a thousand tokens an error is almost certain.

Validity ladder. Switching on the rungs of Table 4(b) one at a time, in the deployed order, lifts validity from 84.17% with analytic primitives alone to 87.69%, while post-processing time grows from 3.95 to 9.30 seconds and the build rate falls by one point, because repair leaves some samples unexportable. The repair rungs add only 3.52 percentage points in total, so the validity comes mainly from the model and the grammar, not from repair.

Complexity prompting. Under the same complexity prompt, GrowBrep's parts are slightly more complex than AutoBrep's (median face counts 8, 32, and 62 versus 6, 30, and 59) and more often valid in every bucket (95.00, 66.73, and 40.33% versus 76.29, 46.59, and 34.72%; Appendix A.2).

Sampling settings. Appendix A.1 sweeps temperature and top-p around the defaults. Unconditionally, a lower temperature raises validity at the cost of diversity, because sampling concentrates on simple shapes. Under point-cloud conditioning, a slightly lower temperature or tighter top-p improves both validity and fidelity, so the conditioned numbers of Section 4.3, taken at the untuned default, are conservative.

5 Conclusion

We presented GrowBrep, an autoregressive B-Rep generator that builds closure into its sequence grammar: a solid grows face by face along a frontier of open edges, and every sequence allowed to complete describes a closed, consistently oriented shell. Under the same strict checker, 81.7% of GrowBrep's attempts on ABC-1M yield valid solids, versus 49.0% for the strongest baseline. On the distribution-level metrics it is comparable to the strongest baselines. Conditioned on point clouds, it reaches 80.0% validity versus 51.6% for the closest prior method at 65× lower cost per cloud, and multiple independent samples raise validity to 95.8%. The next steps are to reduce the geometric error that causes most remaining failures (Appendix E) and to use the low sampling cost for post-training on validity feedback.

LLM USAGE STATEMENT

Large language models were used to polish the language of this paper. All experimental design, results, and conclusions are the authors' own and were verified by them.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. ComplexGen: CAD reconstruction by B-Rep chain complex generation. *ACM Transactions on Graphics*, 41(4):129:1–129:18, 2022.

Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. SolidGen: An autoregressive model for direct B-rep synthesis. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=ZR2CDgADRo.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating sequential CAD designs from beginner-to-expert level text prompts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 7552–7579, 2024. URL https://openreview.net/forum?id=5k9XeHIK3L.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9593–9603, 2019.

Mingi Lee, Dongsu Zhang, Clément Jambon, and Young Min Kim. BrepDiff: Single-stage B-rep diffusion model. In *ACM SIGGRAPH 2025 Conference Papers*. Association for Computing Machinery, 2025. URL https://doi.org/10.1145/3721238.3730698. Article 103, 11 pages.

Jiahao Li, Yunpeng Bai, Yongkang Dai, Hao Guo, Hongping Gan, and Yilei Shi. AutoRegressive generation with B-rep holistic token sequence representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24363–24372, 2026a.

Jing Li, Yihang Fu, and Falai Chen. DTGBrepGen: A novel B-rep generative model through decoupling topology and geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21438–21447, 2025a.

Jing Li, Yihang Fu, and Falai Chen. BrepForge: Factorized B-rep synthesis via wireframe composition and boundary-conditioned surface instantiation. In *ACM SIGGRAPH 2026 Conference Papers*. Association for Computing Machinery, 2026b. URL https://doi.org/10.1145/3799902.3811207. Article 18, 11 pages.

Pu Li, Wenhao Zhang, Jinglu Chen, and Dong-Ming Yan. Stitch-A-Shape: Bottom-up learning for B-Rep generation. In *ACM SIGGRAPH 2025 Conference Papers*. Association for Computing Machinery, 2025b. URL https://doi.org/10.1145/3721238.3730661. Article 104.

Pu Li, Wenhao Zhang, Weize Quan, Biao Zhang, Peter Wonka, and Dong-Ming Yan. BrepGPT: Autoregressive B-rep generation with Voronoi half-patch. *ACM Transactions on Graphics*, 44(6): 226:1–226:18, 2025c.

Shurui Liu, Weide Chen, and Ancong Wu. HiDiGen: Hierarchical diffusion for B-Rep generation with explicit topological constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Yilin Liu, Duoteng Xu, Xingyao Yu, Xiang Xu, Daniel Cohen-Or, Hao Zhang, and Hui Huang. HoLa: B-Rep generation using a holistic latent representation. *ACM Transactions on Graphics*, 44(4):1–25, 2025.

Yilin Liu, Pradeep Jayaraman, Chinthala Reddy, Xiang Xu, and Hooman Shayani. DualBrep: A dual-field continuous representation for B-rep modelling. In *ACM SIGGRAPH 2026 Conference Papers*. Association for Computing Machinery, 2026b. URL https://doi.org/10.1145/3799902.3811057.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Junran Lu, Yuanqi Li, Hengji Li, Jie Guo, and Yanwen Guo. Flatten the complex: Joint B-Rep generation via compositional k-cell particles. In *ACM SIGGRAPH 2026 Conference Papers*. Association for Computing Machinery, 2026. URL https://doi.org/10.1145/3799902.3811197.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations (ICLR)*, 2024.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. PolyGen: An autoregressive generative model of 3D meshes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Open Cascade SAS. Open CASCADE Technology (OCCT). https://github.com/Open-Cascade-SAS/OCCT, 2026. URL https://github.com/Open-Cascade-SAS/OCCT. Version 7.9.3. Accessed 2026-08-11.

Dafei Qin, Rui Xu, Zeyu Shen, Kaichun Qiao, Hongyang Lin, Qixuan Zhang, Huaijin Pi, Lan Xu, Jingyi Yu, Wenping Wang, and Taku Komura. Autoregressive B-Rep shape generation with parametric surfaces. In *ACM SIGGRAPH 2026 Conference Papers*. Association for Computing Machinery, 2026. URL https://doi.org/10.1145/3799902.3811040.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-Recode: Reverse engineering CAD code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9801–9811, 2025.

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P. Adams. Vitruvion: A generative model of parametric CAD sketches. In *International Conference on Learning Representations (ICLR)*, 2022.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

MingZe Sun, Haiyong Jiang, Bingchen Yang, Haoxuan Song, Yidi Li, Jun Xiao, and Peter Wonka. TG-Diff: Coupling discrete topology diffusion and topology-conditioned geometry diffusions for B-Rep generation. *arXiv preprint arXiv:2607.21928*, 2026.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6752–6762, 2021.

Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

Xiang Xu, Joseph G. Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl D. D. Willis, and Yasutaka Furukawa. BrepGen: A B-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics*, 43(4):1–14, 2024.

Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Yilin Liu, Durvesh Malpure, and Pete Meltzer. AutoBrep: Autoregressive B-Rep generation with unified topology and geometry. In *ACM SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12. Association for Computing Machinery, 2025. URL https://doi.org/10.1145/3757377.3763814.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12): 3848–3860, 2023a.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A Additional results

A.1 Sampling sensitivity

Table 5: Sampling sensitivity of unconditional generation.

Setting	Build%↑	Valid%↑	COV%↑	MMD↓ ($\times 10^2$)	JSD↓ ($\times 10^2$)	Uniq%↑
(a) Temperature, top-p fixed at $p = 0.9$						
$T = 0.8$	**99.71**	**96.03**	40.01	1.883	5.282	93.79
$T = 0.9$	98.59	93.10	45.08	1.801	2.592	97.35
$T = 1.0$ **(default)**	97.50	82.69	53.21	1.661	1.537	98.35
$T = 1.1$	93.76	72.21	**57.58**	**1.593**	**1.295**	**98.76**
(b) Top-p, temperature fixed at $T = 1.0$						
$p = 0.9$ **(default)**	**97.50**	**82.69**	53.21	1.661	1.537	98.35
$p = 0.95$	95.75	78.70	55.86	1.644	1.380	**98.99**
$p = 1.0$	87.02	62.98	**57.50**	**1.583**	**1.295**	98.74

In Tables 5 and 6 we sweep temperature and top-p one at a time around the defaults $T = 1.0$ and top-$p = 0.9$, for unconditional and point-cloud-conditioned generation, respectively. In the unconditional setting, validity and diversity trade off. Lowering the temperature to $T = 0.8$ pushes validity from 82.69% to 96.03%, but coverage falls from 53.21 to 40.01, JSD rises from 1.537 to 5.282, and uniqueness drops from 98.35% to 93.79%. Raising the temperature or widening top-p does the opposite; $T = 1.1$ and top-$p = 1.0$ give the best distribution metrics, with validity down to 72.21% and 62.98%. The extra validity comes from simpler shapes: a lower temperature concentrates sampling on high-probability simple shapes, and at $T = 0.8$ the median generated face count drops from 14 to 6. Such shapes are easier to close, which raises validity, but the generated set collapses onto them. Under point-cloud conditioning the behavior differs, and the default is not the best setting in the sweep. Lowering the temperature to $T = 0.9$ raises validity from 79.33% to

Table 6: Sampling sensitivity of point-cloud-conditioned generation.

Setting	Build%↑	Valid%↑	CD↓	F-score↑
(a) Temperature, top-p fixed at $p = 0.9$				
$T = 0.8$	**97.33**	83.67	0.03697	0.4354
$T = 0.9$	95.33	**84.33**	**0.03617**	**0.4415**
$T = 1.0$ **(default)**	95.67	79.33	0.03754	0.4359
$T = 1.1$	91.00	78.33	0.03918	0.4343
(b) Top-p, temperature fixed at $T = 1.0$				
$p = 0.8$	**98.33**	**86.67**	**0.03713**	**0.4377**
$p = 0.9$ **(default)**	95.67	79.33	0.03754	0.4359
$p = 1.0$	86.00	74.33	0.03834	0.4327

84.33% and gives the best CD (0.03617) and F-score (0.4415) in that panel. Tightening top-p to 0.8 raises validity to 86.67% and the build rate from 95.67% to 98.33%, with CD and F-score improving as well. A slightly lower temperature or tighter top-p improves both validity and fidelity in the conditioned setting, so the numbers in Section 4.3, taken at the untuned default, are conservative.

We nevertheless keep the default for every reported number, for two reasons. First, sharpening the sampling distribution is not free. The point-cloud protocol has no distribution-level metric: CD and F-score compare each sample with its own target cloud and cannot detect a loss of diversity, so unchanged or better fidelity is no evidence that nothing is lost. In the unconditional setting, where distribution metrics exist, the corresponding sharpening ($T = 0.8$) raises JSD by a factor of 3.4 and gains its validity from simpler shapes. Second, reporting every number at one untuned default keeps the comparison with the baselines free of sampler tuning on our side; we prefer this to a validity gain found by tuning the sampler. Both sweeps use their own sample pools, so their default rows differ from Tables 1 and 2.

A.2 Complexity prompting

Table 7: Validity and generated face count under each complexity prompt, AutoBrep versus GrowBrep.

Complexity	AutoBrep			GrowBrep		
	Median $\|F\|$	Mean $\|F\|$	Valid%↑	Median $\|F\|$	Mean $\|F\|$	Valid%↑
EASY (< 25)	6	9.4	76.29	8	11.5	**95.00**
MID (25–49)	30	33.3	46.59	32	40.6	**66.73**
HARD (≥ 50)	59	57.4	34.72	62	68.2	**40.33**

The complexity token of a solid is `EASY` if $|F| < 25$, `MID` if $25 \leq |F| < 50$, and `HARD` otherwise. In Table 7 we prompt both models with each of the three tokens and report the median and mean face count each prompt produces, together with the validity rate. AutoBrep's released model accepts the same three prompts. The prompt does shift how complex the generated parts are. Our median face counts are 8, 32, and 62 (14 without a prompt), and AutoBrep's are 6, 30, and 59, each 2–3 faces lower, so the buckets are matched by prompt, not by produced face count. Under the same prompt GrowBrep's validity is higher in every bucket, 95.00%, 66.73%, and 40.33% versus 76.29%, 46.59%, and 34.72% for AutoBrep. Validity falls with complexity for both models; the token steers the complexity of the output and does not itself raise validity.

B Implementation details

B.1 Model and codec

The sequence model is a GPT-style decoder-only transformer (Brown et al., 2020) with 24 layers, width 1,024, a SwiGLU feed-forward block (Shazeer, 2020) of width 4,096, and 16 query heads that share 4 key/value heads of dimension 64. Positions are encoded with rotary embeddings (Su

et al., 2024) of base 10^4, dropout 0.1 is applied on the embedding and feed-forward paths, and the maximum sequence length is $8{,}192$ tokens, including the point-cloud prefix. The model has 406 M parameters. The geometry codec has the same architecture as AutoBrep's (Xu et al., 2025). Its face and edge decoders, which turn geometry tokens into face grids and edge curves, add 47 M parameters, so Table 1 lists 453 M in total. For point-cloud conditioning, the frozen Michelangelo shape encoder (Zhao et al., 2023b) maps the input cloud to a fixed block of 257 latent tokens of width 768, which are projected to the model width and prepended to the sequence as a prefix.

B.2 Token costs and caps

Each FSQ token takes one of $1{,}000$ codes ($8 \cdot 5 \cdot 5 \cdot 5$), and each bounding-box token one of $1{,}024$ levels. Opening a new edge costs 11 tokens: `NEW_EDGE`, 6 bounding-box and 2 FSQ tokens, `DIR`, and `LABEL`. Closing an open edge with `REUSE` costs 1 token. A face costs 11 tokens for its opening token and 10 geometry tokens, plus `F_END` and the delimiters of its loops. Measured on generated sequences, the amortized cost is 44.1 tokens per face for unconditional generation and 43.8 under point-cloud conditioning.

The representation is bounded by $N = 512$ primitives per solid (vertices, edges, and faces together), 64 edges and 64 loops per face, and 68 half-edges per loop. During sampling the mask enforces the last two caps: once a loop holds 68 half-edges, only `L_END` remains permitted, and once a face holds 64 loops, only `F_END` remains.

Edge cap. A closed, connected shell whose faces are simply connected satisfies the Euler relation $|V|-|E|+|F| = 2-2g$ with genus $g \geq 0$, so it consists of $|V|+|E|+|F| = 2|E|+2-2g \leq 2|E|+2$ primitives. Requiring $2|E| + 2 \leq N$ gives $E_{\max} = \lfloor (N-2)/2 \rfloor = 255$ for $N = 512$. By the Euler–Poincaré formula, each inner loop adds one to the right-hand side of the Euler relation, so a shell with holes in its faces can exceed the budget at $E_{\max}$ edges; such a sequence fails at parse time and counts as a failed attempt.

B.3 Growth order

We seed from the face with the fewest edges. The candidates at each step are the unplaced faces whose outer loop touches the frontier; among them we place the one that leaves the smallest frontier and break ties in favor of the face that closes the longest-open edge. If no candidate exists, which can happen on shapes with holes, the traversal restarts from the same seed in breadth-first order, so every solid that breadth-first growth can serialize is still serialized.

B.4 Post-processing

Vertices are positioned at the decoded curve endpoints, and any offset between endpoint positions is a geometric error left to post-processing. The validity ladder first builds the solid with analytic primitives, then applies surface reconciliation, which rigidly pulls each face's surface toward its incident edge curves, and then refits all faces as B-splines. The first result that passes the strict validity check (Appendix D) is returned. If none passes, a final boundary snap moves the boundary samples of each face grid onto its incident edges and rebuilds. Sewing and shape-healing tolerances are not absolute; they scale with the extent of the decoded shape, which need not fill the normalized cube.

B.5 Training and inference

We train the sequence model with AdamW (Loshchilov & Hutter, 2019), weight decay 0.05, and $\beta = (0.9, 0.95)$, for $300{,}000$ steps at a global batch of 64 sequences (16 per GPU on 4 GPUs). The learning rate warms up linearly over $1{,}000$ steps to a peak of 2×10^{-4} and then decays in a single step to one tenth of its peak after two thirds of the steps, instead of following a cosine schedule. The cross-entropy on structural (non-geometry) tokens, i.e., the grammar tokens, `DIR`, and `REUSE`, is weighted by 2, and we add a z-loss (Zoph et al., 2022) with weight 10^{-4}. Training uses fully sharded data parallelism (Zhao et al., 2023a) with bf16 mixed precision on NVIDIA GPUs. The data-side choices, complexity rebalancing and rotation augmentation, are described in Appendix C.

At inference we sample with temperature $T = 1.0$ and nucleus sampling with top-$p = 0.9$, without classifier-free guidance, under a total sequence budget of $8{,}192$ tokens. The prompt is `BOS` followed by a complexity token, which is `UNCOND` unless a complexity bucket is requested (Appendix A.2).

C Dataset

Corpus. We prepare the ABC-1M split (Koch et al., 2019; Xu et al., 2025). Each solid is converted into a fixed-size tensor representation: a 32×32 grid of points sampled from each face's parametric surface together with a per-cell trim mask, 32 points sampled along each edge, a position for each vertex, and the full loop and half-edge incidence. A solid is accepted if it has at most 100 faces, at most 400 primitives in total, at most 64 edges per face, and at most 68 half-edges per loop. The converted training split contains $492{,}820$ solids and the validation split $8{,}707$.

Filtering. Following AutoBrep (Xu et al., 2025), we reject solids with a face that has no edge, with an edge that does not bound exactly two faces, and with a tiny edge or face, that is, one whose bounding box is smaller than $1/1{,}024$ of the solid's largest axis extent along all three axes. We also reject solids with an edge whose two endpoints are the same vertex and solids with a face whose trim mask is empty.

Complexity rebalancing. Simple parts dominate the corpus, and a model trained on the raw distribution underfits the long tail of complex parts. Following AutoBrep (Xu et al., 2025), we downsample simple shapes during training: a solid with fewer than 25 faces is kept with probability 0.25 each time the data loader reaches it, so simple shapes are seen less often but are not removed. Validation and evaluation data are never downsampled.

Augmentation. After normalization to $[-1, 1]^3$, each training solid is serialized four times: once as is and once for each quarter turn about the z axis ($90°$, $180°$, and $270°$). These rotations map the cube to itself and preserve the axis alignment typical of mechanical CAD; we do not use arbitrary rotations. After each rotation the face grids and edge orientations are re-canonicalized before serialization, because the canonical form is defined geometrically and a rotation without re-canonicalization would silently change the training target.

D Metrics and evaluation protocol

Unit of comparison. Every method writes STEP files, so the STEP file is the unit of comparison: we load it, mesh it, sample its surface, and score it. Scoring a method's internal representation instead would evaluate each method on a different object. The baseline numbers we report are therefore our own measurements, not the values published in the respective papers.

Validity. A shape is valid if `BRepCheck_Analyzer` accepts it at the shape's native tolerances, it contains at least one `TopAbs_SOLID`, and no `TopAbs_FACE` or `TopAbs_SHELL` lies outside a solid. The last two conditions make this a watertightness check: `BRepCheck` alone verifies that each sub-shape is well formed but not that the shell is closed, so a set of clean but entirely unsewn faces passes it. We apply the same predicate and tolerances to every method. Each STEP file is scored in a forked subprocess that is killed after a wall-clock budget, so a hang in the CAD kernel can neither stall the evaluation nor silently raise any method's rate.

D.1 Point-cloud-conditioned protocol

DualBrep's post-processing relies on an OCCT repair routine whose behavior differs across OCCT versions, so DualBrep's validity rate depends on the installed kernel. We installed the version DualBrep specifies, took the $1{,}000$ test clouds from its published test list, and measured the 51.6% of Table 2 in that environment. GrowBrep's post-processing does not use this routine. DualBrep's post-processing refits the faces under each of 24 axis-aligned rotations of the shape and keeps the first that sews into a solid.

E LIMITATIONS

Our guarantee stops at the combinatorial level, and apart from the attempts that fail to parse (Section 4.1), the remaining failures are geometric: acceptance as a watertight solid still depends on how accurately surfaces and curves are fitted. Because the grammar describes a single connected shell, solids with internal voids or non-manifold edges lie outside the grammar and cannot be serialized. Figure 6 shows three classes of failure. In the first, the fitted surface of a face departs from the edge curves that bound it, sewing fails, and the shell is left with an opening. In the second, the shell sews into a closed solid with pairwise opposite half-edge orientations, as the grammar guarantees, but the boundary curve of a face self-intersects on its surface or crosses another boundary of the same face. The trimmed region is then invalid and the kernel rejects the face. This class is the most common. In the third, the solid passes all checks yet is of poor quality. Passing the watertight check only says that the shell is closed; it does not rule out intersections between faces, degenerate sliver faces, or shapes that are hard to edit further in CAD. This class is most common among complex shapes.

The validity gap to the baselines in Table 1 reflects the serialization, the model, and the training together; we have not compared the growth order with an alternative layout on the same model.

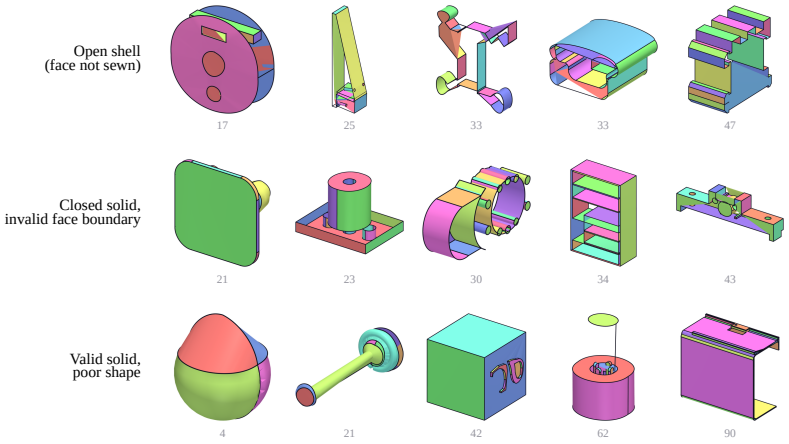

Figure 6: Failure cases of GrowBrep, one class per row: open shell (a face not sewn), closed solid with an invalid face boundary, and valid solid of poor shape.