

AI-Assisted Preconstruction System

Bryon Leland | Construction AI

September 2026

There are so many small and hidden items in a plan set that it is difficult to catch them all. There are discrepancies between the architectural drawings, the structural drawings, the civil, the landscaping pages, and the Title 24. The architect sends different iterations and revisions. Those small details and changes are hard to track and hard to identify, and I only have a small amount of time to review a set. That's the problem this system works on.

There wasn't a single a-ha moment where I decided to use AI for it. It was an evolution. I started by getting AI to read the plans, then added more direction to the prompts, then asked the AI to help me build better systems, until it became a set of interconnected pieces that extract elements from the drawings, read them, and evaluate the results. The short version I give people: AI has been really good at focusing all of our accumulated knowledge on the plan reading and estimating process.

What it produces

A sheet inventory, a quantity takeoff, scope organized by trade, an estimate structure with the prices left blank, an RFI log for what the plans don't say, and a record of every disagreement the readers had and how it was settled. Nothing in that set is priced. Pricing is mine.

Where I spend my time now

There are things the models can do much more accurately and faster than I can. I have shifted my focus to giving them as much of the raw data as I can, building the systems that tell the AI to display what it found in a format I can digest, and reviewing all of the deliverables.

It catches details that I would have missed on every single plan set. I only have a small amount of time to review the plans, and the AI makes that review time much more accurate. Twenty reconciliation checks run before I see anything: window and door schedules against the tags on every floor, the water heater reconciled across the energy form, the plumbing plan, the electrical load, and the gas riser, fuel source across every document, and a sweep for every utility reference with an in-or-out signal. A failed check is never quietly resolved by picking one value.

The area they still struggle with is symbol reading. Receptacles, fixtures, detectors. Those counts stay at medium confidence no matter how many models agree, and verifying them has become part of my routine.

What stays by hand

The cost estimating is still a bit of an art. Not every aspect of an estimate, as I build them, breaks down into quantifiable pieces. I still need to look at the plans as a whole and estimate what the subcontractors will charge. Demolition is a strong example. The AI has gotten much better at listing the scope that needs to be demoed, but there isn't the granularity there to price those pieces off a list. So the system owns the plan-observable inputs, counts, areas, schedules, lengths, and I own the pricing basis, the selections, and the field knowledge like duration.

How a plan set gets read

Every page is rendered once at 600 DPI, and that render is the only image the system is allowed to treat as evidence. A smaller full-sheet image exists for orientation, and it's never used for a number. From the 600 DPI

source the system cuts overlapping tiles so nothing gets lost at a seam, with a tighter grid over the dense areas like device layouts and schedules. Symbol counting is the one exception. It gets a single whole-drawing crop, because a model that's handed overlapping tiles tends to drop the symbols on the seam rather than double count them.

Renders are cached under a contract, not a name. The key is the source file's hash plus a hash of everything that affects a pixel. I added that after a cache served one project's render for another. Before any quantity work, an identity check compares the project name, address, parcel, permit number, and preparer against the cover sheet. A contradiction stops the run.

Three reading methods are used, depending on the sheet. The text layer is pulled with layout preserved, and it's the primary source for notes, code references, and specifications. It is not trusted for numbers. On one set, six of six title-block numbers in the text layer were wrong while the image was right, because a revised drawing was sitting on top of a stale text layer. A vector reader pulls schedules and printed callouts straight from the character map at exact coordinates, in targeted regions only. And a vision model, Gemini 3.6 Flash, reads the tiles with a preamble that ends by telling it to answer NOT FOUND rather than guess. Every vision call is constrained to a response schema for that sheet type, 20 schemas across 11 prompts, where every quantity field is allowed to be empty and every value carries its own confidence. If you require a number, the model will give you a number. Absence has to be legal. A scanned sheet has no usable text layer, so vision carries it end to end, and a text-only sheet like a cover or a notes page skips vision unless its text comes back garbled.

Numbers don't lock until they're verified

No area, count, span, or load gets into the output without a check. On a page with a real text layer, the value the vision model read is searched as an exact string in the character map, and if it's found it's tagged TEXT-VERIFIED with its coordinates. If it isn't found, the region is re-read on a 600 DPI tile. On a scanned page every load-bearing number is re-read on a tile before it locks. Where the text layer and the image disagree, the image wins. The text layer is there to confirm what the model saw, not to overrule the picture.

Having the models check each other

One of the things I have done since I first started using AI is to have the AI review and check its work. This is best done by a new session, or preferably by a different model. It seems to me that the models really like to show off what they have discovered, so they are really good at critically looking at the work of other agents or models.

So every quantity-bearing sheet gets a second read from a different model family, GPT-5 mini, with the same images and the same schema, and the two readings are compared field by field. Agreement never releases a number on its own. Disagreement is settled by evidence: whichever value the text layer or a targeted tile supports wins, even when it's the minority answer. If neither can be confirmed, the item carries a CONFLICT flag with both values shown. Nothing gets averaged.

The gates

Gate	What has to be true
R1 numeric lock	Every number is verified against the text layer or a 600 DPI tile before it locks

Gate	What has to be true
R2 helper scaffolding	Parallel helpers identify sheet IDs and titles only; no quantity ever comes from a helper
R3 basis triangulation	The square footage and unit count are confirmed from two independent places on the plans before the checklist runs
R4 quantity consensus	A second AI reads the plans cold and the two records are reconciled before any document is built

R2 is there because a helper reading at low resolution doesn't say it can't read. It invents a plausible rafter size. R3 is there because every downstream check keys off the square footage, so none of them can catch it when it's wrong.

The cold read

R4 is the gate the models can't pass by themselves. The producer freezes its numbers and evidence. A second AI, running in its own lane, reads the same plans without ever seeing those numbers and freezes its own. Only then are the two compared, item by item, where agreement means the same quantity, unit, sheet, and derivation, not just the same number. Two rounds of written back-and-forth settle what they can. Whatever's left comes to me, or becomes an RFI when the drawings can't answer it. No document gets built until every quantity is consensus verified, approved by me, or waiting on an RFI.

Each model has its own things that it focuses on and addresses well. By using Claude and Codex to review each other's work, I gain much higher confidence that the correct numbers, measurements, and quantities are being produced. And if the models don't know, they have gotten much better at saying so.

The public demonstrations

To show the method without anyone's private drawings, I ran two public ADU plan sets through it.

The County of San Diego Standard ADU Plan A run confirmed all 13 supplied pages, evaluated 89 selected quantity and scope targets, found 42, and recorded 47 as not present. That produced 24 work items, 19 supported by the drawings and 5 held behind one RFI, because the cover sheet referenced an energy package that wasn't in the set. Unit costs stayed blank. The Encinitas Plan B run used a larger public set with several design options and produced 142 rows: 90 ready for pricing, 12 pending RFI, and 40 reference-only rows that carry totals and duplicate schedule entries without creating anything new to price.

Estimate status	San Diego Plan A	Encinitas Plan B
Ready for review and pricing	19	90
Pending RFI	5	12
Reference only	0	40

Estimate status	San Diego Plan A	Encinitas Plan B
Total work items or rows	24	142

Source credits: County of San Diego Planning & Development Services for Plan A; City of Encinitas and plan preparer DZN Partners for Plan B. Both packages include the source plan set and every output so a reader can check the evidence.

Can you just run your plans through it?

People ask that, and it's too simple a question. Yes, you can run plans through AI readers. The real differentiating factor is what the output looks like. What do you need the AI to tell you, and will you be able to use what it produced?