A WORLD MODEL FOR ASSEMBLY SEQUENCE PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes a world model for planning assembly sequences for LEGO structures. The model predicts placement feasibility by looking ahead in a learned latent space, without invoking a physical simulator inside the look-ahead search. The primary technical challenge lies in judging whether committing to a placement leaves the remaining structure buildable, which requires reasoning over future steps. To address this, we represent the assembly with a multi-granularity heterogeneous graph, in which a master node aggregates the overall geometry of the assembled structure as a set of anchor tokens, while a brick-level sub-graph carries the interlocking topology and the static attributes of each brick. A latent dynamics model advances the anchor tokens locally after each placement, which lets the world model roll the geometry forward over several steps and anticipate whether a candidate placement obstructs the assembly of the remaining bricks. We validate the framework on held-out test structures from the StableText2Lego (Pun et al., 2025) benchmark, comparing it against ablations and recent assembly-sequence methods. Our planner achieves near-oracle solutions while issuing the fewest exact-oracle calls per brick and avoiding the later-infeasible placements incurred by the baselines.

1 INTRODUCTION

An essential task for embodied AI and robotics is automated assembly: enabling one or more robots to compose a set of individual parts into a target structure. The key to this problem is *assembly sequence planning* (ASP), determining the order in which parts are placed, as this order decides whether the assembly is physically realizable (Tian et al., 2024). We study ASP on brick assembly, a reproducible testbed with readily available components and existing benchmarks; brick assembly is easily scaled to hundreds of parts and exposes the core physical constraints of other assembly problems, such as interlocking connectivity, per-step stability, and collision-free reachability (Liu et al., 2025; Chung et al., 2021; Pun et al., 2025).

A core challenge in ASP is the need to *look ahead* over future steps (Shu et al., 2025), as a placement that looks feasible now may leave the remaining structure unbuildable later. Existing methods either ignore look-ahead and place parts greedily one step at a time (Liu et al., 2025), or obtain look-ahead by querying physical ground truth through time-consuming, unscalable simulation and motion planning (Shu et al., 2025; Atad et al., 2023). This leaves a gap in designing a feasibility judge that can look ahead over many steps without paying the per-step cost of a physical oracle.

We address this gap with a *manipulation-level world model* and solve the brick assembly sequence problem by assembly-by-disassembly, recursively removing bricks from the completed structure to obtain a feasible order (Figure 1). We train the world model to predict whether a placement is feasible and to imagine how that feasibility evolves as further bricks are removed, so that the planner can look many steps ahead before committing to a brick. To keep this prediction faithful, the model represents the evolving assembly as a multi-granularity heterogeneous graph, reasoning jointly about the global geometry that governs collisions and the local connectivity that governs stability. Concretely, two learned predicates read the overall geometry and inner brick structure and two deterministic gates handle the structure-independent reachability and press checks, whereas an encoder summarizes the built geometry into a set of latents and a dynamics model advances the latents after each rollout. The planner rolls this world model forward to score candidates and commit a removal order, which we detail in Section 3.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

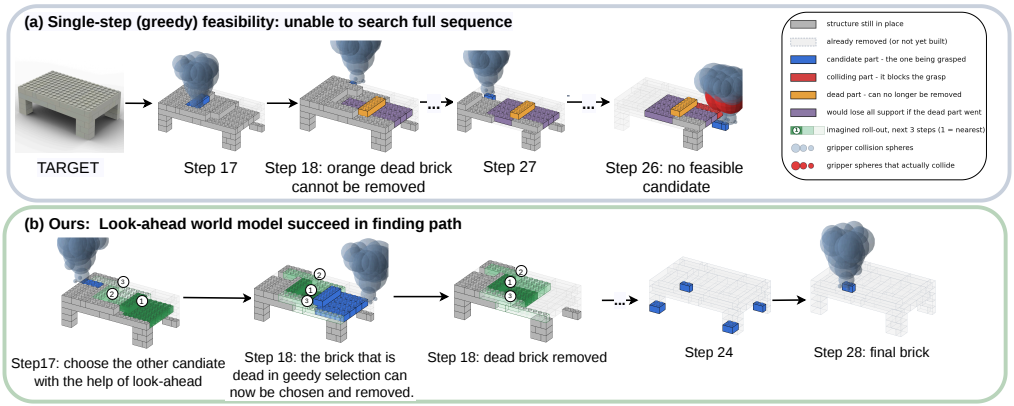

Figure 1: A LEGO structure with a dead-end assembly order(upper) vs a correct one(lower). Our world model succeed by rolling feasibility forward and avoid committing an early candidate disassembly(blue candidate in upper row) which would blocks a later one(orange dead brick). The spheres in the figure denote the gripper's collision geometry.

On the StableText2Lego benchmark, this look-ahead raises buildable-assembly success to 95.4%, above greedy ordering (83.6%) and even above a single-step exact-oracle predicate (92.0%), while reducing commit-time oracle calls by more than half. Under a shared exact static-stability ruler in a single-brick action space, it outperforms recent assembly-sequence methods (WorkBenchMark (Ma et al., 2026) and ASAP (Tian et al., 2024)) by a wide margin. These results show that a learned world model can provide multi-step feasibility look-ahead at oracle-level quality without the per-step cost that makes exact look-ahead intractable.

In summary, we contribute: 1) a manipulation-level world model to imagine and evaluate assembly steps; 2) a multi-granularity heterogeneous graph structured model serving a collision judge and a stability judge that operate at different structural levels; 3) a brick assembly sequence planner that outperforms prior assembly-sequence methods and ablations under a shared physical ruler.

2 Related Work

Assembly Sequence Planning. Assembly sequence planning determines the placement order that makes a structure physically realizable, and prior work spans a spectrum from greedy ordering to look-ahead that relies on an exact-physics oracle. At one end, methods solve for an order while encoding physics only as a deterministic mask or a single-step validity check: placing bricks under a data-free action mask (Liu et al., 2025), filtering an indefinite action space with a learned validity predictor (Chung et al., 2021; Chen & Liu, 2024), imitating ground-truth orders with graph transformers (Ma et al., 2023; Guo et al., 2024), masking illegal actions (Oh et al., 2025), or solving an optimal sequential decision with dynamic programming or reinforcement learning (Nagpal & Mehr, 2024). Robotic-building methods learn placement policies (Wang et al., 2026) or reason over sequence and contact jointly (Zhu et al., 2024; Lai et al., 2026), yet treat gripper reachability as a downstream check or future work. Others predict single-step feasibility directly, such as directional grasp feasibility (Bouhsain et al., 2025) or collision and clearance surrogates (Das & Yip, 2020; Kew et al., 2020), but these are static and reversible and never roll feasibility forward over the irreversible multi-step process we address. Moving toward look-ahead, another line evaluates each candidate against physical ground truth at deployment through simulation and motion planning (Shu et al., 2025; Atad et al., 2023; Tian et al., 2024), which is exact but pays a per-step oracle cost that scales poorly. When focusing on brick assembly tasks specifically, prior systems are largely confined to generation (Pun et al., 2025; Xu et al., 2026) or to deterministic reconstruction planners that check reachability by explicit collision testing (Barghi et al., 2024; Ma et al., 2026; Huang et al., 2025), none performing learned multi-step feasibility look-ahead. We address this gap by learning the physical feasibility judgment and rolling it forward in latent space, so the planner anticipates multi-step consequences without paying a per-step oracle cost at inference.

World Models. Learning a model to replace an expensive physical simulator goes back to learned interaction engines and graph-network simulators (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020; Ha & Schmidhuber, 2018), which latent world models extend to multi-step imagination for

planning (Hafner et al., 2019; 2023; Hansen et al., 2024; Micheli et al., 2023). Their look-ahead, however, targets a task reward or visual future rather than physical feasibility, and generic latent rollouts do not by themselves preserve physical validity (Terver et al., 2026; Cui & Ma, 2026). Object-centric and graph-structured world models make look-ahead compositional (Kipf et al., 2020; Feng et al., 2025; Vakhitov et al., 2026), but the graph-world-model survey's physical role collects only generic collision, friction, and particle models, none addressing assembly legality (Liu et al., 2026a). Recent methods that use a world model as a critic over candidate actions did not consider physical feasibility as well (Qi et al., 2026; Liu et al., 2026b); we share this critic mechanism but predict physical placement feasibility rather than a task reward or a rendered future, which is precisely the physical validity that generic latent rollouts fail to preserve.

Graph Representation and Stability Prediction. Our world model couples a geometry encoder over a multi-granularity heterogeneous graph with a learned stability predictor. The anchor-token encoder follows the point-proxy design of PoinTr (Yu et al., 2021), and its multi-level structure builds on hierarchical graph pooling and heterogeneous graph networks (Ying et al., 2018; Hu et al., 2020). The stability predictor is a learned structural judge, an ability with ample precedent, from predicting stability from appearance (Li et al., 2017) to regressing structural response with graph networks (Cai & Jelovica, 2023; Chang et al., 2026). We adopt these representations for a different role: rather than encoding a shape for reconstruction or scoring stability in isolation, our multi-granularity graph lets a single world model reason jointly about global collision geometry (through the master-node anchor latents) and local interlocking connectivity (through the brick sub-graph), and exposes stability as a readout that the planner queries at every look-ahead step.

3 Assembly Planning with a World Model

3.1 Problem Formulation

We study assembly sequence planning for brick assembly: given a target structure, we must order its bricks so that every placement is physically executable by an embodiment and leaves a stable structure. We solve this with a learned world model that predicts placement feasibility and looks ahead over multiple steps. We first formalize what makes a step feasible. The input is a target structure specified as a set of M axis-aligned bricks $B = \{b_1, \ldots, b_M\}$ on an integer stud grid, where each brick is given by its footprint dimensions and its discrete position. The output is an ordered assembly sequence: a list of steps, each of which places a single brick or a rigidly-coupled group of bricks (of size $K \leq 5$) onto the partially-built structure, using a specified manipulation motion and a designated gripped brick. The placements are executed by a dual-arm embodiment (APEX-MR (Huang et al., 2025)). At any step we write b_{built} for the aggregate of already-placed bricks and b_{active} for the brick or group added in that step.

A step is admissible only if it can be physically executed and leaves a stable structure. We describe a step by an action $a = (b_{\text{active}}, m, b_{\text{grip}})$, where b_{active} is the brick or rigidly-coupled group being moved, m is the manipulation motion (top-down placement or bottom-up insertion), and b_{grip} is the brick within the group that is in direct contact with the gripper. The action is applied to the current aggregate b_{built}. We formalize feasibility as a conjunction of five predicates: an action a is feasible if and only if

$$\begin{aligned}\text{feasible}(a) \iff & \underbrace{\text{press}(a) \wedge \text{reach}(a)}_{\text{deterministic gates}} \wedge \underbrace{\text{collision}(a)}_{\text{learned collision}} \\ & \wedge \underbrace{\text{stable}(b_{\text{built}}) \wedge \text{stable}(b_{\text{built}} \setminus b_{\text{active}})}_{\text{learned stability}}.\end{aligned} \tag{1}$$

Each action expands into a finite set of discrete gripper configurations that realize it, differing in the arm assignment, the approach side, and the tool offset. Manipulation dependent predicates quantify existentially over this set: $\text{press}(a)$ holds if there exists a configuration whose gripper force point, the stud through which force is applied, lies within the admissible bounds; $\text{reach}(a)$ when some configuration admits inverse-kinematics solutions for its motion waypoints; and $\text{collision}(a)$ when some configuration admits a collision-free placement against the assembled structure. The stability predicate $\text{stable}(\cdot)$ requires the aggregate to remain stable under gravity, checked both before and after b_{active} is detached so that the structure is stable throughout. The first two predicates are structure-independent, depending only on the action and its target pose, whereas the latter three

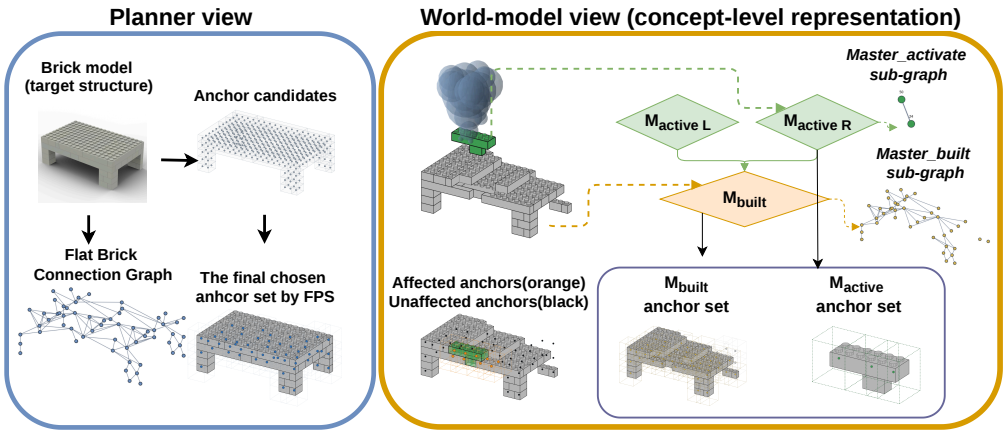

Figure 2: **Structural abstraction:** flat brick connection graph and anchor set selection (planner view) vs. hierarchical heterogeneous latent graph and anchor tokens renewing (world-model view).

depend on the assembled geometry. Our objective is to find a disassembly sequence in which every removal step is feasible, so that the reversed order is a feasible assembly sequence.

3.2 Structural Abstraction

A brick structure admits a natural graph abstraction in which each node is a brick and each edge encodes a stud-level interlock between two vertically adjacent bricks. We build our system on this abstraction while separating the *planner* from the *world model*, each operating on its own view of the graph. The planner reasons over a single flat brick connection graph and solves the assembly order by *disassembly*: starting from the complete target and repeatedly removing one brick or rigidly-coupled group at a time until nothing remains, after which the reversed removal order gives a valid assembly sequence. Planning in this direction is what makes the problem tractable, since it constrains the search to orders that reach the known target and lets each step reason about detaching from a fully specified structure rather than searching forward over exponentially many partial builds (Shu et al., 2025). The world model, in contrast, maintains a hierarchical heterogeneous graph, in which a master node summarizes the assembled structure while a brick-level sub-graph is attached beneath it.

Since the stability predicates depend only on the interlocking topology among the placed bricks, whereas collision-freeness depends only on the exposed surface geometry of the assembled aggregate, the heads that predict feasibility in Section 3.1 consume information at different structural levels. We therefore give the world model a hierarchical heterogeneous graph: a master node M_{built} aggregates the overall geometry of the assembled aggregate b_{built} into a geometric latent, while a brick-level sub-graph beneath it carries the interlocking topology together with a per-brick attribute vector (Figure 2). Latent geometry avoids tracking the full pose of every brick throughout a rollout. The attribute vector is an 11-dimensional summary of each brick's footprint, mass, and current support relations within the structure (its height, and the number of bricks it rests on and supports); the same vector is shared by the encoder, the dynamics model, and the stability predictor, and we give its full specification in supplementary material.

As shown in Figure 2, the geometric latent carried by the master node is not a single vector but a set of *anchor tokens*, attached to a fixed spatial location within the structure. The anchors are generated by farthest-point sampling over all occupied cells of the target structure, and the anchor set remains fixed throughout the assembly. Each anchor carries a token summarizing the geometry of its local neighborhood. We describe each cell in that neighborhood by a *cell token*: whether it is occupied, which of its six faces are exposed, and the attribute vector of the brick occupying it. These anchor tokens are produced by the encoder and rolled forward as the geometric state of the world model.

3.3 World-Model Architecture

The world model realizes the five predicates of Section 3.1 (Figure 3, left). Two of them, $\text{press}(a)$ and $\text{reach}(a)$, are deterministic gates over structure-independent geometry, while $\text{collision}(a)$ and the two stability terms are learned. We describe the encoder and the dynamics model first, then the two gates, and finally the two learned predicates collision and stable.

Encoder. The encoder maps each anchor's neighborhood to a latent representation. After each committed removal, the imagined state is replaced by the actual structure and the latents are recomputed. During look-ahead rollouts between commits, the dynamics model below instead updates the latents incrementally without re-encoding the structure. For anchor i, it computes $z_i = \mathrm{Enc}(\mathbf{c}_i)$ from the cell tokens $\mathbf{c}_i$ in the cube centered at i (Section 3.2), by embedding each cell token, applying local 3D convolutions over the cube, and pooling them with a mask, so that z_i summarizes only the geometry in its own neighborhood. The encoder is trained self-supervised. The training signal comes from an MLP decoder discarded when inferring. From each cell token the decoder Dec must recover the cell's neighborhood occupancy o_i: $\mathcal{L}_{\mathrm{enc}} = \mathrm{BCE}\big(\mathrm{Dec}(z_i),\, o_i\big)$. The encoder is then frozen when training other heads.

Dynamics. During a rollout we do not re-encode the whole structure after every placement. Instead, a dynamics model renews only the latents of the anchors near the placement. Writing z_i for the latent of anchor i and z_i' for its value after the update, it applies a gated residual update

$$z_i' = z_i + \mathbb{1}[i \in \mathcal{A}(a)] \cdot \mathrm{MLP}\big(\,[\,z_i,\ \psi_i(a)\,]\,\big), \tag{2}$$

where $\mathcal{A}(a)$ is the set of anchors whose neighborhood is geometrically affected by action a, $\mathbb{1}[\cdot]$ is the indicator that gates the update to those anchors, $[\,\cdot\,,\,\cdot\,]$ denotes concatenation, and $\psi_i(a)$ is a fixed featurization of the placed brick relative to anchor i: the brick's occupancy footprint within that anchor's cube concatenated with its attribute vector. The final MLP layer is zero-initialized, and is trained self-supervised on $\mathcal{A}(a)$: each updated latent z_i' must decode to the true next occupancy o_i^+ and stay close to the true next-state latent $\bar{z}_i$ through a relative term with $\lambda_z{=}10$ and sg stop-gradient applied to $\bar{z}_i$.

$$\mathcal{L}_{\mathrm{dyn}} = \mathrm{BCE}\big(\mathrm{Dec}(z_i'),\, o_i^+\big) + \lambda_z \frac{\|z_i' - \mathrm{sg}(\bar{z}_i)\|^2}{\|\mathrm{sg}(\bar{z}_i)\|^2}, \quad i \in \mathcal{A}(a), \tag{3}$$

Deterministic gates. The two gates are training-free exact checks that depend only on the action and its target pose, not on the assembled structure, so they are evaluated first and filter out candidates cheaply. Because the motion waypoints are given in closed form over a finite pose domain, we precompute $\mathrm{reach}(a)$, the existence of inverse-kinematics solutions for those waypoints, into an exact lookup table answered in a single query. $\mathrm{press}(a)$ analytically checks that the press point, the stud through which the gripper applies the assembly force, lies within its admissible bounds.

Collision. The $\mathrm{collision}$ predicate predicts whether placing the candidate group b_{active} onto b_{built} can be carried out without geometric collision. As an action expands into a finite set of discrete gripper configurations $\mathcal{G}(a)$ (Section 3.1), the head scores the whole set in a single forward pass: for each configuration $g \in \mathcal{G}(a)$ it outputs a feasibility logit ℓ_g and, as an auxiliary output, a signed distance d_g between the moved group and the structure.

We instantiate $\mathrm{collision}$ as an edge biased cross attention module. Collision depends on how the placed group and the arm sweeping it into position interact with the exposed geometry of the assembled structure. We therefore treat each candidate gripper configuration as a query and let it attend to the anchor latents of M_{built} and M_{active}, which serve as keys and values. This allows each configuration to focus on the small subset of anchors most likely to contend for the same space. We augment the keys and values with a learnable null token, ensuring that attention remains well defined when M_{built} contains few or no anchors. Because the relative position of an anchor with respect to the placed group provides a strong geometric prior for contact, we inject this information explicitly as a per anchor bias on the attention logits rather than requiring the attention module to recover it from the latent representations alone. Importantly, this bias is computed once per anchor, rather than separately for every configuration and anchor pair. Finally, the two motion modes induce qualitatively different collision geometries, namely top-down placement and bottom-up insertion, so we condition the module on the motion m using FiLM 3.1. The collision module is trained on top of the frozen encoder. The model is trained supervisedly using an exact collision oracle 3.4 to generate ground truth signed distance $\hat{d}_g$ and logit y_g with $\lambda_d{=}0.3$:

$$\mathcal{L}_{\mathrm{col}} = \mathrm{BCE}(\ell_g,\, y_g) \;+\; \lambda_d\, \mathrm{Huber}_{\delta=1}(d_g,\, \hat{d}_g). \tag{4}$$

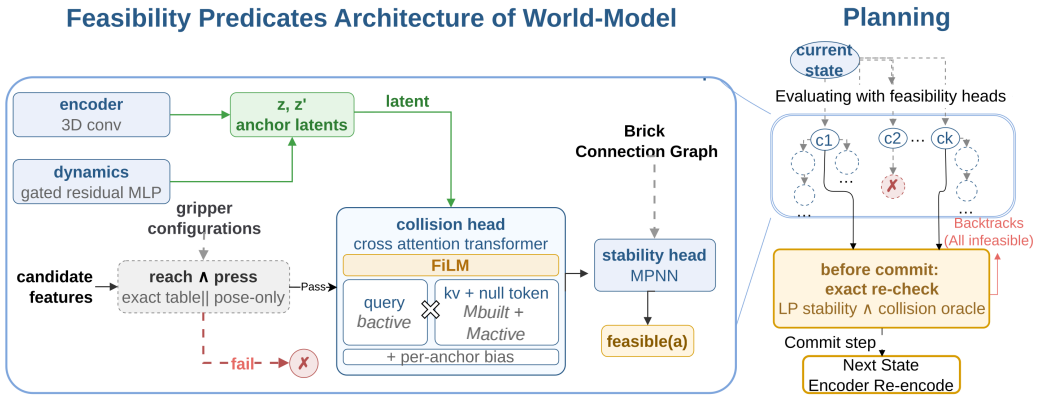

Figure 3: **Feasibility predictor** of world model (left) and planner (right). Left: a candidate passes through the feasibility predicates; blue blocks are learned modules, the dashed block is the deterministic reach,∧,press gate, and the green arrow marks the anchor latents, read only by the collision head. Right: one step of the disassembly search, with solid arrows for committed steps and dashed arrows for the imagined depth-d rollout; a red cross marks a first-layer dead end, and the best-scoring candidate is exactly re-checked before commit. *LP* denotes the exact linear-program stability check applied at commit time.

Stability. The stability predicate evaluates $\mathrm{stable}(b_{\text{built}})$ and $\mathrm{stable}(b_{\text{built}} \setminus b_{\text{active}})$, requiring the sub-assembly to stand both before and after the group is detached. Unlike collision, the stability head does not read the anchor latents; it is a message-passing neural network (MPNN) that operates directly on the brick adjacency graph, predicting a per-brick instability logit s_b, which is passed through a sigmoid to output an instability probability, and we declare a state stable when every brick's predicted instability probability stays below a calibrated threshold τ. We build the training dataset from the StableLego brick assemblies with per-brick stability labels (Liu et al., 2024). Since the structures in the data are complete, their unstable bricks are mostly unsupported, whereas our disassembly planner more often faces overloaded studs and sparse remnants that tip over. We therefore add self-generated partial structures by repeatedly stripping bricks from complete brick assemblies while keeping all remaining bricks in structures connected to other bricks or the ground. We then form a supplementary set of unstable remnants and label them by an exact stability solver. The stability head is trained jointly on the combined dataset with labeled ground-truth stability logit y_b: $\mathcal{L}_{\text{stab}} = \mathrm{BCE}_w(s_b, y_b)$.

3.4 Planning with the World Model

We plan in the disassembly direction (Figure 3, right): starting from the complete target structure, at each step we enumerate the candidates that can currently be removed (a single brick or a rigidly-coupled group), filter them with the feasibility predicates of Section 3.3, and commit one removal before repeating until the structure is empty; executing the committed removals in reverse yields the assembly sequence. Among the candidates that pass the predicates, the planner commits the highest-scoring b_{active} that is not a dead end, where the score is produced by the look-ahead below and ties are broken by a deterministic candidate order. If a state has no committable candidate, the planner backtracks to the previous commit and tries its next-best candidate.

For each passing candidate, the world model performs a depth-d look-ahead in the latent space, a layered rollout advanced by the dynamics model. The roots of the rollout are the currently feasible candidates of the current state; at each subsequent layer, every imagined placement is expanded only along the bricks geometrically adjacent to the group just removed, since removing a group can only alter the feasibility of its immediate neighbors. All placements within a layer are scored with a single forward pass of the collision head, and placements that reach the same state through different orders are merged, which keeps the number of candidates evaluated per step in the single digits. The score of a root is the total number of feasible continuations exposed across its whole subtree: the more feasible continuations a candidate opens up, the higher its score. A candidate is a dead end if its successor state is non-empty yet exposes no feasible continuation at all. A dead end at the first layer vetoes the root directly, whereas at deeper layers an infeasible branch simply contributes nothing to the score rather than ruling the root out.

Before a candidate is committed, the selected action is re-checked by two exact solvers. The stability check is a linear-program relaxation of the StableLego force-equilibrium analysis (Liu et al., 2024), which is provably sound and matches the original verdicts under a per-instance certificate (more details in supplementary material). The collision oracle verifies the placement through analytic inverse-kinematics and geometric collision computation; to speed it up, the collision head's per-placement scores from the look-ahead are reused so that the oracle first examines the placements the head ranks most feasible. A candidate that fails either check is recorded as a false pass and skipped, so every step of the returned assembly sequence is exactly verified.

4 Experiments

The goal of our experiments is to verify whether 1) look-ahead brings a measurable benefit over greedy single-step planning; 2) our world-model based planner with look-ahead is strong both in ablations and in comparison with state-of-the-art baselines; and 3) our collision head needs to read the full geometry latent anchor set.

4.1 Datasets and Setup

Data and representation. We evaluate on the StableText2Lego corpus (Liu et al., 2024), a collection of stud-level LEGO structures spanning 21 object categories. Every structure is built from a library of *eight brick sizes* (1×1, 1×2, 1×4, 1×6, 1×8, 2×2, 2×4, 2×6); all bricks are axis-aligned rectangular cuboids. Each object may appear as several distinct *structures*, where a same shape comes with alternative brick layouts. We apply disassembly sequence planning on these structures.

We split the corpus at the *object* level: $24{,}711$ objects for training and 697 for validation. For evaluation we draw a fixed, stratified sample of $1{,}200$ structures from the official held-out test split ($2{,}851$ objects, $4{,}785$ structures), preserving the split's category and structure-size distribution.

Embodiment. All planning runs use a single, fixed embodiment: the dual-arm LEGO workcell distributed with VAMP-MR (Huang et al., 2026) (its `dual_gp4` environment), i.e. two 6-DoF Yaskawa Motoman GP4 arms, each carrying that repository's LEGO end-of-arm tool (a custom LEGO gripper with a wrist-mounted RGB-D camera). Robot geometry is the sphere-based collision model shipped with the same environment.

Exact references. Two exact, deterministic components provide ground truth. (i) An *exact collision oracle*: closed-form analytic inverse kinematics plus sphere-based geometric collision checking over the manipulation waypoint chain. (ii) *Exact statics*: the StableLego force-equilibrium program (Liu et al., 2024), which we solve with **HiGHS** (Huangfu & Hall, 2018) as a linear program, whose tightness we verify per instance with a complementarity certificate (experiment result in supplementary material). The collision oracle serves as the label source for the learned collision head, while exact statics provides the labels for training the stability head (Section 3.3). Both components serve as the commit-time re-check inside planning (Section 3).

4.2 Metrics

Metrics of full method performance:

- **Succ.** (success rate): Fraction of evaluable structures whose recorded sequence is feasible at every step and removes all bricks: each step must satisfy $\mathrm{feasible}(a)$ (Eq. 1) instantiated with the *exact* collision oracle and *exact* statics; any infeasible step or incomplete disassembly is a failure. The criterion is identical across methods, k is the number of structures found a feasible disassembly sequence and n is the number of total evaluable structures.
- **Orc/br.** (oracle calls per brick): The median number of per-structure exact collision-oracle calls a method spends at commit time, per brick. This is a measure of how much exact checking a method needs.
- **Imagined/br.** (imagined states per brick): The median number of per-structure latent look-ahead states the method advances per brick. This is a measure of search effort.
- **Wall-clock.** The median per-structure wall-clock time(s).
- **McNemar.** As methods time out on different structures, success is compared on paired subsets, using the exact-binomial McNemar test. We report the true p-value or $p < 10^{-k}$.

Metric of collision head analysis:

- **FP@R99.** False-positive rate at 99% recall: at the threshold that still accepts 99% of the feasible cells, the fraction of infeasible cells it lets through; lower is better.

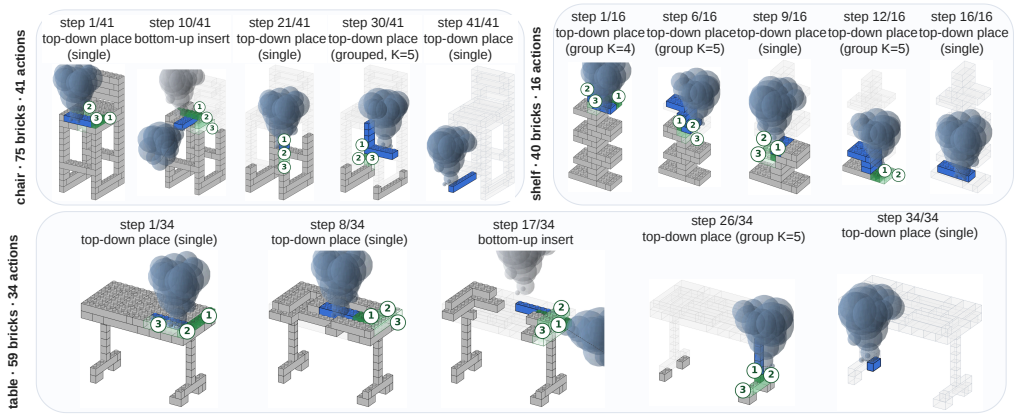

Figure 4: **Qualitative Results.** Qualitative disassembly galleries for the three models. In each keyframe the current candidate brick is blue and the look-ahead bricks are green, with darker shades marking deeper look-ahead steps. For bottom-up insertion the holding arm is shown in steel-blue and the top-pressing arm that stabilizes the sub-assembly in light-gray.

Table 1: **Look-ahead depth ablation analysis.** McNemar compares each row with the row above. The first-to-last $0 \rightarrow 3$ is significant (p=0.0027). 3^* is the re-sampled result on the subset evaluable against d=5 (n=327), and it shares the same configuration as the 3 row (n=1109). "—" = not applicable / not recorded.

Depth	n	Succ.	McNemar	Orc/br.	Imagined/br.	Wall-clock (s)
0	1109	0.805	—	1.26	0	43
1	1109	0.796	p=0.44	0.48	7	23
2	1109	0.819	p=0.017	0.47	33	54
3 (ours)	1109	0.839	p=0.023	0.50	130	137
3^*	327	0.844	—	0.56	74	47
5	327	0.832	p=0.50	0.61	—	371

4.3 Look-Ahead Evaluation

We isolate the effect of look-ahead by fixing the predicate and the planner and varying only the roll-out depth d (Table 1), which shows that a greedy single-step planner is not sufficient. Success improves with depth. The success rate dips from d=0 to d=1, but p-value shows this difference is indistinguishable. From d=2 onward look-ahead both improves success rate and reduces the commit-time exact-oracle calls per brick. Its cost is computation and wall-clock time, which grow with the number of imagined states expanded per brick. Increasing the depth further to d=5 yields no significant gain over d=3. The result indicates that d=1 cannot see the traps, which need at least two steps to expose. The gain plateaues at d=3, which we adopt for our full-method cross-comparison.

4.4 Cross-Method Comparison

We study whether the world model can act as the feasibility predicate during planning without losing solution quality. Under an identical commit-time verifier and look-ahead depth, we build a ladder of methods, each adding new components over the previous one, named as PREDICATE USED/WHETHER USING ROLLOUT (Table 2, left): no feasibility predicate and no rollout, relying entirely on per-step commit-time verification (*seq/none*); a reachability gate without rollout (*reach/none*); a reachability gate with look-ahead (*reach/rollout*); and finally our full method, the learned collision and stability heads paired with rollout (*wm/rollout*). As the Succ. and McNemar p-values in the table show, our method succeeds on most structures, and the improvement is significant compared to reach/rollout, indicating that adding look-ahead and the learned predicates each gives a significant paired improvement. Our method also requires the fewest commit-time verifier calls per brick (Orc/br.), showing that its brick selection is accurate and relies little on backtracking to recover. For reference we also report an exact-oracle predicate (*oracle/none*), whose verdicts are exact but which cannot afford look-ahead: rolling it forward times out on 26 of 75 structures (worst case 48,315s against a 3,600s budget), the very cost the world model removes.

The success rates in Table 2 (left) are higher than those in the look ahead study (Table 1) because the two evaluations are computed over different paired subsets. The deeper rollouts in Table 2 (left) re-

Table 2: **Left:** predicate ladder under an identical commit-time verifier; McNemar compares each row with the row above; the last row is an exact-oracle reference (depth 0), which is off the ladder. **Right:** comparison with external methods under a single-brick action space; McNemar is against Ours (group_max=1); Succ. gives solved/evaluable in parentheses; Exact-oracle + look. times out on 26/75 structures, hence n=49.

PREDICATE LADDER (n=925, group_max=5)

Predicate	Succ.	McNemar	Orc/br.
seq/none	0.836	—	2.06
reach/none	0.836	0:0, $p>0.05$	2.06
reach/rollout	0.886	77:30, $p<10^{-4}$	0.57
wm/rollout (ours)	**0.954**	73:11, $p<10^{-11}$	0.51
oracle/none	0.920	—	1.96

EXTERNAL COMPARISON ($n\leq$75, group_max=1)

Method	Succ. (k/n)	McNemar
WorkBenchMark (Ma et al., 2026)	0.480 (36/75)	13:0, $p<10^{-3}$
ASAP (GNN) (Tian et al., 2024)	0.027 (2/75)	47:0, $p<10^{-13}$
ASAP (heuristic)	0.000 (0/75)	49:0, $p<10^{-14}$
Exact-oracle + look.	1.000 (49/49)	—
Ours (group_max=1)	0.653 (49/75)	—
Ours (group_max=5)	**1.000 (75/75)**	—

quire more commit time verifier calls, causing different predicates to time out on different structures. Restricting the comparison to structures for which no predicate times out leaves 925 paired structures. In contrast, the shallower look ahead study retains 1,109 paired structures. This larger subset includes harder structures that are disproportionately excluded by timeouts in the deeper rollout setting, lowering the overall success rates. The absolute success rates in the two tables are therefore not directly comparable.

Finally, we compare against external assembly-sequence methods under a deliberately unfavorable protocol (Table 2, right). We compared against two baselines. For ASAP (Tian et al., 2024), we run the authors' released pre-trained weights on our structures, plus its heuristic ASP generator. For WorkBenchMark (Ma et al., 2026), we reimplement its Algorithm 1 removal gate. We use the same structures, the same embodiment, and the same exact static-stability ruler, with our grouping ability switched off (group_max=1) so that every method operates in the single-brick action space the external methods assume. Even so constrained, our method significantly outperforms WorkBenchMark and both ASAP variants (paired McNemar). Our advantage within this shared single-brick action space comes from the planner itself; grouping is a separate capability, when we allowed grouped bricks to at most 5 (group-max=5), our method additionally solves every structure. Some qualitative results of our approach are shown in Figure 4.

4.5 COLLISION HEAD ANALYSIS

We ask how much of the assembled geometry the head must read. We compare three configurations that share the architecture, the labels, and the training budget, and differ only in the amount of anchor latents input: no anchors at all (*None*, a floor), only anchors within three cells of the target in the L_∞ norm (*Local*), or all of the anchors (*Full*). Beyond this, every configuration still receives the brick type, the target pose, and the scalar scale features. We show the results of two motions separately (top-down placement and bottom-up insertion). As Table 3 shows, withholding geometry stops the head from filtering: FP@R99 rises sharply for both motions. Restricting attention to a local neighborhood is also 3.4–3.8× worse than the full set. Distant anchors are therefore signal: genuine collisions occur far from the target cell, and a local view cannot anticipate them.

Table 3: **Geometry visibility of the collision head.** FP@R99 (lower is better) for the two manipulation motions.

Motion	None	Local	Full (ours)
top-down	0.768	0.303	**0.081**
bottom-up	0.453	0.178	**0.053**

5 CONCLUSION

We presented a manipulation-level feasibility world model for brick assembly sequence planning. The model learns the geometry-dependent collision feasibility of a placement over a set of anchor-token latents and rolls them forward with a latent dynamics model, letting the planner look ahead over multiple assembly-by-disassembly steps without querying a physical oracle inside the look-ahead search. On StableText2Lego, this yields higher buildable-assembly success than greedy and heuristic-ordering baselines and stronger results than external assembly-sequence methods under a shared exact stability ruler, while issuing far fewer commit-time oracle calls.

AI USE STATEMENT

We used generative AI tools for the following tasks in this work: implementing parts of the method code; writing testing and evaluation code and monitoring the status of training runs; designing and running several of the architecture and capacity ablations reported in the supplementary material; creating and editing figures; drafting and editing the text; and summarizing related literature.

We did not use generative AI tools to interpret experimental results, to clean or reformat datasets, to generate synthetic data, or to formulate or prove mathematical claims.

We have reviewed all AI-assisted work: AI-generated code was tested by the authors, AI-assisted experimental results were checked against the exact oracles and ground-truth labels used throughout the paper, and AI-drafted text was reviewed and edited by the authors. We take responsibility for the final content of this work.

REFERENCES

Matan Atad, Jianxiang Feng, Ismael Rodríguez, Maximilian Durner, and Rudolph Triebel. Efficient and feasible robotic assembly sequence planning via graph representation learning. In *IROS*, 2023. 1, 2

Arman Barghi, Navid Pasiar, Mehdi Tale Masouleh, and Ahmad Kalhor. From bricks to bots: Automated collision-aware sequence planning for lego reconstruction with a two-finger gripper. In *10th International Conference on Control, Instrumentation and Automation (ICCIA)*, 2024. doi: 10.1109/ICCIA65044.2024.10768173. 2

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016. 2

Smail Ait Bouhsain, Rachid Alami, and Thierry Siméon. Learning geometric reasoning networks for robot task and motion planning. In *ICLR*, 2025. 2

Yuecheng Cai and Jasmin Jelovica. Graph neural network for stress predictions in stiffened panels under uniform loading. *arXiv*, 2023. 3

Hung-Fu Chang, Tzu-Kang Lin, and Yung-Li Cheng. Gnn for structural displacement prediction. *arXiv*, 2026. 3

Alan Chen and Changliu Liu. Assemblycomplete: 3d combinatorial construction with deep reinforcement learning. *arXiv*, 2024. 2

Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W. Taylor, Jaesik Park, and Minsu Cho. Brick-by-brick: Combinatorial construction with deep reinforcement learning. In *NeurIPS*, 2021. 1, 2

Sen Cui and Jingheng Ma. Physically native world models: A hamiltonian perspective on generative world modeling. *arXiv*, 2026. 3

Nikhil Das and Michael C. Yip. Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics*, 2020. 2

Fan Feng, Phillip Lippe, and Sara Magliacane. Learning interactive world model for object-centric reinforcement learning. In *NeurIPS*, 2025. 3

Mengqi Guo, Chen Li, Yuyang Zhao, and Gim Hee Lee. Treesba: Tree-transformer for self-supervised sequential brick assembly. In *ECCV*, 2024. 2

David Ha and Jürgen Schmidhuber. World models. In *NeurIPS*, 2018. 2

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019. 3

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv*, 2023. 3

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *ICLR*, 2024. 3

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *WWW*, 2020. 3

Philip Huang, Ruixuan Liu, Shobhit Aggarwal, Changliu Liu, and Jiaoyang Li. Apex-mr: Multi-robot asynchronous planning and execution for cooperative assembly. In *RSS*, 2025. 2, 3

Philip Huang, Chenrui Gao, and Jiaoyang Li. Vamp-mr: Vector-accelerated motion planning and execution for multi-robot-arms. In *IROS*, 2026. 7

Qi Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5. 7

J. Chase Kew, Brian Ichter, Maryam Bandari, Tsang-Wei Edward Lee, and Aleksandra Faust. Neural collision clearance estimator for batched motion planning. *arXiv*, 2020. 2

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *ICLR*, 2020. 3

Shih-Yu Lai, Chia-Ching Yen, Yang-Ting Shen, Peter Yichen Chen, Yu-Lun Liu, and Bing-Yu Chen. Euphoria: Efficient universal planning via hybrid optimization for robust industrial robotic assembly. *arXiv*, 2026. 2

Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction and its application to manipulation. In *AAAI Spring Symposia*, 2017. 3

Jiawei Liu, Senqiao Yang, Mingjun Wang, Yu Wang, and Bei Yu. Graph world models: Concepts, taxonomy, and future directions. *arXiv*, 2026a. 3

Ruixuan Liu, Kangle Deng, Ziwei Wang, and Changliu Liu. Stablelego: Stability analysis of block stacking assembly. *IEEE RA-L*, 2024. 6, 7

Ruixuan Liu, Alan Chen, Weiye Zhao, and Changliu Liu. Physics-aware combinatorial assembly sequence planning using data-free action masking. *arXiv*, 2025. 1, 2

Yuejiang Liu, Fan Feng, Lingjing Kong, Weifeng Lu, Jinzhou Tang, Kun Zhang, Kevin Murphy, Chelsea Finn, and Yilun Du. World action verifier: Self-improving world models via forward-inverse asymmetry. *arXiv*, 2026b. 3

Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. Planning assembly sequence with graph transformer. In *ICRA*, 2023. 2

Wenbo Ma, Daniel Swoboda, Matteo Tschesche, and Till Hofmann. Workbenchmark: A lego-based assembly benchmark with an assembly-by-disassembly baseline for the smart manufacturing league. *arXiv*, 2026. 2, 9

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *ICLR*, 2023. 3

Kartik Nagpal and Negar Mehr. Optimal robotic assembly sequence planning: A sequential decision-making approach. In *IROS*, 2024. 2

Jaehong Oh, Seungjun Jung, and Sawoong Kim. Learning to assemble the soma cube with legal-action masked dqn and safe zyz regrasp on a doosan m0609. *arXiv*, 2025. 2

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating physically stable and buildable brick structures from text. In *ICCV*, 2025. 1, 2

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters (RAL)*, 2026. Previously titled: Strengthening Generative Robot Policies through Predictive World Modeling. 3

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *ICML*, 2020. 2

Chang Shu, Anton Kim, and Shinkyu Park. Subassembly to full assembly: Effective assembly sequence planning through graph-based reinforcement learning. In *ICRA*, 2025. 1, 2, 4

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv*, 2026. 3

Yunsheng Tian, Karl D. D. Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, Hui Li, Sachin Chitta, and Wojciech Matusik. ASAP: automated sequence planning for complex robotic assembly with physical feasibility. In *ICRA*, 2024. 1, 2, 9

Rodion Vakhitov, Leonid Ugadiarov, and Aleksandr Panov. Object-centric world models meet monte carlo tree search. *arXiv*, 2026. 3

Jingwen Wang, Johannes Kirschner, Paul Rolland, Luis Salamanca, and Stefana Parascho. Learning to build: Autonomous robotic assembly of stable structures without predefined plans. *arXiv*, 2026. 2

Chenhui Xu, Ziyue Bai, Fuxun Yu, Heng Huang, and Jinjun Xiong. Rollback-free stable brick structures generation. *arXiv*, 2026. 2

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018. 3

Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, 2021. 3

Xinghao Zhu, Devesh K. Jha, Diego Romeres, Lingfeng Sun, Masayoshi Tomizuka, and Anoop Cherian. Multi-level reasoning for robotic assembly: From sequence inference to contact selection. In *ICRA*, 2024. 2