Are Agentic CAD Model Generation Near Reality?

Er. Pravakar Bogati, Autolab Technologies Pvt.Ltd., April 2026
Kupondol-10, Lalitpur, Nepal
info@autolabtechnologies.coms

"CAD automation," were usually about scripts that renamed exported files, generated drawings, or automated repetitive clicks inside a CAD package. Today, the same phrase means something entirely different. A user can upload a hand-drawn sketch, a photograph of a bracket, a dimensioned engineering drawing, or simply type:

"40 mm steel bracket with two M6 mounting holes."

Within seconds, our system generates executable CAD code, validates it, executes it, and returns a manufacturable STEP model. This article explains what we built, where the real difficulties were, and why AI-generated CAD is becoming one of the most interesting areas in engineering software.

CAD is becoming programmable

For decades, CAD software has been designed around direct human interaction. Engineers sketch profiles. They constrain geometry. They extrude, fillet, chamfer, pattern, and assemble parts manually. Even though modern CAD packages have extensive APIs, most design work is still driven by mouse clicks.

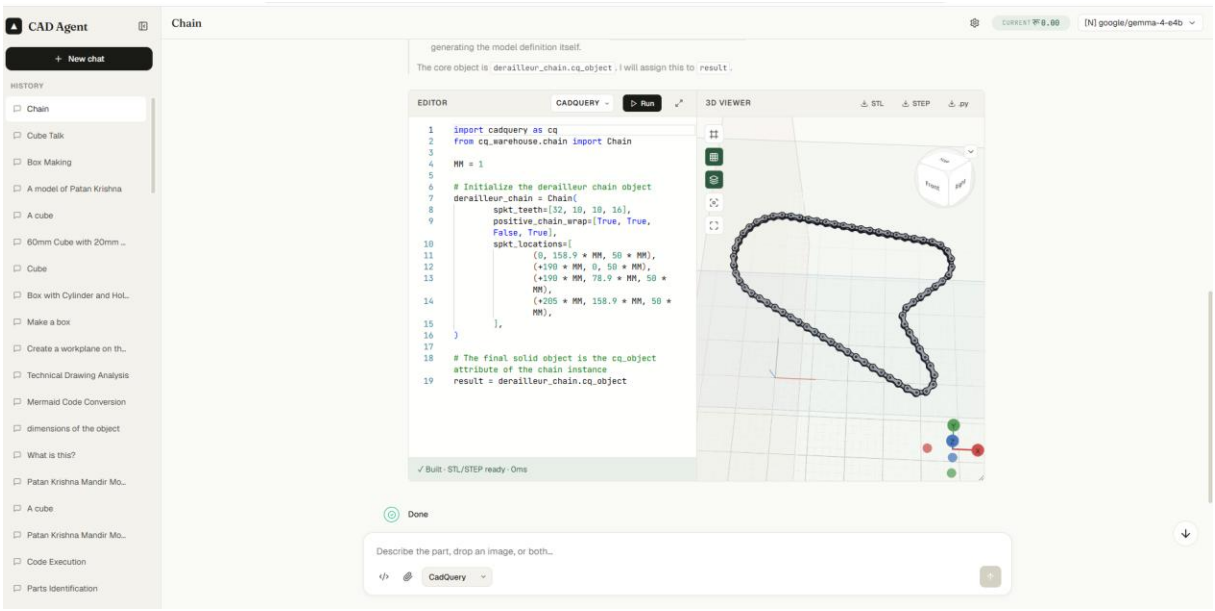
Artificial intelligence is changing that workflow.

Instead of manually constructing geometry feature by feature, engineers can increasingly describe what they want in natural language or provide an image as input.

Large language models now understand engineering intent well enough to generate parametric CAD code, while vision-language models can interpret technical drawings and infer geometric relationships directly from images.

Major CAD vendors are already incorporating generative design, AI-assisted feature creation, and automated documentation into production workflows. What was experimental research only a few years ago is rapidly becoming part of everyday engineering.

Rather than waiting for those capabilities to mature elsewhere, with our specialty in CAD automation invested in building our own system from the inference pipeline to the CAD execution engine.



AI-generated parametric CAD model produced directly from a natural-language prompt.

The surprising part wasn't generating CAD

Before starting this project, we assumed the biggest challenge would be convincing an AI model to generate valid geometry.

It turned out that wasn't the bottleneck.

We experimented with multiple CAD backends, including:

- CadQuery
- Build123d
- PythonOCC
- OpenSCAD
- Onshape APIs
- SolidWorks automation

Each has different strengths. CadQuery and Build123d offer expressive Python-based parametric modeling. OpenSCAD provides deterministic constructive solid geometry. Onshape exposes a cloud-native CAD API. SolidWorks remains deeply integrated into manufacturing workflows. Once the translation layer for each backend was built, producing a valid STEP file became relatively reliable.

The difficult problem was deciding which system should solve which request.

Compute is an engineering problem too

One lesson became obvious very quickly. Not every CAD request deserves the same amount of compute.

Generating a simple L-bracket with two holes is fundamentally different from reconstructing a complex mechanical assembly containing dozens of interacting features.

Running a frontier language model for every request produces good results but it is slow and unnecessarily expensive. Running everything through a lightweight model is fast but complex geometry begins to fail in subtle ways.

Reliability matters more than impressive demos

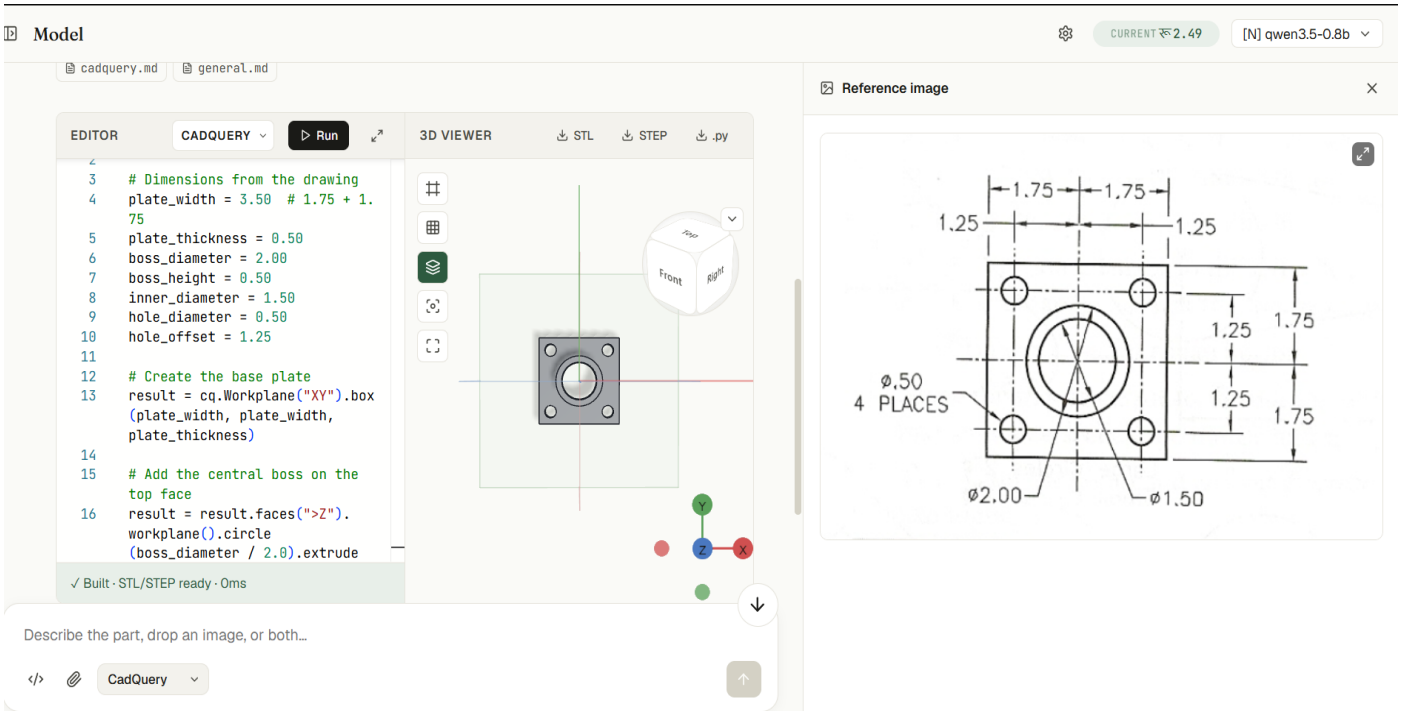
Generating CAD code is only the first step. Trusting it is the real challenge. Current text-to-CAD systems are remarkably good at producing visually convincing models. They are much less reliable when manufacturing constraints enter the picture.

Questions such as:

- Are dimensions actually respected?
- Are hole locations correct?
- Does the part remain fully parametric?
- Are wall thicknesses manufacturable?
- Will downstream CAD software import the STEP correctly?

remain active research problems across the industry.

A model that "looks right" is not necessarily manufacturable.



Manufacturability checks catch issues that are invisible in a purely visual review.

To reduce hallucinations, we built our generation pipeline around retrieval-augmented generation (RAG), grounding model outputs in verified examples rather than relying entirely on model memory.

Instead of asking the language model to invent geometry from scratch every time, it retrieves structurally similar examples, references known- good implementations, and generates code from a much more reliable foundation.

That single architectural decision dramatically improved consistency.

Vision became just as important as language

Many real engineering workflows don't begin with text.

They begin with photographs.

Or scanned engineering drawings.

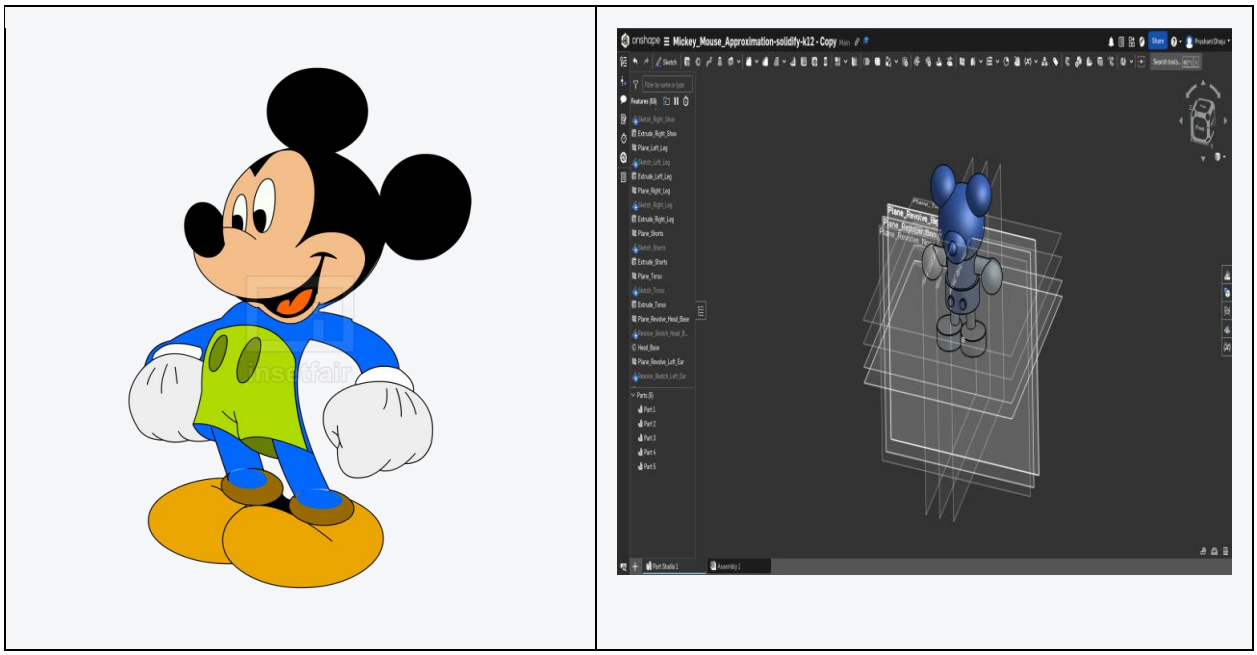
Or sketches on paper.

To support those workflows, we trained our own vision-language models capable of translating dimensioned engineering drawings directly into executable CAD code.

Owning the training pipeline mattered.

Rather than depending entirely on third-party APIs, we fine-tuned our models on our own infrastructure, allowing us to continuously improve performance as new datasets become available.

That also gives us flexibility to experiment with architecture, retrieval systems, prompting strategies, and evaluation methods without waiting for external model providers.



Vision-language models translating hand sketches and dimensioned drawings into executable CAD code.

What happens inside the CAD agent

At a high level, the pipeline looks like this:

1. Accept text, images, sketches, or engineering drawings.
2. Estimate geometric complexity.
3. Retrieve similar verified CAD examples using RAG.
4. Generate parametric CAD code.
5. Execute the code inside the appropriate CAD backend.
6. Validate geometry.
7. Export manufacturable STEP files.

Every stage includes checks designed to reduce hallucinations before the final model reaches the user.

The objective isn't simply producing geometry.

It's producing geometry that engineers can trust.

Why we're building this

The goal isn't replacing CAD engineers. It is reducing the distance between an idea and something that can actually be manufactured. For experienced engineers, that means eliminating repetitive modeling work so they can focus on engineering decisions instead of feature creation.

For students, makers, and first-time users, it means removing years of software learning before they can turn an idea into a physical object. Someone should be able to sketch a phone stand on paper, upload it, and receive a fully editable CAD model. Someone else should be able to upload a production engineering drawing and receive a manufacturable STEP file ready for downstream workflows.

Those are very different users. Our aim is to build one system capable of serving both.

Where AI CAD is heading

We're still at the beginning. Today's systems are becoming good at generating parts.

Tomorrow's systems will understand assemblies, tolerances, manufacturing processes, simulation constraints, cost optimization, and design intent—not just geometry.

The CAD software of the future won't simply draw what engineers ask for.

It will collaborate with them.

We're excited to help build that future.