CRAFT: Corrective and Robust Multi-Agent Framework for Text-to-Parametric CAD

Mohammad Musthafa Rafi[a] Anushrut Jignasu[a] Mahdi Saraeian[a] Chinmay Hegde[b] Aditya Balu[a]
Adarsh Krishnamurthy[a]
Email: [a] (mohd7 — anushrutg — mehdi — baditya — adarsh)@iastate.edu , [b] chinmay.h@nyu.edu

***Abstract*—Text-to-CAD generation has the potential to make mechanical CAD more accessible, but existing approaches face a trade-off between data dependence and output fidelity. Supervised methods rely on large paired text–model datasets, while zero-shot prompting strategies are often brittle and tend to lose the parametric structure needed for downstream editing and customization. We present CRAFT, a corrective and robust multi-agent framework that produces parametric OpenSCAD programs from natural language without task-specific training. At its core is a JSON intermediate representation that preserves symbolic parametric expressions throughout the generation process, enabling models to remain editable rather than collapsing into hard-coded geometry. CRAFT combines semantic understanding, parametric planning, compilation, rendering, self-correction, and component-level verification within a layered recovery process, using multi-view visual feedback to detect and repair geometric errors. Our results show that structured corrective generation produces reliable and customizable CAD models that are competitive with more heavily supervised alternatives.**

***Index Terms*—Text-to-CAD, parametric modeling, multi-agent systems, large language models, zero-shot generation, OpenSCAD**

I. Introduction

Text-driven geometric modeling has seen rapid progress in recent years, with large language and vision models enabling increasingly capable systems for generating 3D shapes directly from natural language descriptions [1–4]. These advances have raised the prospect of making mechanical computer-aided design (CAD) accessible to users without specialized CAD expertise, reducing the barrier between a design intent expressed in words and a manufacturable geometric model. Despite this progress, existing approaches remain brittle in practice. Supervised methods that fine-tune models on paired text–CAD datasets can produce geometrically plausible outputs within their training distribution, but generalize poorly to novel shapes and require substantial curated data to train [5–8]. Zero-shot prompting strategies avoid this data dependency but are prone to generating syntactically invalid code, non-renderable geometry, and outputs that fail to faithfully reflect the original prompt. Critically, both families of approaches tend to produce hard-coded numeric geometry rather than editable parametric models, limiting their utility for downstream engineering and customization tasks [9, 10].

Agentic systems offer a natural path toward addressing these limitations [11, 12]. Rather than treating CAD generation as a single forward pass, a multi-agent pipeline can decompose the problem into discrete stages, including semantic understanding, structured planning, code synthesis, and verification, each of which can be independently validated and repaired. This staged architecture allows errors to be detected and corrected locally rather than propagating silently into the final output, improving both reliability and geometric fidelity. A particularly well-suited target for such a pipeline is OpenSCAD [13], a declarative scripting language in which geometry is expressed as a program over symbolic parameters. Generating OpenSCAD code rather than mesh geometry preserves the parametric relationships that make CAD models editable and reusable. However, verifying the correctness of generated OpenSCAD programs is non-trivial: a program may compile successfully yet produce geometry that is visually or dimensionally incorrect relative to the prompt. Effective verification, therefore, requires grounding generated outputs in external reference knowledge, both to check that recognized components conform to standard specifications and to confirm that expected parts are geometrically present. NopSCADlib [14] is a parametric OpenSCAD component library that provides catalog-defined specifications for standard mechanical parts. These specifications ground generated outputs in physically meaningful dimensions and enable verification of component completeness.

We present CRAFT (Fig. 1), a corrective and robust multi-agent framework for text-to-parametric CAD generation. CRAFT produces executable and editable OpenSCAD programs from natural language without task-specific training through three core mechanisms: a JSON-based intermediate representation that preserves symbolic parametric expressions, multi-view visual feedback to detect and correct geometric errors, and a layered recovery strategy to handle failures at every stage of the pipeline. Our main contributions are as follows:

- We develop a corrective and robust six-stage multi-agent framework for zero-shot text-to-parametric CAD generation that combines structured planning, multi-view visual verification, and layered error recovery to produce executable OpenSCAD programs from natural language.
- We propose a JSON-based intermediate representation that preserves symbolic parametric expressions throughout the generation process, enabling the produced CAD programs to remain editable and reusable rather than collapsing into hard-coded geometry.
- We evaluate CRAFT across three datasets (NopSCADlib,

979-8-3195-1246-8-0/26/$31.00

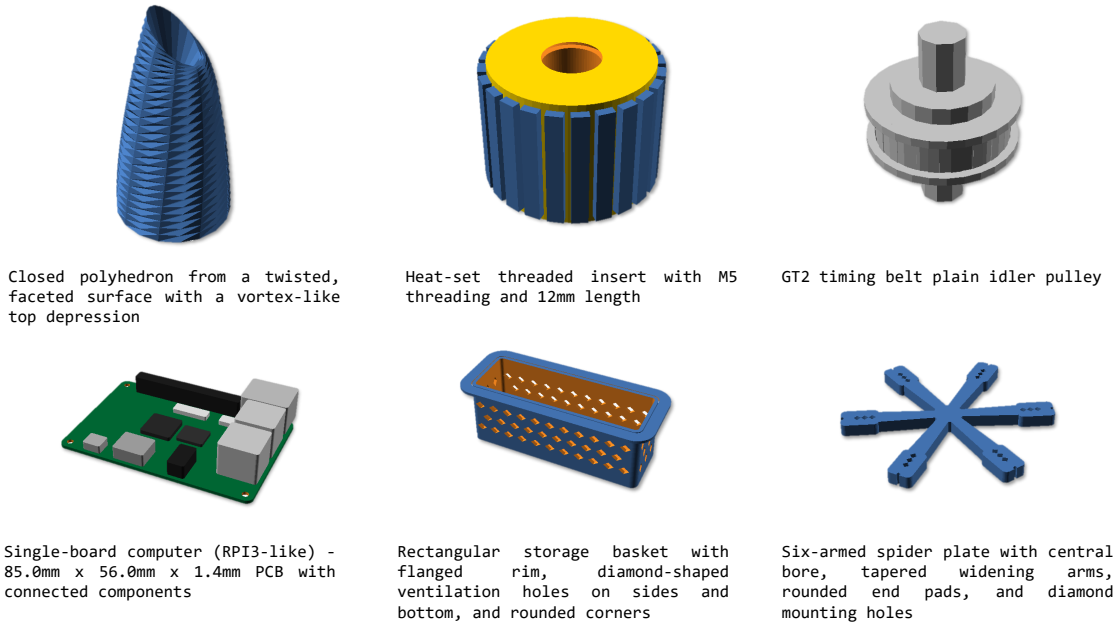

Fig. 1: *Examples of different CAD models created by CRAFT from the prompts.*

ABC, and Slice-100K) and show that it achieves competitive perceptual and geometric quality relative to direct generation baselines.

II. Related Work

Text-to-3D and Text-to-CAD Generation. Generative approaches to 3D modeling have advanced rapidly, spanning neural implicit representations [2], point cloud synthesis [4], and shape generation from text [3]. While these methods produce visually compelling outputs, they typically yield fixed mesh geometry that cannot be edited parametrically or reused in engineering workflows. A parallel line of work targets CAD-specific representations. Text2CAD [5] generates sequential CAD construction sequences from natural language, while LLM4CAD [8] explores multimodal models for 3D CAD generation. CAD-Coder [6] fine-tunes a language model on text–CadQuery pairs with reinforcement learning using geometric rewards, and Text-to-CadQuery [7] scales this paradigm through larger paired datasets. These supervised approaches achieve strong in-distribution results but require curated paired training data and produce outputs that are often not directly editable as parametric programs. CRAFT avoids task-specific training entirely and instead uses structured zero-shot generation to produce symbolic, editable OpenSCAD programs.

Program Synthesis and Code Generation. Generating executable code from natural language has become a well-studied problem, with large language models demonstrating strong performance across programming tasks [15]. Techniques such as chain-of-thought prompting [16] and self-refinement [12] have shown that decomposing generation into reasoning steps improves correctness on structured outputs. These ideas are particularly relevant for CAD, where generated programs must satisfy multiple simultaneous constraints: syntactic validity, successful compilation, geometric correctness, and semantic alignment with the input prompt. Direct single-shot generation is often insufficient in this setting, motivating the use of intermediate representations and iterative repair. CRAFT operationalizes this insight through a JSON-based intermediate representation that decouples planning from code synthesis, enabling schema-level validation before any OpenSCAD code is produced.

Verification and Visual Feedback. Recent work has begun to address correctness verification as a first-class concern in CAD generation. CADCodeVerify [17] introduces vision-language model-based verification and iterative correction as a post-generation step, achieving measurable quality improvements without retraining. More broadly, visual feedback has emerged as a powerful signal for program repair: rendering a generated program and comparing the output against a reference provides information that is difficult to obtain from code analysis alone. CRAFT extends this direction by rendering six orthographic views rather than a single image, enabling the VLM to detect geometric errors that are invisible from any single viewpoint, and integrating this feedback within a bounded iterative correction loop rather than as a one-shot post-processing step.

Retrieval-Augmented Generation. Retrieval-augmented generation (RAG) has shown that grounding language model outputs in external knowledge sources improves both factual accuracy and domain specificity [18]. In the code-generation setting, retrieving relevant library components or API signatures at generation time can reduce hallucination and improve structural correctness. CRAFT applies this principle to parametric CAD by retrieving component specifications from

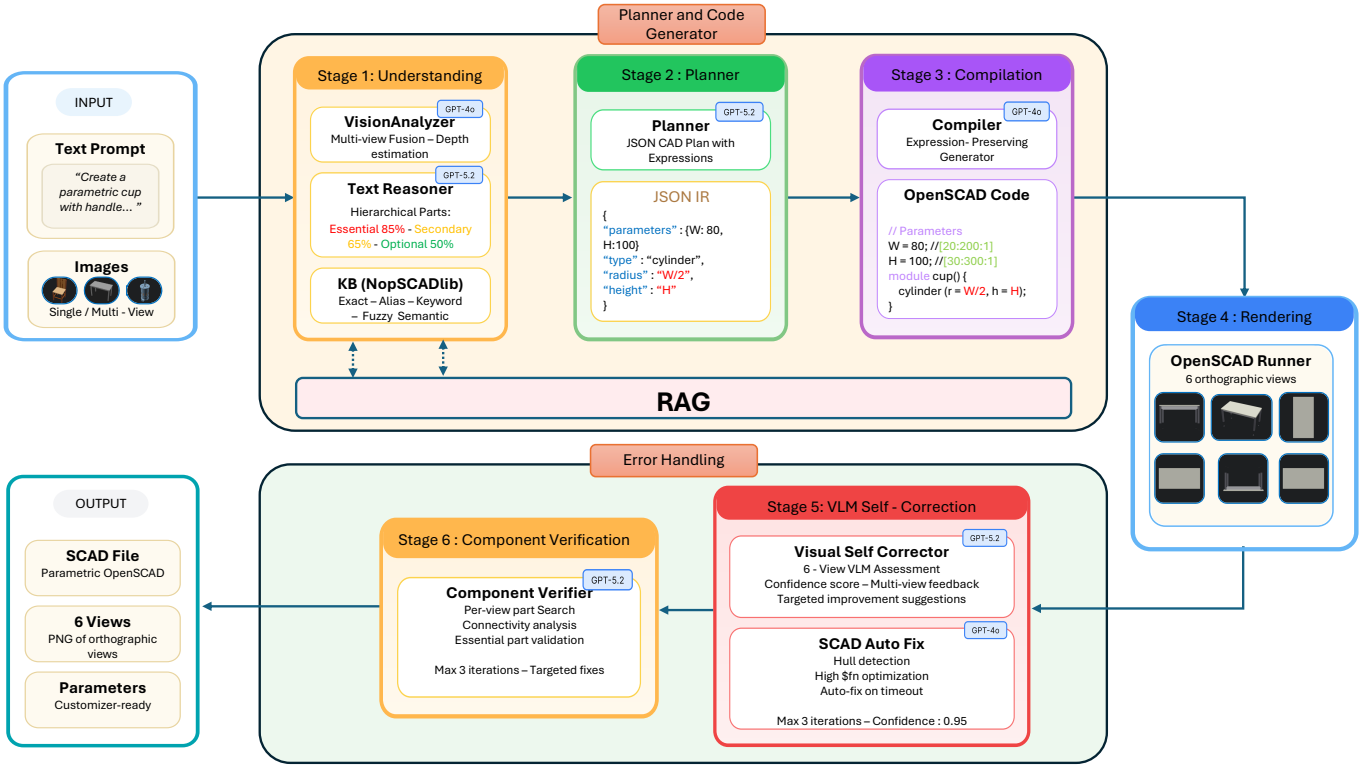

Fig. 2: *Overview of the CRAFT pipeline. The system maps a text prompt and optional reference images to a parametric OpenSCAD program through six stages: understanding, planning, compilation, rendering, multi-view visual correction, and component verification. Reasoning-intensive stages use GPT-5.2, while deterministic translation and repair steps use GPT-4o.*

NopSCADlib [14] during the understanding stage, grounding generated geometry in catalog-defined dimensions rather than relying on the model's implicit knowledge of part geometries.

Multi-Agent Systems. Multi-agent pipelines have demonstrated advantages over single-model generation in tasks requiring planning, execution, and verification across multiple steps [11]. Such systems can isolate failure modes and apply targeted repair without regenerating earlier stages by assigning specialized roles to distinct agents. This architecture is well-suited to CAD generation, where failures can occur at the planning, compilation, rendering, or geometric correctness levels, each requiring a different repair strategy. CRAFT adopts a six-stage multi-agent design in which each stage is independently recoverable, and coordinates these stages through a five-layer recovery strategy that escalates from cheap structural fixes to more costly semantic correction only when necessary.

Parametric Modeling. Parametric CAD represents geometry through symbolic variables and editable relationships rather than fixed coordinate values, enabling downstream customization, reuse, and engineering analysis [9, 10]. Despite this importance, most text-to-CAD systems emit hard-coded numeric geometry, discarding the symbolic structure that makes models practically useful. OpenSCAD [13] provides a natural target for preserving this structure through top-level parameter declarations and symbolic expressions. CRAFT is designed around this requirement: its JSON intermediate representation explicitly stores dimensions as symbolic strings, which are carried through compilation unchanged, ensuring that the final OpenSCAD programs remain fully editable.

III. Methods

CRAFT transforms natural language descriptions and optional reference images into parametric OpenSCAD code through six sequential stages. In our implementation, reasoning-intensive stages (understanding, planning, multi-view assessment, and component verification) are handled by a reasoning LLM (GPT-5.2), while deterministic translation and repair steps are handled by a traditional LLM (GPT-4o). The six stages describe the forward generation path, whereas the five recovery layers introduced in Section III-C describe how the system escalates repair when intermediate artifacts are invalid or the rendered result is unsatisfactory.

The pipeline (Fig. 2) begins with understanding (Stage 1), where image and text inputs are processed in parallel. The image-analysis path extracts geometric cues from any provided reference images and fuses information across views when multiple images are available. The text-reasoning parses the prompt to identify required components, classify them by importance (essential, secondary, optional), and detect references to standard parts such as motors or fasteners. When such components are recognized, their specifications are retrieved from NopSCADlib [14] using five strategies: exact name matching,

fuzzy name matching, category-based lookup, dimensional attribute matching, and semantic embedding retrieval. If retrieval fails, the pipeline proceeds with LLM-only synthesis using a retrieve-then-generate design pattern [18].

The output of stage 1 is passed to planning (Stage 2), where the planner produces a JSON CAD plan (Section III-A). This intermediate representation encodes geometry, constructive solid geometry (CSG) operations, and symbolic parametric expressions in a structured format that can be validated before any code is generated. In compilation (Stage 3), the validated plan is translated into OpenSCAD that preserves expression strings and generates parameter ranges for the customizer interface.

In rendering (Stage 4), OpenSCAD produces six orthographic views: front, back, left, right, top, and bottom. The renderer also applies timeout-aware simplifications for computationally expensive constructs such as `hull()`, which can become slow when computing enclosing geometry over multiple primitives. These views are then used in VLM self-correction (Stage 5, Section III-B), where a vision-language model compares the rendered output against the original prompt and iteratively proposes fixes for up to three rounds. Finally, component verification (Stage 6) checks whether the expected parts are present using tiered confidence thresholds of 85% for essential parts, 65% for secondary parts, and 50% for optional parts. These thresholds were set empirically during pilot runs to prioritize functionally necessary components over cosmetic details. This reflects the idea that missing mounting holes in a motor bracket constitute a failure, whereas omitting a cosmetic chamfer may still yield an acceptable design. More broadly, this iterative repair process is conceptually related to recent self-refinement approaches for large models [12].

A. JSON Intermediate Representation

CRAFT uses a JSON intermediate representation (IR) between design understanding and OpenSCAD generation. The IR encodes parameters, geometric primitives, and CSG operations in a structured form that can be validated before compilation. Unlike direct code generation, the IR preserves symbolic parametric expressions rather than evaluating them into constants. Expressions such as `"outer_diameter/2"` are passed directly into the compiled OpenSCAD code, allowing the final model to remain editable and fully parametric.

B. Multi-View VLM Assessment

Single-image assessment can miss geometric issues that are visible only from specific viewpoints. A model that appears correct from the front may still have incorrect depth or missing features visible only from the side. CRAFT addresses this by rendering six orthographic views (front, back, left, right, top, and bottom) and presenting them jointly to a vision-language model (VLM) for assessment. The VLM returns a confidence score on a 0–1 scale, a list of identified issues, and suggested corrections. If the confidence falls below 0.95, the system generates targeted fixes for the affected parts rather than regenerating the entire design. This loop runs for up to three iterations, with early stopping once the confidence threshold is reached. In our implementation, this assessment is performed using GPT-5.2 due to its stronger spatial reasoning capability.

C. Layered Error Recovery

The recovery layers describe how CRAFT escalates repair when intermediate artifacts fail validation, rendering, or semantic checks. Failures in text-to-CAD generation can arise at several levels: invalid intermediate plans, non-renderable code, and geometrically incorrect outputs. CRAFT addresses these through a five-layer strategy that orders repair mechanisms from inexpensive structural fixes to more costly semantic correction. The first layer repairs malformed JSON plans through schema validation and automatic repair. The second layer applies deterministic OpenSCAD fixes for common rendering failures, such as expensive operations or excessive mesh resolution. The third layer re-invokes the multi-view VLM assessment from Stage 5 to detect semantic and geometric mismatches and generate targeted corrections. The fourth layer re-invokes the component verification procedure from Stage 6 under the same tiered thresholds. The fifth layer incorporates manual repair hints when automatic recovery remains insufficient. Automatic recovery operates under a fixed retry budget of 10 total attempts across the first four layers, set empirically to balance correction capability against latency and cost. In the reported experiments, staged recovery resolved compilation and rendering failures before final output assessment, allowing the comparison to focus on geometric fidelity and semantic alignment of the final recovered outputs.

IV. Results

We evaluate CRAFT against direct generation baselines using GPT-4o and GPT-5.2 to examine whether its structured design improves semantic alignment with the input prompt, geometric faithfulness, and robustness under challenging generation settings, while retaining editable parametric relationships rather than collapsing into fixed geometry. Experiments are conducted on the NopSCADlib benchmark and two additional datasets (ABC and Slice-100K) to assess both in-domain performance and cross-dataset generalization. Detailed experimental settings are provided in Section A.i.

A. Perceptual Alignment

We first evaluate how well generated models match their text descriptions using CLIP Score and Fréchet Inception Distance (FID). CLIP Score measures semantic alignment between rendered CAD images and their prompts, while FID measures how closely the generated renderings match the distribution of ground-truth CAD images.

Table I shows that CRAFT achieves the strongest overall perceptual alignment on NopSCADlib. CRAFT reaches a mean CLIP score of 0.2223, which is the highest among the compared methods and closest to the ground-truth reference score of 0.2333. This advantage is maintained across all three complexity tiers, indicating that the structured pipeline

TABLE I: *Perceptual evaluation on NopSCADlib. CLIP Score measures text–image alignment (higher is better), while FID measures distributional similarity to ground-truth renders (lower is better). GT denotes the ground-truth CLIP score as a reference upper bound.*

Metric	CRAFT	GPT-4o	GPT-5.2	GT
CLIP Score	**0.2223**	0.2033	0.2145	0.2333
Simple	**0.2260**	0.2098	0.2174	0.2279
Medium	**0.2159**	0.2014	0.2144	0.2317
Complex	**0.2251**	0.1983	0.2116	0.2409
FID	**94.47**	118.01	104.63	—
Simple	154.65	144.85	**136.43**	—
Medium	**130.33**	155.06	143.33	—
Complex	**136.90**	184.38	160.51	—

TABLE II: *Geometric evaluation on NopSCADlib, including overall results and breakdown by prompt complexity. F1 is reported at a 1% distance threshold. Best results in each row group are shown in bold.*

Split	Method	CD↓	F1↑	Voxel IoU↑
Overall	CRAFT	0.0704	0.2369	**0.2242**
	GPT-4o	0.0742	0.2467	0.2182
	GPT-5.2	**0.0671**	**0.2585**	0.1870
Simple	CRAFT	0.0941	0.3054	0.2529
	GPT-4o	0.0840	0.3424	**0.3475**
	GPT-5.2	**0.0788**	**0.3480**	0.1567
Medium	CRAFT	**0.0563**	0.2256	0.2450
	GPT-4o	0.0650	0.2253	0.1807
	GPT-5.2	0.0583	**0.2342**	**0.2536**
Complex	CRAFT	**0.0593**	0.1744	**0.1712**
	GPT-4o	0.0737	0.1677	0.1207
	GPT-5.2	0.0638	**0.1871**	0.1500

helps preserve the semantic intent of the prompt consistently rather than only on simpler examples. FID follows a similar overall trend. CRAFT achieves the lowest overall FID of 94.47, compared to 104.63 for GPT-5.2 and 118.01 for GPT-4o, suggesting that its generated shapes more closely match the visual distribution of the reference CAD models. The gap becomes more pronounced on medium and complex components, where structured planning, component retrieval, and iterative verification appear to better preserve global shape organization. On simple components, GPT-5.2 achieves the lowest FID, suggesting that direct generation can be effective for basic geometry, while the advantage of CRAFT becomes more apparent as design difficulty increases.

B. Geometric Accuracy

We next evaluate geometric fidelity using Chamfer Distance (CD), F1, and voxel IoU. F1 score is reported at a 1% distance threshold. These metrics measure complementary aspects of reconstruction quality, including surface proximity, point-level matching, and volumetric overlap with ground-truth models.

Table II, Table III, and Table IV summarize geometric results on NopSCADlib, ABC, and Slice-100K, respectively.

TABLE III: *Geometric evaluation on ABC, including overall results and breakdown by prompt complexity. F1 is reported at a 1% distance threshold. Best results in each row group are shown in bold.*

Split	Method	CD↓	F1↑	Voxel IoU↑
Overall	CRAFT	**0.0804**	**0.1303**	**0.0401**
	GPT-4o	0.1009	0.0847	0.0211
	GPT-5.2	0.0950	0.0880	0.0187
Simple	CRAFT	**0.0825**	**0.1538**	**0.0353**
	GPT-4o	0.0978	0.0778	0.0288
	GPT-5.2	0.0903	0.0942	0.0148
Medium	CRAFT	**0.0701**	**0.1137**	**0.0325**
	GPT-4o	0.0994	0.0947	0.0152
	GPT-5.2	0.0936	0.0946	0.0146
Complex	CRAFT	**0.0884**	**0.1244**	**0.0519**
	GPT-4o	0.1059	0.0828	0.0180
	GPT-5.2	0.1006	0.0762	0.0261

TABLE IV: *Geometric evaluation on Slice-100K, including overall results and breakdown by prompt complexity. F1 is reported at a 1% distance threshold. Best results in each row group are shown in bold.*

Split	Method	CD↓	F1↑	Voxel IoU↑
Overall	CRAFT	0.0617	**0.2831**	0.0633
	GPT-4o	0.0743	0.2146	**0.0718**
	GPT-5.2	**0.0607**	0.2828	0.0627
Simple	CRAFT	**0.0414**	**0.4018**	**0.0763**
	GPT-4o	0.0601	0.2976	0.0537
	GPT-5.2	0.0439	0.3770	0.0609
Medium	CRAFT	0.0752	0.1918	0.0428
	GPT-4o	0.0783	0.1026	0.0498
	GPT-5.2	**0.0631**	**0.2364**	**0.0579**
Complex	CRAFT	**0.0682**	**0.2565**	0.0706
	GPT-4o	0.0837	0.2417	**0.1094**
	GPT-5.2	0.0743	0.2366	0.0687

On ABC, CRAFT achieves the strongest overall performance across all three metrics, obtaining the lowest CD, highest F1, and highest voxel IoU, suggesting that the structured intermediate representation and layered recovery improve generalization to unfamiliar geometry. On Slice-100K, results are more comparable: GPT-5.2 achieves the lowest CD, CRAFT achieves the highest F1, and GPT-4o attains the highest voxel IoU despite weaker surface-based metrics. On NopSCADlib, GPT-5.2 leads on CD and F1 while CRAFT achieves the highest voxel IoU, indicating that direct generation can yield slightly better average surface proximity on in-distribution examples, whereas CRAFT produces stronger volumetric consistency. Taken together with the perceptual results, CRAFT is particularly effective at producing geometrically coherent parametric models rather than optimizing any single surface metric in isolation.

C. Effect of Complexity

We break down results by prompt complexity to examine how performance varies with design difficulty. On NopSCADlib, direct generation remains highly competitive on simple components: GPT-5.2 achieves the lowest CD and highest F1, while GPT-4o attains the highest voxel IoU. When prompts describe basic geometry with limited part interactions, direct code synthesis often produces acceptable models. As complexity increases, CRAFT's advantage becomes more apparent. On medium-complexity examples, CRAFT achieves the lowest CD, though GPT-5.2 remains slightly stronger on F1 and voxel IoU. On complex examples involving multiple interacting features such as mounting holes, flanges, or internal cutouts, CRAFT achieves the lowest CD and highest voxel IoU. Although GPT-5.2 retains an advantage in F1 on medium and complex tiers, CRAFT shows more consistent geometric behavior as structural difficulty increases. On ABC, CRAFT achieves the best CD, F1, and voxel IoU at every complexity tier, with especially pronounced margins on complex components, indicating consistent generalization across prompt difficulty on previously unseen geometry.

Overall, these results support the design motivation behind CRAFT. Structured planning and layered recovery offer limited advantage on simple targets, but become increasingly useful when prompts require multiple relations, reusable dimensions, coordinated part composition, and recovery from intermediate errors.

D. Robustness and Recovery Behavior

Beyond final-shape quality, CRAFT is designed to improve robustness through bounded staged recovery rather than relying on a single forward generation pass. In the main evaluation, we report only the quality of the final recovered outputs, not per-stage failure counts. As a result, the geometric and perceptual gains in Tables I–IV should be interpreted primarily as end-to-end outcome quality after recovery, rather than as isolated measurements of intermediate compilation or repair success. Even under this conservative view, the stronger performance of CRAFT on medium-complexity, complex, and out-of-distribution examples is consistent with the role of structured planning and targeted correction in stabilizing generation when direct one-shot synthesis becomes brittle.

This robustness mechanism is especially important for text-to-parametric CAD, where failure can occur at multiple levels: malformed intermediate plans, non-renderable OpenSCAD code, incomplete part composition, or geometry that is syntactically valid but semantically inconsistent with the prompt. CRAFT addresses these failure modes through schema-level repair, deterministic OpenSCAD fixes, multi-view VLM-guided correction, and component-level verification within a bounded retry budget. Expanded qualitative illustrations of this layered recovery process and its role in improving generation stability are provided in the appendix.

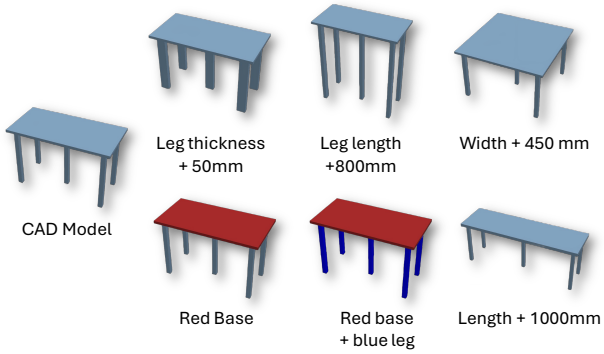

Fig. 3: *Examples of preserved parametric editability in generated CAD models. Changing individual parameters produces coherent geometric and appearance variations, showing that the generated OpenSCAD programs retain interpretable parametric relationships rather than only fixed final shapes.*

E. Preservation of Parametric Relationships

Beyond perceptual and geometric quality, we also examine whether the generated CAD programs preserve editable parametric structure. Fig. 3 shows representative examples where changing individual parameters produces coherent geometric and appearance changes. This indicates that CRAFT generates executable parametric programs rather than only fixed shape approximations.

V. Discussion

CRAFT achieves consistent improvements over direct one-shot generation across multiple datasets and object categories, with the largest gains on medium and complex designs. This result confirms that structured generation becomes more valuable as geometric dependencies and design complexity grow. The JSON intermediate representation is central to this behavior: it preserves symbolic parametric intent before code synthesis, ensuring that generated programs retain meaningful design controls rather than collapsing into fixed geometry. The qualitative parameter-edit results directly support this interpretation.

Several **limitations** should be noted. Performance declines as complexity increases, particularly when shapes involve many interacting parts or implicit structural relationships that are difficult to infer from text alone. OpenSCAD, while well suited for symbolic parametric modeling, does not capture the feature-history richness of industrial CAD systems such as Fusion 360 Gallery [10]. The multi-stage pipeline also depends on external language and vision models, increasing inference cost and introducing the risk of error propagation across stages. Finally, the evaluation covers three datasets, which represents only a subset of real-world engineering designs.

VI. Conclusions

We presented CRAFT, a multi-agent pipeline for text-to-parametric CAD generation that combines structured reasoning, a JSON-based intermediate representation, visual verification, and layered error recovery. Unlike direct code-generation

approaches, CRAFT explicitly preserves symbolic parametric relationships during generation, producing OpenSCAD programs that are executable, editable, and reusable. Experiments across three datasets show that CRAFT achieves strong perceptual and geometric quality while maintaining high reliability. The results demonstrate that preserving parametric intent through an explicit intermediate structure is both feasible and consequential for practical CAD systems.

REFERENCES

[1] L. Zhang, B. Le, N. Akhtar, S.-K. Lam, and T. Ngo, "Large language models for computer-aided design: A survey," *ACM Computing Surveys*, 2026, preprint: arXiv:2505.08137. [Online]. Available: https://doi.org/10.1145/3787499

[2] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "DreamFusion: Text-to-3D using 2D diffusion," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://arxiv.org/abs/2209.14988

[3] H. Jun and A. Nichol, "Shap-E: Generating conditional 3D implicit functions," 2023. [Online]. Available: https://arxiv.org/abs/2305.02463

[4] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, "Point-E: A system for generating 3D point clouds from complex prompts," 2022. [Online]. Available: https://arxiv.org/abs/2212.08751

[5] M. S. Khan, S. Sinha, T. U. Sheikh, D. Stricker, S. A. Ali, and M. Z. Afzal, "Text2CAD: Generating sequential CAD models from beginner-to-expert level text prompts," in *Advances in Neural Information Processing Systems*, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/0e5b96f97c1813bb75f6c28532c2ecc7-Paper-Conference.pdf

[6] Y. Guan, X. Wang, X. Xing, J. Zhang, D. Xu, and Q. Yu, "CAD-Coder: Text-to-CAD generation with chain-of-thought and geometric reward," in *Advances in Neural Information Processing Systems*, 2025, neurIPS 2025 (OpenReview). [Online]. Available: https://openreview.net/forum?id=QoiFdfZUJv

[7] H. Xie and F. Ju, "Text-to-CadQuery: A new paradigm for CAD generation with scalable large model capabilities," arXiv preprint, 2025. [Online]. Available: https://arxiv.org/abs/2505.06507

[8] X. Li, Y. Sun, and Z. Sha, "LLM4CAD: Multimodal large language models for three-dimensional computer-aided design generation," *Journal of Computing and Information Science in Engineering*, vol. 25, no. 2, p. 021005, 2025.

[9] R. Wu, C. Xiao, and C. Zheng, "Deepcad: A deep generative network for computer-aided design models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6772–6782. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Wu_DeepCAD_A_Deep_Generative_Network_for_Computer-Aided_Design_Models_ICCV_2021_paper.html

[10] K. D. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, "Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 54:1–54:24, 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3450626.3459818

[11] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," in *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, 2024, pp. 8052–8060. [Online]. Available: https://arxiv.org/abs/2402.01680

[12] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. [Online]. Available: https://openreview.net/forum?id=S37hOerQLB

[13] M. Kintel and C. Wolf, "OpenSCAD: The programmers' solid 3D CAD modeller," 2024, website. [Online]. Available: https://openscad.org

[14] nophead, "NopSCADlib: An ever expanding library of parts for OpenSCAD," 2024, gitHub repository. [Online]. Available: https://github.com/nophead/NopSCADlib

[15] M. Chen, J. Tworek, H. Jun *et al.*, "Evaluating large language models trained on code," arXiv preprint, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, vol. 35, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903

[17] K. Alrashedy, P. Tambwekar, Z. Zaidi, M. Langwasser, W. Xu, and M. Gombolay, "Generating cad code with vision-language models for 3d designs," *arXiv preprint arXiv:2410.05340*, 2024.

[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474. [Online]. Available: https://arxiv.org/abs/2005.11401

[19] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "ABC: A big CAD model dataset for geometric deep learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9601–9611. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Koch_ABC_A_Big_CAD_Model_Dataset_for_Geometric_Deep_Learning_CVPR_2019_paper.html

[20] A. Jignasu, K. O. Marshall, A. K. Mishra, L. N. Rillo, B. Ganapathysubramanian, A. Balu, C. Hegde, and A. Krishnamurthy, "Slice-100k: A multimodal dataset for extrusion-based 3d printing," in *Advances in Neural Information Processing Systems*, 2024, datasets and Benchmarks Track. [Online]. Available: https://arxiv.org/abs/2407.04180

[21] OpenAI, "GPT-4 technical report," 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[22] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 605–613. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Fan_A_Point_Set_CVPR_2017_paper.html

[23] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. PMLR, vol. 139, 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html

[24] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

A.I. Experimental Setup

Datasets. We evaluate CRAFT on three datasets totaling 668 components to assess both in-domain performance and cross-dataset generalization. Our primary benchmark is NopSCADlib [14], comprising 468 text–CAD pairs derived from parametric OpenSCAD components in the NopSCADlib mechanical library. Ground-truth models are derived from catalog-defined parametric specifications in the library, providing reference geometry anchored in standard mechanical dimensions. To test generalization beyond the primary benchmark library, we also evaluate on 100 shapes from ABC [19] and 100 shapes from Slice-100K [20]. These external datasets contain geometry outside the NopSCADlib library and therefore provide a more challenging test of whether the pipeline can synthesize coherent parametric programs for previously unseen shapes.

Prompt Complexity. For each dataset, components are grouped into Simple, Medium, and Complex tiers to study how generation quality changes with design difficulty. For NopSCADlib, tiers are assigned by source code analysis: a weighted composite of seven structural features—primitive count ($\times 1.0$), boolean operations ($\times 1.5$), CSG depth ($\times 0.5$), parameter count ($\times 0.3$), sub-module calls ($\times 2.0$), lines of code ($\times 0.05$), and transform count ($\times 0.3$)—split at the 33rd and 66th percentiles (boundaries: 36.50 and 66.25). These weights were chosen heuristically based on pilot inspection of code complexity and are intended only to define relative within-dataset difficulty tiers. This yields 159 Simple, 160 Medium, and 149 Complex components. For ABC and Slice-100K, where source code is unavailable, tiers are assigned by mesh geometry analysis using five features: face count (0.25), connected components (0.15), convex hull ratio (0.25), surface-to-bounding-box ratio (0.20), and compactness (0.15), with the same percentile-based splitting. These weights were likewise chosen heuristically to induce relative geometric difficulty groupings within each dataset. Because the NopSCADlib and external datasets use different feature spaces, these tiers should be interpreted as relative within-dataset groupings rather than directly comparable across datasets.

Baselines. We compare CRAFT against single-stage direct generation baselines using GPT-4o and GPT-5.2. Both baselines receive the same text prompt used by CRAFT along with a system prompt instructing the model to output only valid, executable OpenSCAD code. Generation uses temperature 0.0 (deterministic) in a single API call, with no intermediate JSON planning, no component retrieval, no visual feedback, and no error recovery. This comparison isolates the benefit of CRAFT's staged parametric planning and multi-layer recovery over direct code generation from the same underlying model family [21].

CRAFT Configuration. CRAFT uses a hybrid model allocation: GPT-5.2 for reasoning-intensive stages (understanding, planning, VLM self-correction, component verification) and GPT-4o for deterministic stages (compilation, repair). The planner is allowed up to 2 schema-repair attempts. VLM self-correction runs for up to 3 iterations with early stopping at confidence > 0.95, and component verification permits up to 3 iterations with tiered thresholds (Essential 85%, Secondary 65%, Optional 50%). Temperature is set to 0.0 for deterministic compilation and 0.7 for planning-oriented stages.

Metrics. We report both geometric and perceptual metrics. For geometric evaluation, we compute bidirectional Chamfer Distance [22]:

$$\mathrm{CD} = \frac{1}{2}\Big(\mathrm{mean}(d_{\mathrm{gt}\to\mathrm{pred}}) \quad + \mathrm{mean}(d_{\mathrm{pred}\to\mathrm{gt}}) \Big). \qquad (1)$$

which captures both accuracy (predictions close to ground truth) and coverage (ground truth adequately represented); F1 at a 1% distance threshold; and voxel IoU measuring volumetric overlap. Point clouds of 10,000 points are sampled from each mesh for CD and F1 computation; voxelization uses a 64^3 grid along the longest axis. For perceptual evaluation, we report CLIP score using ViT-L/14 [23] to measure text–image semantic alignment, and Fréchet Inception Distance (FID) [24] computed from 2048-dimensional features extracted at the average-pooling layer of Inception-v3 to measure distributional similarity between generated and ground-truth renders.

Rendering and Evaluation Protocol. For each method, the generated OpenSCAD program is compiled and rendered into multi-view images for perceptual evaluation, and the resulting geometry is exported to STL and converted into point clouds and voxel grids for geometric comparison against reference models. OpenSCAD renders all models at 512×512 resolution. STL export uses a 300 s timeout and PNG rendering uses a 120 s timeout to accommodate complex CSG operations. Six orthographic views (front, back, left, right, top, bottom) are rendered for VLM assessment and perceptual metric computation. CRAFT follows the full staged pipeline described in Section III, while baselines output a single OpenSCAD file evaluated using the identical metric pipeline. All methods are evaluated over the same set of prompts. When a method fails to produce a valid output under the evaluation protocol, the example is assigned worst-case metric values so that invalid generations remain reflected in the aggregate results.

A.II. Extended Ablation Studies

In particular, they show that the hybrid configuration improves component completeness, that multi-view correction contributes most strongly in the first two rounds, and that tiered part verification preferentially preserves functionally important components.

A.III. Layered Recovery and Multi-View Assessment

Fig. A.1 and Fig. A.2 illustrate two mechanisms most responsible for CRAFT's robustness: bounded layered repair and six-view visual feedback. These mechanisms are included here in expanded form because they are central to the engineering behavior of the system, but too detailed for the space-limited main paper.

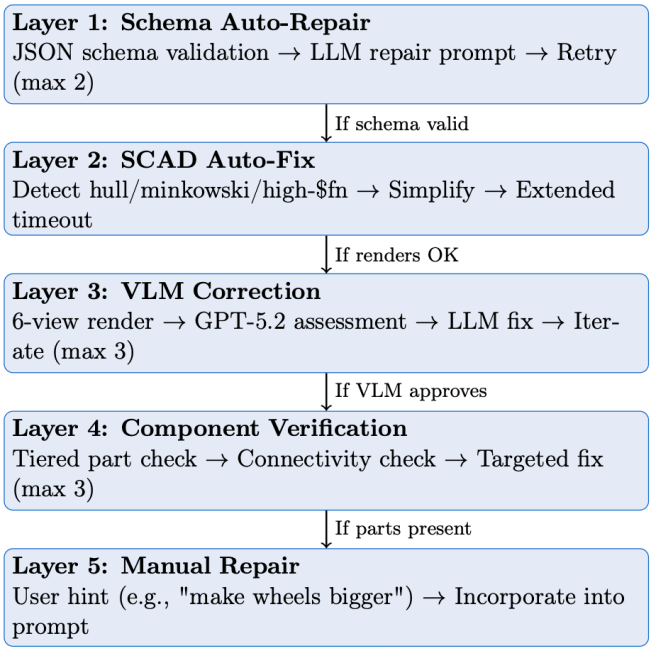

Fig. A.1: *Expanded view of the layered recovery strategy used in CRAFT. The recovery process progresses through five stages: schema auto-repair, SCAD auto-fix, multi-view VLM correction, component verification, and optional manual repair. Repairs proceed from inexpensive structural fixes to increasingly semantic correction, with bounded retry budgets at intermediate stages.*

A.IV. Additional Qualitative Examples

To supplement the benchmark metrics, we include additional qualitative examples across simple, medium, and complex prompts. These examples illustrate both the diversity of generated outputs and the role of preserved parameters in enabling post-generation edits.

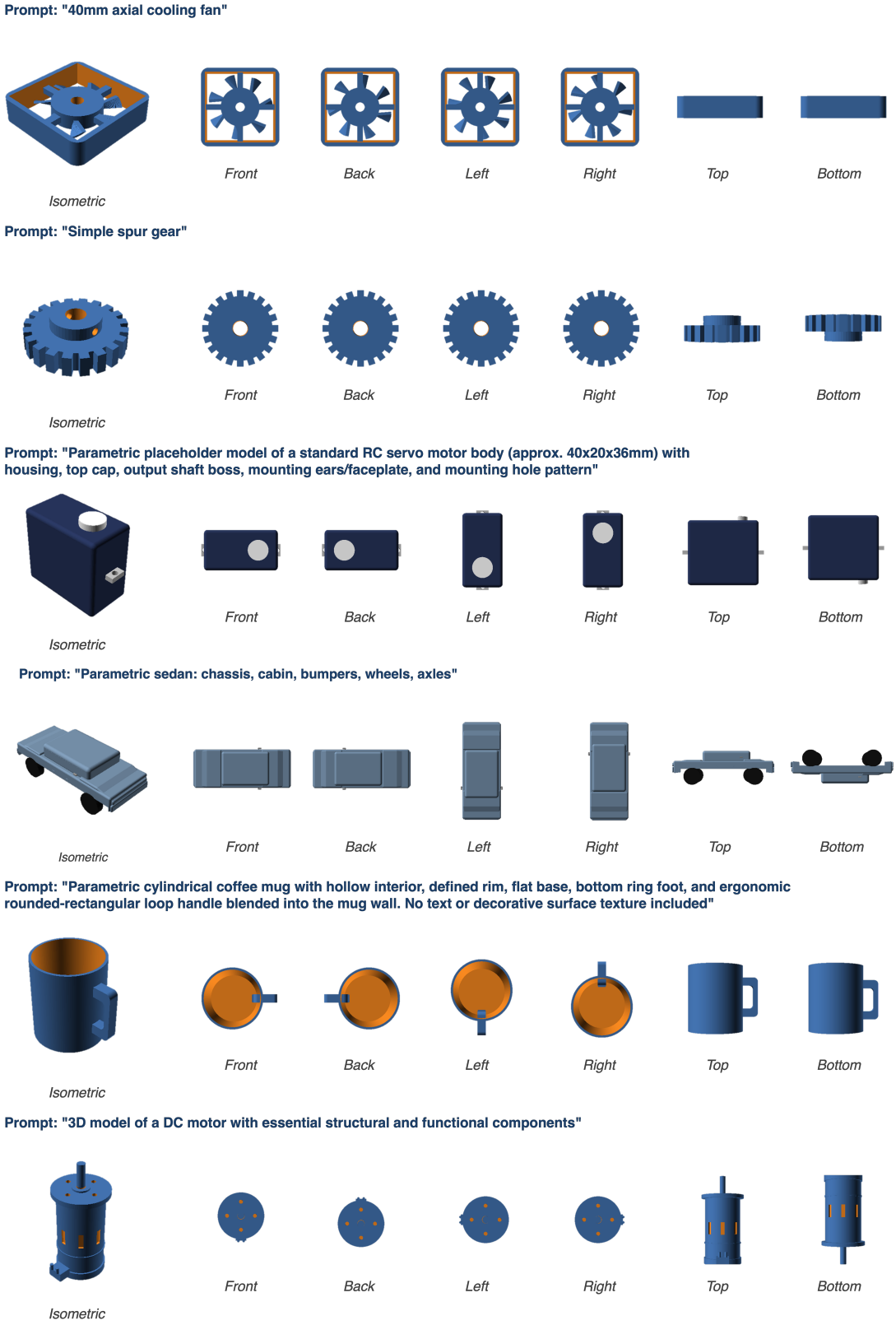

Fig. A.2: *Multi-view rendering outputs from CRAFT. Each row shows a generated model from a text prompt: the isometric output alongside six orthographic views used for VLM assessment. This spatial coverage helps reveal geometric issues that may be hidden from a single viewpoint.*

TABLE R1: *Refreshed geometric evaluation on NopSCADlib (replaces Table II). F1 at a 1% distance threshold. Best in each row group in bold.*

Split	Method	CD↓	F1↑	Voxel IoU↑
Overall	CRAFT	**0.0654**	0.2530	**0.2356**
	GPT-4o	0.0733	0.2467	0.2135
	GPT-5.2	0.0662	**0.2587**	0.1890
Simple	CRAFT	0.0805	**0.3771**	0.2863
	GPT-4o	0.0841	0.3399	**0.3339**
	GPT-5.2	**0.0800**	0.3475	0.1554
Medium	CRAFT	0.0594	0.2277	0.2431
	GPT-4o	0.0652	0.2262	0.1810
	GPT-5.2	**0.0571**	**0.2357**	**0.2591**
Complex	CRAFT	**0.0559**	0.1492	**0.1744**
	GPT-4o	0.0706	0.1694	0.1201
	GPT-5.2	0.0614	**0.1883**	0.1487

A.v. Camera-Ready Additions (staging)

This section stages the revisions requested in review, for redistribution into the main paper. No part of the pipeline was changed: the additions are an ablation isolating each agent (requested by all three reviewers), a quantitative editability metric (Awu5), a short explanation of the "Overall" FID computation (Awu5), and typographic fixes. The geometric results are recomputed on the same models and prompts, and the comparative wording is softened so that claims are stated no more strongly than the numbers support (UJGD). Where the re-run changes a submitted number, the refreshed value supersedes it.

A. Refreshed Geometric Results on NopSCADlib

Table R1 recomputes the NopSCADlib geometric results; it is intended to replace Table II. On the full benchmark, CRAFT attains the best overall Chamfer distance and the best overall voxel IoU, and is comparable to the strongest baseline on F1. As in the submitted version, direct generation remains competitive on the simpler in-distribution parts, while CRAFT's advantage is clearest on volumetric overlap and on the more complex components. We therefore characterize CRAFT's geometry as competitive and tied-or-better rather than dominant.

B. Ablation Study

Table R2 isolates the contribution of each pipeline component by disabling one at a time on the full NopSCADlib benchmark; the final column summarizes the effect of removal relative to the complete system. Removing component retrieval (−retrieval) is the most damaging change, degrading every geometric metric: grounding generation in catalog specifications is the single largest contributor to in-domain geometric fidelity. Disabling multi-view correction (−multi-view) and the recovery layers (−recovery) each reduce Chamfer distance and F1 while leaving voxel IoU essentially unchanged. Two effects are worth noting. Removing the JSON intermediate representation (−JSON-IR) leaves geometry essentially unchanged; its contribution is to editability (§A.v-C), not to raw geometry, which is consistent with its role of preserving symbolic structure rather than altering the final shape. Removing component verification (−verification) slightly lowers Chamfer distance, because verification adds or repairs parts to satisfy the prompt—improving semantic completeness while occasionally moving the surface away from a ground-truth mesh that omits those parts—so we describe it as a completeness mechanism rather than a geometric one.

TABLE R2: *Ablation on NopSCADlib: full pipeline versus single-component-disabled variants (−component). The final column summarizes the effect of removing each component relative to the full system.*

Variant	CD↓	F1@1%↑	IoU↑	Effect of removal
CRAFT (full)	0.0654	0.2530	0.2356	—
− retrieval	0.0764	0.2030	0.1545	large drop on all metrics
− multi-view	0.0738	0.1907	0.2272	lower CD and F1
− recovery	0.0773	0.1878	0.2289	lower CD and F1
− JSON-IR	0.0643	0.2383	0.1999	geometry ≈ unchanged
− verification	0.0637	0.2574	0.2411	CD slightly better

TABLE R3: *Editability on NopSCADlib: exposed parameter count, symbolic-preservation rate, edit-success rate, and post-edit render validity.*

Method	#Params	Symbolic↑	Edit succ.↑	Valid↑
CRAFT	**13.1**	**66.8%**	99.9%	99.1%
GPT-5.2	4.2	47.7%	99.2%	97.7%
GPT-4o	0.1	3.0%	95.0%	95.0%

C. Editability

The paper's central claim—that CRAFT produces editable parametric programs rather than fixed geometry—was previously shown only qualitatively. We quantify it by parsing the exposed top-level parameters of each generated program, perturbing them by $\pm 10\%$ and $\pm 50\%$, re-rendering, and measuring whether the program remains valid and structurally intact. Table R3 shows that CRAFT exposes 13.1 editable parameters per model on average, against 4.2 for GPT-5.2 and 0.1 for GPT-4o, and preserves symbolic expressions in 66.8% of cases versus 47.7% and 3.0%. Edit-success and post-edit render validity are uniformly high, so the difference is in how much editable structure is exposed, not whether edits apply cleanly. The advantage widens with design difficulty: CRAFT's symbolic-preservation rate rises from 59.2% on simple parts to 63.9% on medium and 78.1% on complex parts, with the mean exposed-parameter count increasing from 10.2 to 17.6.

D. Clarification: How "Overall" FID Is Computed

A reviewer asked how the Overall FID is obtained, noting that it is lower than each per-tier value and so cannot be a weighted average. It is not an average. FID is a distance

between two distributions rather than a mean of per-image scores: it fits a multivariate Gaussian to the Inception-v3 feature activations of each image set and measures the Fréchet distance between the generated and reference Gaussians. Two consequences follow. First, the statistic is non-additive across subsets—the FID of a pooled set is not a weighted mean of the FIDs of its parts, because the pooled mean and covariance are re-estimated over all samples rather than combined linearly. Second, FID carries a known finite-sample bias that decreases as the number of images grows, so estimates from small samples are inflated. The per-tier values are each computed on roughly 150 renders, whereas the Overall value is computed on all 468; the Overall FID is therefore both better estimated and, as expected, lower than any single tier. The Overall row should be read as the distributional distance over the full benchmark, not as a summary of the three tier rows.

E. Typographic and Wording Fixes

The following in-body corrections will be applied (raised primarily by Awu5):

- "orthographic" spelled correctly in the Fig. 2 / multi-view caption.
- "OpenSCAD" used consistently (not "OpenScad").
- "Fréchet Inception Distance (FID)" with the correct accent and no stray accent on "Inception".
- "GitHub" capitalized correctly.
- "Datasets and Benchmarks Track" capitalized where referenced.
- Reword "OpenSCAD renders all models at 512×512 resolution" to "All views are rendered at 512×512 resolution," since images, not models, carry a resolution.
- Promote the ablation (Table R2) from the appendix into the main Results section, as requested.