Learning to Build: Stability-Aware Generative Design for Modular Robotic Assembly

Zhenghao Jin
ECE & Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
zhenghao@andrew.cmu.edu

Ruixuan Liu
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
ruixuanl@andrew.cmu.edu

Changliu Liu
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
cliu6@andrew.cmu.edu

***Abstract*— Robotic fabrication systems can only build what the design stage hands them. When a generated 3D design is realized as a modular assembly, a structure that looks correct may admit no arrangement of standardized units that stands up under its own weight—and no downstream layout solver or assembly planner can repair a design that was never buildable. We address this by making the design-to-structure step itself learnable. We train a network that maps an input mesh directly to a voxel occupancy grid, a generative counterpart to the fixed algorithmic discretization used in the classical pipeline. We then fine-tune that network with Direct Preference Optimization against a physics-based stability oracle, so that it places material where the structure carries load rather than merely where the surface lies. Finally, we evaluate on buildability rather than geometric similarity alone, sweeping the occupancy threshold to measure stability as a function of material budget. On designs the algorithmic pipeline cannot render stable, our generator reaches a 40.4% stable rate and preference fine-tuning raises it to 65.0%. On designs it does render stable, our generator reaches 63.3% and fine-tuning raises it to 74.3%. The gain concentrates under aggressive thresholding: at $\tau = 0.95$, where only the highest-confidence cells survive, the stable rate rises from 6.4% to 54.0%.**

I. Introduction

Prefabricated and modular construction builds structures from standardized units, which is what makes automated assembly tractable: a robot places discrete components rather than shaping material in place. Recent systems demonstrate the full path from a design specification to a physically constructed artifact, including bimanual robotic assembly of brick structures with hundreds of components [6].

Turning a 3D design into such an assembly is classically decomposed into two stages: discretize the geometry onto a grid, then solve for a layout of units that fills it [8], [4]. The second stage has received most of the attention, and modern layout solvers and assembly planners are strong; a parallel line of work skips the intermediate grid entirely, generating assemblies directly from a text specification [5]. The first is usually treated as a preprocessing detail—a fixed, hand-designed discretization of the geometry onto the grid.

That stage deserves more attention than it gets, because it decides buildability before any planner runs. The failure mode is documented: a 3D shape produced by a generative model may admit no design-to-assembly solution that is simultaneously within the available unit inventory and stable [6]. Algorithmic discretization optimizes for one thing, geometric coverage of the surface, and is blind to the structural consequences of the grid it emits. It will produce thin unsupported spans, isolated islands, and cantilevers with nothing beneath them, because from a coverage standpoint those cells are correct. No downstream solver can repair an occupancy grid that admits no stable assignment; by then the damage is upstream and irreversible, and the cost is paid in wasted fabrication time.

Our contributions follow from taking the first stage seriously:

1) **A learned design-to-occupancy generator.** We train a network to map an input mesh directly to a voxel occupancy grid. This is the generative counterpart to the algorithmic discretization of [8], and we treat the two as complementary: the algorithmic route is deterministic and geometry-faithful, the learned route is trainable against objectives geometry alone cannot express.
2) **Stability-aware preference fine-tuning.** We fine-tune the generator with DPO [1] using a physics-based stability analysis as the preference oracle, so the model learns which cells are structurally critical.
3) **Buildability measured as a function of material budget.** Recent assembly work reports a single buildable-rate alongside a similarity score [6]. We argue that one number hides what matters: we sweep the occupancy threshold and report the stability curve across it, which separates designs that are stable because they are dense from designs that are stable because they place material well. For a fabrication system the difference is the number of units a robot must place.

II. Method

A. From Design to Occupancy Grid

The algorithmic route [8] discretizes the mesh by a fixed rule, marking cells whose centers fall inside the surface. The rule is deterministic and independent of what the grid will be used for. Our generator instead learns the map. We sample 8,192 points with normals from the input mesh and encode them with PointNet++ [2]; a six-layer Transformer decoder [3] produces per-voxel occupancy logits on a $20 \times$

20×20 grid. Occupancy is obtained by thresholding the predicted probabilities at τ, where larger τ yields sparser structures. Because the mapping is learned, it is trainable against objectives the algorithmic rule cannot express—which is what the next subsection exploits.

B. Buildability as a Preference Signal

Stability is decided by a combinatorial solver and is not differentiable, so it cannot be a loss. But it can rank: given two candidate grids for the same design, we can ask which is buildable. For each training shape we sample candidates across thresholds, score each with the stability analysis, and rank them by a three-tier criterion—certified stability, then unstable-unit count, then deformation from the reference discretization. This yields 1,400 preference pairs with 250 held out; in 76.4% of pairs the chosen candidate is certified stable and the rejected one is not.

Since the generator emits per-voxel occupancy distributions rather than an autoregressive sequence, we take the sequence log-likelihood of [1] to be the sum of per-voxel log-probabilities of the binarized grid, and otherwise use the standard DPO loss with $\beta = 0.1$. To prevent drift on designs the base model already handles, we add an anchor term—the original training loss on a pool of 2,000 stable shapes—weighted $\lambda_{\text{anchor}} = 0.5$ against $\lambda_{\text{DPO}} = 1.0$. We train for 3 epochs at learning rate 10^{-5} with batch size 4, initializing from the base checkpoint, which also serves as the frozen reference policy.

III. Evaluation Protocol

Generative 3D models are scored against a reference shape—Chamfer distance, IoU, or a learned perceptual score. That convention encodes an assumption that fails here: that a design resembling the target is a good design. An assembly that reproduces the input geometry faithfully and collapses under its own weight is not a partial success; for a robot it is a failed build.

We propose scoring two axes jointly. **Buildability**, which this paper reports as its primary result: units are assigned to the occupancy grid, and the assembly is evaluated with the stability analysis of [4], solved with Gurobi [9]. The analysis computes a force distribution satisfying static equilibrium and identifies units that fail—those for which no equilibrium exists, or for which the required friction exceeds the capacity of their connections. A structure is buildable only if no unit fails. We collapse this to a binary verdict and report the *stable rate*, the fraction of (design, τ) tasks certified stable. **Fidelity**: alongside geometric deformation from the reference discretization, we score perceptual correspondence with a locally hosted Qwen2-VL-7B-Instruct [10], comparing renders of the assembly against that reference.

Neither axis is sufficient alone—filling the grid solid is stable, and ignoring stability maximizes fidelity. Reporting stability at a single operating point is likewise insufficient, since it cannot distinguish the two. The threshold sweep is what makes the axis diagnostic, and it separates models whose structures survive only when densely filled from models whose confidence is concentrated where the structure carries load.

TABLE I
STABLE RATE (%) BY OCCUPANCY THRESHOLD τ

τ	Base	+DPO	Δ
Matched-stable subset ($n = 1{,}650$; algorithmic: 100%)			
0.55	89.7	80.6	−9.1
0.70	81.7	79.6	−2.2
0.85	61.8	74.3	+12.5
0.95	19.9	62.6	+42.6
All	**63.3**	**74.3**	**+11.0**
Unstable subset, held out ($n = 250$; algorithmic: 0%)			
0.55	70.0	70.4	+0.4
0.70	57.6	71.6	+14.0
0.85	27.6	64.0	+36.4
0.95	6.4	54.0	+47.6
All	**40.4**	**65.0**	**+24.6**

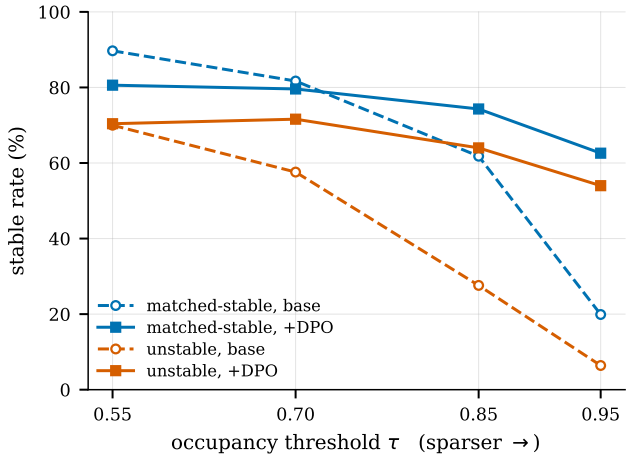

Fig. 1. Stable rate against occupancy threshold τ. The baseline (dashed) degrades sharply as the material budget tightens, while the fine-tuned model (solid) stays comparatively flat; the same crossover appears independently in both subsets.

IV. Experiments

We evaluate on two disjoint subsets of ShapeNet [7] shapes at thresholds $\tau \in \{0.55, 0.70, 0.85, 0.95\}$. Both are defined by the outcome of the algorithmic conversion [8]. The *unstable* subset holds designs the algorithmic pipeline cannot render stable, so its rate there is zero by construction and any success of ours is a design recovered from outright failure. The *matched-stable* subset holds designs it does render stable, where it sits at 100%; this is the stricter test for us. The learned map alone already changes what is buildable: before any fine-tuning, our generator makes 70.0% of the unstable subset stable at $\tau = 0.55$, on designs the algorithmic rule cannot render stable at any setting. We report the held-out designs only. On the 1,400 training shapes the same evaluation gives $42.2\% \rightarrow 66.5\%$ (+24.3), within 0.3 points of the held-out gain, so the model has learned transferable placement rather than memorized layouts.

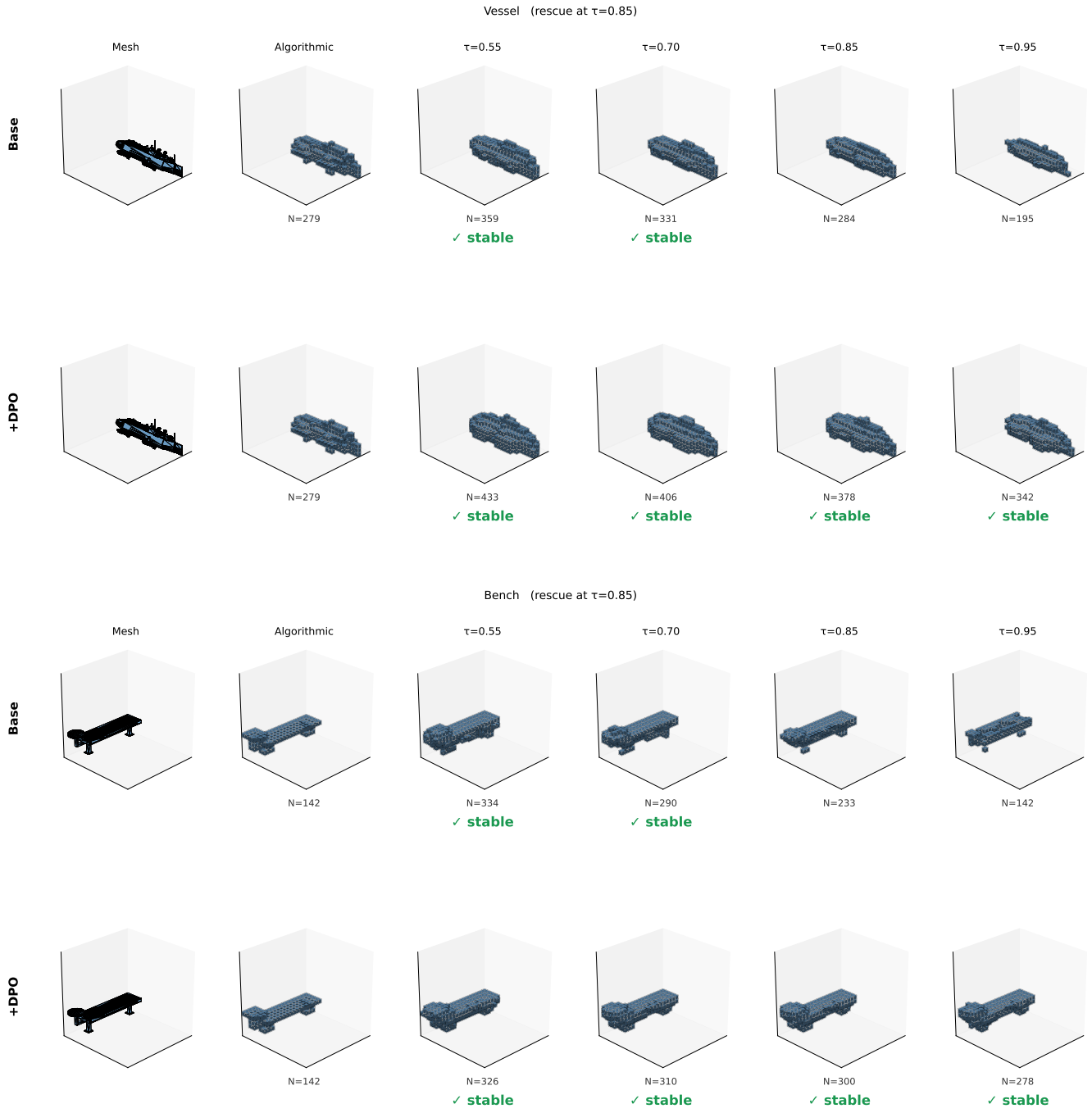

Fig. 2. From input geometry to assembly. The algorithmic route discretizes for surface coverage; at a strict threshold level the base model's grid admits no stable assignment, while the preference-tuned model places material where the structure carries load.

On the unstable subset the stable rate rises from 40.4% to 65.0% (+24.6), recovering two in five of the designs the base model still fails. On the matched-stable subset it rises from 63.3% to 74.3% (+11.0), so the gain is not purchased by sacrificing designs that already worked. We note plainly that on this second subset the algorithmic route remains ahead of us by construction; the two methods recover different designs, which is the sense in which we regard them as complementary rather than competing. Aggregated over the full dataset the algorithmic route remains the stronger converter overall; what it cannot do is recover the designs where it fails outright, which is the regime our method addresses.

The per-threshold breakdown (Fig. 1) explains both with one mechanism. At low τ the grid is dense, the base model is near ceiling, and fine-tuning costs a few points. At high τ the grid is sparse and material is scarce: the base model collapses to 19.9% on matched-stable designs and 6.4% on unstable ones, while the fine-tuned model holds 62.6% and 54.0% respectively. The mechanism is not a smaller material budget but a sharper one. Between $\tau = 0.55$ and $\tau = 0.95$

the base model sheds roughly half its occupied cells and its structures fail, while the fine-tuned model sheds far fewer: its confidence has concentrated on a structurally sufficient core, so aggressive thresholding no longer removes load-bearing cells. Under a tight unit budget that distinction is the whole difference between standing and collapsing. Fig. 2 shows this on two shapes: at τ = 0.85 the base grid admits no stable assignment while the tuned one holds.

On the fidelity axis, similarity to the reference discretization is nearly unchanged. On the unstable subset the fine-tuned model scores 3.38, 3.42 and 3.48 against the base model's 3.50, 3.57 and 3.53 at $\tau = 0.55$, 0.70 and 0.85 (0–5 scale, $n = 250$). The small deficit narrows as τ rises, so the regime where the stability gain is largest is also where the fidelity cost is smallest: at $\tau = 0.85$ the fine-tuned model gains 36.4 points of stability for 0.05 points of similarity.

V. Discussion and Limitations

Making the discretization learnable is what enables everything after it: once the grid is produced by a network, any criterion that can rank two candidates—stability, unit count, assembly sequenceability, robot reachability—can shape it, without a differentiable relaxation. That is the general form of the method, and brick assemblies are one instance.

For robotic fabrication the practical consequence is a shift in where infeasibility is detected. A design certified buildable before fabrication begins cannot strand a robot mid-assembly, and the threshold sweep gives a direct handle on material efficiency: our fine-tuned model remains buildable under aggressive thresholding, where the baseline's structures collapse. We use brick assemblies as a controlled testbed because they make the unit inventory and the stability oracle explicit, but the method assumes only a discrete unit inventory and a ranking oracle—conditions shared by modular and prefabricated construction.

Three limits bound the claim. The stability analysis is static and does not model the assembly process, so a certified structure may still defeat a robot that must place units in sequence; coupling the preference signal to an assembly-sequence feasibility check of [6] is the natural next step. The preference signal is binary, so a marginally stable structure and a robust one are indistinguishable to the objective, even though the analysis already computes a continuous per-unit margin. And we evaluate at a single grid resolution on a single dataset of human-authored meshes; whether structurally critical placement transfers to finer grids, or to designs from generative models, is open.

References

[1] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Adv. Neural Inf. Process. Syst.*, 2023.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Adv. Neural Inf. Process. Syst.*, 2017.

[3] A. Vaswani *et al.*, "Attention is all you need," in *Adv. Neural Inf. Process. Syst.*, 2017.

[4] R. Liu, K. Deng, Z. Wang, and C. Liu, "StableLego: Stability analysis of block stacking assembly," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 9383–9390, 2024.

[5] A. Pun, K. Deng, R. Liu, D. Ramanan, C. Liu, and J.-Y. Zhu, "Generating physically stable and buildable brick structures from text," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2025.

[6] R. Liu, P. Huang, A. Pun, K. Deng, S. Aggarwal, K. Tang, M. Liu, D. Ramanan, J.-Y. Zhu, J. Li, and C. Liu, "Prompt-to-Product: Generative assembly via bimanual manipulation," arXiv:2508.21063, 2025.

[7] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," arXiv:1512.03012, 2015.

[8] S.-J. Luo, Y. Yue, C.-K. Huang, Y.-H. Chung, S. Imai, T. Nishita, and B.-Y. Chen, "Legolization: Optimizing LEGO designs," *ACM Trans. Graph.*, vol. 34, no. 6, 2015.

[9] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024.

[10] P. Wang *et al.*, "Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution," arXiv:2409.12191, 2024.