Title:
Vision-Language Assisted CAD Querying: ROI Localization and Iterative Program Synthesis for Feature Recognition

Authors:
Tathagata Chakraborty, tathagata_chakr@hcl-software.com, HCLSoftware
Christine Zuzart, christine.zuzart@hcl-software.com, HCLSoftware
Baburaj Iyer, baburaj.iyer@hcl-software.com, HCLSoftware

Keywords:
CAD query, feature recognition, vision-language models, agentic workflow, program synthesis

DOI: 10.14733/cadconfP.2026.100-105

Introduction:
Downstream engineering tasks such as computer-aided manufacturing (CAM) and design for manufacturing (DFM) depend on reliable access to manufacturing and design features (e.g., holes, pockets, counterbores, and adjacency patterns) that are often not explicitly represented and are difficult to recover robustly from CAD models [1, 2]. Classical feature-recognition pipelines are typically heuristic and rule-based, and can be slow or brittle; in contrast, supervised learning approaches often require large labeled datasets and continue to face representation challenges in CAD/CAM [3, 4, 6, 5].

Recent vision-language models (VLMs) have demonstrated strong zero-/few-shot perception and reasoning capabilities, motivating their application to CAD-relevant visual tasks such as manufacturing feature recognition from multi-view renderings [11]. In parallel, related model families have enabled rapid progress in CAD code generation and iterative self-correction using visual feedback [9, 10].

Motivated by these trends and practical CAM/DFM requirements, we present a unified workflow for "CAD querying" that combines (i) vision-driven region-of-interest (ROI) localization from CAD screenshots and (ii) iterative, feedback-driven program synthesis over a CAD API. Our objective is to accelerate and generalize feature and ROI discovery for CAM/DFM-style queries with minimal labeled data, while preserving the exactness of geometry-kernel computations.

Method:
Our approach integrates two complementary mechanisms:
(1) Vision-based ROI localization from screenshots. We render one or more standardized views of a CAD part or assembly and overlay a lightweight grid of markup with unique alphanumeric tags (Fig. 1). In addition to the marked-up image, the VLM is provided with a textual description of the target feature and one or more highlighted and annotated screenshots illustrating the feature. A vision model is prompted to (a) identify queried features in the image (e.g., holes, pockets, bosses, and more complex features) and (b) return the nearest tags to each detected instance. This vision-based ROI localization is needed because current VLMs are not yet sufficiently robust to localize exact feature positions (e.g., precise pixel coordinates) reliably in CAD screenshots. Given VLMs' strong OCR and semantic recognition capabilities, the returned tags provide an effective model-to-CAD "pointer" to a

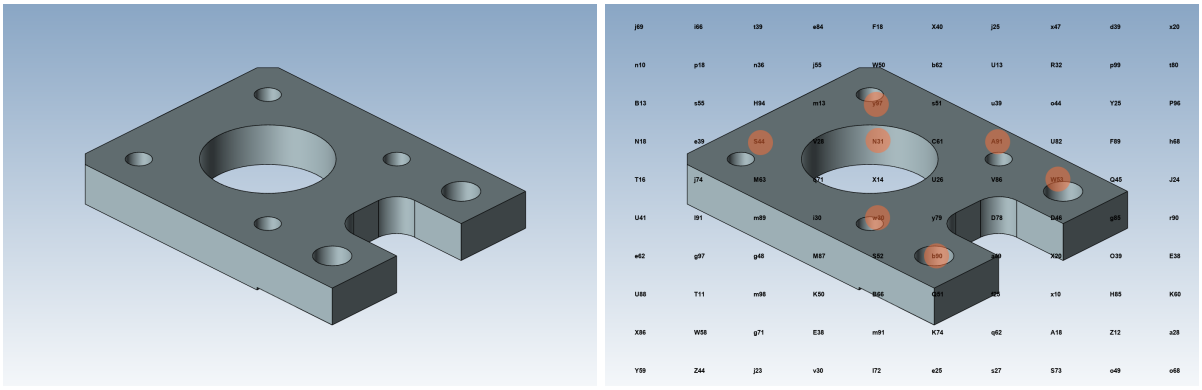

Fig. 1: Screenshot-based ROI localization using unique markup tags: the vision model identifies feature instances and returns the nearest tags, which are then mapped to local CAD neighborhoods for exact recognition/parameterization.

small neighborhood. The CAD system stores a mapping from grid cells to visible face IDs; thus, each returned tag corresponds to a compact candidate set of faces.

VLM output format. When the vision model is provided the marked-up screenshot (right side of Fig. 1), we instruct it to return detections in a strict JSON schema, enabling the downstream system to deterministically parse ROIs and tags:

```
{
"holes": 7,
"regions": ["N31", "S44", "y97", "A91", "W53", "w30", "b90"]
}
```

Multi-view and scale-aware queries. To mitigate occlusion and viewpoint dependence, we issue the same query across multiple predefined orientations (e.g., axis-aligned and isometric views) and aggregate results across views. For size-constrained queries, a simple scale marker can be embedded in the tagged image; the VLM can then filter features using approximate size thresholds, reducing downstream search and enabling rapid triage before exact measurement.

Zoom-in refinement (agentic interaction). The workflow supports an agentic "zoom" action: the model selects one or more grid cells, requests a zoomed screenshot, and repeats tagging at higher resolution to further tighten ROI bounds. This enables a coarse-to-fine localization strategy without requiring the VLM to output precise pixel coordinates. The overall vision-based ROI localization loop is summarized in Fig. 2.

(2) Iterative CAD-query program synthesis over CAD APIs. Given a natural-language feature definition, optional labeled screenshots (few-shot examples; Fig. 3), and relevant CAD API documentation and examples, a multimodal model synthesizes code to locate and parameterize the queried feature. The synthesized program is executed on test models; feedback is returned to the model both visually (annotated screenshots showing expected versus identified, missed, and spurious instances) and textually (counts and failure modes). The model then refines the program in an agentic loop until performance stabilizes (Fig. 4).

Prompt composition. The program-synthesis prompt can incorporate four information sources: (i) a precise feature definition, (ii) labeled screenshots illustrating positives/negatives, (iii) CAD API documentation for geometry/topology access, and (iv) a small code example demonstrating API usage. This

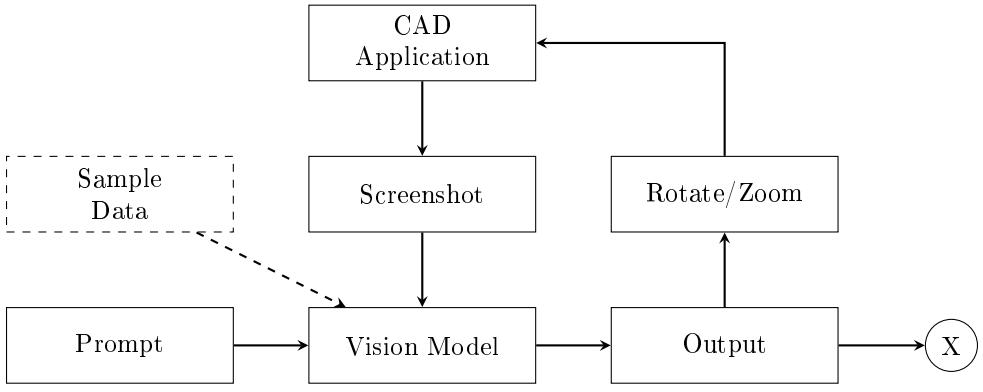

Fig. 2: Vision-based ROI localization workflow: screenshots are analyzed by a vision model guided by a prompt and optional sample data; the output can trigger rotate/zoom actions to acquire additional views.

structured prompt is particularly helpful for composite queries (e.g., proximity constraints, multi-feature patterns) that are hard to solve from images alone.

Automatic feedback and evaluation sets. To train the agentic refinement loop to be both effective and robust, we execute each generated CAD-query program on two disjoint sets of parts: an *open* set and a *withheld* set.

Open-set feedback (visual + textual). On the open set, the system returns rich feedback to the vision-language model after each execution, including (i) visual feedback (e.g., screenshots with predicted feature instances highlighted and mismatches annotated) and (ii) textual feedback (e.g., pass/fail counts, runtime, and error summaries). This enables rapid debugging of feature logic and fast iteration when the program fails on specific geometric edge cases.

Withheld-set feedback (textual/results only). On the withheld set, we restrict feedback to textual/results signals (e.g., aggregate success metrics and failure categories) without returning any visual evidence. This separation discourages overfitting to a fixed set of exemplars and provides a more realistic signal of generalization before deployment.

Tying vision localization to program execution. A practical challenge in CAD querying is that geometry-kernel API calls (e.g., face traversal, adjacency checks, Boolean tests, exact measurements) can be slow and expensive on large models if executed globally. We therefore use the vision-based ROI localization output (tags $\rightarrow$ candidate face sets) to *constrain* the search space of synthesized CAD-query programs: the program first restricts traversal/feature tests to the localized neighborhood, and only falls back to broader search when confidence is low. This coupling preserves exactness (final checks remain kernel-based) while significantly reducing the number of kernel operations required for many queries.

This unification leverages the strengths of foundation models while still relying on precise CAD-kernel computations for final validation/parameter extraction, addressing limitations of purely image- or graph-based learning for feature recognition [2, 4, 5]. Related advances in vision-language CAD generation and verification further motivate the feasibility of model-driven CAD interactions [9, 10, 13, 12].

Results / Use Cases:

We demonstrate two representative CAD querying patterns:

(i) Fast ROI identification for complicated features and contexts. The tagged-screenshot strategy is especially useful when the queried feature is difficult to define purely textually or when the CAD

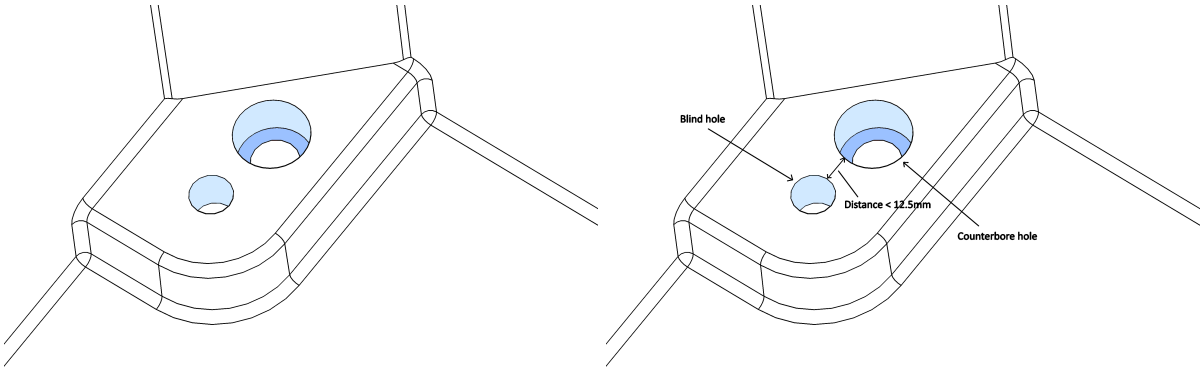

Fig. 3: Few-shot labeled data for composite-feature definitions: a manually highlighted example (left) and an annotated variant (right) that can be embedded into the prompt for program synthesis.

model is large and cluttered (e.g., composite features, localized adjacency patterns, features on dense assemblies). In these settings, the VLM can quickly return a small set of candidate regions, after which exact, kernel-based checks can be applied only locally. The same query can be issued across multiple standard views and optionally refined via zoom-in to improve robustness under occlusion.

(ii) Synthesized CAD-API programs for feature queries and DFM rule validation. Beyond composite-feature definitions (e.g., "a counterbore hole adjacent to a blind hole within 20mm"), the iterative program synthesis loop can be used to implement executable DFM checks (rule evaluation, thresholding, reporting of violations) over CAD APIs. Visual and textual feedback (Fig. 4) provides an interpretable way to debug and refine these programs without requiring the end user to be a CAD API expert.

Coupling ROI localization with synthesized program execution. A key practical benefit is that fast ROI localization can be used in conjunction with the synthesized program to restrict program execution to localized regions of the CAD model (tags $\rightarrow$ candidate face sets), reducing traversal cost while preserving exactness via final kernel-based validation. Finally, these capabilities can be surfaced through a configurable user interface that lets end users compose and iterate on feature-recognition (FR) and design-for-manufacturing (DFM) rules interactively, reducing reliance on bespoke development and shortening the iteration cycle from weeks to minutes.

Target use cases include accelerating CAM/DFM preprocessing (feature extraction and rule checks), manufacturing cost estimation, tolerance analysis, and quality-control style inspection queries over assemblies. The workflow is designed to be practical in real CAD environments: it uses screenshots and existing CAD APIs, supports user-provided labeled examples for custom features and rules, and provides interpretable intermediate outputs (tags, ROIs, highlighted feedback) to keep humans in the loop.

Implementation Notes:

Controlling cost and latency. The agent can be constrained by limiting the number of screenshots/zoom levels and by caching rendered views and face-to-grid mappings. **Robustness.** False positives/negatives from VLM detection can be filtered by repeated queries across viewpoints and by verifying candidates with CAD-kernel computations. **Extensibility.** New feature types can be introduced by attaching a few annotated examples and a textual definition; no retraining is required for the base workflow.

Conclusions:

We propose a practical, unified CAD querying pipeline that couples vision-language ROI localization with

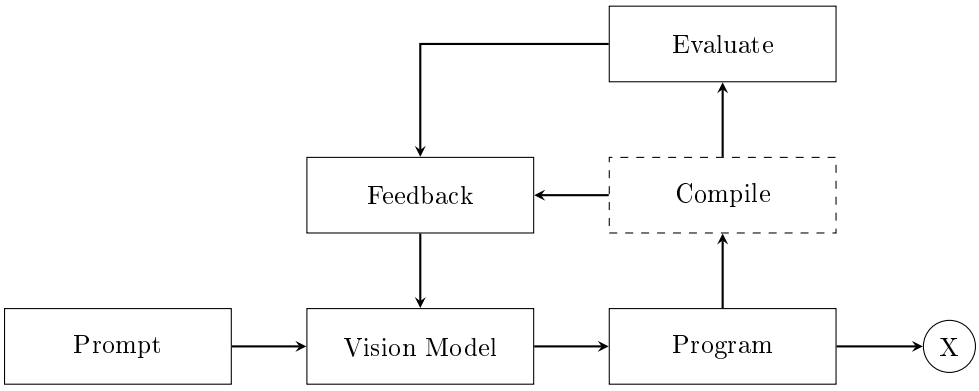

Fig. 4: Agentic workflow for feedback-driven CAD-query program synthesis.

feedback-driven program synthesis over CAD APIs. The approach aims to reduce brittleness and search cost in feature recognition while keeping exactness via CAD-kernel checks, enabling faster deployment of both standard and custom CAM/DFM queries. We expect this workflow to accelerate feature-centric downstream automation and to complement emerging research on foundation-model-driven CAD interactions.

Acknowledgement:
This research was conducted with support from the FR and DFM teams at HCLSoftware. We are thankful for their support and encouragement.

Tathagata Chakraborty, https://orcid.org/0000-0002-2752-2533
Christine Zuzart, https://orcid.org/0009-0005-3128-4418
Baburaj Iyer, https://orcid.org/0009-0009-9801-4550

References:

[1] J. Han, M. Pratt, and W. C. Regli: Manufacturing feature recognition from solid models: a status report, *IEEE Transactions on Robotics and Automation*, 16(6), 782–796, 2000. https://doi.org/10.1109/70.897777

[2] Y. Shi, Y. Zhang, K. Xia, and R. Harik: A critical review of feature recognition techniques, *Computer-Aided Design and Applications*, 17(5), 861–899, 2020.

[3] Y. Shi, Y. Zhang, and R. Harik: Manufacturing feature recognition with a 2D convolutional neural network, *CIRP Journal of Manufacturing Science and Technology*, 30, 36–57, 2020.

[4] P. Wang, W.-A. Yang, and Y. You: A hybrid learning framework for manufacturing feature recognition using graph neural networks, *Journal of Manufacturing Processes*, 85, 387–404, 2023.

[5] X. Fu et al.: Boundary representation compatible feature recognition for manufacturing CAD models, *Manufacturing Letters*, 35, 895–903, 2023.

[6] A. Purwar et al.: Machine learning and representation issues in CAD/CAM, *Journal of Computing and Information Science in Engineering*, 24(1), 2024.

[7] H. Naveed et al.: A comprehensive overview of large language models, *ACM Transactions on Intelligent Systems and Technology*, 2023.

[8] T. Coignion, C. Quinton, and R. Rouvoy: A performance study of LLM-generated code on LeetCode, *Proc. 28th Int. Conf. on Evaluation and Assessment in Software Engineering (EASE)*, 79–89, 2024.

[9] K. Alrashedy et al.: Generating CAD code with vision-language models for 3D designs, *arXiv preprint arXiv:2410.05340*, 2024. https://arxiv.org/abs/2410.05340

[10] A. C. Doris et al.: CAD-Coder: an open-source vision-language model for computer-aided design code generation, *arXiv preprint arXiv:2505.14646*, 2025. https://arxiv.org/abs/2505.14646

[11] M. T. Khan et al.: Leveraging vision-language models for manufacturing feature recognition in CAD designs, *arXiv preprint arXiv:2411.02810*, 2024. https://arxiv.org/abs/2411.02810

[12] T. Preintner et al.: EvoCAD: evolutionary CAD code generation with vision language models, *arXiv preprint arXiv:2510.11631*, 2025. https://arxiv.org/abs/2510.11631

[13] J. Li et al.: ReCAD: reinforcement learning enhanced parametric CAD model generation with vision-language models, *arXiv preprint arXiv:2512.06328*, 2025. https://arxiv.org/abs/2512.06328