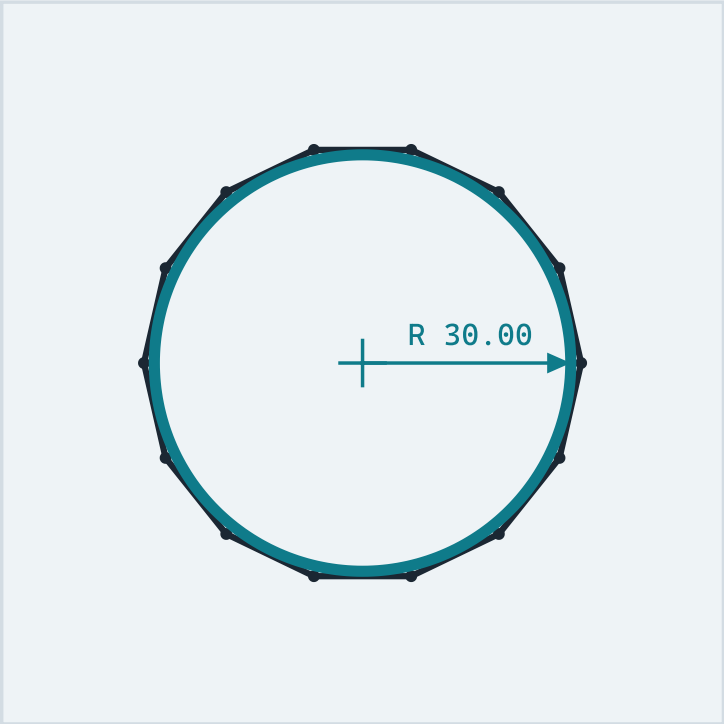


MESH: 14 flat segments. Radius unknown.
B-REP: one exact face. Radius measurable.

The CAD, AI and Robotics Guide

Geometry, agentic CAD and physical AI:
from first principles to what to build.

Samujjal Choudhury

33 LESSONS · 6 PARTS · 9 FIGURES · 4 APPENDICES

Compiled for the SuperHuymn team · 8 September 2026

The CAD, AI and Robotics Guide

Geometry, agentic CAD and physical AI, from first principles to what to build

Written by: Samujjal Choudhury

Date: 2026-09-08

This guide was compiled for the SuperHuymn team from web research and direct repository inspection.

Copyright

Every claim traces to a source listed in Appendix D. A number marked verified was read from its source on 8 September 2026. Everything else is marked search-reported and was not independently confirmed. Nothing here is stated more firmly than it was found.

How to use this guide

Reading the tags

Some lessons include inventory tables that list a tool, a paper or a library next to a tag. The tag says what to do about the resource, not how good it is in the abstract.

- `ADOPT` means the team already uses it, or should start now. It is in the current stack.
- `STUDY` means read the source before deciding. It might be useful, but nobody on the team has tested it against SuperHuymn's own work yet.
- `WATCH` means it is too young, too far from a current need, or too small a project to depend on today. Check back later.

- AVOID means the cost or the risk outweighs the benefit for this team right now.

These tags are the spine of the inventory lessons in Part III. They are kept exactly as the research packs wrote them, table by table, so a reader can trust that "ADOPT" always means the same thing wherever it appears.

Reading the numbers

A star count or a claim in this guide carries one of two labels.

- **(verified)** means someone on the research team opened the source directly this session, usually by browsing the GitHub page or fetching the document, and read the number off it.
- **search-reported** means the number came from a search result or a summary, and nobody opened the primary source to confirm it. Treat it as a starting point, not a fact.

Every number in this guide keeps its original label. If a table says "3.0k stars (verified)", that count was checked directly. If it says "search-reported", it was not, and it should be verified before it goes into a pitch deck or a customer conversation.

Three ways to read this guide

- **New to CAD.** Start at Part I and read straight through. It assumes no background and builds the vocabulary the rest of the guide depends on.
- **Choosing tools.** Go to Part III, which surveys the field as it stands: agent harnesses, MCP servers, kernels, verification tools, physics engines, and more, each carrying a tag.
- **Deciding what to build.** Go to Part VI, which lays out where a small team can win, the ranked action list, and what the research could not confirm.

Contents

PART I. Foundations: what geometry is

Lesson 1. Why geometry is the whole problem

WHAT YOU WILL LEARN

- What a CAD system actually computes, underneath the drawing on screen.
- Why an AI writing CAD code is making a claim, not just producing a picture.
- Why SuperHuymn treats geometry, not the AI model, as the part of the system that has to be right.

1.1 A CAD system answers questions about shape

A CAD system is a machine for answering questions about shape. Where does this surface sit? Do these two parts touch? How heavy is this? Can a drill reach that hole? Every feature a user sees is one of these questions, wearing a friendly name.

1.2 An AI that writes CAD code is making a claim

When an AI writes CAD code, it is not really writing code. It is making a claim about shape. "Here is a bracket with four M6 holes" is a claim. The claim is either true of the resulting geometry or it is not. The only way to find out is to ask the geometry itself, not the AI.

1.3 Why this sits at the centre of SuperHuymn's work

The AI model underneath will change more than once this year. Model providers ship new versions often, and SuperHuymn's tools should not depend on any single one of them staying still. The geometry does not change that way. It is where the truth lives, and truth is what a factory pays for.

KEY POINTS

- A CAD feature is a question about shape, wearing a friendly name.
- An AI-generated CAD script is a set of claims that must be checked against the geometry it actually produces, not assumed from the code.
- The AI model underneath is replaceable. The geometry layer is where SuperHuymn's product has to stay trustworthy.

WHAT SUPERHUYMN DOES WITH THIS

This lesson sets the principle the rest of Part I and Part II build on. The source research names no specific file or tool here; its job is to establish why the checks described in Lessons 4, 6, 7 and 11 matter, and why SuperHuymn's pitch rests on geometry rather than on whichever model happens to sit under Forge Studio this quarter.

Lesson 2. Two ways to describe a shape

WHAT YOU WILL LEARN

- The difference between a mesh and a B-rep, and why almost every AI 3D generator makes a mesh.
- Why turning a B-rep into a mesh is easy, and turning a mesh into a B-rep is not.
- Why this one fact separates useful engineering output from a nice-looking dead end.

2.1 A mesh is a shell of flat triangles

Picture covering a football with thousands of tiny flat stickers. From a distance it looks round. Up close it is made of flat faces. That is a mesh: a list of corner points, called vertices, and a list of triangles joining them.

Meshes are what games, film and phone screens use. They are fast to draw, easy to store, and easy to generate. Almost every AI 3D generator, including TRELLIS, Hunyuan3D and Tripo, produces meshes.

A mesh cannot tell you whether a round-looking shape is exactly a cylinder of diameter 20.00 mm. It only knows it is 400 triangles arranged in roughly a tube. There is no circle stored inside it to measure.

2.2 A B-rep is a blueprint made of exact surfaces

B-rep is short for boundary representation. Instead of triangles, it stores real mathematical surfaces: this face is a flat plane, that face is a cylinder of radius 10 mm, this edge is the exact circle where the two meet. It also stores how the pieces connect: which edges bound which faces, and which faces bound the solid.

This is what engineering CAD uses. It is what OpenCASCADE gives SuperHuymn through build123d. STEP files, the standard neutral file format for exchanging exact 3D models, carry B-rep data. That is why a machinist can open a STEP file and trust that a hole is 20.00 mm, not 19.87 mm.

Figure 2. One of them can be measured and machined, the other cannot.

2.3 Why the direction matters so much

Going from B-rep to mesh is easy. The exact surfaces get chopped into triangles. Every CAD system does this constantly, because a screen can only draw triangles. This step is called tessellation, and how fine the triangles are is controlled by a setting called deflection.

Going from mesh to B-rep is hard. In the general case it has no reliable solution today. Given a pile of triangles, deciding which ones were meant to form one exact cylinder, and what its true radius should be, is guesswork. It is unreliable and slow, and two runs on the same mesh can give different answers.

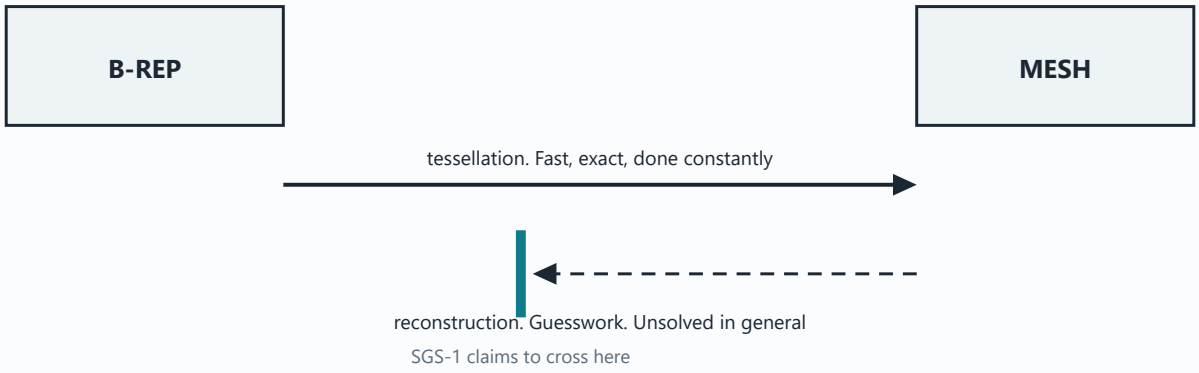


Figure 3. The market splits on this arrow.

The consequence for SuperHuymn. Any pipeline that ends in a mesh has ended in a dead end for engineering. A customer cannot machine from a mesh, cannot change one dimension in it, and cannot check a tolerance against it. This one fact separates the serious half of the AI-CAD market from the toy half, and it is the strongest line available in a pitch. Spectral Labs' SGS-1 is notable because it claims to cross this exact gap: mesh in, editable STEP out.

KEY POINTS

- A mesh is a shell of triangles. A B-rep is exact surfaces plus how they connect.
- B-rep to mesh (tessellation) is easy and happens constantly. Mesh to B-rep is unreliable and, in the general case, unsolved.
- Almost every AI 3D generator produces a mesh, which is why most of them cannot feed an engineering pipeline without a further, unreliable conversion step.
- A tool that produces editable STEP directly from a mesh input, as Spectral Labs' SGS-1 claims to do, is solving the harder and more valuable problem.

WHAT SUPERHUYMN DOES WITH THIS

Any generative 3D tool SuperHuymn evaluates gets asked one question first: does it output B-rep, or does it output a mesh that still needs the unreliable mesh-to-B-rep conversion? That question, more than any demo video, decides whether a tool belongs in an engineering pipeline or only in a visualization pipeline.

Lesson 3. Geometry, topology, and the naming problem

> WHAT YOU WILL LEARN
>
> - The difference between geometry (shape data) and topology (connection data) inside a B-rep.
> - Why an agent's reference to "Face 6" can silently point at the wrong face after an edit.
> - Six real mitigations for this problem, and why property-based selection is the one to enforce everywhere in Forge Studio.

3.1 Geometry and topology are different things

Inside a B-rep there are two layers, and separating them explains most of what follows.

Geometry is the shape data: this plane sits here, facing this way. This cylinder has this axis and this radius. It is pure numbers.

Topology is the connection data: this solid has 6 faces, this face is bounded by 4 edges, this edge is shared by 2 faces, and these 2 edges meet at this vertex. It is pure structure, with no numbers attached.

A useful comparison: geometry is where the houses are. Topology is which streets connect to which. You can move a house without changing the street map, and you can rewire the street map without moving a house.

OpenCASCADE 8.0 added BRepGraph, a graph view of B-rep topology with two-way traversal and history tracking. That is directly useful to an AI system, because history tracking answers the question "which operation produced this face?", which is exactly what an agent needs when it wants to modify something it made earlier.

3.2 The topological naming problem, in one example

This is the most important technical idea in this guide.

An agent builds a box. It wants a hole in the top face, so it refers to the top face. In most CAD systems, faces are identified by an index, something like Face 6, assigned when the shape was built.

Now the agent adds a fillet (a rounded edge) somewhere else on the box. The kernel rebuilds the solid. The box now has more faces, and they get renumbered. Face 6 is no longer the top. It might now be a sliver of rounded edge on the opposite side.

The agent's next instruction, "put the hole in Face 6", now drills into the wrong place. Nothing threw an error. The code ran. The output is silently wrong.

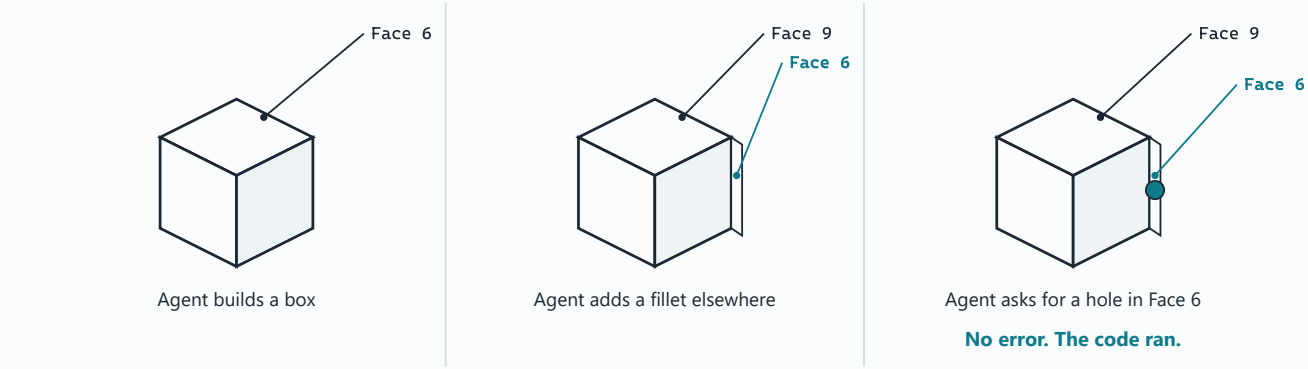


Figure 4. Silent wrongness, which is why it is dangerous.

This is called the topological naming problem, sometimes shortened to persistent naming. It has been a known hard problem in CAD for thirty years, and FreeCAD users have complained about it for most of that time.

3.3 Why it is worse for agents than for humans

A human clicks the face they want and sees the result immediately. Their eyes are the error check.

An agent writes a whole script, runs it once, and gets back either a file or an exception. If a reference drifted but the code still ran, the agent gets no signal that anything went wrong. It will confidently report success. This is the exact failure mode that makes AI-generated CAD untrustworthy, and it is invisible in a screenshot of a nice-looking part.

This is not a hypothetical concern about the future. A 2026 paper on agentic CAD, AgentsCAD (arXiv 2607.02448), documents hitting exactly this failure: reinterpreting a STEP file after a rotation reallocates its `face_id` values, which would invalidate any earlier face-specific operation. The paper's workaround is to force the reorient step to always run last, avoiding the problem through operation ordering rather than solving it.

3.4 The mitigations that actually work

Six mitigations are worth knowing, from real projects and real research. Some avoid the problem, some fix it, and none of them are free.

1. **Select by property, not by index.** Instead of "Face 6", say "the face whose normal points up and whose area is largest". build123d supports this style of query, called a predicate-based selector, and it survives rebuilds because it describes what you want rather than where it happened to be. build123d's own tutorial states that this approach identifies features that stay "robust to features changing within the part," and separately warns against relying on list position, since for methods like `edges()`, "the order within the list is not guaranteed to remain the same" (source: build123d, Selector Tutorial). This is the single most valuable coding convention to enforce in every prompt and every generated script, and it is the mitigation SuperHuymn gets for free from its own kernel choice.

1. **Kernel- or application-level persistent naming.** Give each face and edge an identifier that survives edits, usually derived from the operation history rather than from its position in a list. The `armpro24-blip/cad-cae-copilot` project advertises exactly this, and OpenCASCADE's BRepGraph history tracking (section 3.1 above) is the kernel-level building block for it. FreeCAD shipped a version of this fix at the application level: developer Realthunder built a name-mapping algorithm that attaches persistent names to topological entities at creation time and updates references automatically on regeneration. After years as an unmerged fork, it was folded into official FreeCAD and shipped in FreeCAD 1.0, with further improvements in 1.1 (sources: Ondsel, "FreeCAD's topological naming problem is (officially) history"; the `realthunder/FreeCAD_assembly3` wiki page on the Topological Naming Algorithm). This is the "fix it in the kernel or the app" approach: expensive to build, but transparent to the caller once it exists.

1. **Explicit, caller-supplied deterministic IDs.** Onshape does not silently reassign IDs. FeatureScript operations require an explicit `id` argument on every feature, and the Assembly REST API resolves a mate connector to a face, edge or vertex through an explicit `deterministicId`, obtained via `getBodyDetails`, rather than an implicit position in a list (sources: Onshape FeatureScript forum, "why do op* functions require explicit id"; Onshape Assemblies API docs). This pushes the discipline onto whoever issues the operation, which in Forge Studio's case would be the agent or the MCP tool schema it calls.

1. **Rebuild from parameters, never patch the result.** Keep the script as the source of truth, and regenerate the whole solid on every change. This is slower, and far safer. Code-CAD gives you this for free, and click-based CAD struggles with it.

1. **Check after every rebuild.** Assert the properties you expected: face count, volume, the hole's position. A cheap assertion catches a silent drift that no rendering will reveal.

1. **Academic persistent-identification research.** A body of CAD research, surveyed in "Mechanisms of Persistent Identification of Topological Entities in CAD Systems: A Review", proposes recording a model's creation and modification history in a neutral format and re-deriving entity identity from that history rather than from kernel-internal IDs. This is study material only. Nothing in this research found evidence that any of these academic mechanisms shipped in a tool SuperHuymn could adopt directly, but it is useful background if the team ever needs to justify a from-scratch naming scheme.

KEY POINTS

- Geometry is shape data (numbers). Topology is connection data (structure). A B-rep needs both.
- A reference to a face by index, like "Face 6", can silently point at the wrong face after any edit that triggers a rebuild. The code still runs. The output is wrong.
- This hits AI agents harder than humans, because an agent has no visual check and will report success on a silently wrong part. A 2026 paper on agentic CAD documents exactly this failure.
- Six mitigations exist: predicate-based selection, kernel-level persistent naming, caller-supplied deterministic IDs, rebuild-from-parameters, post-rebuild assertions, and academic persistent-identification schemes. Only the first is free with SuperHuymn's current kernel.

WHAT SUPERHUYMN DOES WITH THIS

Forge Studio treats this as a first-class design constraint, not a bug to fix later. Every prompt that tells the model to select geometry must require property-based selection, mitigation 1 above, since build123d already supports it and it costs nothing to enforce. Every generated part should assert its own expected properties, mitigation 5. This should be written into Forge Studio's MCP tool schemas directly: any tool call that lets an agent "select a face" should force a predicate or a tag as its argument, never a raw index or a cached ID. A small public project, PartMode, builds a browser-native CAD tool around exactly this problem, with source files named `studio-topo-naming.js` and `studio-brep-evidence.js`. It is worth reading for data-structure ideas before Forge Studio designs its own solution, though at 522 stars under a copyleft license it is not something to depend on directly.

Lesson 4. What makes geometry valid

> WHAT YOU WILL LEARN
>
> - Five yes-or-no checks that separate a real, usable solid from something that only looks like one.
> - Why a machine can run every one of these checks automatically.
> - Why a cheap validity gate turns "the AI made a shape" into "the AI made a solid".

4.1 The five checks

An AI can produce something that looks like a part but is not a usable solid. These checks separate the two. Each is a yes-or-no question a machine can answer, which makes them ideal gates.

Manifold. Every edge is shared by exactly two faces. Picture the surface as a proper skin, with no loose flaps and no place where three walls meet along one line. A non-manifold shape cannot be 3D printed or machined, and most solvers refuse to process it. `manifold3d`, already in SuperHuymn's stack, exists for exactly this check.

Watertight, or closed. There are no gaps in the skin. If you poured water in, none would leak out. A shape with one missing face has no inside, so it has no volume and no weight, and any question about its mass is meaningless.

No self-intersection. The shape does not pass through itself. This happens easily when a sweep or a thickening operation turns a corner too tight.

Positive, sensible volume. Volume greater than zero, and within the range you expected. A part that should weigh 200 grams but computes to 40 kilograms usually means a units mistake, not a design mistake.

Sewing tolerance respected. When faces are joined into a solid, the kernel allows a tiny gap, typically around 0.001 mm, and stitches across it. Knowing this tolerance exists explains many strange failures: two surfaces that look like they touch can be 0.01 mm apart and refuse to sew.

4.2 Why this matters as a gate

These checks are fast, and together they are the foundation of a verification gate. Running all of them on every generated part costs little, and it turns "the AI made a shape" into "the AI made a solid".

KEY POINTS

- Manifold, watertight, no self-intersection, positive sensible volume, and sewing tolerance are five yes-or-no checks a machine can run on any solid.
- A shape that fails any of these can look fine on screen but cannot be printed, machined, or trusted for a mass or a volume calculation.
- A units mistake (grams versus kilograms, millimetres versus metres) usually shows up first as an absurd volume, not as an obviously wrong shape.
- All five checks are cheap to run, which makes them a natural automated gate rather than a manual review step.

WHAT SUPERHUYMN DOES WITH THIS

This is the highest-value single thing to automate: one gate that runs all five checks on every part Forge Studio generates, before it is shown to a user or handed downstream. `manifold3d` and `trimesh`, already in the stack, cover the manifold and watertight checks directly. The gate costs little to build and changes the product's claim from "the AI made a shape" to "the AI made a solid you can trust".

Lesson 5. Units, frames and transforms

WHAT YOU WILL LEARN

- Why a factor-of-1000 units mistake is one of the most common real bugs in this field.
- What a coordinate frame and a transform are, and why every part needs both.
- The three ways to write a rotation, and why robotics prefers the least intuitive one.

5.1 Units

CAD usually works in millimetres. Physics engines and the GLB file format work in metres. That is a factor of 1000. A robot arm that appears 1000 times too large in simulation is almost always this mistake. Pick one internal unit, convert only at the boundaries of the system, and write the conversion down in one place.

5.2 Coordinate frames

A frame is an origin point plus three axes. Every part has its own local frame, and the assembly it belongs to has a world frame. Placing a part means saying how its local frame relates to the world frame.

5.3 Transforms

A transform is the recipe for that relationship: a rotation plus a translation (a move). You chain transforms to go from a fingertip's frame all the way back to a robot's base frame. Two conventions cause most confusion: whether rotation happens before or after translation, and whether Z or Y points up. CAD usually has Z up. Many graphics tools have Y up, and GLB is Y up. Decide which convention you use, write it down, and convert at the boundary between systems.

5.4 Rotation representations

A rotation can be written three ways. Euler angles use three numbers and are intuitive to read, but they can lock up in a way called gimbal lock, where the mechanism loses one axis of freedom. A 3x3 matrix uses nine numbers and is safe but bulky. A quaternion uses four numbers, is safe and compact, and is hard for a person to read directly.

Robotics uses quaternions. The W16 engine demo already hit this in practice: it had to build a rotation through a raw OCP quaternion, because build123d had no public constructor for one at the time.

KEY POINTS

- A units mismatch, usually millimetres versus metres, is the most common cause of a part that ends up 1000 times the wrong size.
- A frame is an origin plus three axes. A transform is a rotation plus a translation that relates one frame to another.
- Two silent conventions, rotation-then-translation order and which axis points up, cause most frame confusion. Decide both, and write them down.
- Quaternions (four numbers) are what robotics uses for rotation, because they avoid gimbal lock and stay compact, even though they are hard to read by eye.

WHAT SUPERHUYMN DOES WITH THIS

The W16 engine demo is the concrete case study: build123d had no public constructor for a quaternion, so the demo built one through a raw OCP call instead. Any CAD-to-robotics exporter Forge Studio builds needs a single, documented internal unit and a single, documented up-axis convention, converted only at the two boundaries (import and export), rather than guessed at every step of the pipeline.

PART II. From shape to behaviour

Lesson 6. Mass properties and the bridge to physics

> WHAT YOU WILL LEARN
>
> - The four numbers that turn a CAD drawing into something a physics simulator can move.
> - Why material choice has to be part of the CAD model, not an afterthought.
> - Why computing these numbers automatically from CAD, instead of typing estimates by hand, is the difference between a real pipeline and a demo.

6.1 The four numbers

Once a shape is a valid, closed solid (Lesson 4), you can compute physical facts from it. These are what turn a drawing into something a simulator can move.

Volume. How much space the shape occupies.

Mass. Volume multiplied by material density. Material choice has to be part of the model, not an afterthought, because the same shape in aluminium and in steel has a different mass.

Centre of mass. The single point where the object balances.

Inertia tensor. A 3x3 table describing how hard the object is to spin about each axis. Physics engines need this number. If it is wrong, the simulation is wrong in a way that looks plausible on screen, which is the dangerous kind of wrong.

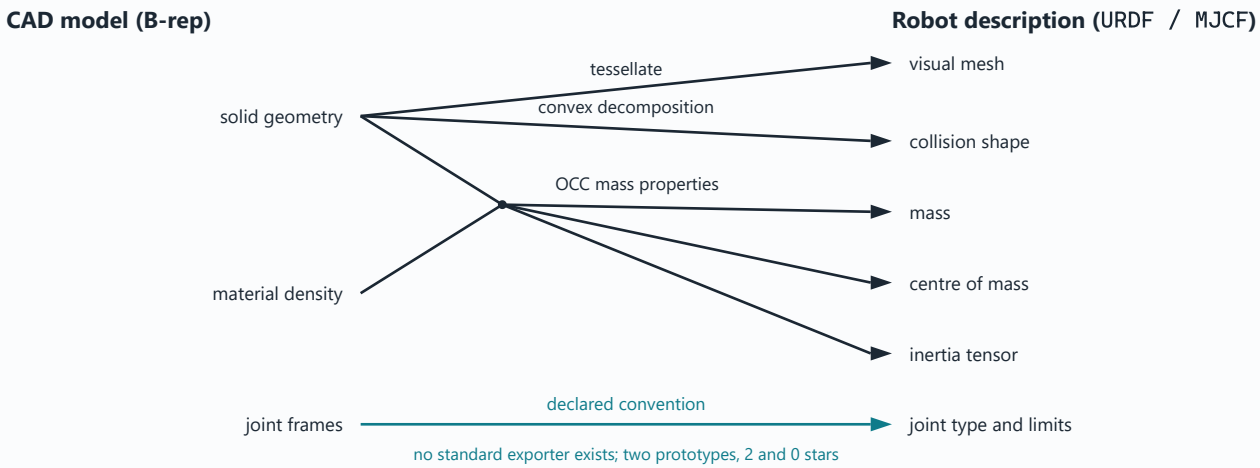


Figure 6. Everything except the joints is computable, and the gap is nameable.

6.2 Where these numbers come from

`trimesh` computes these four numbers from a mesh. OpenCASCADE computes them exactly from a B-rep. Prefer the exact source when you have one, since a mesh approximation compounds any error already present in the tessellation.

Why this matters for SuperHuymn specifically. The pipeline goal is CAD to physics. The handoff between those two worlds is exactly this list of numbers, plus collision shapes (Lesson 7) and joints (Lesson 8). Getting mass properties out of the CAD model automatically, instead of having

a person type estimates into a URDF file by hand, is the difference between a real pipeline and a demo.

KEY POINTS

- Volume, mass, centre of mass, and the inertia tensor are the four numbers that bridge CAD and physics.
- Mass depends on volume and on material density, so material has to be a real property of the CAD model.
- An exact B-rep computation (OpenCASCADE) is preferable to a mesh approximation (`trimesh`), since mesh error compounds into the physics numbers.
- A wrong inertia tensor produces a simulation that looks fine but behaves wrong, which is harder to catch than an obviously broken one.

WHAT SUPERHUYMN DOES WITH THIS

Automatic mass-property extraction from the CAD model, rather than hand-typed estimates in a URDF file, is the concrete difference between a demo and a real CAD-to-physics pipeline. This is the exact number set Forge Studio's export step needs to compute directly from OpenCASCADE, not approximate from a display mesh.

Lesson 7. Collision and clearance

WHAT YOU WILL LEARN

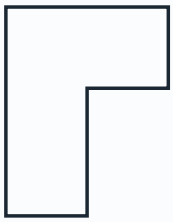
- The difference between checking if two parts overlap right now and checking if they collide during motion.
- Why real-time physics never uses exact geometry for collision, and what it uses instead.
- The one practical rule for not confusing a design-time check with a simulation-time check.

7.1 Do these parts overlap right now?

The exact answer: perform a boolean intersection between the two shapes, and check whether the result has a volume greater than zero. This is slow but exact. This is what the W16 engine demo did to check the piston against the cylinder liner.

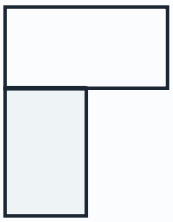
7.2 Do they come close during motion?

Sample the motion at many positions, and check the minimum distance at each one. This is called a sweep, and it is how you prove a mechanism does not crash into itself across its full range of motion, not just at rest.



Exact B-rep boolean.
esign-time verification. Slow, correct

Convex hull.
Wrong: fills the inner corner



Convex decomposition, 2 pieces.
Correct enough, fast

Figure 8. Why a simulation can report no collision when a real part would crash.

7.3 Fast, approximate collision

Real-time physics does not use exact booleans, because they are too slow to run every frame. Instead it wraps each part in a simpler shape: a box, a sphere, a capsule, or a convex hull (the shape you get by shrink-wrapping the part). A complicated part gets split into several convex pieces first, a process called convex decomposition; the common open tool for this is V-HACD. MuJoCo and every game physics engine work this way.

7.4 The practical rule

Use exact booleans for design-time verification, and convex approximations for simulation. Do not confuse the two. A simulation that reports "no collision" may just be using a coarse hull that misses a real, thin overlap.

KEY POINTS

- "Do they overlap now" is answered exactly with a boolean intersection and a volume check.
- "Do they collide during motion" needs a sweep: sampling many positions and checking the minimum distance at each.
- Real-time physics uses fast approximate shapes (boxes, capsules, convex hulls, or several convex pieces from convex decomposition), never exact booleans, because exact checks are too slow to run every frame.
- Exact booleans belong at design time. Convex approximations belong at simulation time. Mixing them up produces false confidence.

WHAT SUPERHUYMN DOES WITH THIS

The W16 engine demo's piston-versus-liner check is the working example of exact, design-time collision checking already in the stack. Extending it to a full-motion sweep, rather than a single rest-pose check, is the concrete next step for any mechanism Forge Studio verifies. A convex-decomposition step (V-HACD) is the missing piece before any part can go into a fast, real-time simulation like MuJoCo.

Lesson 8. Joints, kinematics and robot descriptions

WHAT YOU WILL LEARN

- The four concepts, link, joint, degrees of freedom and kinematics, that describe any robot.
- The difference between forward and inverse kinematics.
- Why three robot-description file formats exist, and why picking one as a source of truth matters.

8.1 Links, joints and degrees of freedom

A robot is a set of rigid bodies connected by joints. That is the whole idea.

Link. One rigid body: a single part, or a welded group of parts.

Joint. The connection that allows motion between two links. Common types are revolute (rotates about an axis, like an elbow), prismatic (slides along an axis), and fixed (no motion, used to attach a sensor).

Degrees of freedom (DOF). How many independent numbers you need to describe the pose of the whole mechanism. A 6-DOF arm has six joints that move.

8.2 Forward and inverse kinematics

Forward kinematics. Given the joint angles, where is the hand? This always has one answer, and it is always directly computable.

Inverse kinematics. Given a desired hand position, what joint angles get there? This can have many answers, or none. It is harder, and it is what a library like Pinocchio is for.

8.3 What a robot description file actually is

A robot description file (URDF, MJCF or SDF) is a list of links and joints. For each link, it records its visual shape, its collision shape, its mass, its centre of mass and its inertia tensor. Notice that everything on that list except the joint structure comes straight out of a CAD model (Lesson 6). That is why a CAD-to-URDF exporter matters so much: without one, a person retypes these numbers by hand, and hand-typed numbers are wrong more often than computed ones.

8.4 Why the three formats differ

URDF is the ROS standard, and it is a tree structure, so it cannot describe a closed loop, like a four-bar linkage, without a workaround. MJCF is MuJoCo's format, and it carries simulation detail like contacts, actuators and sensors. USD is NVIDIA's format, and it carries whole scenes and materials. Converting between formats loses information every time, so a team has to decide which one is its source of truth.

KEY POINTS

- A robot is links (rigid bodies) connected by joints (revolute, prismatic, or fixed).
- Forward kinematics always has one answer. Inverse kinematics can have many answers or none, and needs a dedicated solver like Pinocchio.
- A robot description file records, per link, exactly the numbers a CAD model already produces: visual shape, collision shape, mass, centre of mass, inertia tensor.
- URDF, MJCF and SDF or USD are not interchangeable without loss. Pick one as the source of truth.

WHAT SUPERHUYMN DOES WITH THIS

A real build123d-to-URDF or build123d-to-MJCF exporter is the concrete tool this lesson points to. Without it, a person retypes mass, centre of mass and inertia numbers into a robot description file by hand, exactly the kind of retyped number this guide keeps flagging as an error source. The exporter is what turns Lesson 6's automatic mass properties into a usable robot file.

Lesson 9. Constraints and solvers

WHAT YOU WILL LEARN

- The difference between sketch constraints (2D) and assembly mates (3D), and why they use similar math.
- Why code-CAD like build123d mostly skips constraint solving, and why that suits an AI agent.
- The open constraint-solver projects worth knowing, each with its tag and its verified star count.

9.1 Sketch constraints and assembly mates

Sketch constraints work in 2D: this line is horizontal, these two circles are concentric, this distance is 25 mm. A solver finds positions that satisfy all of them at once. SolveSpace and FreeCAD's planegcs are the open implementations. Dune3D pairs SolveSpace's solver with OpenCASCADE's kernel.

Assembly mates work in 3D, on whole parts: this shaft is coaxial with that hole, these two faces are flush. The idea is the same, applied one level up.

9.2 Why this matters for agentic CAD

Code-CAD like build123d mostly skips constraint solving. You compute positions directly with arithmetic instead. This is simpler, and it is why code-CAD suits large language models: writing a coordinate is easier for a model than declaring a relationship for a solver to satisfy.

The cost shows up at the assembly level. "This bolt goes in that hole" has to become explicit coordinates, worked out by the agent, rather than a relationship it can just declare. Assemblies are where current text-to-CAD systems, SuperHuymn's included, are weakest, and several 2026 papers target exactly this gap.

There is a real link to robotics here. `onshape-to-robot` works because Onshape assemblies carry named mates, so a tool can read degrees of freedom automatically off connectors named `dof_something`. Anything built on build123d has to invent its own convention for declaring which joint is which, because build123d has no mates to read. That convention is worth designing deliberately, not by accident.

9.3 Open constraint solvers, with tags

The table below is carried over from the research pack exactly as written, tags, star counts and all.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
SolveSpace, 4,144 stars (verified)	STUDY	A complete, GPL, from-scratch parametric CAD app with its own constraint solver and file format; read as an architecture reference for how a small team built history-based sketch constraints end to end, not something to embed directly.
planegcs, 99 stars (verified)	ADOPT	A WebAssembly wrapper around FreeCAD's own 2D geometric constraint solver (`DogLeg/Levenberg-Marquardt/BFGS/SQP`), runnable client-side in JS. Directly usable inside the WebGPU browser frontend to give interactive, low-latency sketch-constraint solving (drag a line, watch constraints resolve) without a server round trip, while `build123d/OCCT` still owns the authoritative 3D model server-side.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
FreeCAD's native `Sketcher::PlaneGCS` (C++, inside `FreeCAD/FreeCAD`, 33,338 stars, verified)	STUDY	The upstream source planegcs is compiled from; read before depending on planegcs so version `drift/bugs` can be traced back.
OndselSolver, 6 stars (verified)	WATCH	`Multibody-dynamics/assembly` constraint solver newly folded into FreeCAD's Assembly workbench; extremely young (single-digit stars, small contributor base), worth monitoring for kinematic joint-solving ideas but not production-ready enough to depend on.

KEY POINTS

- Sketch constraints (2D) and assembly mates (3D) solve the same kind of problem at two different scales.
- build123d skips constraint solving in favour of direct arithmetic, which suits an AI agent, but pushes the cost onto assemblies, where positions must be computed explicitly.
- `onshape-to-robot` works because Onshape's named mates carry degrees-of-freedom information automatically. build123d has no equivalent, so that convention has to be designed on purpose.
- planegcs (ADOPT) brings interactive constraint solving into the browser without a server round trip. SolveSpace and FreeCAD's native solver are STUDY references. OndselSolver is WATCH, too young to depend on.

WHAT SUPERHUYMN DOES WITH THIS

Forge Studio's WebGPU browser frontend can use planegcs directly for interactive, low-latency sketch-constraint solving in the browser, while build123d and OpenCASCADE keep authority over the real 3D model on the server. Separately, the assembly-level DOF convention that `onshape-to-robot` gets for free from Onshape's named mates has to be designed deliberately for anything Forge Studio builds on build123d, since no equivalent exists to read off automatically.

Lesson 10. Tolerances, GD&T and machine-readable intent

WHAT YOU WILL LEARN

- Why a dimension without a tolerance is an incomplete instruction to a machinist.
- What GD&T and semantic PMI mean, and what AP242 adds over older STEP versions.
- Why attaching machine-readable tolerances to a generated part is a sharper claim than better-looking geometry alone.

10.1 A dimension is not enough

A drawing that says a hole is 20 mm is incomplete. Real parts are never exact. What a machinist needs is 20 mm plus or minus 0.05 mm, and often more: how round the hole must be, how perpendicular to a reference face, how precisely located relative to two reference edges.

This language is called GD&T, geometric dimensioning and tolerancing. It has historically lived as text and symbols on a 2D drawing, readable by a person only.

10.2 AP242 and semantic PMI

AP242 is the current STEP standard (ISO 10303-242) for carrying GD&T as data attached directly to the 3D model, called semantic PMI (product and manufacturing information). "Semantic" here means a machine can read the tolerance and act on it, instead of just displaying a symbol for a person to read. Earlier STEP versions, AP203 and AP214, only carried graphic PMI: polylines that draw a dimension on screen but carry no machine-readable meaning behind them.

A downstream CAM (computer-aided manufacturing), CAI (computer-aided inspection), or CMM (coordinate measuring machine) program can parse a tolerance directly out of an AP242 STEP file and act on it, instead of a person re-typing it off a drawing.

10.3 The tooling picture

The table below is carried over from the research pack exactly as written.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
ISO 10303-242 (STEP AP242), overview	STUDY	The target export format `if/when` Forge Studio needs to hand a toleranced part to a customer's CAM or CMM pipeline; not needed for the VC demo, but a real gap for any manufacturing customer.
OCCT's own PMI classes (`XCAFDoc_DimTolTool`, `STEPCAFControl_Writer`, part of the OpenCASCADE kernel already in the house stack)	STUDY, unverified depth	OCCT has XCAF `PMI/D`&T data-model classes, but we did not confirm in this session that the specific OCCT version bundled with `cadquery-ocp` can round-trip full AP242 semantic PMI end to end. This needs a hands-on test (write a toleranced STEP file, re-read it, check the D&T objects survive) before claiming the free path works.
HOOPS Exchange (Tech Soft 3D, commercial)	WATCH	The commercial fallback if native OCCT PMI support proves incomplete: explicitly supports `reading/writing` semantic PMI in STEP AP242 across `CATIA/NX/Creo/SolidWorks`/Inventor. Pricing is not public, treat as a cost unknown, only worth pricing out once a real customer needs cross-CAD PMI.
CAD Exchanger SDK (commercial)	WATCH	Same category as HOOPS Exchange, PMI annotations `to/from` STEP AP242 and JT. A second commercial option to quote against HOOPS if that path is ever needed.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
ASME Y14.5-2018 (R2024), the GD&T standard itself (paywalled; only secondary summaries read this session, e.g. Sigmetrix guide)	STUDY	Before Forge Studio (or Gemini) writes any GD&T callout, someone on the team needs to actually read this standard or bring in a GD&T-literate reviewer, the standard itself was not read in this session, only blog summaries of it.

10.4 Why SuperHuymn should care

Tolerances are where an AI-generated part meets reality. A generated model with no tolerance information is an idea, not a specification. If Forge Studio can attach machine-readable tolerances to generated geometry, it produces something a supplier can quote from directly. That is a sharper claim than "better geometry", and far fewer competitors are making it.

KEY POINTS

- A bare dimension, like "20 mm", is incomplete without a tolerance. GD&T is the language for the tolerance.
- AP242 lets a machine read a tolerance directly out of a STEP file (semantic PMI). Older STEP versions only drew the tolerance as a picture (graphic PMI), which a machine cannot act on.
- Whether SuperHuymn's own OpenCASCADE version can round-trip full AP242 PMI has not been tested. This is a hands-on gap, not a documentation gap.
- Commercial fallbacks (HOOPS Exchange, CAD Exchanger SDK) exist if the free OCCT path proves incomplete, but neither has public pricing.

WHAT SUPERHUYMN DOES WITH THIS

Attaching machine-readable AP242 tolerances to generated geometry is a sharper, less-copied claim than "better-looking geometry". The concrete next step, before relying on this, is a hands-on round-trip test: write a toleranced STEP file with OCCT's `STEPCAFControl_Writer`, read it back, and confirm the tolerance data survives in the version of OCCT that ships with `cadquery-ocp`. That test has not been run yet.

Lesson 11. Manufacturability as geometry

WHAT YOU WILL LEARN

- Why "can this part be made" is a question that can be answered from shape alone.
- The specific geometric checks used for machining, 3D printing, and sheet metal.
- Why encoding these checks is cheap and changes what the product is.

11.1 Machining checks

Whether a part can be made is decidable from its shape plus the manufacturing process. This makes it perfect for automated checking, and it is the cheapest credibility SuperHuymn can buy.

Standard machining checks, each one a geometry question:

- Hole depth versus diameter. Beyond roughly five to one, drills wander and break.
- Internal corners. A round cutter cannot cut a sharp internal corner, so every internal corner has a radius whether the designer drew one or not.
- Tool reach. Can a cutter of realistic length and diameter physically reach a surface without its holder hitting something else?
- Deep, narrow pockets and slots. Long, thin tools chatter (vibrate) inside them.
- Flat-bottomed holes. Possible, but they need a specific tool, and they cost more.
- Wall thickness. Too thin, and a wall vibrates, distorts, or breaks during machining.
- Undercuts. Features a tool cannot reach from any single direction, which force a re-fixture (a second setup) or a different process entirely.

11.2 Other processes

For 3D printing, the list changes: overhang angle, minimum feature size, support access, and warping on large flat areas. For sheet metal, it changes again: bend radius versus material thickness, minimum flange length, and hole distance from a bend. There is 2026 research specifically on sheet-metal bending manufacturability (BenDFM), and on estimating bending effort directly from CAD features.

KEY POINTS

- Manufacturability is decidable from shape plus process, which makes it a machine-checkable gate, not a matter of judgment.
- Machining checks include hole depth-to-diameter ratio, internal corner radius, tool reach, wall thickness, and undercuts.
- 3D printing and sheet metal each have their own, different checklist: overhang and support access for printing, bend radius and flange length for sheet metal.
- 2026 research (BenDFM) targets sheet-metal manufacturability specifically, showing this is an active area, not a solved one.

WHAT SUPERHUYMN DOES WITH THIS

The recommendation from the underlying research stands as stated: encoding this checklist as automated checks is roughly half a day of engineering work, and it changes what the product is. A part that passes an automated manufacturability gate is a claim a supplier can act on, not just a shape that looks plausible on screen.

Lesson 12. Meshing for analysis

WHAT YOU WILL LEARN

- Why engineering analysis uses a different kind of mesh than a display mesh.
- What makes an analysis mesh good or bad, and why element shape matters.
- The specific tool chain, from B-rep to a result mapped back onto a part's faces.

12.1 Two different meshes

Confusingly, engineering analysis also uses meshes, but not the same kind used for display (Lesson 2).

A display mesh only needs to look right from the outside. Its triangles can be almost any shape, and only the surface matters.

An analysis mesh, used for finite element analysis (FEA), fills the whole volume with elements, usually tetrahedra or hexahedra (3D shapes with four or six faces). Element quality matters a great deal: long, thin slivers give wrong answers. The mesh needs to be finer where stress changes quickly, such as at a fillet, and can stay coarse everywhere else.

12.2 The tool chain

`gmsh` is the standard open mesher for this job, and it is what both `anvilate` and `stellarmesh` build on. CalculiX then solves the resulting system of equations. If SuperHuymn's FEA tab is to mean anything, this is the chain underneath it: B-rep, then gmsh, then CalculiX, then results mapped back onto the faces a user recognises.

KEY POINTS

- A display mesh only needs to look right from outside. An analysis mesh fills the whole volume and needs good element quality throughout.
- Long, thin elements (slivers) produce wrong analysis results, even if the mesh looks fine.
- An analysis mesh should be finer where stress changes quickly, like at a fillet, and coarser elsewhere.
- The open tool chain is B-rep, then `gmsh` for meshing, then CalculiX for solving.

WHAT SUPERHUYMN DOES WITH THIS

This chain, B-rep to `gmsh` to CalculiX, then results mapped back to the faces a user recognises, is what SuperHuymn's FEA tab needs to actually mean something. Without this specific chain, an "FEA tab" is a label with nothing real underneath it.

PART III. The field as it stands

Part I covered geometry itself. Part II covered how a shape connects to physics, manufacturing and machines. This part covers the field: the open-source projects, the standards and the tools that already exist. It answers one question for each one. What should SuperHuymn do about it.

Lesson 13. The five layers and where the value sits

WHAT YOU WILL LEARN

- The five layers the agentic CAD field split into by September 2026, and which ones are settled versus still open.
- Why the layer with the most public projects is the worst layer for a small team to compete in.
- What it means when four separate teams solve the same small problem without knowing about each other.

13.1 Five layers, one market

"Agentic CAD" means using an AI model to write or edit a 3D design, usually by having the model write code in a library like build123d rather than clicking buttons in a CAD program. By September 2026 that field had split into five layers. A layer here means a distinct piece of the stack that different companies compete on separately, the way a search engine and a web browser are different layers even though most people only see the browser.

A kernel is the core program that stores a shape's exact geometry and calculates operations on it, such as cutting a hole or joining two solids. OpenCASCADE, often shortened to OCCT, is the kernel SuperHuymn already uses, through the Python wrapper build123d.

LAYER	STATE IN SEPTEMBER 2026	WHO WINS
Geometry kernel	Settled. OpenCASCADE, decades of edge cases	Nobody new
Code-CAD library	Settled. build123d and CadQuery	Nobody new
Agent harness	Crowded. 51 public text-to-cad repositories, one at 14,600 stars, plus 116 build123d repositories	Commoditising
Verification	Nearly empty	Open
Product and distribution	Incumbents just moved	Autodesk, Siemens, Dassault, PTC

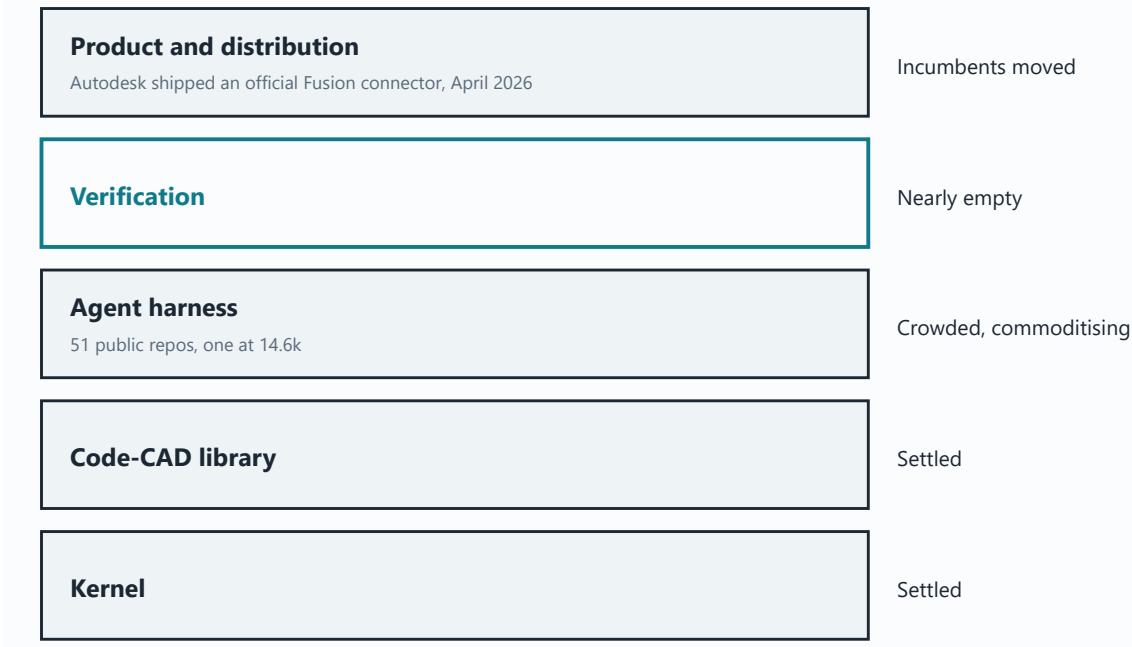


Figure 1. The empty band is the opportunity, and it is visually empty.

A code-CAD library is a Python package used to script a 3D model instead of drawing it by hand. An agent harness is the software that connects a language model, such as Gemini or Claude, to a CAD tool so the model can build something through it. Verification means checking that a generated shape is actually correct and usable, not just that it looks right on screen. Commoditising means a product is losing its scarcity because many people can now build a similar version cheaply.

13.2 Three facts about the crowded layer

Three facts make the agent-harness layer a bad place for a small team to plant a flag.

First, on 28 April 2026 Autodesk shipped an official connector letting Claude drive Autodesk Fusion directly, using MCP. MCP, short for Model Context Protocol, is a standard way of letting an AI model call an outside tool, such as a CAD program, through a fixed set of commands. Once the market leader ships this as a free feature, "an AI can drive CAD" stops being something a small team can sell on its own.

Second, there are now well over a hundred public projects that wrap a CAD program's scripting interface in an MCP server and hand it to an AI model. The hundred-and-first one is worth close to nothing, because the work of wiring an API to an MCP server has become routine.

Third, at least four separate projects arrived, without coordinating, at the same idea: a generated shape must be checked against a written-down set of claims about what it should be. Those four are cad-khana, cad-cae-copilot, `MLYengineering/Text2CAD`, and SuperHuymn's own W16 engine demo validator gate. Full descriptions of the first three are in Lesson 14 and Lesson 17.

13.3 What the convergence means

When four teams that never talked to each other solve the same small problem the same way, that is a signal, not a coincidence. It means the problem is real and still unsolved at scale. Generating a shape from a text prompt is close to a solved problem: dozens of projects do it adequately. Checking that the generated shape is actually correct, watertight, collision-free and machinable is not solved by any of them yet. Lesson 17 covers this gap in full, because it is where a small team can still own something the incumbents have not built.

KEY POINTS

- The field has five layers: kernel, code-CAD library, agent harness, verification, product.
- The kernel and code-CAD layers are already settled in build123d's favour. Nobody is displacing OpenCASCADE.
- The agent-harness layer is the most crowded and the fastest to become worthless as a selling point.
- Four independent teams solving the same checking problem is strong evidence that verification is the open layer.
- A vendor shipping a feature for free (Autodesk plus Claude on Fusion) removes that feature as a differentiator for everyone else overnight.

WHAT SUPERHUYMN DOES WITH THIS

Stop pitching Forge Studio as "an agent that drives CAD." That description is now true of a free Autodesk feature. Every new engineering hour aimed at the harness layer (a better prompt, a nicer chat UI, another MCP tool) should instead be weighed against putting that hour into the verification layer described in Lesson 17, because that is the layer with room left in it.

Lesson 14. Agent harnesses and text-to-CAD

> WHAT YOU WILL LEARN
>
> - How many public projects already exist for turning text into a 3D model, and how concentrated the field is around one leader.
> - The three projects worth reading in full, and exactly what each one gets right.
> - Why writing another MCP-wrapped CAD API is a waste of a week.

14.1 How big the field already is

GitHub groups related projects under a topic tag. The `text-to-cad` topic held 51 public repositories on 2026-09-08. The `build123d` topic held 116. Star counts below were read directly from those pages that day, so they carry the **(verified)** marker used throughout this book for numbers checked by hand rather than pulled from a search summary.

14.2 The verified star table

REPOSITORY	STARS	WHAT IT IS
`earthtojake/text-to-cad`	**14,600**	A library of agent skills for CAD, CAE and CAM. Python, build123d, OpenCASCADE. Exports STEP, STL, 3MF, DXF, GLB, URDF
`Adam-CAD/CADAM`	**5,100**	Open-source text-to-CAD web application. TypeScript, React, WASM, OpenSCAD
`gumyr/build123d`	**3,000**	The library SuperHuymn already builds on
`Pan-Chera/Multi-Agent-CAD`	**934**	Decoupled multi-agent text-to-CAD via constrained test-time compute. build123d, LangGraph, Qwen, aider
`partcad/partcad`	**493**	Package manager for hardware. Digital thread and technical data package tooling, with manufacturability checks
`forgent3d/forgent3d-desktop`	**244**	Desktop AI 3D model generation. Built around build123d and coding agents
`EESJGong/Graph-CAD`	**138**	Hierarchical, geometry-aware graph representations for text-to-CAD
`jdilla1277/agentcad`	**113**	CAD command line tool and MCP server for AI agents
`armpro24-blip/cad-cae-copilot`	**57**	Text-to-CAD and text-to-CAE with real build123d geometry, editable parameters, stable topology pointers, deterministic critique, MCP tools, CalculiX, topology optimisation
`SadilKhan/NURBGen`	**32**	AAAI 2026. Text-to-CAD through LLM-driven NURBS modelling
`Kevoyuan/AgentSCAD`	**10**	OpenSCAD agent with automated geometry repair and manufacturing validation
`vul-os/kerf`	**9**	MIT CAD across disciplines: parametric modelling, drawings, FEM, CAM, PCB, BIM. Chat-driven, offline-first
`SmartAI/Chamfer`	**8**	Text and image to CAD agent harness supporting multiple MCPs and LLMs
`clay-good/anvilate`	**7**	Local-first design agent. Plain English to physics-validated parametric STEP or DXF plus editable Python. CalculiX, gmsh, DFM
`MLYengineering/Text2CAD`	**6**	Contract-driven agentic system for deterministic CAD from natural language, headless
`cyberchitta/cad-khana`	**14**	Claude Code skill plus diagnostics-first build123d wrapper

REPOSITORY	STARS	WHAT IT IS
`baibai2013/build123d-cad`	2	25 sub-skills covering CAD through URDF, MuJoCo, FEA, PCB, firmware. Detailed below

Some file-format terms in that table need a plain definition before they come up again. STEP is a standard file format for exact 3D shapes, readable by most CAD programs. STL and 3MF describe a shape as a mesh of small flat triangles, the format used for 3D printing. DXF is a 2D drawing format used by laser cutters and CNC machines. GLB carries a 3D model with its colour and texture, the format most web viewers expect. URDF, short for Unified Robot Description Format, is a text file describing a robot's parts, joints and limits for a physics simulator. DFM, short for design for manufacturability, means checking that a shape can actually be cut, printed or milled before calling the design finished.

14.3 The three repositories that matter most

`earthtojake/text-to-cad`. ADOPT. **Read it this week.** At 14,600 stars it is the centre of gravity for this whole field. Its own description is worth reading twice. It calls itself not a tool but a library of agent skills for CAD, CAE and CAM (CAE means computer-aided engineering, simulation and analysis of a design; CAM means computer-aided manufacturing, turning a design into machine instructions). Its stack is Python, build123d and OpenCASCADE, and its export list is STEP, STL, 3MF, DXF, GLB and URDF. That list is almost exactly what Forge Studio already targets. It has a fork with its own following, `cedrickchee/text-to-cad` at 23 stars, and its `cad` skill is listed on Claude Skills Hub. If Forge Studio does something this project does not do, name that thing precisely, because it is the whole pitch.

`baibai2013/build123d-cad`. ADOPT **the architecture. It is the closest thing to a blueprint.** Two stars, mostly Chinese-language documentation, and it independently designed close to the same product SuperHuymn is building. It is a hardware-design super-skill: one parent skill routes to 25 sub-skills across two tracks, mechanical CAD and PCB electrical design, joined end to end. The sub-skill list reads like SuperHuymn's own roadmap, already built.

- CAD: `mechanical`, `parts-catalog`, `viewer`, for parametric modelling, reusing standard parts, assembly, an explode animation, a browser preview and screenshot self-checking.
- Robot description: `urdf`, `srdf`, `sdf`, exporting a per-link textured GLB in metric units with movable joints, plus MoveIt planning groups and Gazebo worlds. SRDF is Semantic Robot Description Format, extra robot metadata layered on top of URDF. SDF is Simulation Description Format, a robot and world file used by some physics simulators. MoveIt is motion-planning software for robot arms. Gazebo is a physics simulator often paired with it.
- Manufacturing: `gcode` slicing pre-check, `sendcutsend` laser quoting, `bambu-labs` print upload.
- PCB: `pcb` via tscircuit, `electronics-bom`, `circuit-simulation`, `pcb-mechanical-reliability`, producing Gerber (a standard PCB manufacturing file format), a bill of materials, a component placement list, and JLCPCB quoting.
- Virtual validation: `requirements-verification`, `actuator-sizing`, `fea`, `wear-fatigue`.
- Motion: `simulation`, `mujoco-simulation`, `gait-optimization`, `motion-control`.
- Pre-hardware closure: `firmware` dry-run, `sim2real-calibration`, `integration` bring-up gate, `robot-dog-digital-twin`.

Three engineering rules it enforces are worth copying outright. Capability as files: each sub-skill is a SKILL.md file the agent reads, plus deterministic scripts, reference tables and regression tests. Scripts that fail loudly and assume no language model is present, so they still run correctly in an automated build pipeline with nobody watching. Sub-skills that never call each other's code directly and only communicate through agreed file paths. It also ships a sibling parts library, `build123d-parts-lib`, with 66 parametric standard parts across 8 categories: 21 fasteners, 19 actuators, 9 transmission parts, 7 bearings, 4 pins, 3 servos, 2 retaining rings, 1 seal.

`armpro24-blip/cad-cae-copilot`. STUDY **closely. 57 stars.** Text-to-CAD and text-to-CAE on real build123d and OpenCASCADE geometry, with editable parameters, stable topology pointers and deterministic critique, exposed over MCP, wired to CalculiX (an open finite-element solver) and topology optimisation. Those two phrases name the two hardest problems in agentic CAD. A stable topology pointer is a way to refer to a face or edge of a shape that keeps working even after the shape is edited, solving what is called the naming problem. Deterministic critique means the checker gives the same answer every time on the same input, so it does not hallucinate a pass or a fail. If this project has solved either problem properly, reading its code is worth a week.

14.4 The rest, briefly

- `Pan-Chera/Multi-Agent-CAD`, **934 stars.** STUDY. It splits the work across several agents and spends extra compute at the moment of generation, which is the current best answer for reliability. It uses LangGraph (a framework for chaining agent steps together), the open model Qwen, and aider (a coding-agent tool), so it is a working template for running this pattern without paying frontier-model prices.
- `partcad/partcad`, **493 stars.** ADOPT **for the industrial pitch.** It manages physical parts the way a software package manager manages code libraries, using the language of a "digital thread" and a "technical data package." That is the vocabulary industrial buyers already use. It also checks manufacturability. If SuperHuymn ever sells into a factory rather than a demo, this project's framing is more useful than its code.
- `clay-good/anvilate`, **7 stars.** STUDY. **Punches far above its stars.** Local-first, physics-validated output, using CalculiX and gmsh (a mesh generator), applying DFM checks, and exporting STEP or DXF that opens directly in CATIA, SolidWorks, NX or AutoCAD, plus editable Python. That output choice is exactly right for industrial users, who will not accept a triangle mesh as a deliverable.
- `MLYengineering/Text2CAD`, **6 stars.** STUDY. "Contract-driven" and "deterministic" are the same instinct as cad-khana's claims (Lesson 17) and the W16 demo's validator gate. Three independent projects converging on the same word is a signal, not a coincidence, and Lesson 17 explains why.
- `Adam-CAD/CADAM`, **5,100 stars.** WATCH. Adam is a funded commercial competitor and this is their open web app. It runs on OpenSCAD and WebAssembly rather than a full boundary-representation kernel, which caps its engineering seriousness and explains how it fits inside a browser.
- AVOID**: writing another MCP-wrapped CAD API.** There are well over a hundred already. The marginal one adds nothing.

KEY POINTS

- 51 public text-to-cad repositories and 116 build123d repositories existed as of 2026-09-08, but almost all activity concentrates in the top few.
- `earthtojake/text-to-cad` at 14,600 stars is the project to read first and the one Forge Studio should be compared against by name.
- `baibai2013/build123d-cad`, at only 2 stars, is a near-complete blueprint of SuperHuymn's own roadmap and should be read end to end.
- cad-cae-copilot's "stable topology pointers" and "deterministic critique" name the two hardest unsolved problems in this field.
- Do not spend a week building another MCP wrapper around a CAD API. That category is already full.

WHAT SUPERHUYMN DOES WITH THIS

Read `baibai2013/build123d-cad`'s README and sub-skill list in full, since it is a free blueprint of work SuperHuymn is already doing independently, and check Forge Studio's own roadmap against its 25 sub-skills for gaps (the URDF exporter gap is covered in Lesson 19). Separately, compare Forge Studio's export list against `earthtojake/text-to-cad`'s (STEP, STL, 3MF, DXF, GLB, URDF) and be able to name, in one sentence, what Forge Studio does that this 14,600-star project does not.

Lesson 15. MCP servers and agent skills

WHAT YOU WILL LEARN

- Why an MCP server for a CAD program is now a commodity, and what actually separates a good one from a bad one.
- The agent-skill projects worth adopting today for electronics and robotics.
- The one move that gets SuperHuymn's own work in front of every Claude Code and Codex user for about a day of effort.

15.1 MCP servers

SERVER	STARS	NOTE
Blender MCP	~17,800	The category giant. Mesh, not CAD
`pzfreo/build123d-mcp`	**81 (verified)**	MCP server for build123d, framed as improving AI cognition when modelling
`armpro24-blip/cad-cae-copilot`	**57 (verified)**	CAD and CAE tools over MCP
`Casys-AI/mcp-build123d`	search-reported	Executes parametric models, reports OCCT metrics, exports STEP, STL, GLB
`Alfredoalv13/shapr3d-mcp`	**19 (verified)**	Writes build123d scripts, checks with render preview, opens STEP in Shapr3D
`Svetlana-DAO-LLC/cad-agent`	search-reported	build123d plus MCP, agents create, see, evaluate and iterate unattended
FreeCAD MCP	~617	Several competing implementations
KiCad MCP	~405	Netlist, BOM, DRC, PCB visualisation
PlatformIO MCP v2.2.1	May 2026	Full PlatformIO workflow as MCP tools
Onshape MCP	~11	TypeScript, MIT, cloud REST
ODA MCP Servers	Q3 2026	Open Design Alliance, STEP, IFC, DWG

What this table says. Almost every CAD package already has an MCP wrapper. Most of them just hand a language model a scripting interface and hope it writes correct code. Having an MCP server at all cannot be the differentiator any more, because that is table stakes as of 2026. The real difference has to be whether the geometry coming back is checked: watertight (no gaps or holes in the surface), collision-free, within tolerance, and actually machinable. That checking layer is Lesson 17.

15.2 Agent skills

An agent skill is a packaged set of instructions, usually a file named SKILL.md plus supporting scripts, that teaches a coding agent like Claude Code or Codex how to do one specific job well.

- `NVIDIA/skills`. ADOPT. Official, NVIDIA-verified agent skills for Claude Code, Codex and others, released under Apache-2.0 and CC-BY-4.0, covering physical AI and robotics workflows, simulation, CUDA-X (NVIDIA's GPU software stack), RAG (retrieval-augmented generation) and platform tools. The skills live in NVIDIA's own product repositories and mirror here automatically every day. Install with `npx skills add nvidia/skills`, requiring skills CLI v1.5.16 or newer. Each published skill ships a signature file proving where it came from. For a robotics lab this is free, vendor-checked capability, and its daily-sync design is a good model for how SuperHuymn should publish its own skill.
- `aklofas/kicad-happy`. ADOPT **for electronics.** Agent skills for KiCad: schematic analysis, PCB layout review, EMC pre-compliance (checking a board will not radiate interference before it is built), SPICE circuit simulation, datasheet download, component sourcing, and fabrication prep.
- `VoltAgent/awesome-agent-skills`. STUDY. Over 1,000 community and official skills across Claude Code, Codex, Gemini CLI and Cursor.
- `BenCaunt/SynthCAD`, **16 verified.** STUDY. Skills and tools letting coding agents do CAD work.
- `ppak10/RocketSmith`, **21 verified.** STUDY. Agents designing high-powered rockets with build123d, OpenRocket and PrusaSlicer, driven from Claude Code and a Gemini CLI extension. A working example of an agent driving real engineering tools in a domain with hard physical constraints.
- `Soljourner/claude-engineering-skills`. STUDY. Reported over 100 engineering skills, including integrations with ANSYS, OpenFOAM, SolidWorks and COMSOL. Harvest selectively, and never run a harvested skill's bundled scripts unread.
- `010zx00x1/Awesome-Physical-Engineering-AI`. ADOPT **as a reading list.** A curated list of AI tools for hardware engineering across CAD, simulation and manufacturing.

KEY POINTS

- An MCP server for a CAD program is now a commodity. It cannot be the differentiator on its own.
- NVIDIA's own skills repository is free, verified, and a model worth copying for how SuperHuymn should publish.
- Agent skills for electronics (kicad-happy) and rockets (RocketSmith) already prove the pattern of an agent driving a real engineering tool under hard physical constraints.
- Never run a harvested skill's bundled scripts without reading them first.

WHAT SUPERHUYMN DOES WITH THIS

Publish a public Claude skill that drives Forge Studio's own MCP server. A public skill doubles as open educational material, in a form people can actually run. It distributes into every Claude Code and Codex user searching for CAD skills, it costs roughly a day of work, and it gives away no backend code, only the interface. Package it following `NVIDIA/skills`' conventions above: a SKILL.md, supporting scripts, and a signature file.

Lesson 16. Kernels, libraries and the code-CAD stack

WHAT YOU WILL LEARN

- Why the kernel and the code-CAD library layers are already settled, and what that means for where SuperHuymn should not spend effort.
- Which libraries an agent should reach for before generating any shape from first principles.
- The one architectural question, Python versus a purpose-built language, that nobody at SuperHuymn has answered yet.

16.1 The settled layer: OpenCASCADE and build123d

OpenCASCADE (OCCT) is reported stable at version 8.0.1, released 30 July 2026. Version 8.0 moved the codebase to C++17 and added BRepGraph, a graph representation of a shape's boundary representation (its exact surfaces and edges, as opposed to a triangle mesh) with two-way traversal and history tracking. History tracking matters for an agentic system, because it lets the system explain which operation produced which face. Whether cadquery-ocp, the Python binding SuperHuymn depends on, has been updated to bind against OCCT 8.x is unconfirmed; the Python binding usually lags the kernel by months, so check before planning around the new version.

`gumyr/build123d`, 3.0k stars (verified, browsed 2026-09-08). ADOPT. Already the house kernel wrapper. Every other item in this lesson exists to fill a gap build123d does not already cover.

`CadQuery/cadquery`, 5,709 stars (verified). STUDY. build123d's sibling and ancestor, built on the same OCP binding layer. Its longer-lived issue tracker and its fluent, method-chained API are a useful second reference whenever a build123d operation is missing or buggy.

16.2 Parts you should never model from scratch

- **awesome-build123d, 101 verified, and awesome-cadquery.** ADOPT. Start every new feature by checking these curated lists first.
- **bd_warehouse and cq_warehouse.** ADOPT **now.** Libraries that generate parametric fasteners, bearings, threads and chains on demand. An agent should never model a standard M8 bolt from first principles. Combined with build123d-parts-lib's 66 parts (Lesson 14), this removes a whole class of generation failures at zero research cost.
- **cq-gears.** ADOPT **if any drivetrain appears.** Involute gears, the standard tooth shape used in most mechanical gears, are exactly where an AI-written geometry script tends to fail quietly, producing a gear that looks right but does not mesh correctly.

16.3 The wider geometry-processing toolkit

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
build123d, 3.0k stars (verified)	ADOPT	Already the house kernel wrapper; the rest of this table exists to fill gaps it doesn't cover
CadQuery, 5,709 stars (verified)	STUDY	build123d's sibling and ancestor on the same OCP layer; a second reference when a build123d operation is missing or buggy

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
pythonocc-core, 1,966 stars (verified)	STUDY	A thinner, more direct binding to OCCT than OCP; useful when a job needs a raw OCCT class build123d does not expose yet, such as the PMI writers covered in Part II
CGAL, 6,035 stars (verified)	STUDY	Exact-predicate boolean operations and mesh processing that OCCT's own booleans sometimes choke on; a fallback kernel for the hardest CSG cases, not a replacement
libigl, 5,083 stars (verified)	STUDY	Header-only mesh algorithms (geodesics, parametrization, smoothing, per-vertex normals) for cleanup after a part is turned into a mesh, before it goes to the GLB viewer or a lattice pass
Manifold (manifold3d), 2,258 stars (verified)	ADOPT	Already the house stack's fast, guaranteed-watertight triangle-mesh boolean engine; use it for combining shapes that are already meshes, where a full boundary-representation boolean would be slower or where OCCT rejects a body as invalid
OpenVDB, 3,397 stars (verified)	WATCH	A sparse volume representation used in film visual effects and some generative-design tools for lattice infill and blending; no current use case, revisit if generative lattice parts are ever requested
PyVista, 3,801 stars (verified)	STUDY	A mesh analysis and visualisation library built on VTK; a candidate for server-side headless screenshot checks of a part, separate from the live browser viewer
Gmsh (canonical repository on GitLab, star count not verifiable from that source)	STUDY	A tetrahedral and finite-element mesh generator; relevant once a part needs to be handed to a structural or thermal solver, not needed for CAD authoring alone
fTetWild (search-reported)	WATCH	Reliable tetrahedral meshing for messy or near-broken geometry, that keeps working when other meshers fail; the same "only if FEA is added" gate as Gmsh
ezdxf, 1,432 stars (verified)	ADOPT	Already the house stack's DXF library; used for 2D drawing export, alongside its own PDF and SVG rendering

16.4 OCP.wasm and the browser question

`yeicor/OCP.wasm`, 45 verified. STUDY **, possibly strategic.** This is build123d itself, not just a viewer, compiled to run inside a web browser using WebAssembly (WASM), a way of running compiled code such as C++ directly in a browser at near-native speed, together with Pyodide, a version of Python compiled the same way. Its own author describes it as "100 percent private browser-native code-first CAD." If simple parts could be generated directly on a visitor's browser, the demo would stop needing a warm server running in the background, which is worth a spike given the goal of deploying without a local machine (Lesson 22). At 45 stars and effectively one maintainer, it is a proof of concept, not something to depend on for production work yet.

16.5 Alternative kernels, watched, not adopted

A boundary representation, or B-rep, describes a shape using exact mathematical surfaces and edges, rather than an approximation made of flat triangles. That is what OpenCASCADE stores and calculates.

Several projects are building alternative kernels. Fornjot is an early-stage B-rep kernel written in Rust. Truck is a memory-safe NURBS-based B-rep kernel, also in Rust. CADmium is a browser CAD tool built on Truck. Dune3D v1.4.0, released January 2026 under the codename "Einstein," pairs the SolveSpace constraint solver with OpenCASCADE. Zoo's KCL is a purpose-built language for describing geometry as text, rather than Python. All four are worth watching, none are worth migrating to.

Read KCL's design notes specifically. Zoo made the same bet SuperHuymn made, that text is the source of truth for a shape, then built a language designed only for that purpose instead of reusing Python. That is the best available argument for or against the biggest unexamined design decision in Forge Studio: should the agent write ordinary Python, or should it write a domain-specific language built for CAD.

AVOID **: replacing OCCT.** Nothing found in this research justifies it. Any advantage SuperHuymn can build lives above the kernel, not inside it.

KEY POINTS

- The kernel (OpenCASCADE) and the code-CAD library (build123d) layers are settled. Nobody new is winning here, and nobody should try to displace them.
- bd_warehouse, cq_warehouse and build123d-parts-lib remove a whole category of failure by never letting an agent invent a standard part from scratch.
- Manifold, CGAL, libigl, PyVista and pythonocc-core fill specific gaps build123d does not cover on its own.
- OCP.wasm proves build123d can run entirely inside a browser, which matters for a server-free demo, but it is still a one-person proof of concept.
- The Python-versus-domain-specific-language question, which Zoo answered by building KCL, is Forge Studio's biggest unexamined architecture decision.

WHAT SUPERHUYMN DOES WITH THIS

Adopt bd_warehouse and cq_warehouse immediately for any fastener, bearing or chain, and stop letting the agent derive standard parts from first principles, since this is a zero-cost reliability win available today. Separately, before writing any more agent prompts that produce Python, read Zoo's KCL design notes and decide, on purpose, whether Forge Studio's agent should keep emitting Python or move toward a narrower CAD-specific language, rather than leaving that choice undecided by default.

Lesson 17. Verification and regression, the empty layer

WHAT YOU WILL LEARN

- Why this is the shortest lesson in the field survey, and why that shortness is the whole opportunity.
- The two concrete techniques, automated diffing and render-and-ask checking, that already exist and can be adopted directly.
- Why four separate projects landed on the same idea of checking generated geometry against a written contract.

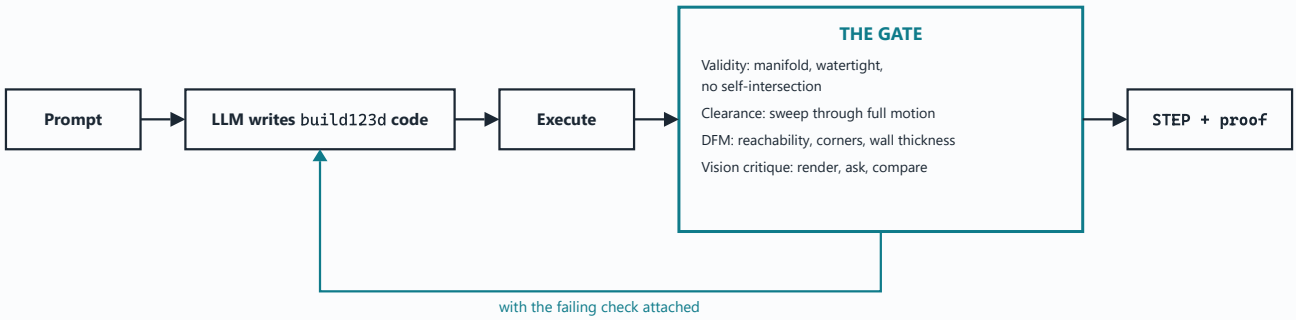


Figure 5. The loop, not the generator, is the product.

17.1 Why this section is short

Lesson 13 named five layers in the agentic CAD field. Four of them are either settled or crowded. This is the fifth: verification, the layer that checks whether a generated shape is actually correct. It is nearly empty. Almost nobody publishing agentic CAD work is systematically checking their own output the way a software team runs automated tests. That gap is the single most important finding in this whole field survey, because it is the one place a four-person team can build something the well-funded incumbents have not built yet.

17.2 diff3d and regression testing

`bdlucas1/diff3d`, 73 verified. ADOPT. A fast visual 3D diff tool for STL, OBJ, 3MF and STEP files. Diffing means comparing two versions of something and showing exactly what changed. This is the missing piece for regression testing an agentic CAD system: after changing a prompt or a model version, diff the new output geometry against a known-good "golden" part, and fail the build if the shape has drifted unexpectedly. Nobody in the agentic CAD field appears to be doing this kind of systematic geometric regression testing yet. It is cheap to add, and it is exactly the kind of engineering discipline that separates a working product from a demo.

17.3 CADCodeVerify: render, ask, repair

CADCodeVerify. STUDY, **then implement.** This approach uses a vision-language model, a model that can look at an image and answer questions about it, to inspect a rendered picture of the generated object. It first generates a list of questions about the object based on the original prompt, then answers its own questions to find places where the object does not match what was

asked for. This is the cheapest available quality win available today: render the part, ask questions about it, repair what is wrong. It catches problems a geometric validator cannot see, such as "that does not look like a bracket," while a geometric validator catches problems the vision model cannot see, such as two parts overlapping inside each other or a surface that is not properly closed. Use both together.

17.4 The contracts pattern, four teams, one idea

The contracts pattern. ADOPT. cad-khana (introduced in Lesson 14's table, 14 stars) declares a set of claims a part must satisfy before it counts as done. `MLYengineering/Text2CAD` (Lesson 14) calls the same idea a contract. cad-cae-copilot (Lesson 14) calls it deterministic critique. And SuperHuymn's own W16 engine demo built the same thing independently, as a validator gate. Four separate arrivals at one idea, in a field where almost nobody talks to anybody else, is strong evidence the idea is both correct and still unbuilt as reusable software. Turning it into a general library, rather than leaving it as four one-off scripts, converts it from an internal habit into a product feature.

17.5 CAD-Judge

CAD-Judge. STUDY. A published approach to morphological grading and verification for text-to-CAD output, built with efficiency as an explicit goal rather than accuracy at any cost.

> KEY POINTS
>
> - Verification is the one layer of the five named in Lesson 13 that is still open, and this lesson matters more than any other in Part III.
> - diff3d gives a cheap, automated way to catch geometric drift the moment a prompt or a model version changes.
> - CADCodeVerify's render-and-ask loop and a geometric validator catch different kinds of mistakes, so use both, not one instead of the other.
> - Four unrelated teams, including SuperHuymn's own W16 demo, independently built a version of the same "contract" idea, which is unusually strong evidence it is the right idea.
> - Generalising the contracts pattern into a reusable library, instead of four separate one-off scripts, turns it into an actual product feature.

> WHAT SUPERHUYMN DOES WITH THIS
>
> Build one shared verification gate that combines a geometric validator, the CADCodeVerify render-and-ask loop, and diff3d regression checks, and run every generated part through it before calling the job done. This is the most concrete, most defensible piece of work named anywhere in Part III, because it sits in the one layer of the five that nobody else has filled yet.

Lesson 18. Generative 3D and the one-way wall

> WHAT YOU WILL LEARN
>
> - Which open generative 3D models exist beyond the TRELLIS model already in the stack, and what each is actually good for.
> - Why turning a generated mesh back into an exact CAD shape is a one-way wall, not a solved conversion.
> - The handful of projects trying to get around that wall, and how far each one has actually got.

18.1 What's already in the stack

Microsoft TRELLIS is already SuperHuymn's generative 3D model. It turns an image or a text prompt into a 3D mesh.

18.2 The open alternatives

- **Hunyuan3D-2 (Tencent).** STUDY. Open weights, mesh-focused, exporting GLB and OBJ, supporting a single image or several images at once through its "2mv" variant, plus a texture-only mode. The strongest open alternative to TRELLIS available today. Full text-to-3D generation, though, stays on Tencent's hosted platform rather than in the released weights.
- **TripoSG / TripoSR.** ADOPT **for previews.** MIT licensed, needing only 6 to 8 GB of GPU memory, generating a shape in under a second, with a very large user community. Good enough for a quick thumbnail preview, not accurate enough for engineering use.
- **Hi3DGen.** STUDY. Estimates a detailed surface-direction map first, then builds geometry from that map. Reported to give the best geometric quality among open image-to-3D models.
- **Step1X-3D (StepFun).** STUDY. Fully open models, including training code and adaptation modules. It combines a hybrid geometry generator with a separate texture-painting step, trained on 2 million curated assets filtered down from over 5 million.
- **TRELLIS 2.** WATCH. Reported to be the current visual-quality leader among open models.

18.3 The wall itself

Turning an exact boundary-representation shape (defined in Lesson 16) into a triangle mesh is trivial; every CAD program does it constantly. The reverse is not. Recovering exact, free-form surfaces from a pile of arbitrary triangles is unreliable, expensive, and gives a different answer each time it is run. Treat any pipeline that ends in a mesh as a dead end for engineering use, not a stepping stone to a real CAD model.

18.4 Partial escapes

- `BlinkingSun/stl2step`. STUDY. A C++ command-line tool, built on OpenCASCADE, that converts an STL mesh to a STEP file.

- **Spectral Labs SGS-1.** WATCH **, benchmark against it.** Announced as the first generative model for structured CAD: an image or mesh goes in, an editable STEP file comes out, described as automating reverse engineering with no human input required. Reported to be weak on organic shapes, thin structures and fine resolution. Whether its weights are actually open is unconfirmed; a Hugging Face demo space exists, which is not the same thing as open weights. This is the most direct threat to SuperHuymn's geometry-side thesis found anywhere in this research.
- **DTGBrepGen and BrepForge.** STUDY **.** Research models that generate a boundary representation directly, keeping a shape's connectivity separate from its exact geometry. This is the research direction that would eventually make mesh-to-B-rep conversion unnecessary, rather than merely improving it.
- **`SadilKhan/NURBGen` , 32 verified (introduced in Lesson 14).** STUDY **.** Published at AAAI 2026. Generates text-to-CAD output directly as NURBS surfaces, a mathematical curve and surface representation CAD kernels already understand, rather than generating a mesh and converting it afterward.

KEY POINTS

- TRELLIS is already in the stack; Hunyuan3D-2 is the strongest open alternative and TRELLIS 2 the current open quality leader.
- TripoSG and TripoSR are fast enough for a thumbnail preview but not accurate enough for engineering work.
- Converting an exact B-rep shape to a mesh is trivial; converting a mesh back to an exact B-rep shape is not, and any pipeline ending in a mesh is a dead end for engineering use.
- Spectral Labs SGS-1 is the most direct known threat to the geometry side of SuperHuymn's plan, though whether its weights are actually open is unconfirmed.
- DTGBrepGen, BrepForge and NURBGen represent the research path that generates the right representation directly, instead of converting into it after the fact.

WHAT SUPERHUYMN DOES WITH THIS

Never build a pipeline where a generative mesh model like TRELLIS is expected to produce an engineering-grade CAD part on its own; treat its output as a visual reference or a starting point for a human-or-agent-authored build123d model, not as the deliverable. Separately, once SGS-1 is reachable for testing, run the same input through both it and Forge Studio's own pipeline and compare the STEP output directly, since that comparison will show exactly where SuperHuymn's checked-geometry approach (Lesson 17) beats a generative shortcut, or does not.

Lesson 19. Physics engines and simulation

WHAT YOU WILL LEARN

- The standard path from a CAD model to a working physics simulation, and the one real gap in that path build123d has not filled.
- Which physics and robotics tools are GPU-mandatory, and why that list matters for a team that deploys on CPU-only servers.
- The wider landscape around the current stack: robot foundation models, the ROS2 motion stack, grasping, digital twins and factory-floor connectivity.

19.1 The CAD-to-simulation bridge

The standard path from a CAD model to a working robot simulation runs like this. Model the part in CAD. Export it as a URDF file (defined in Lesson 14) using a tool such as `sw2urdf` or `Rhoban/onshape-to-robot`. Validate the file with `check_urdf` or `urdfpy`. Convert it to MJCF, the XML format MuJoCo reads, using `mujoco.MjModel.from_xml_path()`. Then hand-tune contacts, actuators, sensors and starting poses. `Rhoban/onshape-to-robot` works because Onshape assemblies carry named mates: a robot's degrees of freedom are mate connectors named things like `dof_something` inside the Onshape file itself.

A correction is due here, including to an assumption made earlier in this same research. A GitHub search for "build123d urdf" returns exactly two repositories: `baibai2013/build123d-cad` (2 stars, the skill described in Lesson 14, which does export URDF with a per-link textured GLB, metric units and movable joints) and `ShubhamGawande191/roboparts` (0 stars, parametric robot mounting parts). So a path from build123d to URDF does exist, but only as a two-star personal skill, not as maintained infrastructure anyone else can depend on.

That is still a real gap, just a more specific one than "nobody has done it." Nobody has done it as a properly tested, production-grade tool. A `build123d` to URDF-or-MJCF exporter that reads joint limits straight from the model, calculates mass and inertia using trimesh, and ships with a real test suite would be a genuine contribution. It would also unblock SuperHuymn's own CAD-to-physics pipeline, and it is publishable as open educational material. One format caveat is worth stating plainly: URDF, MJCF and USD (a newer format used by NVIDIA's tools) each capture different information about a robot, and converting between them always loses something.

19.2 GPU-mandatory, the CPU-first flag

SuperHuymn deploys CPU-first: the target is a demo running on ordinary cloud servers, not a dedicated GPU box. Several tools in this lesson need a CUDA GPU and have no real CPU path, so they are flagged explicitly here rather than buried in a table.

- **NVIDIA Isaac Lab** requires Isaac Sim, Omniverse and an RTX or CUDA GPU, for both training a policy and running its own simulator.
- **cuRobo** is CUDA-only by design; its own name is short for "CUDA Accelerated Robot Library," and it has no CPU path at all.
- **Contact-GraspNet** is a point-cloud deep-learning model whose inference realistically needs a GPU to run at usable speed.

- **NVIDIA Isaac GR00T N1.7** has open weights that could technically run on a CPU, but every published benchmark and its reference hardware, the Jetson Thor board, assume GPU-class compute, so treat it as GPU-bound for now.
- **Genesis** has a CPU backend, but the whole reason to pick it over MuJoCo, its very high parallel-simulation speed, depends on GPU acceleration through its Taichi backend.
- **NVIDIA Omniverse and its "Mega" facility-twin blueprint** are GPU-heavy and carry proprietary licensing on top.

By contrast, MuJoCo itself, MoveIt2, ros2_control and open62541, all covered later in this lesson, run on CPU-only hardware without any of these caveats.

19.3 Simulators

ITEM	STARS / VERIFICATION	LICENSE	TAG	GPU NOTE	WHAT SUPERHUYMN DOES WITH IT
MuJoCo (already in house stack)	15.0k stars, 1.7k forks (verified)	Apache-2.0	ADOPT	Core MuJoCo runs single-core CPU; the `JAX/GPU` variant, MJX, is optional	The engine SuperHuymn already uses is confirmed CPU-native, so nothing about the CPU-first constraint forces a migration away from it
MuJoCo Warp	(not separately verified)	(DeepMind)	ADOPT when scale demands	GPU	DeepMind's GPU reimplementation of MuJoCo, compatible with existing models, for real parallel simulation once one machine's CPU is not enough
MuJoCo Menagerie	(not separately verified)	(DeepMind)	ADOPT	CPU (data, not compute)	Curated, tuned MJCF models of real arms and end-effectors; when the assembly-line arm goal becomes concrete, start from a model here rather than modelling a Franka or a UR arm from scratch
NVIDIA Newton 1.0	(not separately verified)	open source	WATCH seriously	GPU	Open source, GPU-accelerated, differentiable physics built jointly by NVIDIA, Google DeepMind and Disney Research, managed under the Linux Foundation, running on Warp and OpenUSD with MuJoCo Warp as one backend; it is becoming the default physics layer under Isaac Lab and MuJoCo Playground
Genesis (genesis-world)	29.9k stars, 2.9k forks (verified)	Apache-2.0	WATCH	GPU strongly recommended (Taichi backend); has a CPU backend but loses the headline speed advantage	Claims 10 to 80 times the speed of Isaac Sim and MuJoCo on equivalent workloads; watch for its Python-native API and rapid growth, but its performance case depends on a training GPU SuperHuymn does not currently budget for
NVIDIA Isaac Lab	8.1k stars, 3.9k forks (verified)	BSD-3-Clause / Apache-2.0	WATCH	GPU-mandatory	Industry-standard for massively parallel reinforcement-learning training with domain randomisation, reporting 85,000 to 95,000 frames per second on an RTX 4090 for a locomotion policy; it can be the training-time tool of record but cannot be the deploy-time runtime under a CPU-first constraint

ITEM	STARS / VERIFICATION	LICENSE	TAG	GPU NOTE	WHAT SUPERHUYMN DOES WITH IT
PyBullet	(search-reported)	zlib	STUDY	CPU, single-core	Lightweight, pure-Python physics engine, the reinforcement-learning research default before MuJoCo went free; only worth a look as a simpler fallback if MuJoCo is ever unsuitable, no reason to switch today
Webots (Cyberbotics)	(search-reported)	Apache-2.0	WATCH	CPU, multi-threaded	A full robot simulator with built-in industrial-arm models and a friendlier interface than MuJoCo for non-research demos; worth watching as a possible demo visualisation layer, separate from the reinforcement-learning engine choice
Automatic Domain Randomization (ADR), a technique not a single repository	(n/a)	n/a	STUDY	technique, not GPU-specific by itself	A training curriculum that raises the difficulty of randomised conditions as a policy improves; implement the pattern directly against MuJoCo's own randomisation hooks rather than adopting Isaac Lab purely for this
stellarmesh, 82 verified	(verified)	(open)	STUDY	CPU	Meshing for nuclear engineering workflows using gmsh, MOAB and DAGMC on top of build123d; proof that build123d output can already feed a serious downstream solver, and a working reference for the CAD-to-mesh-to-solver path

19.4 Robot foundation models and vision-language-action

A vision-language-action model, or VLA, is a machine-learning model that takes a camera image and a plain-language instruction and outputs a robot's next physical movement.

MODEL	STARS / VERIFICATION	LICENSE	TAG	WHAT SUPERHUYMN DOES WITH IT
OpenVLA	7.0k stars (verified)	MIT	STUDY	A 7-billion-parameter VLA built on Llama2 and `DINOv2/SigLIP` vision models, fine-tunable on around 100 demonstrations; too large to run on an arm's onboard CPU, so study its data format and fine-tuning recipe for a future GPU-side service rather than the CPU control loop
π0 / π0.5 (Physical Intelligence, "openpi")	13.7k stars (verified)	Apache-2.0	STUDY	A flow-matching VLA with reported best-in-class dexterity; read its architecture for how it produces smooth continuous motion instead of discrete language-token actions, relevant only once Forge Studio needs closed-loop arm control
NVIDIA Isaac GR00T N1.7	8.0k stars (verified)	Apache-2.0	WATCH	An open humanoid-reasoning VLA tied to the Jetson Thor board and the wider Isaac product family; humanoid-first and GPU-mandatory in practice (see 19.2); watch for a future arm-only distillation rather than adopting it for a CPU-deployed arm today
Octo	1.8k stars (verified), no commits in roughly 2 years	MIT	AVOID	A transformer policy trained on 800,000 trajectories; effectively superseded by OpenVLA, π0 and GR00T, and no longer actively maintained; keep it only as a historical reference for the general-purpose-policy pattern

MODEL	STARS / VERIFICATION	LICENSE	TAG	WHAT SUPERHUYMN DOES WITH IT
RT-2 (Google DeepMind)	search-reported: 55B parameters, `cloud/TPU`, 1 to 5 Hz inference	not open-weight	AVOID	Weights were never released, and it needs a cloud TPU cluster; cite it as the model that proved the VLA idea works, not as a candidate for any SuperHuymn deployment
SmolVLA (Hugging Face)	ships inside the 27.3k-star LeRobot repository (verified)	Apache-2.0	ADOPT	Explicitly built small enough to fine-tune and run on consumer-grade hardware rather than a datacenter GPU; the one VLA on this list actually worth a real CPU or small-GPU feasibility test on an assembly-line pick task
RDT-1B (RoboticsDiffusionTransformer)	search-reported: roughly 1B parameters	open weights, Apache-2.0	STUDY	A diffusion-based bimanual manipulation policy, an alternative to autoregressive VLAs; worth reading for the tradeoff between diffusion, flow-matching and autoregressive action generation, though not sized for a CPU deployment

19.5 ROS2, MoveIt2 and the motion stack

ROS2, the Robot Operating System, is a widely used software framework for commanding real robot hardware, distinct from MuJoCo and Pinocchio, which SuperHuymn already uses for simulation and kinematics calculation.

ITEM	STARS / VERIFICATION	LICENSE	TAG	WHAT SUPERHUYMN DOES WITH IT
ROS 2 (Jazzy Jalisco LTS / Rolling)	spread across many related repositories, not a single popularity number (search-reported)	Apache-2.0	STUDY	Jazzy is a long-term-support release through 2029; if Forge Studio ever needs to command a physical arm rather than only simulate one, ROS2 is the standard integration layer between planning and real hardware drivers, though the current stack of MuJoCo and Pinocchio can run entirely without it
MoveIt2	2.0k stars, 788 forks (verified)	BSD-3-Clause	ADOPT	The standard motion-planning framework: inverse kinematics, collision-aware planning, grasp generation and trajectory execution; adopt at the point SuperHuymn moves from simulating a pick to planning a collision-free pick on a real assembly-line arm
ros2_control	search-reported	Apache-2.0	ADOPT	The hardware abstraction layer both MoveIt2 and Nav2 sit on; adopt alongside MoveIt2, since it is what turns a URDF file plus a driver into a controllable ROS2 arm with no extra glue code
MoveIt Task Constructor	search-reported	BSD-3-Clause	STUDY	Declaratively composes a multi-stage pick-and-place task, such as approach, grasp, retreat; study it for structuring an assembly-line pick-place-verify sequence instead of hand-coding every stage
Nav2	4.7k stars, 1.9k forks (verified)	Apache-2.0	WATCH	ROS2's navigation stack for mobile robot bases; irrelevant to a fixed assembly-line arm today, worth watching only if the roadmap ever adds a mobile manipulator
Universal Robots ROS2 Driver	search-reported	Apache-2.0	STUDY	An industrial-grade ROS2 driver for UR cobots, co-developed with FZI, covering emergency stop, safeguard stop and speed scaling; the concrete reference for what a real ROS2-to-industrial-arm driver must handle if SuperHuymn ever integrates a UR-class cobot

19.6 Grasping and manipulation

ITEM	STARS / VERIFICATION	LICENSE	TAG	GPU NOTE	WHAT SUPERHUYMN DOES WITH IT
MoveIt Grasps / MoveIt deep-grasp integration	search-reported	BSD-3-Clause	ADOPT	CPU-friendly (grasp generation itself, not necessarily the underlying model)	Plugs any grasp-pose-generation algorithm into MoveIt's pick pipeline; adopt as the integration point regardless of which grasp model is chosen
GPD (Grasp Pose Detection)	768 stars, 250 forks (verified), last commit around 5 years ago	BSD-2-Clause	AVOID	CPU-capable	A classic 6-degree-of-freedom grasp-pose detector from a point cloud; unmaintained for years, cite it only as a historical baseline
Dex-Net 2.0 / 4.0 (Berkeley AUTOLAB)	search-reported	research code, mixed licence	STUDY	CPU-capable for inference	A grasp-quality-from-depth-image approach trained on over 2.5 million synthetic grasps; study its method for generating synthetic training grasps to train a lightweight in-house model sized for CPU inference
Contact-GraspNet (NVIDIA)	search-reported	research code	WATCH	GPU-mandatory	State-of-the-art 6-degree-of-freedom grasp prediction from a single point cloud; flagged in 19.2 as unusable on the CPU-first path as it stands
cuRobo (NVIDIA)	1.8k stars, 341 forks (verified)	Apache-2.0	WATCH	GPU-mandatory	Extremely fast GPU-parallel motion planning and collision checking; flagged explicitly in 19.2 as unusable on the CPU-first path, revisit only if a specific bottleneck forces a GPU into the deploy target

19.7 Digital twins

A digital twin is a live software model of a real machine, kept in sync with the machine's actual current state, rather than a static simulation of a hypothetical one.

ITEM	STARS / VERIFICATION	LICENSE	TAG	WHAT SUPERHUYMN DOES WITH IT
Eclipse Ditto	921 stars, 288 forks (verified)	EPL-2.0	STUDY	IoT digital-twin middleware providing shadow state plus event and command APIs over microservices; a candidate backend for an assembly-line "digital shadow" of an arm or PLC's state, if a persistent twin layer beyond the simulator itself is ever needed
NVIDIA Omniverse / "Mega" Blueprint	search-reported: ABB, FANUC, KUKA and Yaskawa cited as users of Omniverse-based facility digital twins	proprietary NVIDIA licensing	WATCH	Photorealistic multi-robot facility twins; the incumbent-scale benchmark for what "digital twin" means to actual industrial customers, but GPU-heavy and licensed, flagged in 19.2 as not a fit for a bootstrapped CPU-first stack
Unity / Unreal-based twins	not verifiable as one artifact, typically bespoke per project	n/a	AVOID	Common in industry sales pitches but not a single concrete open-source thing to adopt; note it as "how incumbents visualise a twin," not as something to build toward

19.8 Industrial connectivity: OPC UA and MQTT

OPC UA is a standard for describing and exchanging information about factory machines that different vendors' software can all understand. MQTT is a lightweight publish-and-subscribe message protocol used to move small pieces of data quickly between machines and dashboards.

ITEM	STARS / VERIFICATION	LICENSE	TAG	WHAT SUPERHUYMN DOES WITH IT
open62541	3.2k stars, 1.5k forks (verified)	MPL-2.0	ADOPT	A C implementation of the OPC UA standard (IEC 62541); the lightweight, CPU-only, permissively licensed choice for exposing an assembly-line arm or cell's state and commands to PLCs, SCADA and MES systems the way real factories already expect
Eclipse Mosquitto	search-reported (long-standing, widely deployed)	EPL-2.0 / EDL-1.0	ADOPT	The de facto lightweight MQTT broker; adopt as the publish-and-subscribe transport between an arm's control process and any dashboard or cloud-demo layer, trivial to run alongside a CPU-only deployment
OPC UA PubSub over MQTT	search-reported	`n/a` (protocol pattern)	STUDY	OPC UA carries the information model, MQTT carries the transport; study this pairing directly rather than inventing a custom telemetry schema for the assembly-line demo
node-opcua	search-reported	MIT	WATCH	A full OPC UA stack in Node.js and TypeScript; relevant only if the frontend (built in Node) ever needs to speak OPC UA directly instead of proxying through the Python backend

19.9 Research worth reading

RoboMoRe (an LLM co-designs a robot's body shape and its reward function, emitting complete MuJoCo XML), Debate2Create (multiple LLM agents debate to produce a body shape as XML, evaluated on five MuJoCo locomotion tasks through the Brax simulator), RAIL (LLM modules that emit URDF directly), Articraft (agentic articulated 3D assets imported into Isaac Sim with mass and damping values assigned by an LLM), SR-Platform (natural language turned directly into MJCF scenes), URDF-Anything+ (a model that generates articulated 3D objects for physical simulation), and CHIA (an open framework for agentic hardware-and-software co-design). Together these papers show that LLM-authored robot descriptions are already an active, publishable research field, and SuperHuymn is standing right next to it while holding a working CAD kernel.

KEY POINTS

- The path from CAD to a working simulation exists but has a real gap: a properly tested `build123d-to-URDF/MJCF` exporter does not exist yet, only a two-star prototype.
- Isaac Lab, cuRobo, Contact-GraspNet, Isaac GR00T N1.7, Genesis's speed advantage, and NVIDIA Omniverse all require a GPU that a CPU-first deployment does not budget for.
- MuJoCo itself, MoveIt2, ros2_control and open62541 all run cleanly on CPU-only hardware, so the CPU-first constraint does not force a migration away from anything already adopted.
- SmolVLA is the one vision-language-action model on the list small enough to be worth a real feasibility test without a datacenter GPU.
- LLM-authored robot descriptions (RoboMoRe, Debate2Create, RAIL and others) are an active research field SuperHuymn is positioned to draw on directly.

WHAT SUPERHUYMN DOES WITH THIS

Write the production-grade `build123d-to-URDF/MJCF` exporter described in 19.1: joint limits read directly from the model, mass and inertia calculated with trimesh, and a real test suite. Only two-star prototypes exist today, so this both unblocks SuperHuymn's own CAD-to-physics pipeline and counts as a genuine open-source contribution. Everywhere else in this lesson, treat the GPU-mandatory flags in 19.2 as a hard filter: nothing on that list belongs in the CPU-first deploy target described in Lesson 22.

Lesson 20. Electronics and firmware

> WHAT YOU WILL LEARN
>
> - Which text-based circuit-design languages exist beyond SKiDL, and why the electronics field is more mature than expected.
> - Why a generated circuit that has never been simulated is only a guess, not a design.
> - What firmware work an agent can actually be trusted with today, and what it cannot.

20.1 Circuit design languages

The team already runs SKiDL, a Python library that writes KiCad netlists, into KiCad, the open schematic and PCB design program. The neighbouring tools are more mature than expected.

- **atopile.** ADOPT **or study hard.** MIT licensed, from a YC W24 team with backgrounds at Tesla, DJI and Lilium. A declarative language for describing circuits, meaning you state what the circuit should be rather than the steps to build it, and it compiles down to a KiCad project. Because it is plain text, an agent can write it, git can version it, and the compiler's errors are something an agent can act on directly.
- **tscircuit.** STUDY**, and note who already chose it.** MIT licensed, describing circuits as React components in TypeScript, built explicitly to be handed to Claude Code or Codex, with its own command-line tool, syntax and fabrication context. baibai2013's skill set (Lesson 14) picked tscircuit over SKiDL and drove it end to end: writing the component code, running design-rule and manufacturability checks against JLCPCB's process rules, previewing in a browser, then producing Gerber files, a bill of materials, a component placement list, STEP and GLB output, with real ordering held behind a confirmation step. That is a working reference for the whole electronics side of the pipeline.
- `Shaurya-Sethi/circuitron`. ADOPT **as a reference.** Agentic PCB design from natural language, using retrieval-augmented generation with a dedicated MCP server that surfaces SKiDL documentation and examples at every step of the design. The closest public equivalent to SuperHuymn's own electronics work, built on the same SKiDL choice.
- **PCBSchemaGen (2026).** STUDY. Reward-guided generation of SKiDL programs that compile to KiCad schematics, netlists and full projects, with structured verification built in. This validates the choice to use SKiDL and offers a verification design worth copying.

20.2 Simulate before you trust it

ngspice. ADOPT. SPICE is a decades-old standard for simulating how an electronic circuit actually behaves before it is built, and ngspice is a maintained open implementation of it. A generated circuit that has never been run through a simulator is a guess, not a design.

20.3 Firmware

- **PlatformIO MCP server (v2.2.1, May 2026).** ADOPT. Exposes the full PlatformIO embedded-development workflow as MCP tools.
- **Renode, already in use, correctly.** Running Zephyr, an open embedded operating system, through PlatformIO and simulating it in Renode is a documented, working path.

- **Design the firmware agent around configuration, not clever code.** Practitioner reporting puts roughly 75 percent of real embedded firmware work in Kconfig (a configuration system that turns features on and off at build time) and devicetree overlays (files describing what hardware is actually connected) and other build-system files, which is exactly the kind of structured, rule-based work an agent handles well. Writing clever, hand-optimised C is the wrong thing to aim an agent at.
- **"Securing LLM-Generated Embedded Firmware through AI Agent-Driven Validation and Patching."** STUDY **.** Generated firmware needs a security check before it ever touches real hardware.

KEY POINTS

- SKiDL is not an outlier choice; atopile, tscircuit, circuitron and PCBSchemaGen all confirm text-based, agent-writable circuit design is a mature, active pattern.
- baibai2013's skill set is a working, end-to-end reference for driving a circuit language all the way from natural language to a real manufacturing order.
- A circuit that has not been run through ngspice or an equivalent simulator is only a guess about how it will behave.
- Roughly three quarters of real firmware work is configuration (Kconfig, devicetree), which is the part an agent should be aimed at, not hand-written C.
- Any LLM-generated firmware needs a security check before it is loaded onto real hardware.

WHAT SUPERHUYMN DOES WITH THIS

Keep SKiDL as the house electronics language, since `Shaurya-Sethi/circuitron` already validates the same choice with a working MCP-server pattern worth copying directly. Point the firmware agent at Kconfig and devicetree configuration work first, where roughly three quarters of the real effort already lives, rather than at hand-written control code, and never let generated firmware reach real hardware without both an ngspice-equivalent simulation pass and the security validation step named above.

Lesson 21. Analysis, optimisation and manufacturing tools

WHAT YOU WILL LEARN

- The open finite-element and topology-optimisation tools available once a part needs structural or thermal analysis.
- Why manufacturability checking is the step almost every project skips, and how cheap it is to add.
- The specific list of design-for-manufacturability rules worth encoding as automatic checks.

21.1 FEA and topology optimisation

Finite element analysis, or FEA, means breaking a shape into many small pieces and calculating how it bends, heats or fails under a load, rather than guessing from the whole shape at once. Topology optimisation means letting software remove material from a part automatically while keeping it strong enough for its job, to save weight or material.

- **CalculiX.** ADOPT. An open FEA solver, the usual backend for open topology optimisation, and what FreeCAD's own topology optimisation workbench is built on. Both cad-cae-copilot and anvilate (Lesson 14) chose it.
- **Code_Aster (EDF).** STUDY. Broader multiphysics analysis: thermal, fatigue, and nonlinear behaviour.
- **FEniCSx and FEniTop.** STUDY. Research-grade topology optimisation with parallel computing support.
- **TopOpt (pytopopt).** ADOPT **for a first pass.** Solves minimum-compliance and similar problems using the Method of Moving Asymptotes, a standard optimisation technique.
- **gmsh.** ADOPT. The meshing layer both anvilate and stellarmesh (Lesson 19) rely on.
- **STORX (2026).** STUDY. Object-oriented shape and topology optimisation, written in MATLAB.

21.2 Manufacturability, the part everyone skips

DFM, design for manufacturability, was defined in Lesson 14 as checking a shape can actually be cut, printed or milled, not only that it looks correct.

- **Kiri:Moto.** ADOPT. MIT licensed, browser-based, runs entirely locally, keeps data inside the browser, slices and nests 2D parts for laser-cut assemblies.
- **PyCAM.** STUDY. Open computer-aided manufacturing software for 3-axis milling toolpaths.
- **BenDFM (2026).** STUDY. A taxonomy and synthetic CAD dataset for manufacturability specifically in sheet-metal bending, alongside separate 2026 work using machine learning to estimate bending effort.
- **Encode the standard DFM rules as validators.** ADOPT **, half a day of work.** Hole depth-to-diameter ratio, features a milling cutter physically cannot reach, sharp internal corners, deep pockets and slots, flat-bottomed holes, and fillets on outside edges.

That last item is the difference between "the AI made a shape" and "the AI made a part someone can actually machine." For an industrial buyer that is the entire difference, and it costs almost nothing to add.

KEY POINTS

- CalculiX is the shared open FEA backend behind both FreeCAD's topology optimisation workbench and the two most promising harness projects from Lesson 14.
- gmsh is the meshing layer feeding both a manufacturability check and a nuclear-grade analysis pipeline (stellarmesh, Lesson 19), proof build123d output is already good enough for serious downstream work.
- Manufacturability checking, not analysis, is the step almost every agentic CAD project skips.
- A short list of standard DFM rules (hole depth-to-diameter ratio, unreachable features, sharp internal corners, deep pockets, flat-bottomed holes, outside-edge fillets) can be encoded as automatic checks in about half a day.
- That half-day of DFM validator work is the single cheapest way to turn a generated shape into something an industrial buyer can actually use.

WHAT SUPERHUYMN DOES WITH THIS

Encode the standard DFM rule list above as automatic validators and add them to the verification gate described in Lesson 17, since this is named directly as half a day of work with an industrial-scale payoff. This single addition is what turns "the AI made a shape" into "the AI made a part you can machine," which is the actual gap between a demo and something a factory can use.

Lesson 22. Viewers, frontend and deployment

> WHAT YOU WILL LEARN
>
> - Which browser-based CAD viewers already exist, and which pieces of Forge Studio's own frontend claim they build on.
> - The difference between WebGL and WebGPU rendering, and why it might already be a real differentiator for the team.
> - The specific deployment traps (headless rendering, Python version pinning) most likely to break a cloud demo.

22.1 Viewers and web delivery

- **occt-import-js, 287 stars (verified).** ADOPT. OpenCASCADE compiled to WebAssembly (defined in Lesson 16), purpose-built to read STEP, IGES and BREP files directly inside a browser and hand a rendering library a tessellated mesh (a mesh converted from the exact geometry), with full boundary-representation parsing, assembly hierarchy, and per-face colours. This is the direct plumbing between Forge Studio's STEP export and its browser viewer, without a round trip to a server just to preview a file.
- `yeicor-3d/yet-another-cad-viewer`, **138 verified.** ADOPT. Displays and edits OCP models directly in a browser, supports static deployment, face and edge inspection, measurement, per-model clipping planes, transparency, and hot reload through its own `yacv-server`.
- `bernhard-42/three-cad-viewer`. ADOPT. A CAD viewer component built on three.js (introduced below in 22.2).
- **vscode-ocp-cad-viewer, jupyter-cadquery, and `jdegenstein/nice123d` (12 verified).** ADOPT **internally.** These make up the developer feedback loop, letting a developer see a model update live while coding it. `nilcons/docker-vscode-ocp-cad-viewer` runs the same viewer standalone inside Docker.
- `Roadinforest/occt-step-viewer-web` and `ubemacapuno/svelte-step`. STUDY as reference implementations.

Searches surfaced WebGL implementations for all of these, not WebGPU ones. WebGL is the older, widely supported browser graphics standard; WebGPU is the newer one, giving lower-level, faster access to the graphics card. Forge Studio's own screenshots describe WebGPU rendering over an OpenCASCADE B-rep kernel (B-rep defined in Lesson 16). If that description is accurate, it is a genuine, currently unclaimed differentiator, and it should be stated plainly on the site rather than left implicit, since none of the projects above have shipped a working WebGPU equivalent yet.

22.2 The browser-rendering picture underneath

- **three.js, 115,233 stars (verified).** ADOPT. Directly matches the brief: the rendering library behind Forge Studio's browser frontend. The largest, most mature option in this entire research pack by a wide margin, two orders of magnitude above every other project's star count.

- **three.js `WebGPURenderer` plus TSL.** STUDY. TSL, Three.js Shading Language, is a JavaScript-native way of writing a shader (a small program that controls how something is drawn on screen) once and having it compile to WGSL for WebGPU and to GLSL for older WebGL, from the same source. Per 2026 GitHub issue activity (search-reported, not an official stability statement), both are under active development and not yet declared stable, so budget for occasional breakage rather than treating the API as frozen.
- **`yeicor/OCP.wasm`, 45 verified.** WATCH. Already introduced in Lesson 16 as build123d compiled to run entirely in-browser. From the frontend side, it confirms the house kernel can run client-side, but at 45 stars and one maintainer it remains a proof of concept, not a dependency.
- **replicad and OpenCascade.js.** STUDY. A different, reportedly more production-tested take on the same idea as OCP.wasm: a full parametric modeller running on OpenCascade.js and WebAssembly, but with its own modelling API rather than build123d's. Worth reading for its patterns around loading the WASM module and managing memory limits, even though its API does not transfer directly.

22.3 Deployment for the first cloud milestone

The stated target is the site and backend deployed with no local machine involved, so the demo can go straight to social media and to funders. A handful of specific things will bite if not planned for.

- **MuJoCo headless in Docker.** Set `MUJOCO_GL=egl`. Headless rendering, meaning rendering with no visible display attached, then works without needing X11 forwarding, and reported performance is comparable to running on a real desktop. This is the most commonly hit blocker in this whole stack.
- **build123d and ocp_vscode are pure Python** and do not pull in graphics libraries like xrender or libgl, so the CAD half of the deployment container is lighter than expected. The real weight comes from cadquery-ocp itself.
- **Pin the Python version.** The W16 engine demo established that build123d 0.11.1 with cadquery-ocp 7.9.3 pins to Python 3.10 on the team's own machine. The same constraint will hit the deployment container. Pin it directly in the Dockerfile rather than discovering it later inside a broken continuous-integration run.
- **build123d-docker** publishes auto-updating container images. Starting from a working reference Dockerfile beats writing one from nothing.
- **OCP.wasm** (22.2) is the fallback plan if server-side CAD proves too heavy for the demo's compute budget.
- **TRELLIS needs a CUDA GPU.** For a launch demo, running CAD and physics on CPU only, with TRELLIS either stubbed out or moved to a GPU-on-demand host, is the sane split, consistent with the GPU-mandatory flags carried through Lesson 19.

KEY POINTS

- occt-import-js and yet-another-cad-viewer already cover the STEP-to-browser pipeline that Forge Studio's own viewer needs.
- Every viewer found in this research runs on WebGL; if Forge Studio's WebGPU rendering claim is accurate, nobody else in this field has shipped the equivalent yet.
- three.js is the correct, overwhelmingly dominant choice of rendering library, and TSL is worth watching but not yet a stable dependency.
- `MUJOCO_GL=egl` and pinning Python to the version cadquery-ocp actually supports are the two deployment details most likely to break a "no local machine" demo.
- TRELLIS's GPU requirement means the sane first deployment runs CAD and physics on CPU only, with TRELLIS stubbed or hosted separately.

WHAT SUPERHUYMN DOES WITH THIS

State the WebGPU-over-OpenCASCADE rendering claim explicitly on the public site if it is accurate, since no other project surveyed here has shipped that combination, which makes it a real, currently-unclaimed differentiator. For the deployment itself, write the Dockerfile with `MUJOCO_GL=egl` set, Python pinned to the version cadquery-ocp actually supports, and TRELLIS either stubbed out or routed to a separate GPU-on-demand host, so the CPU-only core of the demo cannot be blocked by the one GPU-dependent piece.

Lesson 23. Backend and agent infrastructure

WHAT YOU WILL LEARN

- Why running an AI-written CAD script directly on the same machine that serves MCP traffic is a real security risk, and what to run it in instead.
- Which tools make a long-running CAD or simulation job survive a crash instead of silently losing the user's work.
- The specific patterns (schemas, progress reporting, structured errors) that separate an MCP server an agent can actually use from one it fights against.

23.1 Sandboxed code execution for LLM-written Python

Forge Studio's agent writes and runs build123d and OpenCASCADE Python code generated by Gemini. That code must never run directly on the same machine that also serves MCP traffic, because a buggy or malicious script could then reach everything else on that machine. A sandbox is an isolated environment built so that code running inside it cannot affect anything outside it. A microVM is a very small, fast-starting virtual machine used to build that isolation cheaply.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
E2B / `e2b-dev/E2B`, 13,697 stars (verified)	ADOPT	A Firecracker microVM sandbox with a Python and JavaScript SDK; run every agent-generated CAD script here instead of in-process, with a custom template pre-loaded with build123d, OCCT and MuJoCo so cold starts stay short
Modal Sandboxes	ADOPT	A Python-first serverless platform combining gVisor isolation with on-demand GPU access; a fallback for a CAD job that needs GPU access, such as MuJoCo rendering or TRELLIS, since E2B templates are CPU-only by default (search-reported)
Daytona	STUDY	An agent-native sandbox with persistent workspaces (27 to 90 millisecond provisioning, search-reported); worth a look if a session needs to survive across several agent turns of iterative CAD edits, rather than one-shot execution
gVisor / `google/gvisor`	STUDY	A user-space kernel that intercepts system calls; the drop-in Docker runtime (`runsc`) if SuperHuymn ever self-hosts sandboxing instead of buying E2B or Modal, with lower overhead than a full microVM but imperfect system-call coverage (search-reported)
Firecracker (AWS)	WATCH	The microVM technology E2B is built on; relevant only if SuperHuymn ever runs its own sandbox fleet instead of paying a vendor, not an early concern
Kata Containers	WATCH	A Kubernetes-integrated microVM runtime, relevant only if the backend ever moves to self-managed Kubernetes instead of a managed sandbox vendor
RestrictedPython (Zope)	AVOID	Restricts Python at the level of its parsed syntax tree, not a real security boundary against a determined or buggy LLM-written script, since it has no system-call or network isolation; use it at most as a cheap pre-filter in front of a real sandbox, never as the only defence
Riza Code Interpreter API	AVOID	The hosted service shut down on 1 October 2025, and Riza now points users at Daytona instead (search-reported); a dead end, do not build against it
Judge0 / `judge0/judge0`	WATCH	Open-source code-execution-as-a-service built for competitive programming judging; its isolation model is coarser than E2B or gVisor and its interface is judge-oriented rather than agent-oriented, worth reconsidering only if E2B or Modal pricing becomes a real blocker
Northflank sandboxes	WATCH	Another managed sandbox vendor found in comparisons; no differentiator found over E2B or Modal for this use case (search-reported)

23.2 Job queues and workers for long-running CAD and simulation jobs

A build123d extrude-and-fillet pass, a MuJoCo simulation run, or a TRELLIS generation can take seconds to minutes, too long to hold an MCP tool call open synchronously while waiting for the answer. Each of these jobs also needs to survive a worker crashing partway through, without silently losing the user's CAD state. Durable execution means a system checkpoints its progress as it goes, so it can pick back up from the last completed step after a crash, instead of starting over from nothing.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Temporal / `temporalio/temporal`	ADOPT	Durable execution: if a worker dies mid-simulation or mid-build, Temporal reconstructs the job's state and resumes from the last completed step instead of restarting the whole thing, which is the right fit for the "survives a crash" requirement in the brief; has a first-class Python SDK (search-reported)
Hatchet / `hatchet-dev/hatchet`, 7,892 stars (verified)	ADOPT	A lighter-weight durable task queue purpose-built for AI-agent workloads; every step is checkpointed so a retried Gemini call is never duplicated, and Python tasks run async natively, a simpler starting point than Temporal if less operational overhead is wanted
Celery / `celery/celery`	STUDY	The default Python task-queue choice, well understood by most Python teams, but with no built-in checkpoint semantics; fine for "fire a render job, poll until done" but weaker than Temporal or Hatchet for crash recovery mid-job
RQ (Redis Queue)	STUDY	The simplest possible Redis-backed queue, good for a throwaway demo such as queueing a MuJoCo render, but not something to build the hardened backend on, since it has no retry or checkpoint sophistication
Ray / Ray Jobs	STUDY	A distributed compute framework, relevant if SuperHuymn ever needs to run many independent simulation or CAD jobs in parallel across a cluster, such as batch-generating design variants; overkill for a single-agent minimum viable product
Inngest	WATCH	An event-driven durable workflow engine, TypeScript-first with a Python SDK, comparable to Hatchet or Temporal with no clear edge for this Python-heavy stack (search-reported); track as an alternative if Hatchet's Python support disappoints
Prefect	WATCH	A data and machine-learning pipeline orchestrator from the same team as FastMCP; a better fit for a future training or data pipeline than for the agent-tool-call path itself
AWS Step Functions + AWS Batch	WATCH	A cloud-native alternative to Temporal or Hatchet if the team ever standardises on a single cloud provider, adding vendor lock-in the open-source options avoid

23.3 Artifact storage for generated files

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Cloudflare R2	ADOPT	S3-compatible object storage with zero egress fees (search-reported, holding as of July 2026); store every generated STEP, STL and GLB file here and let the frontend viewer fetch meshes directly without SuperHuymn paying per byte as demo traffic grows
MinIO / `minio/minio`	STUDY	A self-hosted, S3-compatible store; only worth adopting over R2 if SuperHuymn needs on-premises or air-gapped storage for a client's proprietary CAD data, not needed for the funder-facing demo
DVC (Data Version Control)	STUDY	Git-friendly versioning for large binary artifacts, such as CAD files, MuJoCo checkpoints and TRELLIS outputs, that keeps the actual files out of git while tracking which model version produced which file; worth adopting once there are more than a handful of generated assets to keep straight, not yet a blocker
Git LFS	AVOID for this use case	Search-reported guidance is to use Git LFS only for small binaries under 100MB, such as model weights, and to use DVC or object storage for larger, evolving datasets; CAD exports and simulation traces fit the latter, so do not default to LFS here
Backblaze B2	WATCH	Cheaper than R2 per gigabyte but charges for data leaving it (search-reported); only worth it for a write-heavy, read-light traffic pattern, unlikely for a CAD asset store serving a live viewer

23.4 MCP server hardening

This is the layer Forge Studio's Gemini agent talks to build123d and MuJoCo through, so any failure here becomes a failure the agent experiences directly.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Model Context Protocol spec	ADOPT	The canonical reference for tool schemas, the progress-reporting mechanism, and the structured-error flag; the MCP server for build123d and MuJoCo tools should be built and reviewed directly against this specification, not against a blog's paraphrase of it
FastMCP / `PrefectHQ/fastmcp`, 27,553 stars (verified), now FastMCP 2.0 under PrefectHQ	ADOPT	A Python framework that generates tool schemas straight from function signatures and type hints, and ships built-in progress reporting, authentication, and server composition; very likely the fastest path to a hardened MCP server for the CAD tools, rather than hand-rolling the underlying protocol messages
MCP Inspector / `modelcontextprotocol/inspector`, 10,838 stars (verified), maintained by the MCP team at Anthropic	ADOPT	Run it against the Forge Studio MCP server before wiring it to Gemini; it isolates and tests each CAD tool call, shows the raw underlying messages, and catches schema mistakes before they ever reach the agent
MCP progress notifications	ADOPT	A build123d extrude-and-fillet chain, or a MuJoCo simulation run, can take tens of seconds; report incremental progress through the client-supplied token rather than leaving Gemini's chat interface silent, directly answering the requirement for streaming progress
Structured tool errors	ADOPT	Design every CAD or simulation tool's failure path to return a structured, readable error, such as "fillet radius exceeds edge length" rather than a raw OCCT stack trace, so Gemini can correct itself on the next turn instead of retrying blindly
asyncio timeouts on every tool handler	ADOPT	Search-reported best practice: wrap every tool's body in a timeout and push CPU-bound OCCT or MuJoCo work onto a separate thread, since a blocking call inside an asynchronous handler is a common cause of an MCP server freezing an entire agent session
FastAPI-MCP template patterns	STUDY	If any Forge Studio endpoints are already built in FastAPI, this pattern auto-exposes them as MCP tools; worth a look only if it actually saves real duplication over writing FastMCP tools directly

23.5 Agent eval and tracing tooling

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Langfuse / `langfuse/langfuse`, 34,300 stars (verified)	ADOPT	Open-source, self-hostable in roughly 5 minutes with Docker Compose per its own documentation, MIT-licensed except one folder; trace every Gemini call and tool call in the CAD agent loop, replay a failed session if a demo misbehaves in front of a funder, and use its evaluation features to regression-test prompt changes against known-good CAD tasks
Arize Phoenix	STUDY	Built on OpenTelemetry's own conventions and framework-agnostic, free to self-host; a legitimate Langfuse alternative, but Arize the company is under a pending $915 million Dynatrace acquisition announced 13 August 2026 (search-reported) with no stated continuity commitment for standalone open-source Phoenix, so do not build a dependency on it until that resolves
OpenTelemetry GenAI semantic conventions + Traceloop OpenLLMetry	STUDY	A vendor-neutral set of tracing conventions for AI systems now folding into core OpenTelemetry; instrumenting the agent loop against this standard, rather than only a vendor SDK, keeps SuperHuymn free to swap Langfuse for another backend later without re-instrumenting everything
LangSmith	AVOID for now	Purpose-built for the LangChain and LangGraph frameworks, and self-hosting is enterprise-only (search-reported); SuperHuymn's stack is Gemini plus a custom MCP server, not LangChain, so the tight coupling buys nothing and the pricing is worse than Langfuse's for a bootstrapped team

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Braintrust, Helicone, PromptLayer	WATCH	Named in comparisons alongside Langfuse and LangSmith; no evidence found in this research that any solves a problem Langfuse does not already cover for this stack, revisit only if Langfuse's evaluation depth proves insufficient

23.6 Structured output and repair libraries

Gemini's tool-call arguments, and any free-text CAD parameter it extracts, both need to arrive as valid, type-checked Python objects before they touch build123d. Malformed output at this step is a direct path to a corrupted or crashing CAD script.

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Instructor / `567-labs/instructor`, 13,835 stars (verified), now under the 567-labs organisation	ADOPT	Wraps Pydantic data models around Gemini's responses with automatic retry on a validation failure; use this as the default for turning Gemini's plain-language CAD intent into typed parameters, such as radius, dimensions or material, before they reach any build123d call, search-reported as "the safest default, works everywhere, covers the 80 percent case"
Outlines / `dottxt-ai/outlines`, 15,759 stars (verified)	STUDY	Constrained decoding that guarantees schema-valid output at the moment of generation, rather than retrying after a failure; worth adopting later for the highest-volume, highest-cost tool calls once traffic justifies the integration work, search-reported as the choice "for high-throughput production pipelines where every failed retry costs real money," not an early problem
Guardrails AI	WATCH	A broader input-and-output validation framework with schema, structural and business-rule layers; more machinery than a four-person team validating CAD parameters currently needs, revisit if validation rules grow to cover things like GD&T tolerance business rules
BAML	WATCH	A contract-first, code-generated approach that search-reported sources say handles "fundamentally broken" LLM output better than Instructor's strict JSON parsing; worth a look if Gemini's raw output turns out messier than Instructor's retry loop can fix, not a day-one need
Pydantic AI	STUDY	Pydantic's own agent framework with native structured output; the option to consider if SuperHuymn ever wants a full agent framework instead of hand-rolled Gemini-and-MCP orchestration, since it shares a type system with Instructor
Marvin (PrefectHQ)	WATCH	A structured-extraction library from the Prefect team, the same organisation behind FastMCP; no clear advantage over Instructor found in this research, track only

23.7 Docs and GitBook tooling

The brief did not say who the documentation is for; this section treats it as developer and API documentation for MCP tool users and internal onboarding, the closest reading of "MCP server" plus "docs tooling."

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Docusaurus	ADOPT	Free, self-hosted, React-based, with no ongoing licence cost (search-reported); the right choice for a bootstrapped team's internal engineering docs and any public MCP-tool reference, since the team already has React experience in-house
Mintlify	STUDY	AI-ready hosted documentation with OpenAPI and AsyncAPI support and a built-in MCP server for the docs themselves (search-reported); worth it once Forge Studio has an external developer audience for its API and MCP tools and wants a polished surface without maintaining the site, at the cost of a recurring fee Docusaurus does not carry
GitBook	AVOID for `API/MCP` reference	Search-reported as better suited to non-technical collaborative writing than to fast-loading, large technical documentation sets, with reports of slow page loads and sluggish navigation on large sets; the wrong tool for CAD-tool API reference documentation specifically

RESOURCE	TAG	WHAT SUPERHUYMN DOES WITH IT
ReadMe.com	WATCH	An API-documentation specialist, relevant only if Forge Studio ever ships a public REST API alongside MCP, with no evidence yet that it does

KEY POINTS

- LLM-written CAD code must run inside a real sandbox such as E2B or Modal, never directly on the MCP-serving machine, and never protected by RestrictedPython alone.
- Temporal or Hatchet give a long-running CAD or simulation job durable execution, so a worker crash does not silently lose the user's state.
- FastMCP plus the MCP Inspector, progress notifications, structured errors and asyncio timeouts together are what separate a hardened MCP server from one that hangs or gives an agent an error it cannot act on.
- Langfuse gives the team session replay and prompt regression testing at no cost, which matters directly when a demo misbehaves in front of a funder.
- Instructor should be the default way Gemini's free-text CAD intent becomes typed, validated parameters before it ever reaches build123d.

WHAT SUPERHUYMN DOES WITH THIS

Build the MCP server against FastMCP, reviewed directly against the Model Context Protocol specification, with progress notifications, structured tool errors and asyncio timeouts on every handler from the start, since these four items together are what make the difference between an agent that can recover from a mistake and one that silently hangs. Run every agent-generated build123d script inside an E2B sandbox rather than in-process, and instrument the whole agent loop with Langfuse from day one so a bad run in front of a funder can be replayed and understood afterward, not just apologised for.

Dropped for lack of a source

Nothing in the source packs used for this part (the v1 document's Part II, `02-STRATEGIC-ASSESSMENT.md` section 1, and packs R-1, R-2 and R-4) asserted a fact with no attached source and no hedge. Every number either carries a **(verified)** marker, a **search-reported** marker, an approximate `~` figure left exactly as written, or is explicitly attributed to a named report, blog, forum comment or practitioner claim, and each of those attributions was carried through unchanged rather than upgraded to a firmer claim. No content was removed from this part for lacking a source.

Two sections of `packs/R-2-robotics.md` were deliberately left out of Lesson 19 because the ticket's lesson mapping did not name them: section 2, LeRobot and open-source arm hardware, and section 8, robot-learning datasets. This is a scope decision, not a drop for lack of a source; both sections are sourced the same way as the rest of R-2 and may belong in a later part covering open hardware and benchmarks.

PART IV. Competitors and what they teach

Part III listed what exists. This part is different. It asks how these systems actually work, where each one breaks, and what SuperHuymn should copy or refuse.

Three lessons: how text-to-CAD works and how the field splits, a teardown of the people building it, and the lessons that follow.

Lesson 24. How text-to-CAD actually works

WHAT YOU WILL LEARN

- The four different technical approaches hiding behind the same phrase "text to CAD".
- The five stages every one of these systems has, whether or not its builders named them.
- Which stage each approach is weak at, so you can judge a competitor from its architecture alone.

24.1 One phrase, four different machines

"Text to CAD" describes at least four unrelated designs. People compare them as if they were one product, which is why public comparisons are usually confused.

Approach 1: the model writes code. An LLM writes Python that calls a CAD library, usually CadQuery or build123d. The code runs, and out comes a solid. This is what SuperHuymn does, and what most of the open projects do.

Strength: the output is a program. It is editable, diffable, versionable in git, and a human can read it. The LLM is doing what LLMs are best at, which is writing code in a language with plenty of public examples.

Weakness: the model has no eyes. It cannot see that its bracket looks wrong. It also has to hold the whole shape in its head as text, and it refers to faces and edges by name, which brings the naming problem from Lesson 3.

Approach 2: the model predicts CAD operations. Instead of Python, the model emits a sequence of modelling steps: sketch a rectangle, extrude 10 mm, fillet these edges. This is the DeepCAD and Text2CAD research lineage, and models such as CAD-Llama and cadrille sit here.

Strength: the output is close to how CAD systems store history, so it can be replayed and edited by feature.

Weakness: it needs a large training set of construction sequences, which barely exists. The Fusion 360 Gallery has 8,625 human-designed objects with their sequences, and DeepCAD has 176,017 models built only from sketch and extrude. Sketch and extrude alone will not build a gearbox.

Approach 3: the model generates B-rep directly. The model outputs exact surfaces and their topology, skipping code entirely. Spectral Labs' SGS-1 works this way, taking an image or a mesh and producing STEP. Research systems DTGBrepGen and BrepForge separate topology from geometry and generate each.

Strength: no code to run, no library to fail, and the output is native engineering geometry.

Weakness: it is the newest approach and the hardest. SGS-1 is reported weak on organic shapes, thin structures and fine resolution. There is also no program left behind, so changing a dimension later means regenerating rather than editing.

Approach 4: a mesh generator with a CAD label. A model such as TRELLIS, Hunyuan3D or Tripo makes a triangle mesh, and the product calls it CAD.

This is not CAD. Lesson 2 explained why: a mesh has no exact surfaces, so nothing in it can be measured or machined to a tolerance. These tools are good at what they are for, which is game assets and visual prototypes. AVOID treating them as an engineering path.

24.2 The five stages every system has

Whatever the approach, the same five stages appear. Naming them lets you find the weak one fast.

1. **Intent.** Turn a sentence into a specification. "A bracket for a NEMA 17 motor" implies a bolt pattern the user never stated.
2. **Representation.** Decide what to emit: code, an operation sequence, or B-rep.
3. **Execution.** Run it. Code can throw. Geometry operations can fail. Fillets fail often.
4. **Verification.** Ask whether the result matches the intent, and whether it is a valid, makeable solid.
5. **Repair.** When a check fails, feed the failure back and try again.

Most public projects are strong at stages 2 and 3, weak at stage 1, and nearly absent at stages 4 and 5. That is the whole finding of this research, stated as a pipeline.

24.3 Why stage 4 is so often missing

Verification is unglamorous and it makes demos look worse before it makes products better. A generator with no checks produces a nice picture every time, because the failures are invisible. Add checks and suddenly a third of your outputs are marked as failures, which is the truth, but it is a harder slide to show a funder.

There is also a skills reason. Writing a generator needs prompt engineering. Writing a verifier needs geometry knowledge: what manifold means, what sewing tolerance is, how to sweep a clearance check through a motion range. Fewer teams have the second skill set.

KEY POINTS

- Four unrelated approaches share one name, so most public comparisons are meaningless.
- Code generation gives you an editable program but no eyes; direct B-rep gives geometry but no program.
- Every system has five stages, and the field is weakest at verification and repair.
- Mesh generators are not a CAD path, no matter how the marketing reads.

WHAT SUPERHUYMN DOES WITH THIS

Forge Studio is approach 1, which is the right choice for now, because the output is a program the team can inspect and a customer can edit. The gap to attack is stages 4 and 5. Write the verification and repair loop, and Forge Studio becomes a different category of product from the projects that stop at stage 3.

Lesson 25. The competitors, pulled apart

WHAT YOU WILL LEARN

- Who is actually building in this field, open source and commercial.
- What each one got right, in a sentence.
- Where each one is weak, which is where a competitor can be beaten.

Star counts marked (verified) were read from GitHub on 2026-09-08.

25.1 The open-source builders

`earthtojake/text-to-cad` **, 14,600 stars (verified).** The centre of gravity of the whole field. Describes itself as a library of agent skills for CAD, CAE and CAM, built on Python, build123d and OpenCASCADE, exporting STEP, STL, 3MF, DXF, GLB and URDF. *Got right:* shipped as skills rather than as an application, so it plugs into whatever agent the user already has. Distribution through the skill registries, not through a website. *Weak:* being a skill library, it inherits whatever verification the host agent brings, which is usually none. *Take:* the packaging model. Skills travel further than apps.

`Adam-CAD/CADAM` **, 5,100 stars (verified).** An open text-to-CAD web application in TypeScript and React, running OpenSCAD compiled to WebAssembly. *Got right:* a funded commercial company gave away a full working app, which buys enormous reach. *Weak:* OpenSCAD is not B-rep. It is constructive solid geometry that meshes for display, so the output is not the exact-surface geometry a machinist wants. That choice is what lets it run in a browser tab. *Take:* the reach strategy. Note the cost they paid in geometry quality, and do not copy that part.

`Pan-Chera/Multi-Agent-CAD` **, 934 stars (verified).** A decoupled multi-agent framework using LangGraph, Qwen and aider, described as text-to-CAD through constrained test-time compute. *Got right:* separating the agents, and spending more compute at generation time instead of hoping a single pass succeeds. Also runs on an open model, so cost per part is controllable. *Weak:* more agents means more places to fail, and multi-agent systems are hard to debug. *Take:* the test-time compute idea. Retrying with structure beats retrying blindly.

`partcad/partcad` **, 493 stars (verified).** A package manager for physical parts, framed around the digital thread and the technical data package. *Got right:* the vocabulary. Digital thread and technical data package are the words industrial buyers and their auditors already use. It also covers manufacturability. *Weak:* it is infrastructure, not a generator, so it does not compete on output quality. *Take:* the language. When SuperHuymn sells to a factory rather than a funder, this is how to describe the product.

`armpro24-blip/cad-cae-copilot` **, 57 stars (verified).** Text-to-CAD and text-to-CAE on real build123d and OpenCASCADE geometry, with editable parameters, stable topology pointers, deterministic critique, MCP tools, CalculiX and topology optimisation. *Got right:* the two hardest problems are named in its own description. Stable topology pointers address the naming problem from Lesson 3. Deterministic critique means the checker does not hallucinate. *Weak:* at 57 stars it has almost no users, so none of this is battle-tested. *Take:* both phrases, and check whether the implementation behind them is real. This is the closest public project to where SuperHuymn should be going.

`clay-good/anvilate`, **7 stars (verified).** A local-first design agent producing physics-validated parametric STEP or DXF plus the editable Python, using CalculiX, gmsh and DFM checks. *Got right:* the output contract. It hands over a file that drops into CATIA, SolidWorks, NX or AutoCAD, plus the source. That is exactly what an engineering customer wants and it is rarely offered. *Weak:* tiny, and local-first limits how heavy the analysis can get. *Take:* the output contract, word for word. "Validated STEP plus the source that made it" is a strong promise.

`baibai2013/build123d-cad`, **2 stars (verified).** Twenty-five sub-skills covering CAD modelling, a parts catalogue, a browser viewer, URDF, SRDF and SDF export, slicing and laser quoting, tscircuit PCB with fabrication output and JLCPCB quoting, requirements verification, actuator sizing, FEA, wear and fatigue, PyBullet and MuJoCo scene gates, gait optimisation, motion control, firmware dry-run, sim2real calibration, an integration bring-up gate, and a robot-dog digital twin. *Got right:* nearly everything about the architecture. Capability as files, each skill a document plus deterministic scripts plus tests. Scripts that fail loudly and assume no LLM is present, so they run in plain continuous integration. Skills that never call each other's functions and communicate only through agreed file paths. *Weak:* two stars, and documented mostly in Chinese, so it has no reach. *Take:* the architecture, in full. Someone independently designed SuperHuymn's roadmap and published it.

`BOMWiki/partmode`, **522 stars (verified).** A close public analogue to the agentic-CAD-in-a-browser problem. *Take:* WATCH it. If it grows, it is competing for the same users.

25.2 The commercial players

Zoo, formerly KittyCAD. Text-to-CAD with KCL, its own open programming language for parametric CAD, plus Design Studio, the ML-ephant machine learning API and the Zookeeper generator. Already scored on the public CAD Arena benchmark. *Got right:* building a purpose-designed language instead of using Python, and open-sourcing it. If KCL becomes how people write CAD for machines, Zoo owns the layer everyone builds on. *Weak:* a new language is a large adoption ask. Python has the examples and the training data. *Take:* read the KCL design notes before writing more prompts. Python versus a dedicated language is the biggest unexamined decision in Forge Studio's design.

Spectral Labs, SGS-1. Announced as the first generative model for structured CAD. Image or mesh in, editable B-rep STEP out, automating reverse engineering with no human in the loop. *Got right:* attacking the one-way wall from Lesson 2 directly, which is the highest-value unsolved problem in the field. *Weak:* reported struggles with organic shapes, thin structures and resolution. Whether the weights are open is unconfirmed. *Take:* benchmark against it on the same mesh inputs. Whatever the result, you learn where your moat is.

Adam, and CADAgent Pro. Named alongside Zoo and Spectral as the tools with real traction. *Take:* WATCH.

Leo AI. Does not generate geometry at all. It searches an existing parts vault using natural language and geometry-aware matching. *Got right:* noticing that finding the right existing part is often more valuable than generating a new one, and much easier to get correct. *Take:* a real strategic warning. Some customers do not want generation. Ask before assuming.

Autodesk with Anthropic. On 28 April 2026 Autodesk shipped an official connector letting Claude act inside Fusion, alongside Trimble SketchUp and Blender connectors, plus a free help-content server covering more than 110 products. Autodesk has previewed the same for Revit. The

Open Design Alliance has connectors planned for the third quarter of 2026 covering STEP, IFC and DWG. *Got right:* distribution. Every Fusion seat gets AI without installing anything. *What it means:* a Tier-1 CAD vendor publicly conceded that the AI interface layer will not be built in-house. That is an opening for whoever does the part vendors are not doing, which is generating and proving geometry rather than operating a CAD package. *Take:* stop competing on "AI can drive CAD". That is now a vendor feature.

25.3 The research groups

These are competitors too, because they publish and they set the standards a funder will ask about.

CADCodeVerify. A vision-language model inspects the rendered object, generates a catalogue of questions about it, then answers them to find deviations from the specification. *Take:* implement this. It is the cheapest quality improvement available.

EvoCAD. Evolutionary generation with vision-language models. Mutate and select programs instead of one-shot generation. *Take:* better than blind retry when a part keeps failing.

BenchCAD, Text2CAD-Bench, P3D-Bench, CAD Arena. The measuring instruments. BenchCAD evaluates across the whole prior field. CAD Arena is public. *Take:* run one privately, publish the number when it is good.

AgentsCAD and the naming-problem literature. A 2026 paper documents face identifiers being reallocated between edits, which is exactly the failure in Lesson 3. *Take:* the problem is known and named in the literature. Cite it, and show your mitigation.

> KEY POINTS
>
> - The field's leader is a skill library, not an application, which says something about distribution.
> - The projects with the best ideas have the fewest stars, so good ideas are available cheaply.
> - Commercial players are split between generating geometry, searching for it, and operating CAD tools.
> - Autodesk's connector removed "AI drives CAD" from the list of things worth claiming.

> WHAT SUPERHUYMN DOES WITH THIS
>
> Read three repositories properly: `earthtojake/text-to-cad` for packaging, `baibai2013/build123d-cad` for architecture, and `armpro24-blip/cad-cae-copilot` for stable topology pointers and deterministic critique. Then benchmark against SGS-1 on mesh inputs. Those four actions cost days, not months, and they will tell the team more than another quarter of building will.

Lesson 26. Twelve lessons, and what to do about each

WHAT YOU WILL LEARN

- The pattern behind what the winners and the abandoned projects did.
- Twelve specific lessons, each with a concrete action for SuperHuymn.

26.1 The lessons

1. Ship skills, not only an application. The field's most-starred project is a library of agent skills. Skills install into whatever agent someone already uses, so they travel. An application has to be found, chosen and adopted. *Do:* publish a Forge Studio skill that drives the MCP server. About a day of work. No backend given away.

2. The generator is not the product. Every project can generate. Almost none can prove. Four separate teams arrived independently at the same idea of declared checks, which means the need is real and the slot is open. *Do:* build the verification gate and make it the headline.

3. Name the hard problems in your own description. cad-cae-copilot's description says "stable topology pointers" and "deterministic critique". Those phrases tell a knowledgeable reader that the team understands the domain. Most descriptions say "AI-powered". *Do:* write Forge Studio's description in the vocabulary of the problems it solves.

4. Choose the output contract deliberately. anvilate promises validated STEP or DXF plus the editable Python. That is a contract a mechanical engineer can evaluate in one sentence. *Do:* state Forge Studio's output contract explicitly, and include the proof file in it.

5. Do not confuse a mesh with CAD. CADAM reaches a browser because OpenSCAD compiles to WebAssembly, and pays for it with geometry that is not exact. That trade is fine for hobby parts and fatal for machined ones. *Do:* keep B-rep as the source of truth, and say so as a differentiator.

6. Capability as files beats capability as code. baibai2013's design keeps each skill as a document, deterministic scripts, reference tables and tests, with skills talking only through agreed file paths. That survives changing the agent, the model, or the language. *Do:* adopt the file-interface rule between Forge Studio's stages.

7. Scripts must run without an LLM present. The same project insists scripts fail loudly and assume no model is in the loop, so they work in plain continuous integration. *Do:* make every check runnable from a command line with no API key.

8. Spend compute at generation time, with structure. Multi-Agent-CAD's constrained test-time compute beats a single hopeful pass. EvoCAD's evolutionary selection beats blind retry. *Do:* when a part fails, retry with the failing check attached, not with the same prompt.

9. Distribution beats model quality. Autodesk reaches every Fusion seat by shipping a connector. No bootstrapped lab can match that by being better. *Do:* pick a distribution path that does not depend on people finding a website.

10. Sometimes the customer does not want generation. Leo AI searches an existing vault instead of generating, because for many customers the right part already exists. *Do:* ask early customers whether they want a new part or the right existing one. Do not assume.

11. Speak the buyer's vocabulary. partcad uses digital thread and technical data package. Those words appear in industrial procurement documents. *Do:* keep two vocabularies, one for funders and one for factories, and never mix them up.

12. Publish a number before someone else measures you. CAD Arena is a public benchmark and it already lists Zoo. Benchmarks arrive whether or not you are ready. *Do:* run BenchCAD or Text2CAD-Bench privately, fix what it exposes, then publish.

26.2 The pattern underneath

Read together, these twelve say one thing. The teams making progress are the ones treating CAD generation as an engineering problem with tests, not as a prompting problem with demos.

That is good news for a small team. Prompting talent is abundant and cheap. Geometry knowledge, test discipline and the patience to build a gate are not, and SuperHuymn already has a working example of that discipline in its own W16 engine demo, where a validator gate ran on every export.

KEY POINTS

- The field's best ideas sit in projects with almost no users, so they are cheap to acquire.
- Verification, packaging and vocabulary are where small teams can beat funded ones.
- Distribution is the constraint no amount of model quality removes.
- Benchmarks are coming regardless, so it is better to be measured on your own timing.

WHAT SUPERHUYMN DOES WITH THIS

Pick three of the twelve and do them this month. The cheapest three with the largest effect are publishing a skill, attaching the failing check to every retry, and running a benchmark privately. None needs new research, and together they change how the project is judged.

PART V. Evidence and market

Lesson 27. Benchmarks, datasets and proving it works

WHAT YOU WILL LEARN

- Which benchmarks and datasets exist for testing text-to-CAD systems, and what each one measures.
- Why generating a single part is close to solved, while generating a whole assembly is not.
- The one-week test that turns a demo video into a claim backed by numbers.

27.1 The benchmarks and datasets

A benchmark is a standard test that lets different systems be measured on the same tasks, so results can be compared fairly. A dataset is a collection of example models used to train or test a system. The table below lists what exists for text-to-CAD and programmatic CAD, meaning CAD built as a sequence of code steps rather than mouse clicks.

RESOURCE	WHAT IT GIVES
BenchCAD (2026)	Industry-standard programmatic-CAD benchmark, evaluating across DeepCAD, Fusion 360 Gallery, Text2CAD, CADPrompt, CAD-Coder, CAD-Recode, cadrille and CADEvolve
Text2CAD-Bench (2026)	Benchmark for LLM text-to-parametric-CAD
P3D-Bench (2026)	Executes and renders programs, reports validity, scores on four dimensions. Built from Text2CAD v1.1 and Fusion 360 Gallery, deduplicated and complexity-balanced
CADPrompt (2025)	Expert-written instructions for 200 DeepCAD samples. An out-of-distribution test: examples that look different from what a system trained on, checking whether it truly generalises
CAD Arena (cadarena.dev)	Public open benchmark for AI-generated parametric CAD. Already lists Zoo
DeepCAD	176,017 models from sketch and extrude only
Fusion 360 Gallery	8,625 human-designed objects with construction sequences
Text2CAD	DeepCAD augmented with descriptions at four prompt levels
GenCAD-Code	About 163,000 image-to-CadQuery script pairs, from CAD-Coder

27.2 Where research is heading

Several approaches are worth reading directly. CAD-Recode turns a point cloud, a set of 3D dots measured off a real object, into readable Python code. Cadrille rebuilds CAD from multiple views using reinforcement learning, a training method that rewards good outputs. EvoCAD checks evolutionary generation with a vision-language model, and does better than blindly retrying when a part keeps failing. Zero-to-CAD synthesises readable CAD programs at a million-model scale with no real training data. CADDesigner, Embodied CAD (agents grounded in a solver, working on parametric assemblies of exact surfaces), and Agent-Aided Design for Dynamic CAD Models round out the list.

27.3 Where the wall is

Assembly-level reasoning is where Forge Studio will hit its wall. A single part is close to a solved problem. An assembly, meaning several parts connected by mates, tolerances and moving joints, is not solved, and that is where the last three research approaches above are aimed.

27.4 The concrete move

Take twenty prompts from Text2CAD-Bench or BenchCAD. Run them through Forge Studio. Publish three numbers: the valid-solid rate, meaning the share of runs that produce a real, usable solid; the mean wall-clock time, meaning the real time elapsed per run rather than computer processing time; and the mean number of retries needed. This is about one week of work, and it turns a demo into a claim backed by numbers. CAD Arena is where a public score would eventually live, and that cuts both ways: a good score is marketing, a bad one is permanent.

KEY POINTS

- Single-part generation is close to solved across current benchmarks; whole assemblies are not.
- No published number yet exists for how often Forge Studio's own generation succeeds.
- A one-week test against twenty public benchmark prompts produces three concrete numbers: valid-solid rate, wall-clock time and retries.
- Nine named benchmarks and datasets exist for this work; BenchCAD is the closest to an industry standard as of 2026.

WHAT SUPERHUYMN DOES WITH THIS

Run twenty prompts from Text2CAD-Bench or BenchCAD through Forge Studio and publish the three numbers above. This is a backend task, roughly a week of work, and it is the cheapest way to turn a demo video into a claim a funder can check.

Lesson 28. The commercial field and what the incumbents did

WHAT YOU WILL LEARN

- Who sells AI-CAD today, at what price, and which of them is a real competitor versus an adjacent market.
- What Autodesk, PTC, Siemens and Dassault each did in 2026, and why two of them chose the same protocol Forge Studio already uses.
- Why "AI can drive CAD" stopped being a claim worth making on its own.

28.1 The vendor landscape

MCP, short for Model Context Protocol, is a standard way for an AI system to read data from and take actions inside another piece of software. It is the connection method several vendors below already ship or are adopting, and it is the same one Forge Studio uses.

PLAYER	WHAT IT IS	PRICING (WHERE PUBLIC)	TAG	WHY IT MATTERS
Zoo.dev (Zoo Design Studio, formerly KittyCAD)	Text-to-CAD with the open KCL language, a Design Studio, an ML API and Zookeeper	Free tier: 40 Text-to-CAD credits a month. Pro: `$99/month` (search-reported). API and MCP usage metered separately, about $0.0083 a second (search-reported). Team: `$399/month` per user (search-reported)	STUDY	Closest philosophical competitor: LLM-driven parametric CAD, an open engine, API and MCP first. Already scored on CAD Arena. Study the free-in-app, paid-API split as a pricing template.
Spectral Labs SGS-1	Generative model for structured CAD. Mesh or image in, editable STEP out	Not public	WATCH, benchmark against it	Strongest technical threat to the geometry layer. Claims to cross the mesh-to-B-rep wall (Lesson 18). Whether its weights are open is unconfirmed (Lesson 33).
Adam (Datagrok, product name adam.new)	Text-to-3D-to-parametric-CAD copilot from a Y Combinator W25 startup; also gives away the 5,100-star (verified) open CADAM web app	Standard `$5.99/month`, Pro `$17.99/month` (search-reported). Raised a $4.1M seed round in October 2025 (search-reported)	STUDY	One of three vendors here with real traction, and the only one giving away a full app. The low price suggests the prosumer tier is racing to the bottom; Forge Studio should aim at the industrial tier instead.
Leo AI	Not generative. Natural-language, geometry-aware search over a company's existing part-data vault	Not public	not tagged	Shows the adjacent market: finding a part that already exists often beats generating a new one.
CADAgent Pro	A dedicated text-to-CAD tool	Not public	not tagged	Direct category competitor.
nTop (nTopology)	Implicit-modelling CAD, simulation and 3D-printing platform	Quote-only across pricing sites (search-reported)	WATCH	Enterprise pricing, likely five to six figures per seat a year, is the tier Forge Studio should target once it has customer references, not a near-term comparison.
Neural Concept	3D deep-learning models that predict a part's physics from its shape in seconds instead of the hours or weeks a full simulation takes	Not public; raised $100M for this work (search-reported)	WATCH	A different layer, a simulation shortcut rather than geometry generation. A plausible future add-on to Forge Studio's physics loop, not a competitor.

PLAYER	WHAT IT IS	PRICING (WHERE PUBLIC)	TAG	WHY IT MATTERS
Physna	A geometric search engine: a REST API, a way for one program to ask another for data over the web, that finds duplicate or similar parts in a design library	Quote-only (search-reported)	WATCH	Duplicate-part detection could be a feature built on top of a customer's own CAD library. A small differentiator, not core.
Sloyd.ai	A parametric text- or slider-to-3D generator for game assets	Plus `$15/month`, Pro `$50/month` (search-reported)	WATCH	Adjacent market, game assets rather than engineering CAD. A useful price anchor for cheap AI-3D, not a direct comparison.
Meshy, Tripo, Rodin, Hitem3D, CSM (Common Sense Machines), Hyper3D Rodin	Mesh generators: they output a shell of flat triangles for games, film or 3D printing, not exact machinable surfaces	Rodin Pro `$15/month`, the cheapest of this group (search-reported)	AVOID	Wrong output type: a decorative mesh, not manufacturable parametric geometry with GD&T (geometric dimensioning and tolerancing, the language of allowed variation) or a BOM (bill of materials). Not a comparison for industrial CAD; useful only to rule out in a pitch.
Hadrian	Not a CAD product. An AI-native precision-machining factory operator: software, robotics and real machine shops, for aerospace and defence parts	$1.37B Series D round, about $7.9B valuation, August 2026 (search-reported)	STUDY	Shows where the capital is going: AI plus robotics plus real manufacturing is fundable at billion-dollar scale in the US. Not a competitor or customer, but the strongest available signal of where investor appetite sits, useful in a funding conversation.
L&T Technology Services, AgenticIQ	An end-to-end agentic AI platform for engineering and manufacturing, launched by a large Indian engineering-services firm, August 2026	Not public (search-reported)	STUDY	India's large engineering-services firms are already packaging agentic AI for engineering as a service. Watch these as possible channel partners or acquirers, and as competitors if the plan ever shifts toward services.
HCLTech Agentic AI, TCS "AI-led ER&D"	The same move at India's other two large IT and engineering-services majors	Not public (search-reported)	WATCH	Confirms the trend is sector-wide in Indian engineering services, not one company. Forge Studio's pitch needs to say product, not services wrapper.

What this pricing tells a four-person team: the market has split into two tiers. One is consumer and prosumer text-to-3D, priced $6 to $100 a month with metered add-ons for heavier API use (Adam, Zoo, Sloyd). The other is enterprise engineering platforms, priced by quote only and assumed to run five to six figures per seat a year (nTop, and, as shown below, Onshape and Fusion's paid tiers). Nobody has published a price for agentic CAD sold to an industrial manufacturer. That gap is Forge Studio's opening, and it means the team will likely have to set its own price rather than undercut a visible competitor number.

28.2 What the incumbents just did

On 28 April 2026, a date reported by several outlets though Autodesk's own page does not itself state it, so treat it as unconfirmed, Autodesk and Anthropic announced that Claude connects directly to Autodesk Fusion through MCP. Autodesk's own marketing page, checked directly this session, confirms Claude can perform sketch, extrude and fillet-level operations inside Fusion, run multi-step manufacturing workflows, and support custom agentic workflows. This shipped as part of a wider "Claude for Creative Work" push covering nine creative tools including Blender,

Autodesk and Adobe. Two components exist, reported by third parties and not independently confirmed: a local Fusion MCP server for live-session control, and a cloud-hosted Fusion Data MCP server for querying project data without Fusion running.

The same wave brought a Trimble SketchUp connector and an official Blender connector, plus Autodesk's Product Help MCP Server, giving free read-only access to help content across more than 110 products. Autodesk has previewed MCP support for Revit. The Open Design Alliance plans MCP servers for the third quarter of 2026, covering the STEP, IFC and DWG file formats.

Two more vendors independently landed on the same protocol. PTC's Onshape shipped an AI Advisor and, in July 2026, opened Onshape Labs, an early-access programme adding a FeatureScript MCP Server, automation agents and AI-assisted drawing checks. Siemens NX has a Design Copilot built with Microsoft, but it searches documentation rather than generating geometry, a materially less aggressive move than Autodesk's. Dassault Systemes announced a new Nvidia collaboration and new "AI assistants" at 3DEXPERIENCE World 2026, with fewer public details than the other three.

One industry read of the Autodesk move is worth stating plainly. A Tier-1 CAD vendor publicly conceded that it will not build the AI interface layer in-house, and let a third party plug into it instead. Siemens, Dassault and PTC now face the same choice, and two of them have already answered with MCP too.

28.3 The real price ceiling

Autodesk's own pricing page, checked directly this session, gives real numbers. Base Fusion costs $510 to $680 a year. Fusion for Manufacturing costs $1,530 to $2,040 a year. Fusion for Design costs $1,643 to $2,190 a year. Pricing-aggregator sites report Onshape Standard at about $1,500 per user a year and Professional at about $2,500 per user a year, search-reported and not confirmed on Onshape's own page. Two vendors landing in roughly the same $500 to $2,200-per-seat-per-year band, one of them directly confirmed, gives Forge Studio a real number to argue its own price against, instead of a guess.

28.4 What it means for SuperHuymn

Two things pull in opposite directions. The bad news is that "AI can drive CAD" stopped being a differentiator the day Autodesk shipped it themselves, and incumbents have distribution no bootstrapped lab can match. The good news is larger: the incumbents just declared the interface layer open, which moves value toward whoever does the part they did not, generating and checking geometry rather than merely operating a CAD package. Forge Studio should be positioned against that gap, not against Fusion.

Every major CAD vendor shipped an AI-copilot or agent story in 2026, and two of the four, Autodesk and PTC, have independently converged on MCP as the protocol for agent-to-kernel control. This is the strongest single validation point in this guide for Forge Studio's own architecture choice. It also means "MCP-driven CAD agent" will not defend a position by itself within twelve to twenty-four months; the defence has to be in what Forge Studio does with agentic CAD for physical-AI and robotics-specific workflows that none of the big four are targeting.

Vendor-integrated assistants worth knowing by name: SolidWorks AURA, Onshape AI Advisor, Autodesk Assistant, Siemens NX AI Chat.

KEY POINTS

- Autodesk connected Claude to Fusion through MCP; PTC's Onshape shipped a similar FeatureScript MCP server. Two of four major CAD vendors converged on the protocol Forge Studio already uses.
- "AI can drive CAD" is no longer a differentiator; the incumbents shipped it themselves.
- The interface layer just opened, which pushes value toward whoever generates and checks geometry, not who merely operates the CAD package.
- AI-CAD pricing splits into a $6 to $100-a-month consumer tier and a quote-only enterprise tier around $500 to $2,200 a seat a year; no public price exists yet for agentic CAD sold to an industrial buyer.
- Indian engineering-services majors (LTTS, HCLTech, TCS) already sell agentic AI for engineering as a service, which is a different business from a product.

WHAT SUPERHUYMN DOES WITH THIS

Position Forge Studio against the verification gap the incumbents left open, not against Fusion itself. Use the Autodesk Fusion MCP and Onshape FeatureScript MCP tool schemas as design references when hardening Forge Studio's own MCP server (Lesson 31). Anchor any future price conversation to the $500 to $2,200-per-seat-per-year band Autodesk and Onshape already charge.

Lesson 29. India, funding and go-to-market

WHAT YOU WILL LEARN

- How big India's engineering-services and robotics markets are, and why that context matters even before SuperHuymn sells into either one.
- Which Indian and global grants, credit programmes and incubators a four-person, pre-revenue team can realistically apply to.
- Where agentic CAD gets published and who hires for it, useful for building credibility before a first sale.

29.1 How big the opportunity is

Numbers below are search-reported, useful for scale rather than for quoting as audited fact.

India's ER&D market, engineering research and development services, was $63 billion in the 2026 financial year, projected to exceed $100 billion by 2030, according to NASSCOM, India's software industry association, reported through Business Standard. Growth is attributed to AI moving into engineering workflows, with AI-as-a-Service and Robotics-as-a-Service named as new revenue lines.

India's robotics market was about $480 million in 2026, growing about 34% a year, described as Asia's fastest-growing major robotics market. This figure comes from a single source, RoboticsCenter.ai, and was not checked against a second estimate, so treat it as a rough scale only.

The PLI (Production Linked Incentive) scheme has committed 1.91 lakh crore rupees across 14 sectors. A lakh is 100,000 and a crore is 10 million, so one lakh crore rupees is 10 to the power of 12 rupees, one trillion. As of December 2025, 2.16 lakh crore rupees in approved investment had produced 20.41 lakh crore rupees in sales and 1.439 million jobs. Robotics is one of 27 priority sub-sectors under Make in India 2.0. A robotics-specific PLI tranche is expected in the FY2027 budget but is not yet confirmed. This scheme rewards manufacturers for output and investment. It is not a grant a small software team applies to directly, but it explains why Indian manufacturers are motivated to automate.

A draft National Strategy on Robotics from MeitY, the Ministry of Electronics and Information Technology, aims to make India a robotics leader by 2030, through R&D funding, Centres of Excellence, regulatory sandboxes and "Moonshot Projects" in manufacturing, agriculture, healthcare and security. It is still a draft; treat any funding window from it as unconfirmed until MeitY publishes implementation rules.

29.2 Grants and programmes a small team could actually apply to

Several of these require DPIIT recognition first. DPIIT, the Department for Promotion of Industry and Internal Trade, is the Indian government office that officially registers a company as a startup for schemes like the ones below.

PROGRAMME	WHAT IT IS	ELIGIBILITY / AMOUNT	TAG	IMPLICATION
Startup India Seed Fund Scheme (SISFS)	A DPIIT scheme, 945 crore rupee outlay, routed through approved incubators rather than applied to directly	DPIIT-recognised startup, incorporated under 2 years; up to 20 lakh rupees for a proof of concept, prototype or trials (search-reported)	ADOPT	A realistic near-term target once DPIIT recognition exists, applied for through an approved incubator such as an IIT technology business incubator or ARTPARK, not the government directly. Small, but enough to cover cloud and compute for a demo phase.
ARTPARK @ IISc, Technology Innovators in Residence and Venture Incubation Programme	Deep-tech (AI plus robotics) incubation at IISc (the Indian Institute of Science) in Bangalore, seeded by the government's NM-ICPS programme, including a 75,000 square foot robotics manufacturing and testing facility called ARTGarage	Confirmed to exist on artpark.in, browsed directly; funding amounts, equity terms and eligibility detail were not published on the page checked	ADOPT	Best India-specific fit by subject overlap, AI plus robotics plus a physical prototyping facility, of anything found in this research. Check artpark.in directly for the next application window rather than relying on this summary.
RDI (Research, Development and Innovation) Fund	A 1 lakh crore rupee government fund under the Department of Science and Technology; launched November 2025	Targets higher-readiness projects and critical-technology purchases; cannot be applied to directly, flows through approved fund managers	WATCH, not yet ADOPT	Explicitly not for idea-stage ventures by the scheme's own design. SuperHuymn is pre-revenue and would not qualify yet. Revisit once Forge Studio has a paid pilot.
Karnataka Deep-Tech Elevate Programme, "LEAP"	A state-level deep-tech grant programme, 2026 rollout	40 startups to receive 50 lakh to 1 crore rupees (AI, semiconductors, quantum, robotics, materials); a 300 crore rupee fund-of-funds plus a 100 crore rupee LEAP allocation (search-reported)	ADOPT	If SuperHuymn is Karnataka-based, a concrete, sizeable, robotics-eligible state grant to track for the 2026 cohort. Needs a check against the state's own startup portal before relying on the figures.
Telangana T-Hub / T-Fund	A large innovation campus with a state co-investment fund alongside angels and venture capital, focused on business and enterprise technology	search-reported, no specific amounts found	WATCH	An alternate-state option if not Karnataka-based; needs its own primary check.
IndiaAI Mission, subsidised GPU compute	A 10,372 crore rupee, five-year national AI compute, dataset and startup-support programme	34,000 GPUs available at 115 to 150 rupees a GPU-hour, roughly 42% below market (search-reported), scaling to 100,000 GPUs by end of 2026	ADOPT for the subsidised-compute track, WATCH for the direct-grant track	The general subsidised compute access is realistically usable by a four-person lab needing compute for a demo. The direct grant track (already used by four foundation-model startups) does not apply, since SuperHuymn orchestrates Gemini rather than training a foundation model.
NVIDIA Inception Program	A free global startup programme: training credits, hardware and software preferred pricing, venture-network introductions, partner cloud credits	Free to join; needs at least one developer, a working website, and to be an incorporated company under 10 years old (search-reported); over 2,000 Indian member startups already, including 25 or more robotics companies	ADOPT	Free, low effort, and directly on-thesis. Should be one of the first things the team applies for.

PROGRAMME	WHAT IT IS	ELIGIBILITY / AMOUNT	TAG	IMPLICATION
Google for Startups Cloud Program	Google Cloud credits for AI-first startups	Up to $350,000 in credits over 2 years for Vertex AI or Gemini-based startups (search-reported); India-specific API pricing up to 70% lower	ADOPT	Directly on-thesis since Forge Studio already runs on Gemini. The single highest-value free-money item here, because it subsidises exactly the model calls the product depends on.
Microsoft for Startups Founders Hub	A no-funding-required startup credit bundle	Up to $150,000 in Azure credits, 20 seats of GitHub Enterprise and Copilot for a year, Microsoft 365, LinkedIn Premium	ADOPT	No investor backing needed. A good parallel credit source alongside Google's, useful for the hosting side if the team does not standardise on one cloud alone.
CHAMPIONS portal / Digital MSME (ICT promotion) scheme	A Ministry of MSME portal for grievance handling, technology and finance support, and an ICT-adoption subsidy scheme for small manufacturers	search-reported; specific subsidy amounts not retrieved	STUDY, as a go-to-market channel, not a grant for SuperHuymn itself	Relevant as demand-side signal: if Forge Studio is ever packaged as an affordable design tool for small manufacturers, this is the group of schemes the customer already uses.
PM Vishwakarma Yojana	An artisan and traditional-trade digitisation and micro-loan scheme covering 18 family-based trades	Scoped to self-employed artisans in the unorganised sector, not to software companies	AVOID	Wrong beneficiary class; included only to rule it out.
GeM (Government e-Marketplace)	India's public-procurement platform; a GeM Startup Runway exists for innovative or unique-design products	DPIIT-recognised startups and MSMEs eligible; startups reportedly secured over 19,000 crore rupees in orders, growing more than 36% a year (search-reported, single source)	STUDY	A real, low-cost channel to sell Forge Studio, or bespoke CAD-agent services, to Indian government and public-sector manufacturing once the product is sellable. Not useful pre-revenue.

Net read on India-specific funding: SISFS, ARTPARK, Karnataka's Deep-Tech Elevate grant, NVIDIA Inception, Google for Startups Cloud and Microsoft Founders Hub form a realistic, low-dilution stack a four-person team can pursue in parallel without a revenue history. The RDI Fund and IndiaAI Mission's direct foundation-model grants are not yet a fit. DPIIT recognition is a common early prerequisite and should be treated as a cheap, early step rather than optional paperwork.

29.3 Who publishes and hires in agentic CAD

Research venues, all search-reported and abstract-level only unless noted: Text2CAD-Bench (May 2026), a benchmark for text-to-parametric-CAD finding that current models handle basic shapes but degrade on complex topology, the connection structure of a shape; CADDesigner (August 2026), conceptual CAD generation by a general-purpose agent; cadrille, multi-modal CAD reconstruction with reinforcement learning; Foundation Models for Automatic CAD Generation (July 2026); Test-Time Scaling for CAD Generation via Verifier-Free Consensus Selection (August 2026); FutureCAD, an LLM paired with a transformer that grounds text in exact B-rep surfaces; and GIFT, image-to-CAD program synthesis using geometric feedback.

The Symposium on Solid and Physical Modeling and Shape Modeling International, held jointly in July 2026 in Istanbul, is run by the Solid Modeling Association. The 2026 edition already happened by the time of this research; track the 2027 call for papers.

ASME IDETC-CIE, a mechanical-engineering design conference, held its 2026 edition in Houston. Directly relevant papers found there include a multi-agent framework for car design, and CAD-Coder, an open vision-language model for CAD code generation that won a best-paper award at the 2025 edition. This is the mechanical-engineering field's own venue for agentic design, distinct from computer-science venues like the one above.

The Journal of Computing and Information Science in Engineering published "Intelligent Design 4.0," a framework for fully automated engineering design through multi-agent systems. Read this paper directly before citing it; it looks like the closest existing academic description of what Forge Studio is trying to build a product version of. Computer-Aided Design, an Elsevier journal, is the field's long-standing flagship publication.

On hiring: Autodesk Research posts open roles in design automation, AI, computational geometry and human-computer interaction, useful for benchmarking pay, not as a partner target. Zoo.dev hires through Hacker News "Who's Hiring" posts among other channels, a channel a bootstrapped Indian lab could use too for remote-friendly technical hires. PTC's Onshape Labs is an early-access programme, which works as a beta-tester and community-credibility channel rather than a hiring channel. It is worth joining early just to see a shipping competitor's MCP tool-schema design.

KEY POINTS

- India's ER&D market is $63 billion now and projected past $100 billion by 2030; that is the addressable services market SuperHuymn is adjacent to, not a market it sells into directly yet.
- A realistic, low-dilution funding stack exists for a four-person pre-revenue team: SISFS, ARTPARK, a Karnataka state grant, NVIDIA Inception, Google for Startups Cloud and Microsoft Founders Hub.
- DPIIT recognition is a prerequisite for several of these programmes and should happen early.
- The RDI Fund and IndiaAI Mission's direct grants are for later-stage or foundation-model work; not a fit yet.
- ASME IDETC-CIE and the JCISE journal are the two venues most directly matched to what Forge Studio is attempting; a case-study submission there builds credibility with engineers, not just investors.

WHAT SUPERHUYMN DOES WITH THIS

Register for DPIIT recognition first, since it unlocks the Startup India Seed Fund Scheme and the GeM Startup Runway later. In parallel, apply to NVIDIA Inception, Google for Startups Cloud and Microsoft for Startups Founders Hub, all free and on-thesis. Check artpark.in directly for the next VIP or TIR application window before assuming this lesson's numbers are current.

Lesson 30. The industrial picture

WHAT YOU WILL LEARN

- How much money moved into physical AI and robotics in 2026, and why most of it is not reachable by a new entrant.
- The real reason manufacturing, not humanoid robots, is where design-tooling revenue shows up first.
- Which open robot arms exist at each price tier, useful if SuperHuymn ever builds toward assembly-line arms.

30.1 Where the money went

All figures below are search-reported, useful for calibration rather than exact citation.

METRIC	REPORTED VALUE
Physical AI equity raised, pure-play dataset	$8.73B
Q1 2026 funding	$6.4B across 27 deals
Year-to-date 2026 versus 2025	About 7.1 times
Top 10 deals' share of capital	78.8%
Robot foundation models and general-purpose robots' share	77.6%
Follow-on rounds' share of 2026 deals	92.0%
Large manufacturers testing or using physical AI by early 2026	67%, with efficiency gains reported up to 40%

Valuations reported: Figure AI about $39B, Skild AI about $14B (close to $1.4B raised in January 2026), Wayve about $8.6B, Physical Intelligence about $5.6B (after a $600M Series B led by CapitalG in November 2025), Apptronik about $5B, Field AI about $2B.

Read this carefully before pitching. Follow-on rounds, meaning later rounds into companies that already raised before, made up 92% of 2026 deals, and the top ten deals took 78.8% of the capital. That is a market funding its existing winners, not new entrants, and it is concentrated in robot foundation models (AI systems trained to control many kinds of robots) and humanoids. A bootstrapped Indian lab in agentic CAD is not competing for that pool of money and should not pretend to be. The honest position is the opposite one: manufacturing is the commercially serious end of physical AI, because factories have structured tasks, a clear labour need and measurable returns, and design-side tooling reaches revenue years before a humanoid robot does.

30.2 Open hardware, for when the arm goal gets real

LeRobot (Hugging Face) is the integration point for open robot hardware, led by Remi Cadene, formerly of Tesla Optimus. It documents hardware support for SO-101, SO-100, Koch v1.1, LeKiwi, Hope Jr, Reachy 2, Unitree G1, Earth Rover Mini, OMX, OpenArm, and ALOHA and ViperX rigs through Trossen.

Desktop tier: Reachy Mini ($299 to $449), SO-100 and SO-101 arms under $500. Research bench tier ($5,000 to $25,000): the ALOHA bimanual rig, meaning two arms working together, Trossen ViperX, OpenArm, Hope Jr.

OpenArm is an open-source 7-degree-of-freedom arm, meaning it has seven independent joints, more than the six a typical industrial arm has, led by Enactic in Japan and sized to human proportions for a 160 to 165 cm person, balancing reach against how hard the arm is to move safely. Seeed reBot Arm B601-DM is a 6-degree-of-freedom arm with a fully open hardware and software stack, supporting ROS (Robot Operating System), LeRobot, Isaac Sim, MoveIt and Pinocchio. It launched around April 2026. Industrial arms to model against: Franka Emika (7 degrees of freedom, senses force at each joint), Universal Robots, xArm, KUKA iiwa.

If SuperHuymn's long-term goal is arms for assembly lines, the cheapest credible first step is generating parts for one of these open arms and simulating them, not designing an arm from nothing.

KEY POINTS

- 2026 physical-AI funding is concentrated: 92% follow-on rounds, top ten deals took 78.8% of capital, mostly into robot foundation models and humanoids.
- SuperHuymn should not pitch into that funding pool; manufacturing-side tooling is the nearer, more honest revenue path.
- Reported valuations run from about $2B (Field AI) to about $39B (Figure AI), useful only as calibration, not as citable fact.
- Open robot arms exist from under $500 (SO-100, SO-101) to $25,000 (ALOHA, OpenArm); LeRobot is the integration point across almost all of them.
- The cheapest credible path toward assembly-line arms is generating and simulating parts for an arm that already exists, not designing a new one.

WHAT SUPERHUYMN DOES WITH THIS

When the arm goal becomes concrete, target an existing open arm such as the SO-101 or OpenArm for the first generated parts and simulation, rather than designing a new arm. Do not include physical-AI funding-pool numbers in a pitch as if SuperHuymn competes for that capital; the manufacturing framing is the honest one.

PART VI. Decisions

Lesson 31. Where a small team wins

WHAT YOU WILL LEARN

- Why "agentic CAD" stopped being a useful way to describe Forge Studio, and what to call it instead.
- Five concrete risks worth naming out loud before they cause a surprise.
- A ranked list of where the backend and agentic-AI seat counts most, and five things not to do.

31.1 What I would tell the team

Stop describing Forge Studio as agentic CAD. Everyone says that now, including Autodesk (Lesson 28). Describe it instead as a system that produces manufacturable geometry with a proof attached. The proof is the product: the part is watertight (its skin has no gaps), it does not self-intersect (pass through itself), it clears its neighbours by a stated margin, it can be cut by a real tool, and here is the machine-checkable evidence for each claim. Nobody in the open field is selling that, and every industrial buyer needs it before a generated part touches a machine.

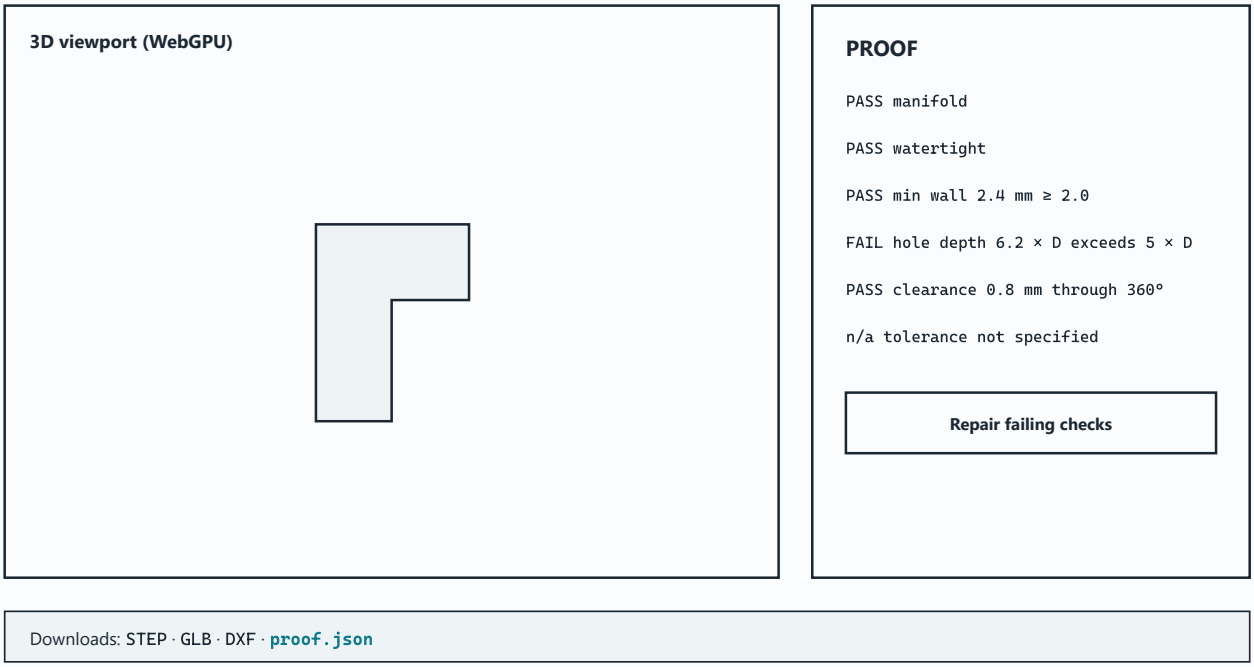


Figure 9. The deliverable is the part plus its evidence, and `proof.json` is the thing competitors

The demo that wins is not a prettier part. It is the same part with a failure injected on purpose, caught by the gate, repaired automatically, and the difference shown. Anyone can screenshot a rendered bracket. Almost nobody can show a system catching its own mistake.

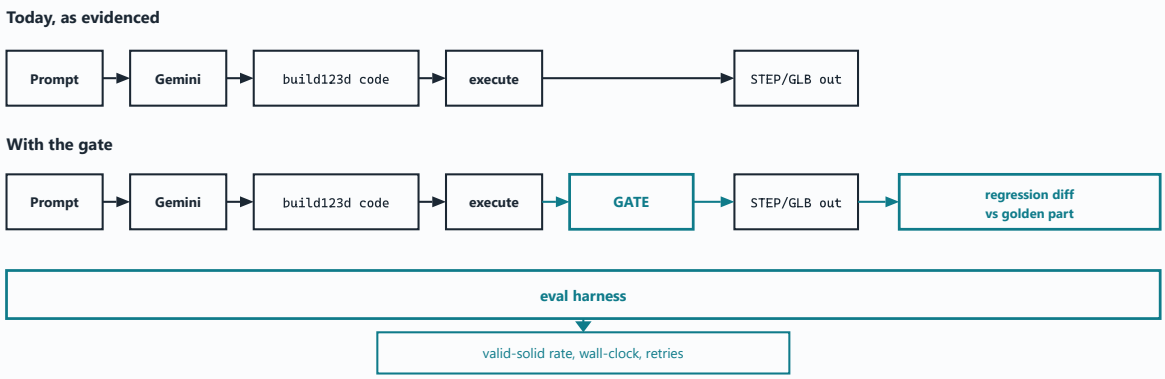


Figure 7. The additions are small and they are the whole difference.

The number matters more than the video. Twenty prompts from a public benchmark, three numbers: valid-solid rate, wall-clock time, retries. A week of work. Funders discount demos and remember rates.

31.2 Risks worth naming out loud

- Distribution asymmetry. Autodesk reaches every Fusion seat by shipping a connector. A bootstrapped lab reaches nobody by shipping a better model. Whatever gets built needs a distribution path that is not "people find our website." Publishing a skill, a packaged set of instructions an AI agent can load, into the Claude Code and Codex skill registries is the cheapest one available.
- Documentation debt. A thinly documented codebase slows every new contributor and makes onboarding expensive. Documentation and an eval harness, a tool that runs a system through standard tests and reports numbers, are therefore the two highest-value backend contributions available, and neither collides with feature work in progress.
- Model dependency. The generation step is built around one commercial model API. A price change, a rate limit or a deprecation moves the cost base and the roadmap at once. Keep the code-generation step swappable behind an interface, and keep at least one open-weight model in the test matrix so the option stays real rather than theoretical.
- GPU dependency creep. TRELLIS needs a CUDA GPU, a graphics processor built for this kind of parallel computation. Newton, Isaac Lab and Genesis, all simulation tools, are GPU-first too. A CPU-first deploy is the right call for a demo, and every additional GPU-required component raises hosting cost for a team with no revenue.
- Benchmark exposure. CAD Arena (Lesson 27) is public. A score there is marketing if good and permanent if bad. Run the numbers privately first.

31.3 Where the backend seat counts most

Ranked for the person doing backend and agentic AI work, not for the company in the abstract.

1. The eval harness. Nobody has one yet, it is backend work, it produces the number the company needs, and it is publishable.

2. The verification gate. Geometric validity checks, plus a render-and-ask vision loop, a system that looks at a rendered part, asks itself questions about it, and checks its own answers, plus DFM rules, design for manufacturing, checking a shape can actually be made, behind one interface. This is the product's strongest defence, and it is squarely backend work.
3. A production-grade `build123d-to-URDF/MJCF` exporter, a tool converting a CAD model into the robot-description formats used by ROS and MuJoCo. Only two-star prototypes exist today. It unblocks the internal CAD-to-physics pipeline, it is a real open-source contribution under the builder's own name, and it doubles as the written material the founder asked for.
4. The deploy itself: a Dockerfile with Python pinned to 3.10, the `MUJOCO_GL=egl` environment setting for headless rendering, a job queue for anything over thirty seconds, and a health-check endpoint. Unglamorous, blocking everyone, and already assigned.
5. MCP server hardening: clear tool descriptions, error messages an agent can recover from, progress updates while a job runs, and timeouts. This is the difference between a server an agent can use well and one it struggles against.

31.4 What I would not do

- Do not migrate off OpenCASCADE. Nothing found justifies it.
- Do not build another MCP-wrapped CAD API as a differentiator. There are over a hundred already.
- Do not chase mesh generative models as the geometry path. Mesh-to-B-rep conversion, turning a triangle shell back into exact surfaces (Lesson 18), is a research problem, not an engineering task, and Spectral Labs' SGS-1 already has a head start on it.
- Do not pitch into the robot-foundation-model funding pool (Lesson 30). In 2026 that pool ran 92% follow-on rounds, with the top ten deals taking 78.8% of capital. It is funding existing winners.
- Do not claim the company does something it does not. Every capability claim in a deck should map to a test that actually runs.

KEY POINTS

- Call Forge Studio a system that proves its geometry, not "agentic CAD."
- The winning demo shows the system catching and repairing its own mistake, not a prettier rendered part.
- Five risks are worth saying out loud: distribution asymmetry, documentation debt, dependency on a single model vendor, GPU dependency creep, and public benchmark exposure.
- The backend priority order is: eval harness first, then the verification gate, then the CAD-to-robot exporter, then the deploy itself, then MCP hardening.
- Five things not to do: migrate off OpenCASCADE, build another MCP wrapper, chase mesh-based generation, chase humanoid funding, or claim a capability with no test behind it.

> **WHAT SUPERHUYMN DOES WITH THIS**
>
> Work the priority list in section 28.3 in order: build the eval harness first, then the verification gate, then the `build123d-to-URDF/MJCF` exporter, then finish the Dockerfile deploy, then harden the MCP server. Each one is backend work, and each one is publishable.

Lesson 32. The ranked action list

WHAT YOU WILL LEARN

- Thirteen concrete build ideas, each traced to a specific gap or resource found in this guide.
- Why the geometry-derived items now rank above the others.
- Which single idea converts a demo video into a number a funder can check.

32.1 Why the order changed

The original ranking put the verification layer first and eval numbers second, which was already close. This version moves every geometry-derived item to the top of the list, including one item not in the original twelve: enforcing property-based face selection (Lesson 3), because it is the cheapest fix in this entire guide and the most expensive one to retrofit later. The reasoning does not change: each item below traces to a specific gap or resource named earlier.

32.2 The list, ranked

1. Build the verification gate and name it in the pitch. Combine geometric validity checks, the render-and-ask vision loop, and DFM rules into one gate every generated part must pass. Verification (Lesson 13) is nearly empty as a field while agent harnesses are crowded. This is the product.
2. Encode the DFM rule set as validators. Hole depth-to-diameter ratio, features a cutting tool cannot reach, sharp internal corners, deep pockets and slots, flat-bottomed holes, fillets on outside edges. About half a day of work, and it turns "the AI made a shape" into "the AI made a part you can machine."
3. Add geometric regression testing with diff3d, a tool that visually compares two 3D files. Diff every generated part against a saved-good version on every prompt or model change, and fail the build on drift. Nobody in this field appears to be doing this systematically. It is cheap, and it is the engineering discipline that separates a demo from a product.
4. Enforce property-based face selection everywhere geometry is chosen. Instead of referring to "Face 6," refer to "the face whose normal points up and whose area is largest." This is the single most valuable coding convention against the topological naming problem (Lesson 3), where a rebuild silently renumbers faces and an agent edits the wrong one with no error raised. Cheap to add now, expensive to retrofit.
5. Publish eval numbers. Twenty prompts from BenchCAD or Text2CAD-Bench, three numbers: valid-solid rate, wall-clock time, retries. One week of work, and it converts a demo into a claim (Lesson 27).
6. Write a production-grade `build123d-to-URDF/MJCF` exporter. Joint limits read from the model, mass and inertia computed from the geometry, and a real test suite. Only two-star prototypes exist. It unblocks the internal CAD-to-physics pipeline and is a genuine open-source contribution and reference-material candidate.
7. Adopt bd_warehouse plus a parts library. Stop letting the model derive standard fasteners from first principles. An immediate cost and reliability win.

8. Read `baibai2013/build123d-cad` end to end. Two stars, but someone independently built SuperHuymn's roadmap as 25 linked skills, so no competitive threat and a free blueprint, including a URDF exporter.
9. Ship a public Claude skill for Forge Studio's MCP server. About a day of work, distribution into every Claude Code and Codex user, and no backend code given away. Follow `NVIDIA/skills`' packaging pattern (Lesson 15).
10. Benchmark against SGS-1. Same mesh inputs, compare STEP outputs. Whatever the result, it shows where the real advantage sits.
11. Read KCL's design notes before writing more agent prompts. Whether the agent should emit Python or a purpose-built language is the biggest unexamined architectural decision in the current design (Lesson 16).
12. Add browser-side CAM, computer-aided manufacturing, generating toolpaths for a machine, through Kiri:Moto. A part that arrives with a toolpath is a different product from a part that arrives as a bare file.
13. Consider point-cloud-to-CAD as a second vertical. Factories already pay for reverse engineering, and it is nearer to revenue than assembly-line arms. CAD-Recode is the starting point; SGS-1 is the competition.

KEY POINTS

- The four geometry-derived items (verification gate, DFM validators, diff3d regression, property-based selection) now sit at the top because they are the cheapest fixes and the most expensive to retrofit.
- Publishing eval numbers stays a top priority: one week of work for a checkable claim.
- Every item traces to a specific gap or resource named earlier in this guide; nothing here is a new idea introduced at this stage.

WHAT SUPERHUYMN DOES WITH THIS

Treat this list as the backlog in order. Items 1 to 4 should be built before items 5 to 13 are even scheduled, because they are cheap now and expensive later.

Lesson 33. What could not be verified

> WHAT YOU WILL LEARN
>
> - Which claims in this guide rest on a single search result rather than a checked primary source.
> - Specifically which dates, prices and eligibility rules should be checked again before they go into a pitch deck.
> - That naming a guide's own blind spots is part of making it trustworthy, not a weakness to hide.

33.1 What v1 already flagged

- Star counts marked (verified) were read directly from GitHub topic pages on 2026-09-08. Everything else is search-reported. Licences and last-commit dates were not checked individually; check the licence file before adopting anything.
- Whether Spectral Labs SGS-1 publishes open weights is unconfirmed.
- Whether cadquery-ocp, the Python binding to OpenCASCADE that build123d depends on, has a binding against OpenCASCADE 8.x is unconfirmed, and it matters for any plan built around the newer kernel version.
- The funding figures in Lesson 30 come from market-analysis blogs, not company filings. Treat them as calibration, not citation.
- The Awesome-Physical-Engineering-AI reading list's own README could not be fetched directly, so its contents are known only from search snippets.
- Forge Studio's internals are known here only from internal project notes and public material. Where this guide says "Forge Studio does not have X," read it as "no evidence of X was available," and confirm it internally before repeating it.
- `baibai2013/build123d-cad` is documented mostly in Chinese. Its capability list, quoted through this guide, is translated from its README and has not been verified by running the skills.

33.2 What R-3 flagged

- The exact announcement date of the `Autodesk/Anthropic` Fusion MCP launch. Third-party sources say 28 April 2026, but Autodesk's own page, read directly this session, does not itself state a launch date. Treat 28 April 2026 as unconfirmed (Lesson 28).
- PTC's own Onshape Labs press release content. The page returned an access error, and a rendering attempt loaded the page shell without the article text. Everything about Onshape Labs in this guide is search-reported through third parties, not PTC's own words.
- Dassault Systemes' specific 2026 AI-assistant capabilities. Only a summary list of "fascinating applications" was found; the keynote or product page itself was not read.
- nTop's and Physna's per-seat pricing. Both are confirmed quote-only across several pricing-aggregator sites, but no actual quote or contract figure was found or obtainable without contacting sales.
- The Karnataka Deep-Tech Elevate Programme's exact application window and portal address for the 2026 cohort. Only secondary press coverage was found; the state's own startup portal was not checked directly.

- ARTPARK's TIR and VIP 2026 funding amounts, equity terms and application deadline. The page, read directly, confirms the two programmes exist and that application links are present in navigation, but states no amount or date.
- Whether the IndiaAI Mission's subsidised-compute track is actually reachable by a four-person, unfunded lab, versus only by larger or already-registered research entities. Eligibility language mentions "researchers, startups, MSMEs, academic institutions," but no application process detail or waitlist status was found.
- A second, independent source for the India robotics market figure of $480 million growing 34% a year (Lesson 29). Only one source, RoboticsCenter.ai, was found.
- The GeM Startup Runway's exact eligibility bar and current order-value figure. The 19,000 crore rupee, 36%-a-year figure came from a single search summary, not GeM's own reporting page.
- The Digital `MSME/CHAMPIONS` scheme's actual subsidy amount and which products qualify. The scheme's existence is confirmed; its terms were not retrieved.

33.3 What was dropped for lack of a source

Nothing in Parts IV and V of this guide was dropped for having no source at all. Every figure carried into Lessons 24 through 29 already arrived tagged verified or search-reported in v1 or R-3, and those tags are preserved rather than upgraded or removed. Where a claim came from a single, uncross-checked source, such as the India robotics market figure, that limit is repeated here rather than smoothed over.

> KEY POINTS
>
> - Two specific figures need a primary-source check before appearing in any funder-facing document: the `Autodesk/Anthropic` launch date, and the India robotics market size.
> - Three funding programmes (Karnataka Deep-Tech Elevate, ARTPARK `TIR/VIP`, IndiaAI Mission subsidised compute) have application windows or eligibility rules that were not confirmed and must be checked directly on the relevant government or programme site.
> - No claim in this guide's evidence-and-market half was invented or left uncredited; the caveats above are the complete list.

> WHAT SUPERHUYMN DOES WITH THIS
>
> Before any of these numbers goes into a pitch deck or a grant application, check it against its primary source: Autodesk's own announcement for the Fusion MCP date, ARTPARK's site for the 2026 `VIP/TIR` window, and karnataka.gov.in for the Deep-Tech Elevate cohort. This is a half-day task, not a research project, and it is cheap insurance against citing a number that turns out to be wrong in front of a funder.

Appendix A. Glossary

Alphabetised from `03-GEOMETRY-FOUNDATIONS.md`, section 16. Definitions are given exactly as that document states them.

AP242: the STEP application protocol carrying semantic PMI. **B-rep:** boundary representation. Exact surfaces plus how they connect. What engineering CAD uses. **Convex decomposition:** splitting a complicated shape into several convex pieces (V-HACD). **Convex hull:** the shrink-wrapped outer shape, used for fast collision. **DFM:** design for manufacturing. Checking a shape can actually be made. **DOF:** degrees of freedom, the number of independent motions. **FEA:** finite element analysis. Filling a volume with elements to compute stress. **Forward kinematics:** joint angles to hand position. **GD&T:** geometric dimensioning and tolerancing, the language of allowed variation. **Inertia tensor:** how hard an object is to spin about each axis. **Inverse kinematics:** hand position to joint angles. **Manifold:** every edge borders exactly two faces. A proper closed skin. **Mesh:** a skin of flat triangles. What screens and AI generators use. **PMI:** product and manufacturing information. Semantic PMI is machine-readable. **Quaternion:** a four-number way of writing a rotation that avoids gimbal lock. **Sewing tolerance:** the small gap the kernel will stitch across when joining faces. **STEP:** the neutral B-rep exchange format. What suppliers accept. **Tessellation:** turning a B-rep into a mesh for display. Controlled by deflection tolerance. **Topological naming problem:** references to faces and edges breaking when the model is rebuilt. **Topology:** the connection structure of a shape, without coordinates. **Transform:** a rotation plus a translation, placing one frame relative to another. **URDF / MJCF / SDF:** robot description formats for ROS, MuJoCo and Gazebo. **Watertight:** no gaps. The solid has a real inside.

Appendix B. Failure checklist

One page, pulled from `03-GEOMETRY-FOUNDATIONS.md`, section 14. Every row is a symptom a machine can catch, collected from the sources in this research and from the W16 engine demo's own record.

SYMPTOM	USUAL CAUSE
Part is 1000 times too big or small	Millimetres versus metres at a format boundary
Boolean operation fails or returns nothing	Faces coincident within sewing tolerance, or a self-intersecting input
Fillet fails	Radius larger than the space available at that edge
Volume is zero	Shape is a shell, not a closed solid
Simulation behaves oddly but looks fine	Wrong inertia tensor, or collision hulls much coarser than the visual mesh
The agent edits the wrong face	Topological naming, see Lesson 3
Export is enormous	Tessellation deflection set too fine
Assembly looks right, parts interfere	Clearance never checked through the full motion range, only at rest

A verification gate that catches all of these is the highest-value thing SuperHuymn can build (Lesson 31, Lesson 32). Every row above is a machine-checkable test, not a matter of judgement.

Appendix C. Robot hardware and demonstration data

This appendix collects material from the robotics research pack that did not belong inside any single lesson. It matters when SuperHuymn's stated long-term goal, robotic arms for assembly lines, stops being a plan and becomes a build.

Two practical points before the tables. First, do not design an arm from nothing. Generating parts for an existing open arm and simulating them is a far cheaper first step, and it produces a demo sooner. Second, if the team ever trains a policy rather than only simulating one, the data format decision should be made once, early, and not revisited.

Star counts marked (verified) were read from GitHub on 2026-09-08.

C.1 Open arm hardware and the LeRobot stack

ITEM	STARS / NOTE	LICENCE	TAG	WHAT SUPERHUYMN DOES WITH IT
Hugging Face LeRobot	27,300 stars, 5,600 forks (verified)	Apache-2.0	ADOPT	The end-to-end stack: hardware interface, dataset format, policies, evaluation. The fastest route from a real arm to a working imitation-learning loop, before anything custom is needed.
LeRobotDataset format	part of the LeRobot repo	Apache-2.0	ADOPT	A Parquet and MP4 schema hosted on the Hugging Face Hub. Adopt it if the team ever records its own assembly demonstrations, instead of inventing a format.
SO-100 / SO-101 arm	7,400 stars, 670 forks (verified)	Apache-2.0	ADOPT	Roughly 200 to 300 US dollars, 6 degrees of freedom, 3D printed with hobby servos, supported by LeRobot out of the box. The obvious bench arm for a pick-and-place prototype before touching an industrial cobot.
ALOHA and Mobile ALOHA (Stanford)	search-reported: bimanual teleoperation rig	open hardware	STUDY	The pattern for collecting good demonstration data with two arms. Study the method, not needed at single-arm scale.
ACT, Action Chunking Transformer	ships inside LeRobot	Apache-2.0	STUDY	The imitation-learning policy behind ALOHA. It predicts a chunk of actions rather than one step at a time. Read it as the baseline before reaching for a full vision-language-action model.
OpenArm	search-reported	open hardware	STUDY	A 7-degree-of-freedom arm from Enactic in Japan, scaled to human proportions for a person of 160 to 165 cm, trading reach against inertia for safe operation.
Seeed reBot-DevArm / Arm B601	search-reported, launched around April 2026	open hardware and software	STUDY	6 degrees of freedom with the full stack open, supporting ROS, LeRobot, Isaac Sim, MoveIt and Pinocchio.
Trossen ViperX, Franka Emika, Universal Robots, xArm, KUKA iiwa	search-reported	commercial	WATCH	The arms an industrial customer already owns. Model against these rather than inventing a target platform.

Price tiers reported in the sources: desktop learning arms under 500 US dollars, including Reachy Mini at 299 to 449, and research bench rigs between 5,000 and 25,000.

C.2 Robot-learning datasets

These matter only if SuperHuymn trains policies. They are listed so the decision is informed rather than accidental.

DATASET	REPORTED SCALE	TAG	WHAT SUPERHUYMN DOES WITH IT
Open X-Embodiment / RT-X	search-reported: over 1M trajectories, 22 embodiments, 60 datasets, 34 labs	ADOPT	The standard corpus behind OpenVLA and Octo. The first place to pull manipulation demonstrations from, before spending months collecting your own.
DROID	search-reported: 76,000 trajectories, 350 hours, 13 institutions, Franka Panda with stereo cameras	STUDY	Notable for standardising on one hardware rig. If the team collects its own data, fix one arm and one camera layout the same way.
RoboMIND and RoboMIND 2.0	search-reported: single and dual arm, up to 6 platforms	WATCH	Cross-embodiment data including UR5e, which is cobot-class and closest to an assembly arm.
AgiBot World	search-reported: over 1M trajectories, 217 tasks, 106 scenes	WATCH	Humanoid scale. Lower near-term relevance for a fixed assembly arm.

C.3 The CPU-first warning

SuperHuymn is deploying without a local machine and without a guaranteed GPU. The robotics research pack flagged these as effectively GPU-mandatory: Isaac Lab, cuRobo, Contact-GraspNet, Isaac GR00T in practice, Omniverse and Mega, and Genesis, which has a CPU backend but loses its speed advantage on it.

Plan around that. Rigid-body simulation in MuJoCo and kinematics in Pinocchio run fine on a processor. Policy training does not. Keep the two on separate deployment tracks so a demo never waits on a graphics card.

> **WHAT SUPERHUYMN DOES WITH THIS**
>
> Nothing this month. This appendix exists so that when the arm goal becomes real, the team starts from an SO-101 on the bench and the LeRobot dataset format, rather than from a blank page and a bespoke schema.

Appendix D. Sources

v1's sources, unchanged, plus every named source from `packs/R-3-commercial.md` not already listed below. Grouped by topic; duplicates dropped.

Verified by direct browsing, 2026-09-08: GitHub `text-to-cad` topic · GitHub `build123d` topic · `baibai2013/build123d-cad` · `NVIDIA/skills` · GitHub search: build123d urdf

Agent harnesses and text-to-CAD: `earthtojake/text-to-cad` · `Adam-CAD/CADAM` · `Pan-Chera/Multi-Agent-CAD` · `partcad/partcad` · `armpro24-blip/cad-cae-copilot` · `clay-good/anvilate` · `cyberchitta/cad-khana` · `jdilla1277/agentcad` · `forgent3d/forgent3d-desktop` · `EESJGong/Graph-CAD` · `SadilKhan/NURBGen` · `Kevoyuan/AgentSCAD` · `vul-os/kerf` · `SmartAI/Chamfer` · `MLYengineering/Text2CAD` · `anniedoris/CAD-Coder`

MCP: `pzfreo/build123d-mcp` · `Casys-AI/mcp-build123d` · `Alfredoalv13/shapr3d-mcp` · `Svetlana-DAO-LLC/cad-agent` · 9 MCP servers for CAD (Snyk) · CAD and 3D modeling MCP servers · ODA MCP servers

Skills: `aklofas/kicad-happy` · `VoltAgent/awesome-agent-skills` · `BenCaunt/SynthCAD` · `ppak10/RocketSmith` · `Soljourner/claude-engineering-skills` · `010zx00x1/Awesome-Physical-Engineering-AI`

Kernels, libraries, viewers: `gumyr/build123d` · CadQuery · OCCT · awesome-build123d · awesome-cadquery · `mlightcad/awesome-cad` · build123d external tools · `yeicor/OCP.wasm` · yet-another-cad-viewer · three-cad-viewer · jupyter-cadquery · vscode-ocp-cad-viewer · `jdegenstein/nice123d` · `bdlucas1/diff3d` · stellarmesh · Truck · CADmium · Dune3D · Zoo KCL

Research and benchmarks: BenchCAD · Text2CAD-Bench · P3D-Bench · CAD-Judge · CAD-MLLM · cadrille · EvoCAD · Zero-to-CAD · Embodied CAD · Agent-Aided Design for Dynamic CAD · CADDesigner · CADmium fine-tuning · CAD Arena · CHIA co-design framework · Foundation Models for Automatic CAD Generation · Test-Time Scaling for CAD Generation via Verifier-Free Consensus Selection · FutureCAD, LLM-driven program generation and B-rep primitive grounding · GIFT, image-to-CAD via geometric feedback

Physics and robots: State of simulation for physical AI (NVIDIA) · Isaac Lab · MuJoCo Playground · onshape-to-robot · mujoco_menagerie · URDF, MJCF, USD · MuJoCo headless in Docker · RoboMoRe · Debate2Create · Articraft · SR-Platform · LeRobot OpenArm docs · Seeed reBot-DevArm · Trossen open-source arm guide · awesome-physical-ai · Awesome-Embodied-Robotics-and-Agent

Electronics and firmware: circuitron · tscircuit · PCBSchemaGen · AI PCB design walkthrough · Codex CLI for embedded, PlatformIO MCP, Zephyr · Zephyr on PlatformIO and Renode · Securing LLM-generated firmware · LLM multi-agent mechatronics

Analysis and manufacturing: TopOpt Python · FEniTop · STORX · BenDFM · Sheet-metal bending effort ML · Open-source CAM in the browser · CNC DFM checks

Generative 3D: Step1X-3D · SGS-1 · The AI CAD Frontier · stl2step · DTGBrepGen · BrepForge · Mesh to B-rep, the hard problem · Hunyuan3D vs TRELLIS vs TripoSR

Commercial and industrial: Claude for CAD, Blender and Fusion connectors (DEVELOP3D) · MCP arrives in CAD · Autodesk Fusion MCP analysis (CoLab) · Zoo ML API · Zoo vs Adam vs SGS-1 · AI CAD software in 2026 · Physical AI funding trends · Physical AI Q1 2026 funding · 20 physical AI

companies to watch · Open-source humanoid buyers guide · Autodesk x Anthropic, Claude for Fusion (browsed) · Zoo.dev pricing (browsed) · Physna public API guide

Commercial and incumbent moves, additional context (R-3, no direct URL retrieved): DEVELOP3D, PromptArmor and onemetrik write-ups on the Fusion MCP server, search-reported · ARC Advisory Group and PTC's own newsroom on Onshape AI Advisor and Onshape Labs, search-reported, PTC's own release returned an access error · G2, Capterra and GetApp pricing-aggregator listings for nTop and Physna, quote-only confirmed, no figure obtained · PR Newswire and Manufacturing Dive on Hadrian's funding · The Wire and Yahoo Finance on L&T Technology Services' AgenticIQ launch.

India-specific market, grants and programmes (R-3): ARTPARK @ IISc startups overview (browsed) · NASSCOM via Business Standard, India ER&D market size, search-reported · RoboticsCenter.ai, "State of Robotics 2026, India," search-reported, single source · MeitY draft National Strategy on Robotics, search-reported summary, PDF not opened · Deccan Herald and BioSpectrum India on Karnataka's Deep-Tech Elevate programme, search-reported · NVIDIA's own blog on the Inception Program's India robotics members, search-reported, no URL retrieved · an Investindia-style summary on GeM Startup Runway order values, search-reported, single source.

Research venues and hiring (R-3, beyond the arXiv papers listed above): Symposium on Solid and Physical Modeling / Shape Modeling International 2026, solidmodeling.org, search-reported · ASME IDETC-CIE 2026, event.asme.org, search-reported · Journal of Computing and Information Science in Engineering, "Intelligent Design 4.0," search-reported · Autodesk Research careers, `research.autodesk.com/careers`, search-reported · Zoo.dev careers, `zoo.dev/careers`, search-reported.

Everyone can generate geometry.
Almost nobody can prove it.

The field of agentic CAD split into five layers. Four are settled or crowded. One is nearly empty, and it is the one that decides whether a generated part can be trusted, machined and shipped.

Product and distribution	incumbents moved
Verification	**nearly empty**
Agent harness	crowded
Code-CAD library	settled
Geometry kernel	settled

INSIDE

Geometry from first principles, with no CAD background assumed.
Every open-source project in the field, with star counts read off GitHub.
A teardown of thirteen competitors and the lessons they teach.
Benchmarks, funding, industrial hardware, and a ranked list of what to build.

HOW ENTRIES ARE TAGGED

ADOPT use it now
STUDY read before deciding
WATCH moving target
AVOID looks relevant, is not

Numbers are marked verified where they were read from the source, and search-reported where they were not. Nothing is stated more firmly than it was found.

Samujjal Choudhury
8 SEPTEMBER 2026