MCAD-BENCH: BENCHMARKING CAD EDITING ON REAL SUPPLIER PARTS WITH A CONTROLLED ANSWER CHANNEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Benchmarks that ask a language model to produce CAD almost always require the answer as a program in a geometry-kernel API and then score the solid that program builds. We show that this choice is part of the measurement, not an implementation detail. CAD systems share features but not syntax, so no canonical API settles the matter, and a representation that moves the result should be controlled rather than ranked. MCAD-Bench is built from 144 real supplier products stored as STEP, with three task families (reconstruction, instruction-guided editing by defect injection, and parameter-specified editing) on single-part and assembly tracks. Alongside the code channel we provide a structured feature-op *answer channel*, in which the model emits an operation and its parameters and we apply verified primitives. Instruction-guided editing has two tiers, one that gives the defect's location and one in which the model must find it. Over five models, editing success in code spans up to 39 points, mostly through executability, and the structured channel narrows this to at most 8 in either tier. A model whose code runs on only 7% of instances when it must find the defect reaches frontier-level accuracy in the channel. The narrow band does not mean the models have converged. Many instances lie beyond what the channel can express and many are passed by every model, so fewer than 50 of the 301 single-part instances separate any two models in either tier, against at least 155 in code. Where the channel can express the repair, the two frontier models are indistinguishable when told the location and differ significantly when they must find it. Assemblies remain limited by the models, with wide spreads and no model near the channel's ceiling. We also report *achievability oracles*, a two-arm protocol run before any near-zero result is published. On our own pipeline they caught results that looked like model findings but were properties of the benchmark, including thirteen instances on which the metric rejects its own reference and 28 on which it accepts the unedited input.

1 INTRODUCTION

Language models are being put to work inside CAD software, where an assistant must not merely describe a change to a part but carry it out. Evaluating that requires deciding what the model is shown, how it may answer, and how the answer is scored. The second decision is usually made silently. Nearly every recent CAD benchmark asks for a program in a geometry-kernel API (CadQuery, OpenSCAD, an OpenCascade binding) and scores the solid it builds (Wang et al., 2026; Doris et al., 2026; Yang et al., 2026; Zhang et al., 2026). The channel is then inside the measurement, and a model that knows the shape but not the constructor is indistinguishable from one that knows neither.

To measure how much of a CAD score comes from knowing the constructor, we removed the constructor as an obstacle. Two frontier models were given a fixed reference for the kernel API, showing how to read a face and giving a generic recipe for each kind of kernel call a correction needs, but naming no instance's target, operation or parameter (Table 15). Paired with their own answers on the 301 editing instances of Sec. 5, Claude Opus 4.8 goes from 39% to 53% edit success and GPT-5.6-sol from 42% to 49%, mostly because more answers run. The model must still locate the defect,

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.

choose the operation and set its parameters, so 7 to 14 points of a code score are knowledge of one kernel's API.

The fix is not a better API, because there is no canonical one. P3D-Bench runs the same models through JSON, OpenSCAD, CadQuery and Three.js, finds that the format itself changes the result, and reports which format scores best (Yang et al., 2026). We draw a different conclusion. A representation that moves the result should be controlled, not ranked. Industry practice does not settle it either, since SolidWorks, Onshape, CATIA and FreeCAD each expose their own API while sharing the same features, from holes and fillets to their removal. The channel also determines which data a benchmark can use. Editing a part by program presupposes a program that builds it, and industrial CAD does not ship that way, since a supplier part arrives as a STEP solid with no feature history. A code-channel benchmark is therefore either confined to corpora generated by programs, the toy-geometry limitation the field has already noted (Kimmel et al., 2025; Sharma et al., 2018), or it burdens the model with reaching the topology through the API. A structured feature-op channel avoids both, which is what lets this benchmark use 144 real supplier products with manufacturing features, multi-solid assemblies and published drawings (Figure 2).

Controlling the channel changes the panel. Over five models at temperature 0 on 301 certified instances, the structured channel brings every model to 77–86% executable and places every score in a 43–52% band, against 3–42% in code at the same tiers. Our second contribution is to show that this flatness is not convergence. A structured channel imposes a ceiling and saturates below it. When the location is given, every model fails 136 of the 301 instances (117 of them beyond the ceiling) and passes 140, and only 25 separate any two models, against 155 in code (49 and 190 when the model must find the defect). On the channel-feasible subset the two frontier models are indistinguishable when told where the defect is and significantly different when they must find it.

Our third contribution is methodological. A near-zero score has at least four possible causes. The metric may not see the property, the instance may be unanswerable, the channel may be unable to express the correction, or the model may have failed. The score alone cannot tell these apart, so before reporting any such axis we run two *achievability oracles*, a PERFECT arm that submits the exact answer and a CHANNEL arm that expresses it in the answer channel. On our own pipeline the protocol repeatedly caught results that looked like findings about models but were properties of the benchmark. Among them were an ill-posed defect axis, a metric that failed perfect submissions because concentric solids share a centroid, and editing instances on which the metric rejects its own reference (thirteen) or accepts the unedited input (28).

Contributions. (i) **MCAD-Bench**, with 150 real supplier products (144 active) plus 149 products with published parametric variants, all in STEP, and three task families routed to single-part and assembly tracks by *measured* separability. (ii) **Geometry-only scoring with a measured noise floor**. The scorer never executes model-produced code, injected defects are admitted only above an empirically measured metric floor, and the editing score is two-sided and symmetric with the injection gate. (iii) **A controlled, backend-agnostic answer channel** whose fairness is measured, not assumed. A per-operator fidelity oracle shows where the channel falls *below* code and keeps a ceiling-flattened panel from being read as convergence, and an API-reference control measures how much of a code score is knowledge of the kernel API. (iv) **Achievability oracles**, a reusable protocol, together with the defects it caught in our own pipeline.

2 Related Work

Multimodal CAD generation and benchmarking. Early work learned and synthesized CAD geometry as CSG programs, construction sequences and boundary representations (Wu et al., 2021; Sharma et al., 2018; Jayaraman et al., 2021; 2023; Xu et al., 2024), drawing on large collections such as ABC (Koch et al., 2019). The Fusion 360 Gallery (Willis et al., 2021b) and SketchGraphs (Seff et al., 2020) added programmatic and relational constraints but stay at part level or in 2D. Language and vision-language models moved the field toward multimodal generation, translating text or images into executable scripts, evaluated by Text2CAD-Bench (Wang et al., 2026), CADBench (Doris et al., 2026), P3D-Bench (Yang et al., 2026) and BenchCAD (Zhang et al., 2026), with concurrent work raising success rates by multi-agent reasoning (Barkley et al., 2026; Alrashedy et al., 2025). These still rest largely on procedural shapes or everyday-object models (Sharma et al., 2018; Chang et al., 2015), lacking manufacturing features, multi-solid assemblies with physical contact

(Willis et al., 2022) and parametric families, and purely visual shapes are insufficient for machining (Kimmel et al., 2025).

Instruction-guided CAD editing and evaluation. Practical CAD workflows (Du et al., 2018) require editing as much as synthesis, including structural shape variation and macro-operation editing (Mo et al., 2020; Jones et al., 2021). Modifying a CAD script demands structural reasoning that alters target parameters without breaking geometric constraints or invalidating downstream assemblies (Lambourne et al., 2021; Yu et al., 2022; Willis et al., 2021a), and while sequential command execution (Ganin et al., 2021; Xu et al., 2022) shows the value of programmatic representations, rigorous evaluation of instruction-guided editing remains rare. Benchmarks tie evaluation to a single scripting backend, mixing geometric reasoning with recall of one kernel's API, while global 3D metrics and compilability checks cannot capture precise topological edits. MCAD-Bench isolates geometric reasoning from API syntax recall with a backend-agnostic edit DSL, and scores STEP two-sidedly so that correct local changes are rewarded and collateral damage penalized. The question is not confined to generalist models. CAD-Llama (Li et al., 2025), CADFusion (Wang et al., 2025) and Text-to-CadQuery (Xie & Ju, 2025) fine-tune on a single CAD representation, and a controlled channel is how one would ask whether such a model learned geometry or one vendor's syntax.

3 The MCAD-Bench Corpus

3.1 Provenance and Preprocessing

MCAD-Bench is built from 3DSource (`lib.3dsource.cn`), a part library aggregating supplier-authored CAD for industrial components, sampled uniformly across its category taxonomy rather than in proportion to catalogue volume. The sample covers 150 products in nine categories and 74 component types. Data use was negotiated with the library operator.

Each product supplies a STEP model, catalogue imagery, a parameter table, and an engineering drawing where one is published. Preprocessing classifies the assets, reads and unit-normalizes the STEP, extracts a B-rep dimension inventory, renders the visual channel and applies quality control, writing a content-addressed manifest keyed by SHA-256 so no stage can silently consume a modified input. Six products whose STEP is unreadable, absent or not in millimetres are excluded, leaving 144 active entries. Whether an entry is a part or an assembly is decided by decomposing its solid, not by the catalogue (Table 6). Fused assemblies route to the single-part track even where the catalogue calls them assemblies, because no component correspondence can be recovered from them, while loose multi-solids are matched geometrically and structured assemblies by component name.

Complexity and modality. Entries span 4.4 mm to 3.0 m (median diagonal 113 mm) and reach 4019 faces and 83 solids, with no duplicates. Every entry's isometric render and three-view projection is generated from the STEP solid under a fixed camera, so what a model sees is a deterministic function of the geometry it is scored against rather than supplier photography. Models also receive a dimension inventory read from the ground-truth B-rep (Appendix A).

3.2 The Parametric Variant Set

A second collection covers 149 products for which the library publishes multiple STEP variants, each with the parameter vector that generated it. Deduplication is on *geometry* rather than file hashes, since STEP headers embed export timestamps. After deduplication the set holds 971 distinct shapes and 3537 variant pairs, from which we evaluate a stratified sample of 217 pairs over 91 products (Appendix A). We admit the set on measured sufficiency rather than on the assumption that a parameter row determines a shape, and 127 of 149 products prove fully injective.

A test-only benchmark. MCAD-Bench defines no training split, and the parametric variants share products with the main corpus, so any few-shot exemplar must come from a different product than the test item. The catalogue is public, so pretraining exposure cannot be excluded by construction. We calibrate against it with a same-product retrieval baseline (Sec. 4.4).

4 BENCHMARK DESIGN

4.1 TASKS

Three families run over one pool, routed to single-part and assembly tracks.

Reconstruction. Rebuild the reference solid R from the generated isometric render, three views and dimension inventory, over the 85 single-part items. We do not pose the 59 assembly entries as reconstruction, since we do not consider multi-part reconstruction from renders well posed. Non-executable output scores zero rather than being filtered.

Instruction-guided editing. This family is the paper's main instrument. We take a real part R from the corpus and programmatically inject exactly one localized defect, such as cutting a hole that should not be there, rounding an edge that should be sharp, deleting a bore or displacing one component of an assembly. The result is P. The model is given P and a one-sentence instruction naming the correction, and must return the repaired solid, which is scored against R. Because *we* made the defect, its exact inverse is known before any model runs. This is what makes a localized two-sided metric, an admission gate and an achievability oracle possible at all. Six operators produce the 342 single-part instances (adding, removing or resizing a hole, adding a fillet or chamfer, and removing a fillet) and three produce the 127 assembly instances (displacing a component, breaking a contact, deleting a member of a symmetry group). Per-operator counts are in Appendix C.

Each instance is posed at three *tiers*, differing only in how much of *where* is given away. *Pointed* states the operation, location and parameters, so in principle only execution is under test. *Selection* withholds coordinates but hands over the faces a user would have clicked, the interface a CAD operator has. This tier ships with the benchmark but is not evaluated here. Our evaluation budget ran out before its arm, scheduled last, returned any answer. *Detection* gives natural language alone, so the model must first find the defect in P itself. The `remove_*` operators are the exception. Their feature is absent from P, so they carry a coarse octant descriptor to keep the task well posed. Localization cost is then pointed minus detection, which is clean in the structured channel but not in code (Sec. 5).

Parameter-specified editing. A source variant and a target parameter row, with the target variant as reference, over 217 evaluated pairs. This is a different regime from injection editing, because most parameter changes are not localized (the median area-weighted preserved-surface fraction is 0.39 for single-parameter pairs). We therefore evaluate it as *spec-conditioned generation* scored globally, which without the source solid is text-to-CAD over real supplier specifications (Sec. 5.3), and report the source-given condition separately (Appendix H). It needs a symbol legend, because 59% of specification key–value pairs are bare drawing symbols, such as `V=108` or `K=135`, whose meaning the catalogue gives only on its drawing. We induce one with no annotator and validate it against a permutation null (Appendix A).

4.2 METRICS

Every score is computed from geometry alone, and no model judges any output at any tier. The scorer ingests STEP and never model-produced code. Execution happens once, in a separate process under a wall-clock timeout, and scoring imports nothing the model wrote, which makes this a security boundary as well as a methodological choice.

Write C for a candidate solid, R for the reference, d for R's bounding-box diagonal, and $P_C, P_R \subset \mathbb{R}^3$ for area-weighted surface samples of N points each. Two STEP files of the same solid do not read as identical, and that residual, the *noise floor* of $0.00363\,d$ for Chamfer, is the sampling spacing itself. It is the unit in which an injected defect and an editing error are measured, and Appendix B gives its measurement and the sampling configuration it depends on.

Single-part. The primary distance is the symmetric L_2-mean Chamfer distance $\mathrm{CD}(C, R)$ between P_C and P_R, reported in mm and, scale-free, as CD/d. Overlap is a voxel IoU computed in *one shared frame* anchored on R's bounding box. Both solids are voxelized on the same G^3 grid by an exact solid classifier, and candidate mass outside the box is clipped and counted as unmatched,

because normalizing each solid into its own unit cube scores a half-size copy as perfect. A single IoU threshold is an arbitrary success line, so we report a curve and summarize it by its normalized area, $\mathrm{mIoU} = \frac{1}{\tau_{\max}-\tau_{\min}} \sum_j (\tau_{j+1} - \tau_j) \frac{\mathrm{Pass@IoU}(\tau_j)+\mathrm{Pass@IoU}(\tau_{j+1})}{2}$, the trapezoidal area over the finite sweep $\mathcal{T} = \{\tau_j\}$ divided by its span, with $\mathrm{Pass@IoU}(\tau) = \frac{1}{n}|\{i : \mathrm{IoU}_i \geq \tau\}|$ over all n attempted items, with non-executing items counted as failures at every τ so the executable rate is absorbed by construction. Over a band spanning $[0, 1]$ this area tends to the mean IoU as the sweep is refined. Over a restricted band it is not, and differences can change sign, so bands are never mixed within a table. The frozen benchmark sweep is $\mathcal{T} = \{.80, .85, .90, .95, .99\}$, and because the best reconstruction model averages 0.46 IoU (Table 4) that sweep does not separate models, so for reporting we add a discriminative sweep $\mathcal{T} = \{.50, .60, .70, .80, .90, .99\}$ and label which is in use. A candidate whose exported solid exceeds a complexity guard (800 KB of STEP for reconstruction, a size ratio against the input for editing) is scored zero rather than voxelized, since voxelizing such candidates is prohibitively slow.

Assembly. A global shape metric is blind to assembly structure, so assemblies are scored per solid, after a Hungarian assignment of candidate to reference solids on their pairwise Chamfer cost. We report *cardinality*, whether the returned solid count equals the reference's, and a scale-relative part accuracy $\mathrm{rel.\,P@CD} = \frac{1}{|H|} \sum_{\eta \in H} \frac{1}{n_R} |\{i : \mathrm{CD}_i \leq \eta\, d_i\}|$ over $H = \{.005, .01, .02, .05\}$, the single-part sweep, where d_i is reference part i's own diagonal and an unmatched part fails. The whole-shape package also sweeps per-part Chamfer at absolute thresholds and scores pose and contact graphs (Appendix J), but on parts spanning 1.4–761 mm its absolute part-accuracy curve stays below 0.07 for every model, so we do not report it. Assembly *editing* is scored pass/fail by the localized scorer below.

Editing. An edit is scored on two sides over a region we know in advance. Given the reference R, the injected input P and a submission P', the points whose nearest neighbour in the other cloud lies farther than $\tau = 3\,\mathrm{floor}$ define the changed surface. Density-filtered and dilated by $2.5\,\tau$, this surface becomes the shell, a ground truth computed without P'. On it $m_{\mathrm{in}} = \mathrm{CD}(P', R)$ passes iff $m_{\mathrm{in}} \leq k \cdot \mathrm{floor}$ with $k = 3$, the threshold that admitted the injection. Collateral is not averaged, since a mean would hide a small stray change. Outside the shell the same cluster detector runs between P and P', and any cluster exceeding $k \cdot \mathrm{floor}$ fails the submission. A no-op fails m_{in} on every instance we report (Sec. 4.4), and a global rebuild fails locality. Assembly edits take the same two-sided form per solid. The targeted solid, member or contact must match R and every other solid must match P, each within its own sampling floor (Appendix J).

4.3 The Answer Channel

How a model is allowed to answer is an experimental variable we manipulate rather than a fixed part of the task. In the **code** channel the model writes kernel code that we execute in a separate process under a wall-clock timeout to obtain STEP. In the **DSL** channel it emits a structured operation against an enumerated menu of the input's own features and *we* apply verified primitives. Both emit the same artifact and are scored by the identical metric. Editing and spec-conditioned generation need different vocabularies and different justifications.

The edit vocabulary is closed. It has seven feature operations for single parts, six of which have admitted instances, and three rigid-motion operations for assemblies, with one operation per answer. It is the dual of the injection operators, not a general CAD interface. We therefore do not claim that it covers a kernel API, which the code channel may call in full. Our claim is a measured one. We call an instance *expressible* when the correct payload names a target the executor can resolve, and *reachable* when some payload a model could emit passes once executed. The operation always exists, since the vocabulary is the injectors' dual, but on 18 of the 301 instances the target does not resolve, mostly blends along straight edges that the executor's matcher does not recognize, so those 18 are not expressible. On the rest the shortfall is one of reachability, set by kernel fidelity and metric tolerance (Table 1). Because the oracle is given the correct answer and up to eight alternatives from the menu, it bounds the channel's panel in aggregate, which is why such a panel cannot be interpreted on its own.

The generation vocabulary is open-ended by comparison, and an answer is a short program, because constructing to a specification has no unique solution to invert. Its oracle therefore certifies coverage from below rather than bounding models from above, so the two oracles are not interchangeable and we never divide the coverage one out of a panel (Appendix F).

Three conditions make a difference between channels attributable to the channel. First, prompts are matched except where the answer format forces a difference, and on single parts it does. Naming a target requires a menu to name it from, so the structured channel sees each feature's centre, axis and radius where the code channel sees a summary of P and the solid itself. The menu is a deterministic function of that solid, so the difference is one of access rather than information, and a code-channel arm given the identical menu measures what that access is worth (Sec. 5). On assemblies both channels already receive the same menu. Second, the vocabulary is certified by a test able to fail, as ours did when an earlier assembly vocabulary could not restore several components at irregular poses and was extended before any result on that axis was reported. Third, the executor, which is our code rather than the model's, is calibrated as an instrument (Appendix F).

4.4 Certifying Tasks and Metrics

Before reporting any near-zero axis we run two arms. The **PERFECT** arm submits the exact answer and asks whether the metric and the instance are sound. It is not a formality, since submitting R itself scores 96%, not 100%. On thirteen of 342 single-part instances the metric rejects its own reference, because the changed surface is too thin to raise a region mask or reaches past it. Doing nothing must fail as well, and submitting P unchanged passes 28 more, mostly injections admitted because they break the solid's validity, which a distance metric cannot see. We report the 301 instances on which the reference passes and the no-op fails, and 126 of 127 assembly instances (Appendix C). Kept, either kind would inflate the unanimity Sec. 5 reads as a ceiling. The **CHANNEL** arm expresses that same answer in the answer channel and asks whether the vocabulary suffices. If both arms fail, the metric or the instance is broken. If PERFECT passes and CHANNEL fails, the vocabulary is at fault. If both pass, a near-zero score is a genuine capability gap.

A third, cheap arm completes certification. If a bounding box can beat a metric, the metric is not measuring what the task claims, so non-generative baselines run on every track and appear in the same table as the models. On spec-conditioned generation a box at the target's dimensions outranks two of seven models on mIoU and four on mean IoU, and retrieving a different real variant of the same product beats all of them (Table 3). Cross-product retrieval scores zero on mIoU, so the signal is real but family-level. Models on such a track must be read relative to that floor. A model that recalled a sibling variant from pretraining would reach it without reading the specification, so scores near the floor cannot separate reading from recall.

5 Experiments

Setup. The panel is five models, `claude-opus-4.8` (Claude), `gpt-5.6-sol` (GPT), `glm-5.2` (GLM), `deepseek-v4-pro` (DS-Pro) and `deepseek-v4-flash` (DS-Flash), answering through one OpenAI-compatible gateway at temperature 0, one sample per instance. The specification-conditioned arm adds `gemini-3.1-pro` (Gemini) and `kimi-k2.7-code` (Kimi), and the detection tier adds `gpt-6-sol` (GPT-6) and `claude-fable-5.1` (Fable), released after the panel was fixed. Reconstruction from renders uses the four panel models that accept images (Claude, GPT, Gemini, Kimi). No request sets a reasoning parameter, so each model runs at its provider's default, which for Claude is without extended thinking. Earlier baselines run at a gateway sampling default are never pooled with these. Whole-shape metrics come from a single package at its fast configuration (Appendix B), which pre-centres assemblies because supplier world frames are arbitrary. Editing uses the localized metric of Sec. 4.2 in the input frame, without centring or pose alignment, because preserving that frame is part of the task. Non-executable answers count as failures, so every denominator is all attempted instances.

Table 1: Model-path oracle on the 301 certified single-part instances. Failures are *unres.* (target not resolved, so not expressible), *kern.* (kernel declined), *reg.* (misses R) or *loc.* (collateral).

operator	n	unres.	kern.	reg.	loc.	pass
add hole	90	0	0	14	**18**	58
resize hole	77	2	1	7	9	58
remove hole	53	0	0	15	4	34
add fillet	21	**7**	0	6	0	8
add chamfer	41	**9**	1	13	2	16
remove fillet	19	0	8	1	0	10
all	301	18	10	56	33	184

Table 2: The code channel given the structured channel's own feature menu, detection tier, paired on each model's covered instances. *disc.* counts discordant pairs (code-only/menu-only) behind an exact McNemar p, code against +menu.

		success			executable			
model	n	code	+menu	DSL	code	+menu	disc.	p
Claude	247	36%	21%	45%	83%	89%	58/20	$<.001$
GPT-5.6	138	38%	37%	43%	65%	68%	22/21	1.000
GPT-6	166	44%	40%	47%	66%	67%	36/30	.539
GLM	168	15%	24%	42%	38%	64%	12/28	.017
DS-Pro	130	5%	8%	38%	9%	25%	3/ 7	.344
DS-Flash	209	4%	10%	40%	15%	44%	2/15	.002

5.1 The Answer Channel Changes What Is Measured

Figure 1 is the same 301 certified instances answered four ways. In the code channel the five models of the full grid span 7%–41% at the pointed tier and 3%–42% at detection, and the spread is mostly *executability*. At detection, executable rates differ elevenfold (7% to 84%) while accuracy on the answers that run differs threefold (22% to 66%). The two weakest return runnable geometry on 7% and 14% of detection-tier instances, so their scores measure kernel-API fluency almost exclusively. In the structured channel every model executes at 77%–86% and scores 49%–52% and 43%–50% by tier, the open models gaining most (DS-Pro $0.03 \to 0.45$ and GLM $0.15 \to 0.44$ on detection), and the residue is geometry that runs but lands outside R. The decomposition also separates profiles a scalar merges. Claude executes 88% of its pointed-tier answers yet scores 23%, because given coordinates it rebuilds 58% of its feature removals as filler solids, which pass 6% of the time. At detection it defeatures instead and scores 39% (Appendix E). Every structured-channel bar sits under the channel's own ceiling, the dashed line, and below it the channel saturates. Only 25 of the 301 instances separate any two models at the pointed tier and 49 at detection (Table 11). The ceiling belongs to the channel, not to the instances, since in code at least one model passes 34 of the 117 instances beyond it (45 at detection). Read on the 184 instances that ceiling admits, the panel spans 4 points at the pointed tier and 12.5 at detection, and the two frontier models are indistinguishable when told the location and separated when they must find it ($p = 1.0$ against $p = 0.003$, Appendix E). What still separates models through the channel is localization. Nor is the flatness the executor's tolerance, since requiring every coordinate to lie within $0.05\,d$ of its target, a fifth of the widest gate, costs no panel model more than 1.6 points (Appendix F).

Two newer models, and the other side of the ceiling. The detection tier adds two models released after the panel was fixed (Table 10). GPT-6 lands within two points of its predecessor on both channels, 44% in code and 51% structured, so the bottleneck is a property of the representation, not of one release. Fable behaves differently. It executes 94% of its code answers and reaches 63%, above the structured channel's own 61% ceiling, so for this model the channel *costs* 13 points. The channel removes a confound for models limited by kernel fluency and imposes one on a model that is not.

Is the menu the advantage? The structured channel names targets from a feature menu the code channel does not receive, so its advantage could be access to pre-extracted coordinates. We ran the code channel with the identical menu at the detection tier, on partial coverage of 130–247 instances per model (Table 2). The menu raises executability for every model but moves `edit_success` far less than the structured channel does, and Claude *falls* 15 points as its wrong-region answers rise from 86 to 155. The menu reports face centroids, which the structured executor snaps to the matching feature but a cut written in code follows off-axis. The advantage therefore lies in the answer format together with an executor that resolves the named feature, not in access to coordinates.

What bounds the structured channel. Table 1 decomposes that ceiling by failure mode, measured down the path a model actually takes, after retrying every failure with up to eight payloads

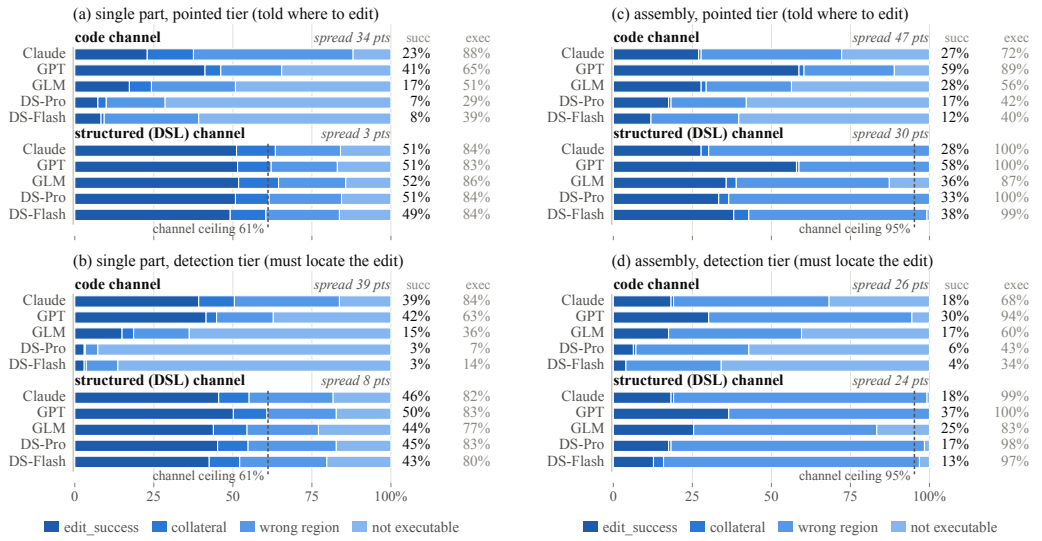

Figure 1: Editing outcomes at temperature 0 for the five models that answer both tiers, over all 301 certified single-part instances (left) and all 126 certified assembly instances (right). Each answer falls into one of four ordered outcomes (not executable, executable but missing the region R, region-correct with collateral over the gate, `edit_success`), and the right-hand columns give `edit_success` and the executable rate. The dashed line is the structured channel's ceiling, the model-path oracle of Table 1 for single parts and, for assemblies, a privileged vocabulary oracle read only as an upper bound.

built only from the menu the model sees. On 89% of the failures nothing passes, so the shortfall belongs to the channel rather than to the parameters the oracle happened to choose (Appendix F). Unresolved targets and kernel refusals concentrate on blends, and on four of the six operators the ceiling sits *below* Fable's detection-tier code answers (`add_fillet` 38% against 76%, `add_chamfer` 39% against 54%, `add_hole` 64% against 73%, `resize_hole` 75% against 78%). We trace the `add_fillet` and `add_hole` gaps to the executor rather than to the vocabulary, namely its blend matcher and the bore extent it infers where a model writing code chooses it. These conclusions are specific to single parts. The assembly vocabulary is certified separately (Sec. 4.3), and there expressiveness *did* bind (Table 14).

5.2 Assemblies

On assemblies the channel effect differs in kind (Figure 1, right). The structured channel takes every model to 83%–100% executable. At the pointed tier it is worth 8 to 26 points to the three weaker models and nothing to the two frontier ones ($0.27 \to 0.28$ and $0.59 \to 0.58$), which could already write runnable assembly code. Nor does it flatten this panel. The spread stays at 30 and 24 points with no bar near the ceiling, so the assembly track is still limited by the models while the single-part track has become limited by the channel. The two newer models lead this track, at 44% and 37% in code at detection against 30% for the best of the original five, and the structured channel lowers both slightly, to 41% and 37%.

5.3 Specification-Conditioned Generation

Table 3 splits the 217 pairs by separability into 140 single-part targets, scored by the whole-shape sweep, and 77 multi-solid targets, scored per reference part after centring and never pooled with the first. On the single-part half the spread is mostly executability, and Gemini leads while GPT is the most reliable and third by score. Same-product retrieval beats every model, so the condition fails the trivial-baseline gate of Sec. 4.4 and we report it as a condition read relative to that floor, not as a fourth family. The multi-solid prompt states the target solid count, so under-production falls to at most four of the 77 and the remaining error is over-production, concentrated in GLM.

Table 3: Specification-conditioned generation (target parameter row, no source solid), seven models at temperature 0, all attempted instances. Both aggregations rank the models identically but not the baselines. The assembly half reports solid count (*card.* exact, *over* too many) and *rel. P@CD* (Sec. 4.2). The last three rows generate nothing. Same-product retrieval returns a nearest sibling variant on the single-part half and the pair's own source variant on the assembly half.

	single part (n=140)			assembly (n=77)			
model	exec.	mIoU	meanIoU	exec.	card.	over	rel. P@CD
Gemini	78%	**0.254**	**0.432**	43%	40%	3%	0.011
Claude	84%	0.231	0.391	75%	65%	8%	0.012
GPT	86%	0.162	0.362	81%	**75%**	4%	**0.022**
GLM	79%	0.155	0.324	83%	57%	25%	0.019
DS-Pro	45%	0.080	0.167	55%	47%	6%	0.010
DS-Flash	49%	0.048	0.149	57%	36%	14%	0.009
Kimi	25%	0.042	0.099	53%	42%	9%	0.013
retrieval, same prod.	—	0.351	0.589	—	**81%**	—	**0.133**
target bbox	—	0.079	0.353	—	—	—	—
retrieval, cross prod.	—	0.000	0.091	—	—	—	—

Table 4: Reconstruction from renders, single-part track, n=85, temperature 0, all attempted instances. *P@.90* is the frozen reference point and mIoU† repeats the reporting band at benchmark configuration. *Box* is a solid at the reference bounding box and *retrieval* the nearest other reference.

model	exec.	mIoU	meanIoU	P@.90	mIoU†
GPT	**85%**	**0.369**	**0.464**	**0.32**	**0.327**
Claude	82%	0.171	0.360	0.07	0.144
Gemini	55%	0.170	0.279	0.11	0.168
Kimi	20%	0.072	0.106	0.04	—
box floor	—	0.053	0.313	0.00	—
retrieval floor	—	0.000	0.143	0.00	—

5.4 Reconstruction from Renders

GPT leads on every column of Table 4, but the margin depends on the aggregation. On the plain per-item mean the panel is ordered but close, while on the reporting band, which separates a roughly right shape from a dimensionally right one, GPT more than doubles either competitor and Claude and Gemini become indistinguishable (Appendix K). The box floor changes how the panel reads, scoring above Gemini on the plain mean but below every model on the band, because a box is never dimensionally right.

6 Discussion

The broader lesson is not specific to CAD. When a benchmark requires answers in a demanding formal language, that language becomes part of the measurement. Controlling it moves a five-model editing panel from a 39-point spread to eight, but that flat panel is set by the channel's ceiling and its saturation below it, and for a model fluent in the kernel the same channel costs 13 points. We therefore recommend reporting both channels, since neither alone separates a model's geometry from its fluency in one kernel. A code-channel score should carry its executable rate and an API-reference control, and a structured-channel score its oracle ceiling, with models compared on the instances that ceiling admits. A near-zero axis is not a finding until the oracle has run. Appendix L states four limitations.

AI USE STATEMENT

Language models are the object of study here. The evaluated panel is listed in Section 5, and its outputs are only ever scored, never used as labels, references or scores. Separately, we used generative AI in producing this work. *Synthetic data.* The prose build descriptions in the text-only reconstruction study (Appendix K) were generated by Gemini from the renders and dimension inventory, which is why Gemini is excluded from that comparison. *Research and engineering assistance.* An LLM coding assistant was used to implement and debug the preprocessing pipeline, task generator, evaluation harness and scorer, and to write the figure- and table-generation scripts. It was also used to propose, run and critique analyses and experimental checks and to help interpret their results. *Retrieval and discovery.* The same assistant searched for and retrieved related work, including the fine-tuned CAD models cited in Section 2, whose bibliographic records were then verified against the primary sources. *Writing.* It drafted and edited parts of the paper text and formatted references. No reference solid, injected defect, instruction or score was produced by a language model. Defects and instructions are generated programmatically, references are the suppliers' STEP files, and scoring is geometric. The authors reviewed and verified all AI-assisted code, analyses and text, and take full responsibility for the content of this paper, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Data. The source CAD comes from a public part library that aggregates supplier-authored models, and data use was negotiated with the library operator (Section 3.1). Rights in the source files are settled between the suppliers and the platform and are not ours to assign, so every file that carries supplier geometry is released only to researchers, under access terms agreed with that operator. That covers the reference STEP files, the parametric variants, the editing inputs P (each is a reference with one feature changed), the solids the models returned, and the catalogue drawings. Everything else is released openly. That comprises the renders, three-view projections and dimension inventories, which we generate from the geometry rather than take from supplier imagery, and the content-addressed manifest with every instruction, injection log and region. It also comprises each model's answer as the code or payload it returned, the scores, and our code, with the code under the MIT licence and the other open artifacts under CC BY 4.0. *People.* No human subjects were involved and no annotation was purchased. Defects, instructions and labels are produced programmatically and admitted only above a measured noise floor. *Executing model output.* Code written by the evaluated models is run in a separate process under a wall-clock timeout to obtain a solid, and the scorer ingests geometry only and never executes model output. Saved answers should be replayed in a container without network access. *Scope.* The benchmark measures geometric agreement with a reference. It makes no claim that a passing solid is manufacturable or safe to use, and it is test-only, with no training split.

REPRODUCIBILITY STATEMENT

Section 5 fixes the evaluation protocol, including model identifiers, a single gateway, temperature 0, one sample per instance, and denominators over all attempted instances. Appendix B gives the sampling configurations of both scorers (point count, tessellation deflection, voxel resolution, per-item seeding) and derives the noise floor from the editing scorer's. Appendices C and D give the injection operators with per-operator counts, the three instruction tiers as shipped, both answer vocabularies, the shared prompt blocks and a worked instance with model output. Appendix J gives the assembly scoring rules and instructions and Appendix F the oracles that certify both vocabularies. Table 10 gives the per-model counts behind Figure 1. Appendix I re-runs one full arm to bound run-to-run variation (2.3% of verdicts flip). That arm is the structured channel at the pointed tier, the one the channel decides most completely, so this is a stability bound for it, not a confidence interval for the code channel. The supplementary material contains the anonymized code for the preprocessing pipeline, task generation, evaluation harness and scorer. The dataset manifest is content-addressed, so a modified input cannot be consumed silently, and saved model answers can be re-executed and re-scored without querying any model. A third party runs the benchmark from the files obtained un-

der the access terms in the Ethics Statement, which include every reference and every editing input, together with the open code and manifest, which names each reference file by its SHA-256 hash.

REFERENCES

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew C. Gombolay. Generating CAD code with vision-language models for 3D designs. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=BLWaTeucYX.

Jesse Barkley, Rumi Loghmani, and Amir Barati Farimani. CADSmith: Multi-agent CAD generation with programmatic geometric validation. *CoRR*, abs/2603.26512, 2026. doi: 10.48550/ARXIV.2603.26512. URL https://doi.org/10.48550/arXiv.2603.26512.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *CoRR*, abs/1512.03012, 2015. URL http://arxiv.org/abs/1512.03012.

Anna C. Doris, Jacob Thomas Sony, Ghadi Nehme, Era Syla, Amin Heyrani Nobari, and Faez Ahmed. CADBench: A multimodal benchmark for AI-assisted CAD program generation. *CoRR*, abs/2605.10873, 2026. doi: 10.48550/ARXIV.2605.10873. URL https://doi.org/10.48550/arXiv.2605.10873.

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. InverseCSG: automatic conversion of 3D models to CSG trees. *ACM Trans. Graph.*, 37(6):213, 2018. doi: 10.1145/3272127.3275006. URL https://doi.org/10.1145/3272127.3275006.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 5885–5897, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/2e92962c0b6996add9517e4242ea9bdc-Abstract.html.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D. D. Willis, Thomas Davies, Hooman Shayani, and Nigel J. W. Morris. UV-Net: Learning from boundary representations. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 11703–11712. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01153. URL https://openaccess.thecvf.com/content/CVPR2021/html/Jayaraman_UV-Net_Learning_From_Boundary_Representations_CVPR_2021_paper.html.

Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. SolidGen: An autoregressive model for direct B-rep synthesis. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=ZR2CDgADRo.

R. Kenny Jones, David Charatan, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeMOD: macro operation discovery for 3D shape programs. *ACM Trans. Graph.*, 40(4):153:1–153:16, 2021. doi: 10.1145/3450626.3459821. URL https://doi.org/10.1145/3450626.3459821.

Ma Kimmel, Mueed Ur Rehman, Yonatan Bisk, and Gary K. Fedder. Position: You can't manufacture a NeRF. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 - Position Paper Track*. OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/kimmel25a.html.

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 *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 9601–9611. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00983. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Koch_ABC_A_Big_CAD_Model_Dataset_for_Geometric_Deep_Learning_CVPR_2019_paper.html.

Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 12773–12782. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01258. URL https://openaccess.thecvf.com/content/CVPR2021/html/Lambourne_BRepNet_A_Topological_Message_Passing_System_for_Solid_Models_CVPR_2021_paper.html.

Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. CAD-Llama: leveraging large language models for computer-aided design parametric 3D model generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025*, 2025. URL https://arxiv.org/abs/2505.04481.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. StructEdit: Learning structural shape variations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 8856–8865. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00888. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Mo_StructEdit_Learning_Structural_Shape_Variations_CVPR_2020_paper.html.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICML 2020 Workshop on Object-Oriented Learning*, 2020.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural shape parser for constructive solid geometry. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 5515–5523. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00578. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sharma_CSGNet_Neural_Shape_CVPR_2018_paper.html.

Liang Wang, Heng Meng, Zekai Xiang, Jin Liu, Pingyi Zhou, Litao Chen, and Yongqiang Tang. Text2CAD-Bench: A benchmark for LLM-based text-to-parametric CAD generation. *CoRR*, abs/2605.18430, 2026. doi: 10.48550/ARXIV.2605.18430. URL https://doi.org/10.48550/arXiv.2605.18430.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-CAD generation through infusing visual feedback in large language models. In *International Conference on Machine Learning, ICML 2025*, 2025. URL https://arxiv.org/abs/2501.19054.

Karl D. D. Willis, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pp. 2105–2114. Computer Vision Foundation / IEEE, 2021a. doi: 10.1109/CVPRW53098.2021.00239. URL https://openaccess.thecvf.com/content/CVPR2021W/SketchDL/html/Willis_Engineering_Sketch_Generation_for_Computer-Aided_Design_CVPRW_2021_paper.html.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: a dataset and environment for programmatic CAD construction from human design sequences. *ACM Trans. Graph.*, 40(4): 54:1–54:24, 2021b. doi: 10.1145/3450626.3459818. URL https://doi.org/10.1145/3450626.3459818.

Karl D. D. Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. JoinABLe: Learning bottom-up assembly of parametric CAD joints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 15828–15839. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01539. URL https://doi.org/10.1109/CVPR52688.2022.01539.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 6752–6762. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00670. URL https://doi.org/10.1109/ICCV48922.2021.00670.

Haoyang Xie and Feng Ju. Text-to-CadQuery: A new paradigm for CAD generation with scalable large model capabilities. *CoRR*, abs/2505.06507, 2025. URL https://arxiv.org/abs/2505.06507.

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 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 24698–24724. PMLR, 2022. URL https://proceedings.mlr.press/v162/xu22k.html.

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 Trans. Graph.*, 43(4):119:1–119:14, 2024. doi: 10.1145/3658129. URL https://doi.org/10.1145/3658129.

Yikang Yang, Zhanpeng Hu, Youtian Lin, Mengqi Zhou, Jingxi Xu, Feihu Zhang, Jiaheng Liu, and Yao Yao. P3D-Bench: Benchmarking MLLMs for parametric 3D generation and structural reasoning. *CoRR*, abs/2606.11152, 2026. doi: 10.48550/ARXIV.2606.11152. URL https://doi.org/10.48550/arXiv.2606.11152.

Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. CAPRI-Net: Learning compact CAD shapes with adaptive primitive assembly. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 11758–11768. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01147. URL https://doi.org/10.1109/CVPR52688.2022.01147.

Haozhe Zhang, Kaichen Liu, Miaomiao Chen, Lei Li, Shaojie Yang, Cheng Peng, and Hanjie Chen. BenchCAD: A comprehensive, industry-standard benchmark for programmatic CAD. *CoRR*, abs/2605.10865, 2026. doi: 10.48550/ARXIV.2605.10865. URL https://doi.org/10.48550/arXiv.2605.10865.

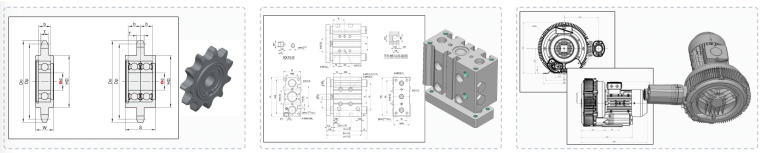

Figure 2: Three products from the source catalogue, each shown as its dimension drawing beside the STEP solid. They are a bearing-mounted chain sprocket (4 solids), a guided pneumatic cylinder (6 solids) and a side-channel blower (11 solids).

APPENDICES

These appendices supplement the main text. Everything shown is transcribed from the shipped artifacts. Instruction strings are the manifest's own fields, per-model counts are generated from the stored predictions rather than typed, and model answers are verbatim output unless marked as abridged.

What each appendix contains. **Appendix A** gives the corpus composition and the conditioning generated for every entry, with one product's specification table and the symbol legend induced from it. **Appendix B** states the sampling configuration and derives the metric noise floor that gates both defect injection and editing success. **Appendix C** lists the injection operators with their per-operator counts and defect classes. **Appendix D** specifies what each answer channel is given and what it must return, the three instruction tiers, and one instance followed from injection to score. **Appendix E** gives the per-model editing counts underlying the main paper's failure-mode figure. **Appendix F** certifies the two structured vocabularies against oracles that bound in opposite directions, and reports the API-reference control. **Appendix G** replicates the channel effect on specification-conditioned generation, on both halves of that corpus. **Appendix H** reports a condition the main paper does not run, re-dimensioning with the source solid loaded. **Appendix I** measures run-to-run stability at temperature zero. **Appendix J** defines how assembly edits and whole-shape assembly axes are scored and gives the assembly instruction text. **Appendix K** compares reconstruction from renders with reconstruction from a prose description of the same renders. **Appendix L** states what all of these results bound, and **Appendix M** concludes the work, main text and appendices together.

A CORPUS COMPOSITION AND CONDITIONING

No two active entries are duplicates. All 144 content hashes differ, and no pair agrees on face count with volume and surface area within 1%. Of the 150 sampled products, six are excluded before any task is built, three because their STEP file does not read as a valid solid, one because the supplier ships only a Parasolid model, and two because the model is not authored in millimetres. Figure 3 gives the category composition of the 144 active entries and Figure 4 the visual conditioning generated for each of them, both computed from the STEP solid under a fixed camera and view convention. Supplier drawings ship with the corpus for 38 entries, but no arm evaluated in this paper receives them.

Table 5 shows one product's published specification table from the parametric variant set, together with the symbol legend induced from it. The legend is needed because 59% of specification key–value pairs are bare drawing symbols (`V=108`, `K=135`) meaningless without the catalogue drawing. It is induced with no annotator by fitting each symbol to a geometric predicate that must hold on *every* variant of the product, so a wrong gloss is rejected mechanically. That makes in-sample validation 100% by construction, so we validate leave-one-out and against a permutation null. Over 127 products the 694 glosses validate at 99.6%, against a chance rate of 19.5%.

The 217 evaluated pairs are a seeded sample from the 3537, stratified by the kind of parameter change into 90 single-parameter, 90 multi-parameter and 37 categorical pairs (the last limited by availability), with at most four pairs per product and change type. They cover 91 products. Each pair is routed to the single-part or multi-solid half by counting the solids of both variants.

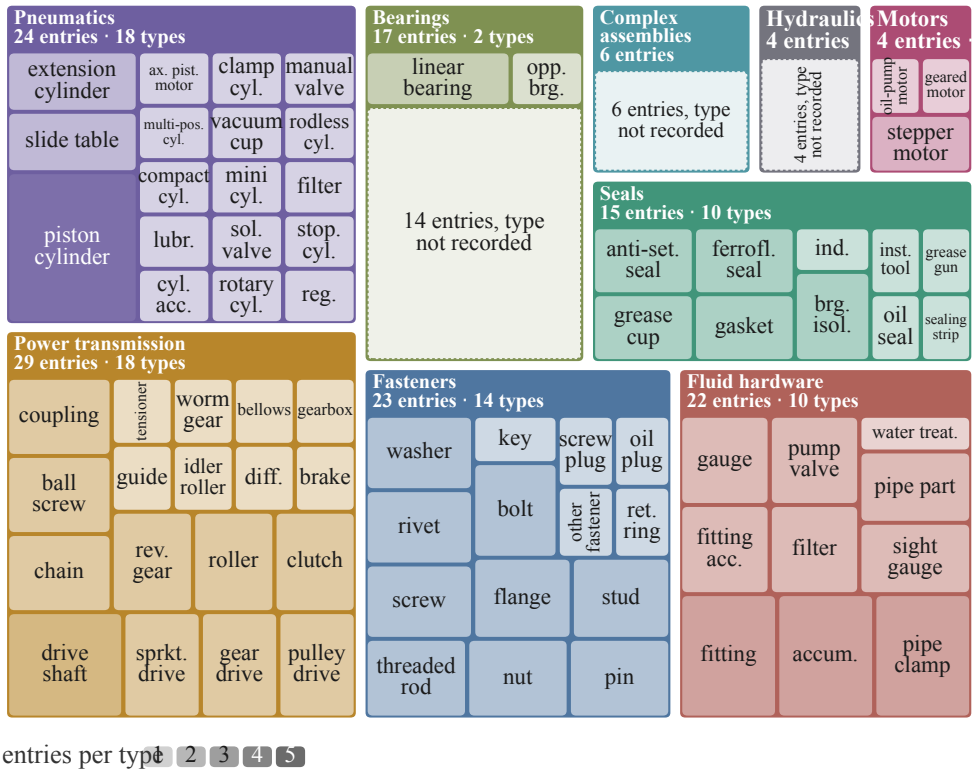

Figure 3: Category composition of the 144 active entries, sampled uniformly over the supplier taxonomy rather than in proportion to catalogue volume. Outer blocks are the nine catalogue categories, inner cells the component types recorded in the manifest, and area and shading both encode entry count. Dotted cells are entries whose type the source metadata leaves unrecorded.

Table 5: One product's specification table as published (self-aligning ball bearing on an adapter sleeve, GB/T 281), the first five of its ten shipped variants. Each row is a STEP variant the library ships with the parameter vector that generated it, and a task instance pairs one row's solid with another row's parameters. The symbols are the catalogue's and carry no attached meaning, so the second row is induced with no annotator, by fitting each symbol to a geometric predicate that must hold on every variant of the product. In the gloss row, *dia.* is the diameter of a cylindrical feature, *gap* the distance between a parallel pair of flat faces, and *rad.* a fillet or spherical radius. Symbols that no predicate fits are left unglossed rather than guessed. Here that is d, the nominal bore designation, which names the bearing size but measures nothing on the solid.

variant	d	d_1	d_2	D	B	B_1	B_2	r
induced gloss	—	dia.	dia.	dia.	gap	gap	gap	rad.
1204 K+H 204	20	17	32	47	14	24	7	1.0
1205 K+H 205	25	20	38	52	15	26	8	1.0
1206 K+H 206	30	25	45	62	16	27	8	1.0
1207 K+H 207	35	30	52	72	17	29	9	1.1
1208 K+H 208	40	35	58	80	18	31	10	1.1

B Sampling and the Noise Floor

Sampling. Two scorers sample surfaces the same way, by a triangle-area-weighted multinomial draw of N points with uniform barycentric coordinates within each drawn triangle, but at different settings. The editing scorer, whose configuration also defines the noise floor, tessellates at linear

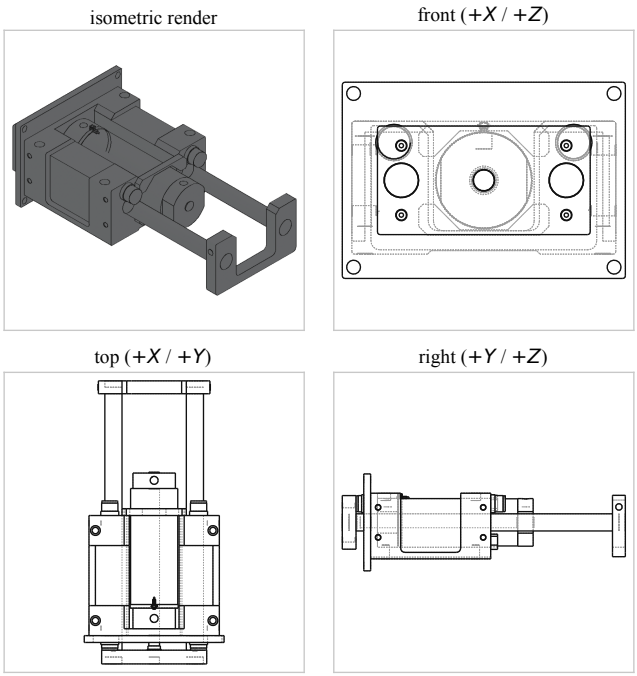

Figure 4: The conditioning generated for every entry, here a guided pneumatic cylinder, consisting of an isometric render and a three-view projection computed from the STEP solid, hidden edges dashed. The axis convention is given in the prompt, since without it the extra views hurt rather than help.

Table 6: Corpus routing and complexity. Assignment is by measured solid decomposition, not catalogue metadata. In all, 79 single parts and 6 fused assemblies route to the single-part track, 25 loose multi-solids and 34 structured assemblies to the assembly one. Counts are medians, except that *drw.* counts entries with an authentic supplier drawing.

track	N	faces	solids	drw.
single-part track	85	36	1	17
assembly track	59	288	5	21
total active	**144**	122	1	38

deflection $0.001\,d$ clamped to $[0.02, 0.2]$ mm and angular deflection 0.25 rad and draws $N = 5 \times 10^4$ points per shape. The whole-shape package, which scores reconstruction and spec-conditioned generation, has a benchmark configuration (0.01 mm, 0.1 rad, $N = 10^4$, voxel resolution $G = 128$) and a fast one (0.05 mm, 0.3 rad, $N = 2048$, $G = 64$). Every whole-shape number in this paper uses the fast configuration, because at benchmark configuration 37% of reconstruction items did not return within a 1500 s per-item budget and would have scored zero. The benchmark configuration appears only as the mIoU† column of Table 4, where GPT still leads and Claude and Gemini stay within 0.03 of each other. Area weighting is not incidental. Vertex-count and face-count proxies over-represent regions the tessellator happened to subdivide finely, which systematically overstates how localized a change is. The area-weighted sampling spacing $\sqrt{A/N}$ over surface area A is what sets the resolution of every distance in the paper. Seeds are fixed per item in both scorers (the whole-shape package derives them from a stable hash of the global seed and the item id), so re-ordering a batch cannot change a score.

The noise floor. Two STEP files describing the same solid do not read as identical, because tessellation density, the sampling draw, and write/read round-trip drift each leave a residual on every distance. We measure that residual directly rather than assuming a tolerance. For each reference R,

Table 7: The injection operators. *sel.* counts instances that admit a selection-tier variant. A seventh single-part operator, `remove_chamfer`, is implemented but produced no admitted instances on this corpus. Counts are the shipped substrate, after two admission gates, the magnitude gate of the main paper and an invertibility gate, so an instance survives only if its magnitude clears $k \cdot \mathrm{floor}$ and its inverse is constructible on P.

operator	defect class	n	sel.	what the injector does
single-part track ($n = 342$)				
`add_hole`	feature add	95	95	cut a new through-hole
`resize_hole`	dimensional	82	82	re-cut a bore coaxially
`add_chamfer`	feature add	58	58	bevel a sharp convex edge
`remove_hole`	feature remove	54	0	defeature (fill) a bore
`add_fillet`	feature add	33	33	round a sharp convex edge
`remove_fillet`	feature remove	20	0	defeature a fillet
assembly track ($n = 127$)				
`pose`	pose	59	—	translate + rotate one solid
`connectivity`	connectivity	42	—	break a contact
`cardinality`	count	26	—	delete a symmetry-group member

at the editing scorer's configuration, we re-tessellate and STEP round-trip it, producing variants that *should* score zero against the original, score them, and take the 95th percentile across the dataset:

$$\mathrm{floor}_{\mathrm{CD}} = 0.00363\, d, \qquad \mathrm{floor}_{\mathrm{conn}} = 0.18.$$

A closed form shows what is being measured, namely the sampling spacing itself:

$$\mathrm{floor}_{\mathrm{mm}} \;\approx\; 0.5\sqrt{A/N},$$

which is also the smoke test we use after any change to the sampling code.

The floor is a property of the exact sampling configuration, so any change to N, to mesh deflection or to voxel resolution requires re-deriving it, and it gates geometry comparisons only, since applying a surface-sampling residual to an exact B-rep parameter is a category error. It enters the benchmark in two places, deliberately at the same threshold. An injected defect is admitted only if its magnitude exceeds $k \cdot \mathrm{floor}$, and an editing submission passes only if its residual falls below $k \cdot \mathrm{floor}$. A defect too small to be measured is therefore never posed as a task, and a repair is never asked to beat the measurement.

C The Injection Operators

Table 7 is the full operator set. Each operator takes the reference solid R and produces the model's input P by making exactly one change, and the model's task is to invert it. Two properties are recorded per instance and used downstream. *Defect class* says what kind of error the edit represents (a feature that should not be there, one that is missing, or a wrong dimension), and the metric's region mask is built from it. *Validity class* records whether the injected P is still a valid solid. Of the 342 single-part instances, 41 are *validity-breaking*, and we keep and label them rather than discard them, because a malformed input is a case a CAD agent meets in practice.

The *selectable* column matters for the selection tier only. An added or resized feature is present in P, so a GUI face-pick can point at it. A *removed* feature is absent from P by construction, so there is nothing to select and those 74 instances have no selection variant.

Certifying the substrate. Before any model result we submit R itself, round-tripped through the same CadQuery writer every submission passes through, and score it. A correct answer must pass, so anything that does not is unpassable by any backend and is excluded. Thirteen single-part instances fail it, and the two mechanisms are distinct. Five return no region reading at all, because the changed surface between R and P raises fewer than the eight dense moved points the mask needs at scoring density, so the injection gate admitted a defect the scorer cannot localize. Eight fail locality, because the genuine difference between R and P extends past the shell's $2.5\,\tau$ catchment,

Table 8: Achievability of the editing substrate. *Region* and *locality* are the two sides of the metric when R itself is submitted through the writer every answer passes through, and R *passes* counts instances where both hold. *No-op passes* counts those among them on which submitting P unchanged also passes. An instance is certified when R passes and the no-op does not. Anything short of the full count is a benchmark defect, not a model result.

operator	n	region	locality	R passes	no-op passes	certified
`add_hole`	95	95 (100%)	91 (96%)	91	1	90
`resize_hole`	82	82 (100%)	78 (95%)	78	1	77
`add_chamfer`	58	55 (95%)	58 (100%)	55	14	41
`remove_hole`	54	54 (100%)	54 (100%)	54	1	53
`add_fillet`	33	32 (97%)	33 (100%)	32	11	21
`remove_fillet`	20	19 (95%)	20 (100%)	19	0	19
single-part total	342	337 (99%)	334 (98%)	329	28	**301**
assembly total	127	127 (100%)	127 (100%)	127	1	**126**

and the detector charges the correct answer with 0.04–1.85 mm of collateral over 10–74 clustered points. The failures are not size-driven, median diagonal 71 mm against 69 mm for those that pass, so this is mask calibration rather than floor scaling, and the fix belongs in the mask rather than in the threshold. All 127 assembly instances pass this test (Table 8). Twelve of the thirteen are passed by nothing we ran, model or oracle, so they are unanimous failures by construction. The thirteenth, `4892623477461__resize_hole__0`, is the defect seen from the other side. Models do pass it while the reference does not, because an answer that disturbs *less* of the surface outside the mask than the reference itself escapes a collateral test the reference trips. An instance where the metric rejects its own ground truth but accepts other answers cannot be scored either way, so it is excluded with the rest. The cost is that certification removes one genuine success from each of six models. Keeping these 13 would cap every model at zero on instances that are unanimous by construction, inflating the count of instances the main paper attributes to the structured channel's ceiling.

The identity test shows that the metric accepts the right answer, not that it rejects a wrong one, so we also submit P unchanged, the degenerate answer of doing nothing. It passes 28 of the 329 instances that survive the identity test. Twenty-four are validity-breaking injections, 13 chamfers and 11 fillets. The injection gate admits a validity-breaking edit on its topology rather than on its distance to R, but the editing metric reads distance only, and on these 24 the distance defect lies below $k \cdot \mathrm{floor}$, so an answer that leaves the defect in place passes. The other four are validity-preserving instances whose defect sits at the threshold, admitted at injection and passed by the no-op under a different sampling draw. On such an instance any executable answer that disturbs nothing above the gate succeeds, whether or not it attempts the edit. We exclude all 28 and report on the 301 instances on which the reference passes and the no-op fails. On the assembly track the no-op passes one instance of 127, a 0.56 mm, 28° pose change of one solid in a 9.4 mm assembly. The solid's centroid moves less than the metric's 1 mm tolerance and its surface only 0.29 mm, within its distance threshold, so the metric cannot see the change. We report on the remaining 126 (Table 8). Excluding the 28 moves no structured-channel rate by more than 1.6 points and Claude's code-channel rate by about 4, because Claude passes 19 of them at the pointed tier and 23 at detection, and it leaves every conclusion in the main paper unchanged.

Why removal is the hard family. Four of the six single-part operators add or resize a feature, so inverting them means deleting or re-cutting something the model can see in P. The two `remove_*` operators invert a *lossy* operation. Defeaturing heals the neighbourhood into a smooth face, so restoring the feature means synthesizing geometry that is not in the input and inferring where it belonged. Removal accordingly carries most of the localization cost in the structured channel. From the pointed tier to detection, `remove_fillet` falls from 42–53% to 0–32% across the five panel models and `remove_hole` from 55–58% to 28–53%, while no other operator loses more than eight points for any model. `remove_fillet` is also the operator the invertibility gate rejects most.

Table 9: The three tiers on one instance. *Pointed* states the location and axis, so only execution is being tested. *Selection* withholds coordinates but hands the model the picked faces through a side channel, emulating a user who clicked the feature. *Detection* gives neither, so the qualifier Ø7.85 mm is the only handle, and the model must find the matching feature in P itself. For `remove_*` operators, where the feature is absent from P, the detection instruction adds a coarse non-coordinate region hint (an octant descriptor) so the task stays well-posed rather than unanswerable.

tier	instruction as shipped
pointed	`Remove the added Ø7.85 mm through-hole with axis direction (0.87, 0.00, 0.50) at (-10.4, -19.0, -6.0) mm and restore the face.`
selection	`Remove the` **`selected`** `added Ø7.85 mm through-hole and restore the face.`
detection	`Remove the added Ø7.85 mm through-hole and restore the face.`

D What the Model Is Given, and What It Returns

D.1 The Three Instruction Tiers

The tiers differ only in how much of *where* is given away, and the correction itself is the same. Table 9 shows all three for one real instance, an added Ø7.85 mm through-hole, exactly as they appear in the shipped manifest.

D.2 The Answer Vocabularies

Code channel. The input solid P is preloaded as a CadQuery object named `part`, accompanied by a compact JSON side channel holding P's bounding box, face and edge counts, surface-type census and the multiset of its cylindrical radii. This conveys scale and feature stock but no coordinates, so the model must locate its target in `part` itself. The model returns one Python block assigning the edited solid to `result`. CadQuery (imported as `cq`) and the underlying OCC bindings are both importable, so the channel is unrestricted and any operation the kernel supports may be used.

Structured channel, single part. One JSON object, with `op` drawn from `add_hole`, `remove_hole`, `resize_hole`, `add_fillet`, `remove_fillet`, `add_chamfer` and `remove_chamfer`, and parameters `center` (target location), `axis` (hole or bore direction), `radius` (hole radius, fillet radius, or the *target* radius for a resize), `distance` (chamfer), and `edge_near` (a point on the edge to blend). Coordinates are world-frame millimetres. The model chooses them from an enumerated feature menu read from P, listing its cylindrical faces (bores and holes), toroidal and conical faces (fillet and chamfer blends), the longest sharp two-face edges, and the largest planar faces, each with centre, axis and radius. The menu contains P's coordinates only and never the reference's.

The executor then re-resolves the named target against P's own geometry, with a tolerance that depends on the operation. A blend to be removed is matched by surface type and radius, and the blend face whose bounding-box centre lies nearest the stated point is accepted if it lies within $\max(0.25\,d, 8\,\mathrm{mm})$. A hole to be filled is matched by radius and axis direction among bores whose wall comes within $\max(5r, 0.12\,d, 8\,\mathrm{mm})$ of the point, a bore to be resized by its axis line within $\max(0.01\,d, 0.5\,\mathrm{mm})$, and a blend to be restored on the nearest edge that accepts it within $\max(0.15\,d, 2\,\mathrm{mm})$. A removed hole is cut exactly where the answer states. Only surface types the matcher recognizes can be resolved. A blend is recognized as a torus or a cone, so a blend along a straight edge, which is a cylinder or a plane, cannot be targeted at all (Appendix F). This is a real asymmetry with the code channel, where a model must select the face itself, and it cuts both ways. At the pointed tier this is the tier working as designed. The instruction already states where the feature is, and the widest tolerance is on the scale of the injected region itself, whose bounding box has median diagonal 0.656 of the part's. Any coordinate near the stated one therefore resolves the same feature, and what remains under test is the operation and its parameters. At the detection

tier the instruction states no location and the targets are not unique (only 39% of `add_fillet` and 14% of `add_chamfer` instances have a single type-and-radius match), so localization is still tested. The structured channel only has to be nearest within tolerance, however, where code must be exact. Appendix F measures how much the tolerance contributes by tightening it after the fact, and finds that it contributes little.

Structured channel, assembly. One JSON object, with `op` drawn from `move_solid`, `duplicate_solid` and `remove_solid`, a `solid` index into an enumerated solid menu, and a rigid motion given as `translate` plus `rotate` (axis, centre, angle). `duplicate_solid` takes either `count`, repeating one motion cumulatively for a regular pattern, or `copies`, giving each new component its own motion. Without the second form, restoring several components at irregular poses is inexpressible in a single operation, which capped the cardinality axis.

Parameter track. The parameter-specified family uses a different, more general vocabulary (sketch, extrude, revolve, loft, sweep, cylinder, boolean, fillet, chamfer, shell, pattern and move), because that task constructs rather than repairs. An answer there is an ordered *op list* with named intermediates, executed in sequence by a separate executor. The two are never pooled.

D.3 A Worked Instance

Figure 5 follows one instance from injection to score. It is the same instance as Table 9, at the detection tier, and both answers are the same model's output. The structured answer is verbatim, and the code answer is abridged.

The two channels receive the same instruction and the same input solid, and differ only in the side channel described above (the preloaded `part` or the enumerated feature menu) and in what must come back. Both prompts carry the identical coordinate-frame block, both produce a STEP solid, and the scorer sees only the solid.

The pointed instruction's coordinate and the menu row's differ by construction, and neither is wrong. The instruction states the injection anchor, which lies on the feature's axis, while the menu reports the face's mesh centroid, which for a cylinder cut by non-parallel ends sits slightly off it. The executor accepts any bore of the stated radius and direction whose wall lies within $\max(5r, 0.12\,d, 8\,\mathrm{mm})$ of the stated point, so either coordinate resolves the same feature, and the radius in the instruction disambiguates the rest.

The structured answer contains no kernel call, so the model's entire contribution is the choice of operation and which enumerated feature is the target. That is how the channel isolates the modelling decision from API fluency. It is also why the channel imposes a ceiling. A payload may state any coordinate, but it takes effect only through a target the executor resolves, so a feature its matcher does not recognize cannot be named by any payload, whether copied from the menu or not.

D.4 Prompt Blocks

Both channels share the coordinate-frame block, which is stated because the metric performs no pose alignment and an answer built in the wrong frame scores near zero regardless of its shape:

> Coordinate frame (right-handed, millimetres) and view convention --- build oriented in THIS world frame: FRONT view: horizontal = world $+X$, vertical = world $+Z$ (depth $= +Y$). TOP view: horizontal $= +X$, vertical $= +Y$ (depth $= +Z$). RIGHT view: horizontal $= +Y$, vertical $= +Z$ (depth $= +X$).

The three views are rendered from $-Y$, $+Z$ and $+X$ respectively (`pipeline/render.py`), so each *depth* entry names the view axis rather than a signed direction into the screen. The horizontal and vertical mappings, which are what a model builds against, are the projections of those viewpoints.

The code channel additionally states the locality requirement, which is what makes the task an *edit* rather than a reconstruction:

1. Injection (ours, hidden)

```
op: add_hole radius: 3.9251 mm location: (-10.39, -19.0, -6.0) axis:
(0.866, 0.0, 0.5)
m(P): 3.26 mm floor: 0.155 mm, k = 3
```

2. Code channel

```
given: part (preloaded), instruction
returns: a Python block
 for face in base.Faces():
  s = BRepAdaptor_Surface(face)
  if s.GetType()==GeomAbs_Cylinder:
    if abs(s.Cylinder().Radius()
           -3.925)<1e-4:
      hole_face = face; break
d = BRepAlgoAPI_Defeaturing()
d.SetShape(base); d.AddFaceToRemove(...)
result = cq.Shape.cast(d.Shape())
```

3. Structured channel

```
given: feature menu of P, instruction
returns: one JSON object
 menu (excerpt):
{center: [10.09,-0.43,-5.83],
 axis: [-0.5,0,-0.866], radius: 1.0}
{center: [0.1,-19.01,0.27],
 axis: [0.866,0,0.5], radius: 3.925}
 answer:
{"op": "remove_hole",
 "center": [0.1, -19.01, 0.27],
 "axis": [0.866, 0.0, 0.5],
 "radius": 3.925}
```

Figure 5: One instance end to end (`1092630161494__add_hole__0`, detection tier). We cut a Ø7.85 mm hole into the reference solid R to obtain the model's input P, and the instruction is "Remove the added Ø7.85 mm through-hole and restore the face." Both answers come from the same model and both restore R. The structured answer is verbatim. The code answer is abridged, with imports and error checks omitted and variables renamed. The code answer must locate the face by iterating the B-rep and matching a radius, then drive the defeaturing API correctly, whereas the structured answer selects a row of the feature menu. Both are executed to STEP and scored identically. An answer is region-correct if the changed-surface shell is within k·floor of R, and local if nothing outside it moved.

```
The current model is ALREADY LOADED as part. Start from it
and modify ONLY the feature the instruction targets. Do NOT
rebuild the part from scratch. Every part of the geometry
the instruction does not mention MUST remain exactly as
in part. Your edit is scored on (a) whether the targeted
region becomes correct AND (b) whether the rest of the part
is left untouched --- global rebuilds fail (b).
```

The structured channel instead states the schema and the instruction-to-operation mapping, and requires the model to copy coordinates from the enumerated menu rather than invent them. Neither prompt names a face index, and neither reveals R.

The reconstruction tasks add the dimension inventory, a JSON object read from the ground-truth B-rep that lists primitive dimensions by surface and curve type rather than a curated set:

```
{"cylindrical_radii_mm": [0.5, 1.0, 1.35, ...],
"cylindrical_radii_counts": {"0.5": 4, ...},
"blend_face_radii_mm": [0.5, 2.0], "cone_half_angles_deg":
[...], "sphere_radii_mm": [...], "planar_face_count": 200,
"surface_type_counts": {...}, "edge_length_histogram":
{...}, "n_edges_measured": 901, "note": "exact B-rep
primitive inventory by surface/curve type; NOT a curated
dimension set"}
```

E Editing Results, Per Model

Figure 1 shows both editing tracks as failure-mode decompositions, and Table 10 gives the underlying counts, so every segment of those figures can be read off exactly. The panel is OpenRouter, temperature 0, one sample per instance. Each row sums to the track's instance count (301 certified single-part, 126 certified assembly), because every answer falls into exactly one class, namely not executable (no P' exists to score), executable but missing the target region, region-correct with collateral over the gate, or `edit_success`. The denominator is every attempted instance, never

Table 10: Editing results per model and arm, over every attempted instance in each track. Counts are not-executable, executable-but-wrong-region, region-correct-with-collateral and `edit_success`, and sum to the track's instance count. The last two columns of each panel give the executable rate and `edit_success` as fractions. GPT-6 and Fable answer the detection tier only. On the assembly track the structured channel nearly empties the first column without closing the gaps in the last one.

	single part, n=301						**assembly**, n=126					
model	not exec.	wrong reg.	collat.	succ.	exec.	succ	not exec.	wrong reg.	collat.	succ.	exec.	succ
pointed tier, code channel												
Claude	36	152	44	69	.88	.23	35	56	1	34	.72	.27
GPT	104	58	15	124	.65	.41	14	36	2	74	.89	.59
GLM	148	80	21	52	.51	.17	55	34	2	35	.56	.28
DS-Pro	215	56	8	22	.29	.07	73	30	1	22	.42	.17
DS-Flash	183	90	3	25	.39	.08	76	35	0	15	.40	.12
pointed tier, structured (DSL) channel												
Claude	48	62	37	154	.84	.51	0	88	3	35	1.00	.28
GPT	51	63	32	155	.83	.51	0	52	1	73	1.00	.58
GLM	43	64	38	156	.86	.52	16	61	4	45	.87	.36
DS-Pro	47	69	32	153	.84	.51	0	80	4	42	1.00	.33
DS-Flash	49	70	34	148	.84	.49	1	71	6	48	.99	.38
detection tier, code channel												
Claude	49	100	34	118	.84	.39	40	62	1	23	.68	.18
GPT	112	54	10	125	.63	.42	7	81	0	38	.94	.30
GLM	192	53	11	45	.36	.15	51	53	0	22	.60	.17
DS-Pro	279	12	1	9	.07	.03	72	45	1	8	.43	.06
DS-Flash	260	30	2	9	.14	.03	83	38	0	5	.34	.04
GPT-6	107	53	10	131	.64	.44	15	55	1	55	.88	.44
Fable	19	73	18	191	.94	.63	4	75	0	47	.97	.37
detection tier, structured (DSL) channel												
Claude	55	80	29	137	.82	.46	1	101	1	23	.99	.18
GPT	52	66	32	151	.83	.50	0	80	0	46	1.00	.37
GLM	69	68	32	132	.77	.44	21	73	0	32	.83	.25
DS-Pro	52	84	29	136	.83	.45	2	101	1	22	.98	.17
DS-Flash	61	83	29	128	.80	.43	4	102	4	16	.97	.13
GPT-6	46	72	30	153	.85	.51	0	73	1	52	1.00	.41
Fable	46	66	36	153	.85	.51	1	79	0	46	.99	.37

the scored subset. The file `scored_*.jsonl` holds only runnable submissions, and dividing by it would reward a model for failing to execute. On the assembly track at the pointed tier in code, for instance, DS-Pro (17%, 42% executable) would overtake Claude (27%, 72%) on the scored subset. No answer in these eight arms was lost to a gateway error, so no instance is dropped from any denominator. Short names follow the Setup paragraph of Section 5. GPT-6 and Fable answer the detection tier only, and Gemini and Kimi appear only in the parametric and reconstruction sections.

Which instances separate the models. In the structured channel 92% of the certified instances receive a unanimous verdict from the five models at the pointed tier and 84% at detection, against 6% and 7% expected if the models passed independently at their own rates. Table 11 splits that unanimity by cause. Instances beyond the channel's ceiling fail for every model by construction, and below the ceiling the channel saturates, with most of the remaining instances passed by every model. The code channel has almost no unanimous passes, and most of its instances separate the models. On the 184 instances the ceiling admits, the two frontier models are indistinguishable in the structured channel when told the location and separated when they must find it, while in code the pattern reverses for the reason given next.

The ceiling is a fixed set of 117 instances, the same in every row of Table 11. What varies is how many of them every model fails, because a model can pass an instance the oracle's search did not. In code at least one panel model passes 34 of the 117 at the pointed tier and 45 at detection (63

Table 11: Which of the 301 certified single-part instances separate the five panel models. The ceiling is the structured channel's (the model-path oracle of Table 1, after the payload search), and the same 117 instances lie beyond it in every row. *Beyond, all fail* counts those of the 117 that every model fails, so it varies by row, and the rest of the 117 count as separating. *All pass* counts instances every model passes, *all fail* those inside the ceiling that every model fails, and *separating* the rest. The last three columns are read on the 184 instances inside the ceiling. They give the spread of success rates across the panel, the discordant pairs between Claude and GPT (Claude only / GPT only) and the exact McNemar p between them.

		all 301 instances				184 inside the ceiling		
tier	channel	beyond, all fail	all pass	all fail	separating	spread (pts)	Claude/ GPT	p
pointed	code	83	5	58	155	44.0	14/57	$<.001$
pointed	structured	117	140	19	25	4.3	6/7	1.000
detection	code	72	2	37	190	50.5	40/55	.151
detection	structured	117	106	29	49	12.5	3/17	.003

once GPT-6 and Fable are counted), so the ceiling is a property of the structured channel, not of the instances. In the structured channel no panel model passes any of them at either tier, and the 117 therefore appear unchanged in both structured rows. The one structured answer beyond the ceiling is Fable's, outside the panel, on `1630111479193__add_hole__1`, where it states a point on the axis between two coaxial menu rows and the executor fills a different extent of the bore, a payload the search never tried. The ceiling therefore bounds the panel in aggregate but not every instance. Before the no-op test of Appendix C, a DS-Flash answer also passed beyond the ceiling, by removing an unrelated small fillet from an instance whose injected fillet lies below the gate. That instance is one of the 28 the no-op test removes.

Why the pointed tier can hurt code. In the code channel Claude scores 23% at the pointed tier and 39% at detection (Table 10). On the 152 certified instances whose correction removes an added hole, fillet or chamfer, the two tiers elicit different programs. Given the defect's coordinates, Claude builds a filler solid at that location and unions it with the part in 58% of its answers, and those pass 6% of the time. It calls the kernel's defeaturing operation in 41%, which pass 33% of the time. Without coordinates it has to find the feature in the B-rep, and it then defeatures in 93% of its answers, which pass 60% of the time. GPT defeatures in 91% and 95% of its answers at the two tiers and scores 41% and 42% overall. We classify strategies by the kernel calls each answer makes (defeaturing, union or fuse, cut). A stated location is therefore not neutral in code, and pointed minus detection measures localization cost cleanly only in the structured channel, whose executor resolves the named feature either way.

E.1 The Information-Matched Control

The structured channel names its target from an enumerated feature menu, whereas the code channel is given the solid and a summary of it. That is a difference in access as well as in answer format, so we ran the code channel with the *identical* menu in place of its summary, with everything else held fixed, including the system prompt, instruction and output contract, at the detection tier (Table 2).

Two caveats apply. First, the arm is *partial*. The budget ran out mid-run, so each model covers 130–247 of the 301 certified instances rather than all of them. Coverage is not selected by difficulty, which we checked. Each model's code-channel rate on its covered subset reproduces its rate on the full arm within four points (36 against 39 for Claude, 38 against 42 for GPT-5.6-sol, 44 against 44 for GPT-6-sol, 15 against 15 for GLM, 5 against 3 and 4 against 3 for the DeepSeek pair), and every comparison below is paired on the covered instances. Second, the arm was collected at high concurrency, which starved the *executor* rather than the model and cost Claude 59 execution timeouts on its 247 covered instances, against 2 on the same instances in the code arm. An executable rate must not depend on the harness's worker count, so the affected answers, which are stored as files, were re-executed at low concurrency with a generous budget, symmetrically in both arms. This recovered 172 of 265 in the menu arm and 39 of 59 in the code arm. Thirty-three of the latter were

Fable's ordinary code-channel answers, which were collected under the same contention and are the ones Table 10 reports. The panel models' code-channel rates barely moved, by three instances for Claude and none for the others. Fable itself was not run with the menu. We dropped that arm for budget, and it is the least informative one to drop, because the control asks whether access to the menu explains the structured channel's advantage, and Fable has no such advantage (63% in code against 51% structured). Afterwards the menu arm has at most one timeout per model and the code arm at most two. Reporting the first reading would have turned a harness artifact into a finding about models, which is exactly the kind of error this benchmark is meant to catch.

The aggregate rows of Table 2 hide one further pattern. The two GPT models are unchanged in aggregate, yet a third or more of their instances change verdict, in both directions (the *disc.* column). That is symmetric churn rather than stability, and it is why the table reports the discordant pairs beside the exact McNemar p.

F Certifying the Two Structured Vocabularies

The editing and generation vocabularies are certified by arguments that bound in opposite directions. The difference is forced by the tasks, not chosen.

An upper bound for editing. An editing instance is produced by inverting a defect we injected, so the ground-truth correction is known and can be handed to the channel directly. The oracle does exactly that, mapping each injection log to its inverse operation and executing it. Being handed the correct answer is hard to improve on, so the result bounds model scores in that channel in aggregate, although a model that guesses well can exceed it on individual operators (Table 1). This is why a structured-channel panel must be read on the channel-feasible subset rather than as a free-standing comparison. A payload of the right operation exists for every instance by construction, but it does not always name a target the executor can resolve. On 18 certified instances the executor finds no target for the ground-truth payload (*unres.* in Table 1). Sixteen of them are blends it does not recognize, and a widened matcher that also accepts the cylinder of a straight-edge fillet and the plane of a straight-edge chamfer resolves 12 of those 16. The other two are bores whose axis line it cannot find. These 18 instances are not expressible in the channel as run. The empirical content of the rest is where the shortfall goes once a payload is supplied, whether to kernel refusals, geometry that misses the target, or collateral over the gate. One residual is concentrated in a single operator. On `add_hole`, 18 of 90 certified instances are region-correct yet fail locality, because undoing the injected hole requires deciding how far along its axis to fill, and the executor infers that extent from the input where a model writing code chooses it itself.

Two corrections separate a ceiling a model could attain from one only an oracle can. First, the oracle may resolve its target through the injection log, a selection side channel no model receives. On the 301 certified instances, running it down the model's own path lowers it from 177 to 170, about 2 points. Second, supplying only the ground-truth payload understates the channel, because a different legal payload sometimes passes where the exact inverse fails. We therefore audit reachability. For every instance on which the ground-truth payload fails, up to eight further payloads are built *only* from the enumerated menu the model is shown, applied and scored, and the instance counts as reachable if any passes (Table 12). Fourteen of 131 are rescued this way, all by a menu payload rather than the ground-truth one. The search is not uniformly deep. Eight instances (six `add_chamfer` and two `add_fillet`) offer no menu candidate of the required type, so only the ground-truth payload was tried on them, and 48 more offer fewer than eight. The eight are among the 18 unresolved targets above, since the menu lists a blend only when the matcher recognizes it. They count as unreachable, which is right for a channel that cannot name the target, but the 89% means "no payload this channel can express passes", not "eight alternatives were each tried". The corrected ceiling is 184 of the 301 certified instances, or 61%, against 177 (59%) for the privileged oracle, so the two corrections partly cancel. The substantive result is the other column. On 89% of the oracle's failures *no* payload the channel can express passes, so the shortfall is a property of the channel rather than of the parameters the oracle happened to supply.

How much the executor's tolerance contributes. The executor accepts a target within a size-relative tolerance, which for blend removal is as wide as a quarter of the part's diagonal (Appendix D). To test whether that tolerance does the localizing for the model, we tighten it after the

Table 12: Reachability audit over the 301 certified single-part instances, with the oracle run down the model's own path. Each instance whose ground-truth payload fails is retried with up to eight payloads built only from the menu the model sees. *Reachable* means some payload passes, so the channel can express the repair and the oracle's single payload understated it. *Unreachable* means none does, which is a genuine channel ceiling.

operator	oracle failures	reachable	unreachable
`add_hole`	38	6 (16%)	32 (84%)
`add_chamfer`	31	6 (19%)	25 (81%)
`add_fillet`	13	0 (0%)	13 (100%)
`resize_hole`	19	0 (0%)	19 (100%)
`remove_hole`	19	0 (0%)	19 (100%)
`remove_fillet`	11	2 (18%)	9 (82%)
all	131	**14 (11%)**	**117 (89%)**

Table 13: Tolerance ablation for the structured channel, 301 certified single-part instances, five panel models. An answer passes at gate g if it passed under the published executor and its stated coordinate lies within $g\,d$ of the target it acted on. The ranges and spreads are `edit_success` in percent across the five models, and the ceiling is gated the same way.

	pointed		detection		
gate	range	spread	range	spread	ceiling
published	49.2–51.8	2.7	42.5–50.2	7.6	61.1
$0.10\,d$	48.5–51.5	3.0	42.5–50.2	7.6	60.1
$0.05\,d$	48.2–51.5	3.3	41.5–48.8	7.3	60.1
$0.02\,d$	47.8–50.2	2.3	40.9–48.2	7.3	58.5

fact. For every successful structured answer we measure how far its stated coordinate lies from the target it acted on, as a fraction of d. For blend removal this is the executor's own distance, from the stated point to the bounding-box centre of the blend face it resolved. For the other operators it is the distance to the injected defect's changed surface, the same point cloud the scorer builds its shell from, less the payload radius for hole operators, since a point on a bore's axis lies one radius from its wall. An answer passes at gate g only if it passed and that distance is at most $g\,d$. A tighter gate can only reject answers, so nothing is re-executed or re-scored. The ceiling is gated the same way, with the oracle allowed to state the target's own menu row as a model does. At $0.05\,d$, a fifth of the published blend gate, no panel model loses more than 1.6 points, the spread stays at 3 points at the pointed tier and 7 at detection, and the ceiling falls by one point (Table 13). Nine in ten successful answers state a coordinate within $0.01\,d$ of their target. Fable is the most affected model and loses 3 points at detection. The flat panel is therefore not produced by the tolerance.

The assembly vocabulary. The assembly oracle is built the same way, from the injection log. A displaced component is returned by the rigid motion that fits its vertices onto its reference pose, a broken contact by the inverse of the logged shift, and a deleted member by duplicating a surviving twin onto each vacant pose. Each payload is applied by the real executor and scored by the assembly scorer (Table 14). Unlike the single-part oracle it is privileged, since a model is not given the reference poses, so it is read only as an upper bound. Expressiveness bound here. Seven of the 26 cardinality instances delete more than one member of a symmetry group, and under the original form of `duplicate_solid`, which repeats a single motion, none of the seven can be restored, because the missing members sit at irregular poses. We added per-copy motions before any model ran, and all seven now pass. The extension matters in practice. On the final panel the structured channel restores six of the seven for GPT at both tiers, and the code channel, which never had the restriction, restores six for GPT at both tiers and for GPT-6 and Fable at detection. The three pose failures fit their vertices exactly yet place the solid elsewhere, which points at the fitting method rather than at the vocabulary. We have not diagnosed the three single-removal cardinality failures.

Table 14: The assembly vocabulary's oracle, 126 certified instances, pointed payloads from the injection log, applied by the executor and scored by the assembly scorer. *Single motion* is the original `duplicate_solid`, which repeats one rigid motion, and *per-copy* the shipped form that gives each restored member its own. The two forms differ only on the seven instances that delete several members, which are the only ones re-run under the single-motion form. *Region* and *local* are the two sides of the metric for the shipped form, and every failure is a region failure.

		pass			
operator	n	single motion	per-copy	region	local
pose	58	55	55	55	58
connectivity	42	42	42	42	42
cardinality, one removed	19	16	16	16	19
cardinality, several removed	7	0	**7**	7	7
all	126	113	**120**	120	126

A lower bound for generation. The generation vocabulary covers sketch, extrude, revolve, loft and sweep, the boolean set, fillet, chamfer, shell, pattern and move, and an answer is a short program rather than a single operation. Constructing a solid to a specification has no unique program, so there is no single answer to supply. The coverage oracle instead detects each target's construction class from its own geometry (rotationally invariant solids are sectioned on the meridian half-plane and revolved, constant-cross-section ones are sectioned at mid-height and extruded) and executes the result through the same executor and metric. A miss means *not reachable by these two canonical constructions*, not *inexpressible*, so the score is a level the vocabulary demonstrably attains rather than a limit it cannot pass. This mechanically derived construction reaches mean IoU 0.866 on the target set, so the vocabulary suffices, but a better program might reach further and no model is capped by it. Unlike the editing oracle it must therefore never be divided out of a panel.

The API-reference control. Table 15 gives the control the introduction cites. The same two models answer the 301 certified detection-tier instances in the code channel with a fixed block appended to the system prompt, identical for every instance and both models, and everything else unchanged. The block shows how to iterate a part's faces and read a face's type, radius and axis, suggests matching a feature by surface type and radius, and gives generic recipes for the three kinds of kernel call the corrections need, namely defeaturing a set of faces, cutting or fusing a cylinder, and filleting or chamfering an edge. It names no instance's target, operation or parameter, but it is more than a list of signatures, since the recipes are the kernel mechanics of the edit. The arm without the block is the one in Figure 1, so the pairs are exact. For GPT the gain is entirely executability, and its accuracy on the answers that run is unchanged (66% and 64%). Without the block, 59 of its 112 non-executing answers call a defeaturing method that does not exist, and with it none do, so the failures it removes are lexical. Claude also becomes more accurate on the answers that run (47% to 58%), which the recipes, not the names, explain. An earlier version of this control, at a development gateway's default temperature on an earlier substrate revision, reported 9% to 42% for Claude. That deployment of the model executed 40% of its answers without the block, against 84% on the final setup, so the gap does not reproduce and we no longer cite it.

G The Channel Effect on Specification-Conditioned Generation

The main paper reports the channel comparison on editing only. Table 16 gives it for the parametric track, scored on the same final panel and at the same fast configuration as the code arm it pairs with. This is a second, independently built replication of the channel effect, with a different task family (from-scratch construction rather than repair), a different vocabulary, and a different executor. Nothing is shared with the editing channel but the finding.

Every model improves, and the size of the improvement is broadly inverse to how well the model scores writing kernel code unaided. The two weakest code arms gain most ($+0.096$ and $+0.082$), while the two strongest gain $+0.048$ and $+0.008$. This matches the editing result and is what the

Table 15: The API-reference control, 301 certified single-part instances, detection tier, code channel, final setup (temperature 0, one sample). *Gain* and *loss* are per-instance transitions in `edit_success` against the row above. The instance sets are identical, so each pair is matched, and an exact McNemar test on the gains and losses gives $p = 2 \times 10^{-9}$ for Claude and $p = 0.014$ for GPT.

model	API ref.	exec.	succ.	gain	loss
Claude Opus 4.8	no	84%	39%	—	—
Claude Opus 4.8	given	93%	53%	49	6
GPT-5.6-sol	no	63%	42%	—	—
GPT-5.6-sol	given	76%	49%	48	26

Table 16: Specification-conditioned generation through the two answer channels, single-part half, n=140, seven models at temperature 0 with one sample per instance. The denominator is all 140 attempted instances in both channels, so a non-executable or unscored answer counts zero. *Executable*, here and throughout, follows the main paper's convention that the answer ran and exported a solid. An answer that the scorer later declined to read is still executable and still scores zero. The *code* arm is the one reported in the main paper, and *struct.* emits a JSON sequence of atomic parametric operations that a verified kernel executes. Metrics are the house package at its fast configuration, aggregated over the $.50 : .99$ reporting sweep. Kimi's executability is bounded by its output cap in both channels and is a length result rather than a geometry one.

	executable		$\text{mIoU}_{.50:.99}$		
model	code	struct.	code	struct.	Δ
Gemini	78%	96%	**0.254**	**0.302**	+0.048
GPT	86%	**98%**	0.162	0.239	+0.077
Claude	84%	93%	0.231	0.238	+0.008
GLM	79%	78%	0.155	0.195	+0.040
DS-Pro	45%	81%	0.080	0.157	+0.077
Kimi	25%	35%	0.042	0.139	+0.096
DS-Flash	49%	77%	0.048	0.130	+0.082
panel mean	64%	80%	0.139	0.200	+0.061
panel spread	—	—	0.212	0.173	−0.039

channel is for. When the answer is a structured payload rather than a program, a model that knows the geometry but not the API is no longer penalized for the latter.

Most of the gain is executability rather than accuracy. The panel's executable rate rises from 64% to 80%, and the two DeepSeek models, whose code arms ran on only 45% and 49% of instances, reach 81% and 77%. There are two exceptions. GLM is flat on executability, 79% against 78%, and still gains $+0.040$ on score, so for that model the channel is buying geometric accuracy rather than runnability. Kimi rises only 25% to 35% because its ceiling is the output cap, not the API. It truncates in both channels (88 of its 140 single-part code answers), and its row should be read as a length result.

The panel spread narrows from 0.212 to 0.173 in the reporting band, the same qualification the main paper places on the editing channel. The structured vocabulary compresses the panel, so a channel comparison removes a confound rather than establishing a ranking.

The multi-solid half tells the same story on its own axis (Table 17). Exact cardinality is **75%** through the structured channel against 52% for code, monolithic answers 1%, and executability 88% against 64%. Every model improves, by between $+5$ and $+52$ points, and the largest gain belongs to the model whose code arm was least executable at 43%.

Table 17: The multi-solid half of the parametric track through both channels, n=77, seven models at temperature 0. Denominator is all 77 attempted instances. *Exact cardinality* is the fraction whose returned solid count equals the reference's. This table compares solid count only. Table 3 adds a per-part shape axis for the code arm. Δ compares the structured arm against code. An earlier structured run under an *asymmetric* prompt, in which only the code side stated the target solid count, scored 14% panel cardinality and so inverted this comparison cleanly enough to be mistaken for a channel result. That figure measures the size of the artifact, not a result, and we report it here rather than beside the corrected arm.

	executable		exact cardinality		
model	code	struct	code	struct	Δ
GPT	81%	**100%**	**75%**	**97%**	+22
Gemini	43%	99%	40%	92%	**+52**
GLM	**83%**	82%	57%	75%	+18
DS-Pro	55%	90%	47%	73%	+26
Claude	75%	94%	65%	70%	+5
DS-Flash	57%	83%	36%	62%	+26
Kimi	53%	65%	42%	55%	+13
panel	64%	88%	52%	75%	+23

H EDITING WITH THE SOURCE SOLID PRESENT

The parametric track in the main paper withholds the source solid. The model receives a target parameter row and an envelope and builds the variant from scratch. That is the artificial half of the condition. Real CAD work starts from an existing file, so we also ran the condition the track was designed around, in which the *source* variant is loaded and bound to a variable named `part`, and the instruction is to re-dimension it to the target catalogue size. Everything else is held fixed, including the 217 pairs, the six models at temperature 0 with one sample per instance, the answer channel, and the executor family. The two conditions differ in whether the source solid is present, so every comparison below is paired at the instance level. Only the code channel is available here, and that is measured rather than scoped. No source file in this corpus carries a feature history for an operation to set, and a closed re-dimensioning vocabulary covers under a third of the pairs, so the property that makes a structured channel a clean instrument is what stops it expressing a global re-dimensioning.

Why not shape overlap. Submitting the source unchanged is a legal answer here, and `result = part` is one line. That degenerate policy scores $\text{mIoU}_{.50:.99}$ 0.204, above nine of the fourteen model-by-channel cells this track reports, so a shape-overlap headline would rank a model that barely edits above most of the panel. A second obstacle rules the axis out entirely, and it is a property of source-given editing rather than of any model. The voxel scorer, at the same configuration and per-item budget as the no-op baseline, failed to return within 600 s on **36%** of this arm's single-part answers, against 7% for the source-withheld arm and 15% for the no-op. Neither denominator is then fair. Counting an unscored answer as zero gives a panel from 0.308 down to 0.021 against the no-op's 0.204, while restricting to answers the scorer returned gives 0.684 down to 0.147 against its 0.240. Two of the six models sit above the baseline under one convention and below it under the other. We report no voxel-overlap axis on this half.

What we score instead. We score spec compliance. Each catalogue symbol is mapped to a geometric predicate by the induced legend, and an answer is credited on a dimension when the measured geometry matches the stated value within 2% or 0.05 mm. Restricting that to the dimensions the instruction asked to change gives a floor-robust axis, the unchanged source satisfying 8.5% of them against 94.1% of the dimensions it was supposed to preserve. Scores come from the shipped spec checker, not the shape-metric package used elsewhere in this work. Validated against the reference's own specification, it gives 0.9899 micro over 1089 checked dimensions and scored every answer. Denominators are every attempted instance, a non-executable or unreadable answer scoring zero. Of the 217 instances, 175 carry a checkable dimension and 125 a checkable *changed* one, the single-part half contributing 122 and 94.

Table 18: Re-dimensioning with the source solid loaded. n=140 single-part and n=77 multi-solid instances, six models at temperature 0, one sample each, denominator all attempted. SCR_Δ is spec compliance restricted to the dimensions the instruction asked to change, scored by the shipped spec checker rather than the shape-metric package. *degen.* is the share of *executable* answers whose bounding box, solid and planar-face counts and radius multiset all match the *source* within the checker's own tolerance, i.e. the degenerate policy of handing back what was given. That test is calibrated at both ends. It fires on 100% of the no-op arm, and a geometrically perfect answer would trip it on 2 of 140 single-part instances. The no-op row is the source variant submitted unchanged.

	single part			multi-solid		
model	exec.	SCR_Δ	degen.	exec.	SCR_Δ	degen.
GPT	87%	**0.287**	3%	91%	**0.316**	1%
Claude	81%	0.198	6%	74%	0.200	4%
Gemini	46%	0.166	6%	48%	0.189	8%
DS-Pro	20%	0.134	4%	10%	0.032	12%
GLM	36%	0.121	6%	26%	0.126	5%
DS-Flash	17%	0.061	4%	9%	0.053	0%
panel (6)	48%	0.161	5%	43%	0.153	5%
no-op (source)	—	0.085	100%	—	0.116	100%
reference	—	1.000	—	—	1.000	—

Table 19: Paired contrast, single-part half, n=140. The two conditions share instances, models, sampling and executor family and differ only in whether the source solid is loaded. Compliance falls for every model. Executability splits the panel instead of shifting it, since the two strongest code authors are unmoved while the remaining four lose 25 to 43 points.

	executable			SCR_Δ		
model	no src.	source	Δ	no src.	source	Δ
GPT	86%	87%	+1	0.466	0.287	−0.178
Claude	84%	81%	−3	0.308	0.198	−0.109
Gemini	78%	46%	−32	0.405	0.166	−0.239
DS-Pro	45%	20%	−25	0.206	0.134	−0.073
GLM	79%	36%	−43	0.348	0.121	−0.227
DS-Flash	49%	17%	−32	0.198	0.061	−0.138

H.1 Compliance Falls, and Executability Splits the Panel

Table 18 gives the condition on its own terms. Every model clears the no-op floor on the single-part half except DS-Flash, which falls below it, and the panel ranking is led by the model that is also the most reliable. Degenerate returns are rare, 3–6% of executable single-part answers against a 1.4% false-positive rate, so handing the source back is not the failure mode that occurs. Set against the source-withheld condition, the result is uniform and negative.

Table 19 separates the two effects. Executability does not fall uniformly. It is unchanged for GPT and Claude and collapses for the other four. Building from scratch is done inside the fluent CadQuery API, whereas with a loaded B-rep the models reach for the underlying kernel bindings, where recall is far weaker. That is a fallback we observe, not an API limit, because CadQuery's selectors do operate on an imported solid and the models largely do not use them. Import failures rise from 3 to 104 across the arm, and a block of kernel symbols (`gp_OX`, `brepbndlib_Add`, `topods`) joins the fluent-API names the source-withheld arm misses. Compliance, unlike executability, falls for every model including the two whose executability is unaffected, so it is not the collapse propagating arithmetically. Conditional on the answer running, GPT loses 0.231 and Claude 0.172, on executable subsets that differ by two instances and one between the conditions.

Table 20: Spec compliance split by whether the instruction asked for the dimension to change, conditional on the answer executing, single-part half. Both axes are calibrated, since the reference scores 1.000 on each, and the unchanged source scores 0.085 and 0.941, the extreme of preserving everything and changing nothing. Conditioning on execution is a diagnostic, not the headline. It drops each model's own failures, and DS-Pro and DS-Flash execute on under a quarter of instances with the source present, so their rows rest on a small self-selected subset.

	changed dims			preserved dims		
model	no src.	source	Δ	no src.	source	Δ
GPT	0.542	0.311	-0.231	0.494	0.670	$+0.176$
Claude	0.444	0.272	-0.172	0.516	0.557	$+0.041$
Gemini	0.559	0.380	-0.179	0.600	0.658	$+0.058$
DS-Pro	0.451	0.493	$+0.041$	0.446	0.500	$+0.054$
GLM	0.470	0.395	-0.075	0.468	0.719	$+0.251$
DS-Flash	0.419	0.429	$+0.010$	0.380	0.571	$+0.192$
no-op (source)		0.085			0.941	
reference		1.000			1.000	

Table 21: The multi-solid half, n=77, denominator all attempted. *Exact card.* is the fraction whose returned solid count equals the reference's. *rel. Pass@CD* is per-part Chamfer normalized by each reference part's own diagonal and swept at the single-part thresholds, so the two halves of the track are commensurable. The package's own assembly sweep is in absolute millimetres over reference parts spanning 1.4–761 mm and carries no signal at this scale. No answer in the source-given arm fused the assembly into one body, against two in the source-withheld arm, both DS-Flash's.

	executable		exact card.		rel. Pass@CD	
model	no src.	source	no src.	source	no src.	source
GPT	81%	**91%**	**75%**	**75%**	0.022	**0.199**
Claude	75%	74%	65%	57%	0.012	0.132
Gemini	43%	48%	40%	29%	0.011	0.067
GLM	**83%**	26%	57%	17%	0.019	0.017
DS-Pro	55%	10%	47%	6%	0.010	0.014
DS-Flash	57%	9%	36%	9%	0.009	0.006
no-op (source)	—	—	—	**81%**	—	—

H.2 The Source Acts as a Preservation Prior

Table 20 splits compliance by whether the instruction asked for the dimension to change. Every model improves on the dimensions it was told to keep, by up to $+0.251$, and the four with the highest execution rates get worse on the dimensions they were told to change, the two with matched executability by 0.231 and 0.172. The two exceptions on the changed axis, $+0.041$ and $+0.010$, are DS-Pro and DS-Flash, whose rows rest on the under-a-quarter of instances they execute at all. The source solid behaves as a preservation prior. It supplies the geometry that should survive and pulls the answer toward the input on the geometry that should not.

This is not literal copying. Scored against the *source* parameter row on the dimensions it was told to change, which measures how much of the input survived where it should not have, the unchanged source gives 0.976 and the models 0.008 to 0.032. They change the dimension and land on the wrong value, more often than when they build the same part with no source in front of them. The two axes are the pair the injection-editing metric imposes, region correctness and collateral preservation, recovered here from a condition scored on a single axis and then split.

Table 21 confirms this on the multi-solid half, using a metric that shares nothing with the compliance checker, namely per-part correspondence and Chamfer distance computed by the shape package. Geometric agreement improves sharply. For GPT and Claude, the two models whose executability

Table 22: Two independent runs of the same arm (pointed tier, structured channel, 342 single-part instances, five models, temperature 0 and one sample per instance in both). Columns are `edit_success` over all 342 attempted instances. *flips* counts instances whose verdict changed between runs in either direction.

model	run 1	run 2	flips	rate
GPT	50.0%	49.4%	4	1.2%
Claude	49.7%	49.4%	5	1.5%
DS-Flash	48.0%	48.0%	8	2.3%
GLM	50.0%	50.0%	10	2.9%
DS-Pro	49.4%	50.3%	13	3.8%
panel	—	—	40	2.3%

the condition leaves intact, relative Pass@CD rises ninefold and elevenfold. This is the preservation prior again, now in per-part shape and pose rather than in dimensional compliance. The three models whose executability collapses gain nothing, because the denominator counts their failures. Solid count moves the other way, because 62 of the 77 pairs already have the target's solid count in the source, so the degenerate policy scores 81% against 75% for the best model, and the axis the main paper reports for the source-withheld condition stops being informative once the source is supplied. We print it with that baseline beside it rather than omitting it.

I Run-to-Run Stability at Temperature Zero

Every number in this work is a single greedy sample, so we measured what that costs. An earlier revision of this benchmark ran at the gateway's default temperature and two nominally identical assembly runs then disagreed on roughly a fifth of instances. We re-ran one arm end to end a second time under identical settings, namely the pointed tier through the structured channel on all 342 single-part instances and all five models (Table 22).

Forty of 1710 verdicts changed, a 2.3% flip rate, against the roughly 20% seen at the default temperature. Per-model scores move by at most 0.9 points and four of the five by 0.6 or less, so this arm's single-sample results are reproducible to within about a point, and its near-tie is a real tie. The five models span 2.0 points in the first run and 2.3 in the second, and the paired comparison between the two frontier models is 7 instances to 6 in the first run and 7 to 7 in the second. The bound should not be carried over to other arms. This is the arm the channel decides most completely (only 25 of the certified instances separate any two models, Table 11), and a short structured answer leaves less room for run-to-run variation than a program does, so the code channel would need its own replicate. This is a stability bound for one arm, not a confidence interval, and it says nothing about variance across corpus resampling.

J Assembly Scoring and Instructions

How an assembly edit is scored. The localized scorer reads R, P and the submission P' as sets of solids in the input frame and never pre-aligns them, since a global shift is itself a failed edit. Each solid is sampled with 4000 area-weighted points, and solids are matched by a Hungarian assignment on centroid distance, with ties broken on (area, volume) because concentric solids share a centroid exactly. A matched solid counts as unchanged when its centroid moves at most $\max(1\,\mathrm{mm}, 0.003\,d)$ and its Chamfer distance to its partner is at most kf_i with $k = 3$, where $f_i = \max(0.5\sqrt{A_i/4000},\ 0.2\,\mathrm{mm})$ is the sampling floor for that solid's own area A_i. Contacts are read at the tolerance the injector used, $\max(0.004\,d,\ 0.05\,\mathrm{mm})$. Each operator then applies the two-sided rule, and a submission succeeds when it is both region-correct and local.

- *Pose.* Region-correct when P' has R's solid count and the solid nearest the displaced component's correct centroid is unchanged relative to R. Local when every other solid is also unchanged relative to R, where those solids coincide with P's.

- *Connectivity.* Region-correct when the solid count matches and, after mapping P''s solids to R's through the correspondence, every contact of R is present. Local when the solid count equals P's, no contact absent from R appears, and every solid other than the moved one is unchanged relative to P.
- *Cardinality.* Region-correct when the solid count is restored and every solid of R is matched and unchanged. Local when every solid already present in P is preserved.

Whole-shape assembly axes. The whole-shape package, which scores the assembly half of spec-conditioned generation, reports four axes. *Cardinality* is the error in solid count. *Per-part geometry* uses a Hungarian assignment on the pairwise Chamfer cost matrix and the part-accuracy curve $\mathrm{PA}(\tau_p) = \frac{1}{n_R}|\{i : \mathrm{CD}_i < \tau_p\}|$, with unmatched parts entered as ∞, summarized by its trapezoidal area divided by the span of the sweep, $\tau_p \in [0.1, 5.0]$ mm in 0.1 mm steps at benchmark configuration and $\{0.2, 0.5, 1, 2, 5\}$ mm at fast. *Pose* is decoupled into a centroid translation error and a rotation error, the minimum geodesic between inertia principal-axis frames over the part's symmetry group, which removes spurious 180° penalties from arbitrary eigenvector signs. *Connectivity* compares contact graphs, where a reference edge exists wherever the minimum surface-to-surface distance falls below $0.01\,d$. Its score is the fraction of *reference* edges whose corresponded candidate solids come within τ_c, recall with no precision term, summarized the same way. On the parametric multi-solid targets, whose parts span 1.4–761 mm, the absolute part-accuracy curve stays below 0.07 for every model, which is why the main paper reports cardinality and the scale-relative rel. P@CD of Section 4.2 instead.

Instructions. The assembly track's three axes take the same three-tier treatment. As shipped, at the pointed and detection tiers respectively:

> **pose:** Return the displaced component near (0.0, 288.0, 49.2) mm to its correct pose (it was translated and rotated). / Return the displaced component to its correct pose.
>
> **connectivity:** Reconnect the separated component (solid 0) to restore its 1 broken contact(s). / Reconnect the separated component to restore its broken contact with the rest of the assembly.
>
> **cardinality:** Restore the 1 missing component(s) --- part of a symmetry group whose remaining identical copies are still present. / Restore the missing component(s) from a symmetry group of identical parts whose other copies are still present.

The cardinality wording follows the ill-posed defect axis named in the introduction. The injector removes a *proper subset* of a symmetry group so that a volume-matched twin survives in P for the instruction to refer to. Before that fix the instruction asked for geometry present nowhere in the input.

K What a Language Bottleneck Costs

The render-conditioned arm. Table 4 in the main paper reports reconstruction from renders over all attempted instances, so a non-executable, over-complex, timed-out or unscored answer counts zero. Fifteen of GPT's answers exceed the 800 KB candidate-complexity guard and score zero, against two for each of the other models. Scoring those answers instead of zeroing them would give GPT 0.418/0.533, Claude 0.172/0.371 and Gemini 0.174/0.287 (mIoU/meanIoU), with seven of GPT's fifteen still not returning. We keep the guard, applied identically to every model. Kimi answered 81 of 85, and 56 of those carried no extractable code, 31 stopping at the output cap and 25 returning prose, so its row is a length result. The *box* floor is a solid at the reference bounding box, which the dimension inventory already gives the model.

Prose instead of renders. Reconstruction from renders admits a controlled comparison the other tracks do not, because the same part can be presented as images or as prose describing those im-

Table 23: Text-only reconstruction, n=85, temperature 0, published convention (a non-executable, over-complex, timed-out or unscored answer counts zero). *box floor* is a solid at the reference bounding box, and *retrieval floor* is the nearest other reference in the set. The dimension-only arm is the control and is run on the two models fluent enough for the contrast to carry signal.

arm	exec.	meanIoU	mIoU	vs. box
box floor	—	0.313	0.053	—
retrieval floor	—	0.143	0.000	−0.170
GPT, description	85%	**0.379**	**0.203**	+0.066
GPT, dimensions only	84%	0.298	0.126	−0.015
Claude, description	73%	0.273	0.106	−0.040
Claude, dimensions only	82%	0.253	0.091	−0.060
GLM, description	64%	0.176	0.061	−0.137
DS-Pro, description	24%	0.062	0.014	−0.251
DS-Flash, description	22%	0.063	0.026	−0.251

ages. We wrote a build description for each of the 85 single-part instances with Gemini, shown the same renders and dimension inventory the reconstruction arm receives, then re-ran the panel on the description alone with no images. Gemini is absent from the table below for that reason, since a model evaluated on its own descriptions would measure self-consistency rather than comprehension. Both sides are the same parts, the same references and the same scorer, so the difference isolates the presentation.

We report three results, in decreasing order of strength. The first, which survives every test, is the modality gap. Against the same models' render-conditioned scores, prose costs GPT 0.085 meanIoU and Claude 0.087, the same magnitude for two models at very different absolute levels, and the paired comparison favours the images on 43 of 69 discriminating instances for GPT and 47 of 70 for Claude ($p = 0.053$ and $p = 0.006$ by sign test). A description written from the renders, by a model that could see them, recovers most but not all of what the renders carry.

The second result is weaker than it first appeared. A description beats a bare dimension list for both models, but by 0.081 meanIoU for GPT and 0.020 for Claude, and only the former is significant by a t-test ($p = 0.024$). The sign test is not (40 wins to 29, $p = 0.23$), so the mean moves on a handful of large wins rather than broadly. A 30-instance pilot of this same comparison put the GPT gain at 0.191 mIoU, two and a half times the full-scale value, which is why we report the full set and treat small pilots as go/no-go gates rather than effect-size estimates.

The third is a caution about the track. No model separates from the trivial floor. GPT's best arm sits 0.066 above a bounding box at $p = 0.11$, Claude's sits below it, and the three weaker models sit significantly below ($p < 0.001$). We therefore report the modality comparison, which is a within-model contrast and immune to the floor, and do not publish a ranking on a task none of the panel demonstrably solves. Claude's lower execution rate on prose (73% against 82%) held from 46 instances to 85 and is real, but it costs nothing. Its description arm still outscores its dimension arm, so the instances it fails to run are ones it answers badly either way.

L Discussion and Limitations

The lesson extends beyond CAD. Whenever a benchmark requires the answer to be expressed in a demanding formal language, the language is part of the measurement, and a benchmark that does not control it reports the language's difficulty as the model's. Appendices F and G show what controlling it looks like in practice, with an achievability oracle on each vocabulary and a second answer channel that removes the API from the answer without removing the geometry. The corollary is that a near-zero axis is not a finding until such an oracle has run, and that the baselines which expose a broken axis (a bounding box, a retrieval lookup, a degenerate policy that resubmits its own input) cost almost nothing next to the inference they audit.

Four limitations bound what we report. *Sampling.* Every number is a single greedy sample on one gateway with no confidence intervals. Appendix I measures rather than assumes what that costs, and

bounds it at about a point for one arm, but that arm is the one the structured channel decides most completely, so it licenses no error bars elsewhere and says nothing about variance across corpus resampling. *Leakage.* The parametric variants and the main corpus are the same catalogue products, so pretraining exposure is structural. We calibrate against it with a retrieval baseline rather than eliminate it, and that baseline would not detect a model that had memorized the exact target variant. *Ceilings.* Executor success below 100% on some operators caps what any structured-channel score can reach, which is why those panels are read on the channel-feasible subset. The ceilings are improvable rather than fundamental. *Metric fit.* The metric suite was designed for localized editing, so specification-conditioned generation violates its assumptions, with most items scoring below the headline band's lower threshold. In the source-given condition the voxel axis is withdrawn entirely, because the scorer does not return on 36% of the answers.

M CONCLUSION

MCAD-Bench treats the answer representation as a variable to control rather than an implementation detail, and measures what controlling it does across three task families on real supplier CAD. The tracks answer differently. Single-part editing turns out to be limited by the representation, since the flat panel the structured channel produces is set by that channel's ceiling and its saturation below it rather than by model convergence, and the one model fluent enough to exceed that ceiling in code is held back by the channel. Assemblies remain limited by the models. Specification-conditioned generation reproduces the channel effect but does not clear its own retrieval baseline, so we report it as a condition to be read floor-relative rather than as a fourth task family, and reconstruction from renders has a clear leader whose margin depends on the aggregation. The appendices point the same way. The channel effect replicates outside editing altogether, and supplying the source solid acts as a preservation prior rather than simply making the task easier, on a condition that admits no structured channel at all. Locality, not modality, decides where the answer channel can be controlled. One practice underlies every track, and we recommend it to anyone extending this benchmark. No evaluation axis should be published without the score a degenerate policy attains on it, and no near-zero axis until an achievability oracle has shown the task answerable and the metric sensitive. Applied to our own pipeline, this practice repeatedly exposed apparent model findings as properties of the benchmark.