Assembly-aware Tiling for Standing Polyominoes

HAONAN ZHANG, Hong Kong University of Science and Technology, China
WENJUN LIU, Hong Kong University of Science and Technology, China
XINGXIN HE, Hong Kong University of Science and Technology, China
JUN HAN, Hong Kong University of Science and Technology, China
ZIQI WANG✉, Hong Kong University of Science and Technology, China

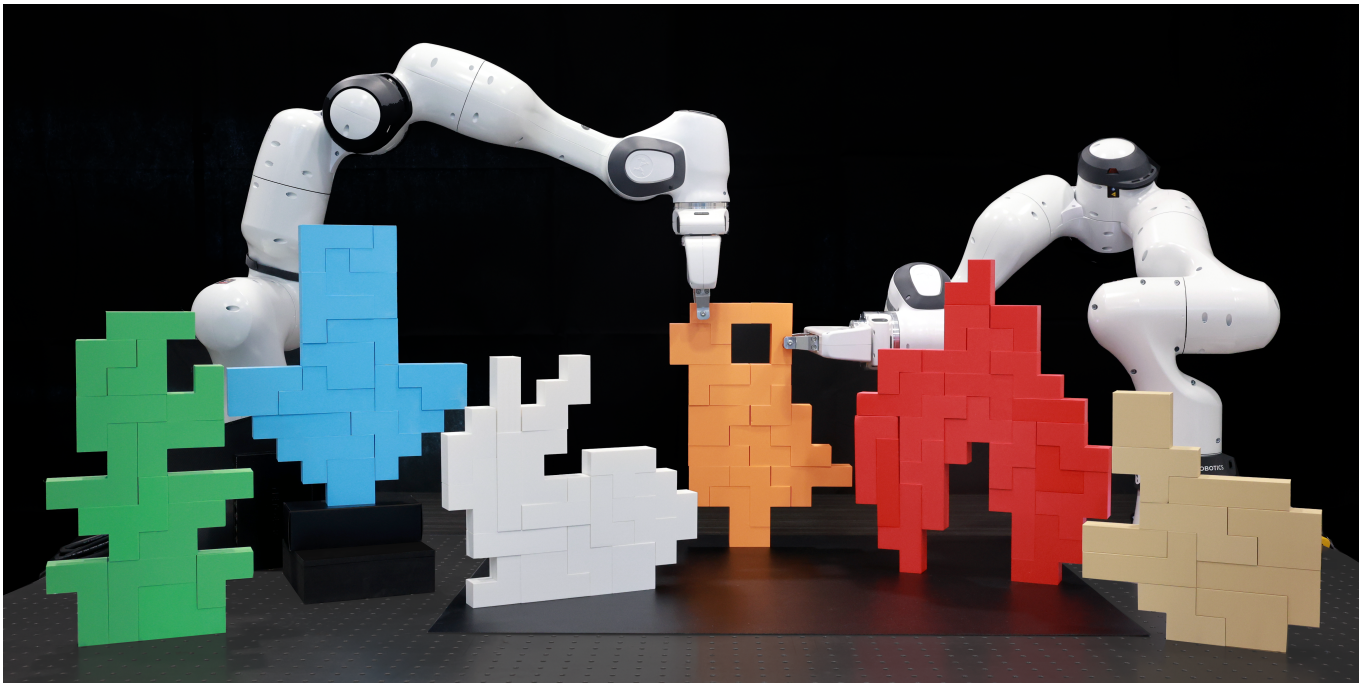


Fig. 1. We propose an assembly-aware tiling framework for generating stable polyomino structures that can be physically assembled by a dual-arm system.

We introduce a novel assembly-aware framework for designing standing polyominoes, a class of 2D equilibrium puzzles that serves as a testbed for robotic assembly. Unlike standard tiling problems that focus solely on geometric coverage, standing polyominoes must admit physically feasible bimanual assembly sequences, in which one robotic arm provides temporary support while the other places a new part. The key challenge lies in incorporating assembly constraints into the tiling process, which necessitates a fast and generalizable sequencing policy. However, this leads to a "chicken-and-egg" problem, as no valid standing polyomino designs are available to train such a sequencing policy. To address this, we propose a three-stage reinforcement learning framework. We first train a stability-aware tiling policy to generate physically stable designs. Next, we perform offline, search-based sequencing to identify designs that admit feasible bimanual assembly sequences and use these designs to train the sequencing policy. Finally, we integrate the learned sequencing policy back into the tiling process to enable assembly-aware design generation. Our approach expands the design space of equilibrium puzzles and establishes a principled framework for incorporating assembly constraints directly into computational design.

CCS Concepts: • **Computing methodologies** → **Planning for deterministic actions**; **Robotic planning**.

Additional Key Words and Phrases: Computational Tiling, Assembly Sequence Planning, Reinforcement Learning, Robotic Assembly

ACM Reference Format:
Haonan Zhang, Wenjun Liu, Xingxin He, Jun Han, and Ziqi Wang. 2026. Assembly-aware Tiling for Standing Polyominoes. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers '26), December 01–04, 2026, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3829340.3842268

Authors' Contact Information: Haonan Zhang, Hong Kong University of Science and Technology, Hong Kong, China, haonan.zhang@connect.ust.hk; Wenjun Liu, Hong Kong University of Science and Technology, Hong Kong, China, wliucy@connect.ust.hk; Xingxin He, Hong Kong University of Science and Technology, Hong Kong, China, xingxin.he@connect.ust.hk; Jun Han, Hong Kong University of Science and Technology, Hong Kong, China, hanjun@ust.hk; Ziqi Wang, Hong Kong University of Science and Technology, Hong Kong, China, ziqiw@ust.hk.

This work is licensed under a Creative Commons Attribution 4.0 International License.
SA Conference Papers '26, Kuala Lumpur, Malaysia
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2842-6/2026/12
https://doi.org/10.1145/3829340.3842268

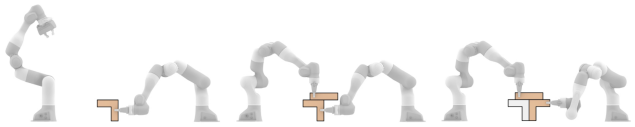


Fig. 2. Bimanual assembly of three polyomino pieces in a vertical orientation. Due to the instability of the base L-shapes when released, the task requires a bimanual approach to prevent collapse during construction.

1 INTRODUCTION

Equilibrium puzzles, such as Jenga and rock balancing, have long served as intriguing games for developing human physical dexterity and spatial reasoning. More recently, they have emerged as challenging benchmarks in robotic learning [Furrer et al. 2017; Goldberg et al. 2025; Sun et al. 2025; Wang et al. 2025], offering rigorous testbeds for evaluating the robustness and generalization capabilities of modern robotic assembly algorithms.

In this work, we investigate standing polyominoes, a class of 2D equilibrium puzzles inspired by the classic game Tetris. A key motivation for our standing polyominoes is to provide a reproducible dry-stacking benchmark for robotic assembly, as their components can be readily fabricated using 3D printing. Assembling such structures is particularly challenging, as it requires coordinated bimanual manipulation: one arm must provide temporary support to maintain equilibrium while the other precisely places new components; see Figure 2. Despite their geometric simplicity, our standing polyominoes capture key challenges in robotic assembly, including stability evaluation, assembly sequencing, and grasp planning. Existing toys, such as Jenga Maker, offer only a limited set of designs with relatively simple assembly sequences. To expand this design space, we propose an assembly-aware approach for generating standing polyominoes from given binary images. Specifically, we study how to decompose a target image into a set of polyomino structures that admit a physically feasible, bimanual robotic assembly process.

Existing tiling algorithms [Chen et al. 2025; Xu et al. 2020] primarily focus on geometric coverage and fail to account for the constraints of bimanual robotic assembly. As a result, many computationally generated designs become unstable the moment they are oriented vertically. Even when a final design is statically stable, it may remain infeasible for dual-arm assembly due to kinematic reachability limitations or self-collisions between the manipulators. Moreover, certain designs may require three or more robots to maintain intermediate structural stability.

These limitations motivate us to develop a reinforcement learning (RL) approach that explicitly incorporates robotic assembly constraints into the tiling process. To efficiently compute RL rewards, we require a fast and accurate method for determining whether a generated tiling admits a valid assembly or disassembly sequence. Search-based sequencing methods are accurate but often computationally prohibitive for RL training. Alternatively, one can train a policy to predict disassembly sequences, which introduces a chicken-and-egg problem: training such a policy requires a dataset of valid standing-polyomino designs, yet obtaining this dataset is difficult without a pre-existing sequencing policy.

To address this, we adopt a three-stage pipeline. We first use AlphaZero-style RL [Silver et al. 2018] to train a stability-aware tiling policy without considering robot-related constraints. This enables us to collect a wide variety of stable tiling designs as an initial dataset. We then apply offline search-based sequencing to filter these designs, identifying those that are truly feasible for bimanual assembly to train a generalizable sequencing policy. Finally, we freeze this sequencing policy and embed it within the tiling framework to enforce assembly constraints during generation. Our results show that integrating the sequencing model directly into the tiling process increases the success rate of the exact cover from approximately 50% to 90%. Specifically, we make the following contributions:

- We develop a novel stability-aware tiling policy that demonstrates strong generalization to unseen target images.
- We introduce a generalizable disassembly planning method that explicitly accounts for robotic graspability constraints.
- We propose an assembly-aware tiling framework for generating stable standing polyomino datasets while incorporating bimanual robotic assembly constraints.

2 RELATED WORK

2.1 Computational Tiling

Tiling is a classical mathematical problem that asks whether a given region can be fully covered using a specified set of shapes, without overlaps or gaps [Cohen et al. 2003; Grünbaum and Shephard 1987; Kaplan 2009; Kaplan and Salesin 2000; Peng et al. 2019, 2018].

In this work, we focus on Polyomino tiling, which provides a grid-based discretization of tiling problems [Golomb 1996]. Polyomino tiling problems have been widely studied using combinatorial optimization methods, including exact-cover search with Algorithm X [Knuth 2000] and ILP-based formulations [Garvie and Burkardt 2020, 2022]. Beyond classical tiling, Kita et al. [2023] combine polyomino tiling with dissection puzzles. While most prior studies assume a predefined set of tile shapes, recent inverse tiling methods automatically discover a minimal set of tiles that can cover a given target image [Chen et al. 2025]. Learning-based approaches have also been explored for tiling problems. For example, Xu et al. [2020] use self-supervised graph neural networks to learn optimal tile placements from a set of candidate locations.

Recent works have also extended polyomino tiling to 3D structures [Kita and Miyata 2021; Lo et al. 2009]. However, these approaches often rely on non-flat polyomino parts composed of many voxels [Chen et al. 2022; Kita and Miyata 2021; Song et al. 2012; Wang et al. 2018] or deformed polyominoes with additional connectors [Lo et al. 2009]. In contrast, our framework generates structurally stable 2D tilings using simple, flat polyomino pieces. Moreover, we integrate robotic assembly constraints directly into the design process, enabling the resulting tiling designs to be physically assembled by a dual-arm robotic system.

2.2 Assembly Sequence Planning

Assembly sequence planning (ASP) aims to determine a feasible sequence of tasks for assembling a target structure [Wang et al. 2021]. Traditionally, ASP is formulated as a search problem [Halperin et al. 1998; Tian et al. 2022; Wilson 1992]. The main challenge of this

formulation is the combinatorial explosion, where the search space expands exponentially as the number of parts increases. Recent learning-based methods have provided a new direction to solve ASP. Graph Neural Networks are the most widely adopted approach for encoding part relations and assembly states [Atad et al. 2023; Ma et al. 2022; Tian et al. 2024; Wang et al. 2025]. Reinforcement learning has also been considered an effective way to tackle assembly planning problems [Ghasemipour et al. 2022; Wang et al. 2025]. These works demonstrate that learning-based approaches can significantly accelerate ASP. However, many reinforcement learning–based ASP methods are designed for a single assembly, and typically require retraining to handle new structures. In contrast, our approach leverages an AlphaZero-style framework trained on a large-scale dataset, enabling the sequencer to generalize effectively to unseen structures.

2.3 Robotic Assembly Planning

Robotic assembly planning extends beyond conventional ASP by considering physical manipulation constraints, such as grasp feasibility, kinematic reachability, and collision avoidance. Recent works have demonstrated the effectiveness of robots in many assembly tasks, including block stacking [Li et al. 2020], timber joint assembly [Apolinarska et al. 2021], LEGO construction [Chen et al. 2023; Chung et al. 2021; Pun et al. 2025], Soma cube assembly [Oh et al. 2025], magnetic assembly [Ghasemipour et al. 2022; Kataoka et al. 2023], irregular stone assembly [Menezes et al. 2021], dry-stone construction [Johns et al. 2023], interlocking block assembly [Zhang et al. 2021], mechanical assembly [Tian et al. 2025], discrete shell structures [Wang et al. 2023], and construction [Funk et al. 2022].

However, the target structures considered in these works vary substantially in geometry, scale, and assembly requirements, making quantitative comparisons across methods difficult. To address this gap, we propose standing polyominoes as a standardized benchmark for robotic assembly. Standing polyominoes offer several advantages. First, different target images can generate a large variety of assembly designs, many of which require challenging bimanual manipulation. Second, the individual parts are simple, flat pieces that can be fabricated with standard 3D printers and reused across different tiling designs. Third, assembling these structures does not require tight insertion motions, such as those involved in joint-based assemblies, thereby reducing the precision requirements on the robotic setup. Together, these properties make standing polyominoes a challenging yet easily reproducible benchmark for evaluating future robotic assembly algorithms.

3 Methodology

3.1 Stability-aware Tiling

This subsection details how we train the tiling policy $\pi_{\texttt{tile}}$ to cover binary images with polyomino pieces while ensuring the resulting structures remain stable. We consider a polyomino inventory comprising Trominoes (2 types), Tetrominoes (5 types), and Pentominoes (12 types), which offers a balance between shape expressiveness and inventory size. Although our framework can be extended to larger polyomino sets such as Hexominoes (35 types) and Heptominoes (108 types), doing so substantially increases the inventory size and leads to longer training times.

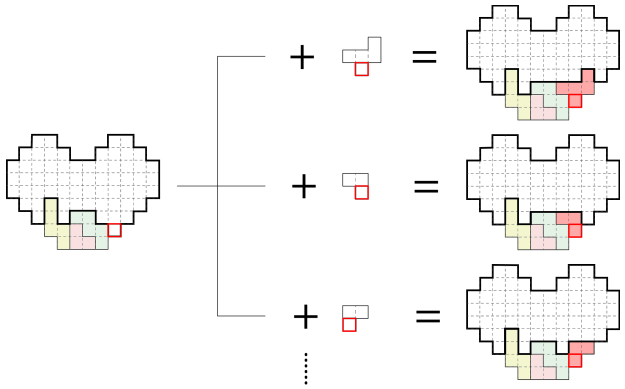


Fig. 3. The bottom-right corner of the polyomino is aligned with the bottom-rightmost uncovered pixel of the tiling design (highlighted in red). The white region with the bold outline denotes the remaining piece S.

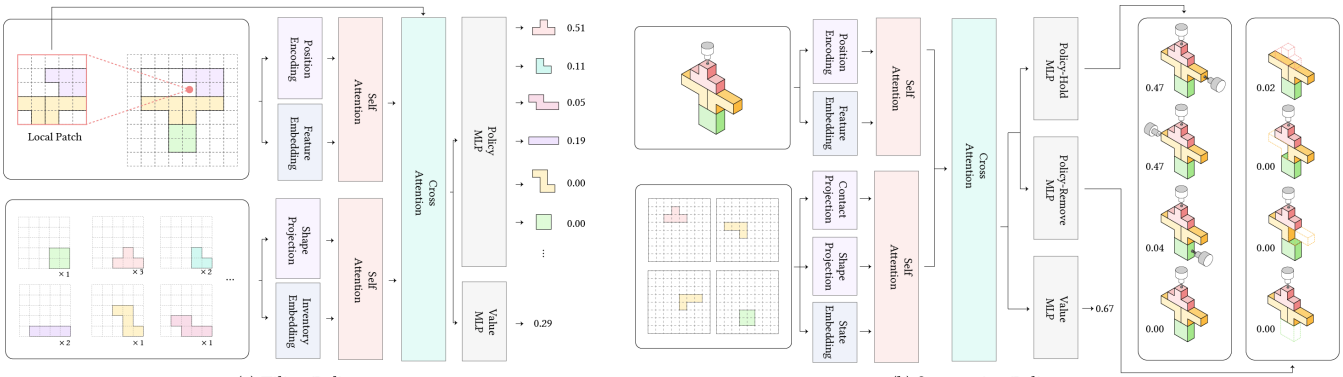
Algorithm 1: AlphaZero with a Surrogate Transition

Input : Initial state s_0, transition μ, and its surrogate $\hat{\mu}$
Output: Policy π, and replay buffer $\mathcal{B}$

for $k = 1$ **to** N **do**
  $s \leftarrow s_0$;
  while ***not*** `Terminate(`s`)` **do**
    $a, p \leftarrow \texttt{MCTS}(s, \pi, \mu, \hat{r}, N_{\text{mcts}})$;
    $s' \leftarrow \mu(s, a)$;
    $r \leftarrow r(s, a)$;
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s, a, p, r)\}$;
    $s \leftarrow s'$;
  `Train(`π`)` using $\mathcal{B}$;
  if $k \mod N_\mu = 0$ **then**
    `Train(`$\hat{\mu}$`)` using $\mathcal{B}$;

Our tiling is a part-decomposition process, similar to other puzzle design problems [Chen et al. 2022; Wang et al. 2018], where a given design domain is progressively decomposed into polyomino pieces. The uncovered remaining piece S is treated as a physical solid and participates in contact interactions with neighboring parts. Our objective is to minimize the area of the remaining piece, ideally reducing it to zero to achieve an exact cover.

A key benefit of treating our tiling as a part-decomposition process is that the structural stability of intermediate designs exhibits a monotonic property: once an intermediate tiling becomes unstable, all subsequent tilings are deemed unstable. This is because components are progressively split from the remaining piece, thereby reducing the structural integrity of the resulting assembly. This property allows us to terminate infeasible designs early by performing a structural stability check at each step of the tiling process. A detailed description of the tiling process is provided in the supplementary material.



(a) Tiling Policy (b) Sequencing Policy

Fig. 4. (a) Tiling policy architecture π_{tile}. The network takes the current tiling design and polyomino inventory as input. The current tiling design consists of 6 channels: four encoding contact geometry ($\pm x, \pm y$) and two representing the design domain mask and occupancy, respectively. Each inventory piece is represented as a 5×5 occupancy mask together with a scalar specifying the remaining number of available pieces of that type. We additionally extract a local patch centered at the next placement location. The network outputs both a state-value estimate and a probability distribution over candidate polyomino placements, with invalid actions masked out to zero probability. (b) Sequencing policy architecture π_{seq}. The network takes the current disassembly state and the set of remaining parts available for disassembly as input. The disassembly state is represented using six channels: four encoding contact geometry and two indicating the design domain mask and the part state. Each candidate part for disassembly is represented as a masked disassembly state containing only the features associated with that part. The network outputs separate policy heads for selecting the holding and removing parts, together with a state-value head. The state-value head estimates $V_{\pi_{\text{seq}}}$, representing the probability that the current disassembly state can lead to a successful complete disassembly sequence. Invalid actions are masked to zero probability. Detailed network architectures are provided in the supplementary material.

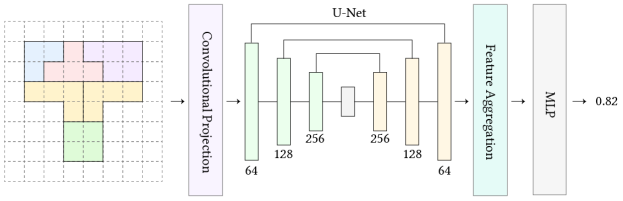


Fig. 5. The stability surrogate network is implemented using a U-Net architecture [Ronneberger et al. 2015]. The input tiling design is represented by six channels: four encoding contact geometry, one indicating the design domain mask, and one encoding the corresponding part states. The network outputs a prediction of the structural stability of the tiling design.

At each tiling step, a polyomino P is selected, transformed, and placed such that its bottom-right corner aligns with the bottom-rightmost valid pixel of the remaining piece S (Figure 3). We unify selection and transformation by augmenting the inventory with all unique variants generated via in-plane rotations and reflections of each base polyomino. We cap the inventory of each base polyomino at 10 instances, with all its transformed variants sharing this quota. The tiling process terminates when no additional polyomino can be placed without overlap or when the resulting intermediate design, including the remaining piece, becomes structurally unstable.

We are inspired by the AlphaZero framework to train our tiling policy $\pi_{\texttt{tile}}$ (Algorithm 1), which leverages Monte Carlo Tree Search (MCTS) to address challenging reinforcement learning problems with discrete action spaces. Here, we focus on the formulation of its four key components: the state, transition, reward, and policy network. The state s includes the current intermediate tiling design and the current piece inventory. At each step, we perform N_{mcts} MCTS simulations from the current state s to determine the next action of selecting a polyomino to place. MCTS produces an improved policy distribution p over candidate actions, from which the action $a = P$ is selected. The transition function $\mu(s, a)$ returns the subsequent state s', while the reward function $r(s, a)$ evaluates the selected action and encourages full coverage of the target image.

$$r = \begin{cases} \alpha(P) & \text{valid action and } s' \text{ is stable} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\alpha(P)$ denotes the area fraction of P relative to the tiling board.

An action P is deemed invalid if P overlaps with already occupied pixels or invalid background pixels, or if the inventory for P has been exhausted. Consequently, the probabilities for these actions are masked to zero. For structurally unstable states, we assign a reward of zero but allow the tiling process to continue during training to collect additional failure cases for surrogate learning. Figure 4(a) illustrates the tiling policy $\pi_{\texttt{tile}}$, which is built upon a transformer-based backbone. To optimize the policy, trajectories are stored in a replay buffer $\mathcal{B}$. The policy parameters of $\pi_{\texttt{tile}}$ are then updated every episode using batches sampled from this buffer.

Training efficiency is largely determined by the throughput of the MCTS loop, which in turn depends on the speed of reward computation. The reward function r requires an expensive physics-based simulation to evaluate the structural stability of the resulting tiling design s'. To accelerate RL, we train a surrogate stability simulator $g(s)$ that predicts the probability of a tiling state s being structurally stable. The surrogate g is implemented using a lightweight U-Net architecture (Figure 5). Once trained, the surrogate reward used in MCTS is defined as $\hat{r} = g(s')\alpha(P)$ for a valid action. The stability

surrogate g is initially pretrained on tilings generated via uniform sampling. Every N_μ training episodes, we sample tiling designs from the replay buffer and verify their stability using a physics-based simulator. These labels are then used to continuously update the surrogate model g, ensuring its predictions remain aligned with ground-truth stability assessments.

3.2 Disassembly Sequencing

The previously generated tiling designs, despite being structurally stable, may still lack a robot-executable assembly sequence and therefore cannot be physically assembled. This is further evidenced in Table 2, where only approximately half of the generated designs are robot-buildable. In dry-stacking contexts, assembly and disassembly are inherently reversible; consequently, reversing a disassembly sequence yields a viable robotic assembly plan. We therefore adopt an assembly-by-disassembly formulation [Wang et al. 2025; Zhou et al. 2014] and train a generalizable disassembly policy π_{seq} to assess the robot-buildability of diverse standing-polyomino designs.

A disassembly sequence is considered feasible if it satisfies two constraints. (1) **stability**: the structural stability of every intermediate state under gravity, and (2) **graspability**: the existence of collision-free solutions for each manipulator to grasp and remove parts without interfering with the structure or the other arm. Given that collisions primarily occur when one arm holds a part while the other performs an extraction, we simplify the second constraint by verifying the existence of collision-free inverse kinematics (IK) solutions at each step; see Figure 6. Furthermore, because our designs are 2D, a valid grasp guarantees a feasible motion for part removal along the direction perpendicular to the primary face.

Our disassembly process mostly follows the framework established by [Wang et al. 2025], and our main contribution is extending it from solving a single design instance to learning a generalizable policy. A disassembly state consists of a tiling design together with the disassembly states of all parts that have not yet been disassembled. Each part is assigned one of three states: "non-removed", "fixed", or "removed". The disassembly process alternates between two discrete actions. First, one of the robots grasps a "non-removed" part, transitioning it to the "fixed" state. This action is valid if a collision-free IK for grasping exists. Second, one of the robots removes the part it is currently grasping, transitioning it to the "removed" state. This action is valid when the remaining assembly maintains structural stability under gravity. For valid actions, we use the same reward model as in Equation 1. Invalid actions, including those that lead to structurally unstable states, immediately terminate the episode.

We train our sequencing policy π_{seq} by extending the same AlphaZero framework to the robotic disassembly setting. However, this extension introduces two primary challenges: dataset generation and surrogate training. First, we must construct a training dataset of polyomino tilings that are compatible with dual-arm manipulation. Not all designs from the replay buffer $\mathcal{B}$ used in Section 3.1 can be physically disassembled by two robotic arms. To bridge this gap, we employ an offline beam search (width $k = 64$) to identify feasible disassembly sequences. This process effectively filters the dataset, ensuring the sequencing policy learns exclusively from valid, kinematically reachable tiling designs. We further develop a novel transformer-based sequencing network to learn a generalizable sequencing policy (Figure 4(b)).



Fig. 6. A disassembly step is considered valid if at least one candidate robot configuration yields a collision-free IK solution for the grasp.

The second challenge is to develop a surrogate transition model to accelerate the MCTS loop. Similar to stability-aware tiling, the state transition in the disassembly process is computationally inexpensive. The main difficulty lies in learning surrogate models that can accurately evaluate graspability and structural stability. First, to assess graspability, we precompute a library of IK solutions for grasping individual voxels with each robot. The graspability model evaluates a disassembly state by checking whether a selected combination of these precomputed IK solutions yields a collision-free grasp configuration; see Figure 6. In practice, we find that this process is efficient enough to exhaustively verify all candidate actions. We therefore reformulate graspability as part of the action mask and filter out infeasible actions during action generation. Second, to evaluate structural stability, we train a dedicated hold-aware stability surrogate model g from scratch, rather than reusing the original stability surrogate in Section 3.1. This is necessary because the disassembly setting introduces an additional condition in which parts may be temporarily supported by robotic arms, leading to a stability distribution different from that in the tiling process.

3.3 Assembly-aware Tiling

This section uses the sequencing policy from Section 3.2 to finetune the stability-aware tiling policy from Section 3.1, yielding an assembly-aware tiling policy. A straightforward approach is to replace the stability reward with a sequencing-based disassemblability reward. However, disassemblability differs from structural stability: the inability to disassemble an intermediate tiling design does not necessarily imply that the final tiling design is invalid. In particular, while splitting the remaining piece may cause structural instability, it may also simplify robot grasping and part removal, thereby improving disassemblability. Therefore, disassemblability lacks the monotonicity property of structural stability and cannot be directly used as a step-wise feasibility criterion; relying only on final-design disassemblability would result in a sparse terminal reward.

To provide the RL with more frequent feedback, we introduce the concept of a modified intermediate tiling design together with a new necessary condition for bimanual robotic assembly:

Proposition 1. *A final tiling design is disassemblable only if every modified intermediate tiling design is disassemblable. In each modified intermediate design, the remaining piece S is treated as a fixed part and excluded from robot–assembly collision checking.*

While this proposition provides only a necessary condition, satisfying it for every intermediate tiling design does not guarantee

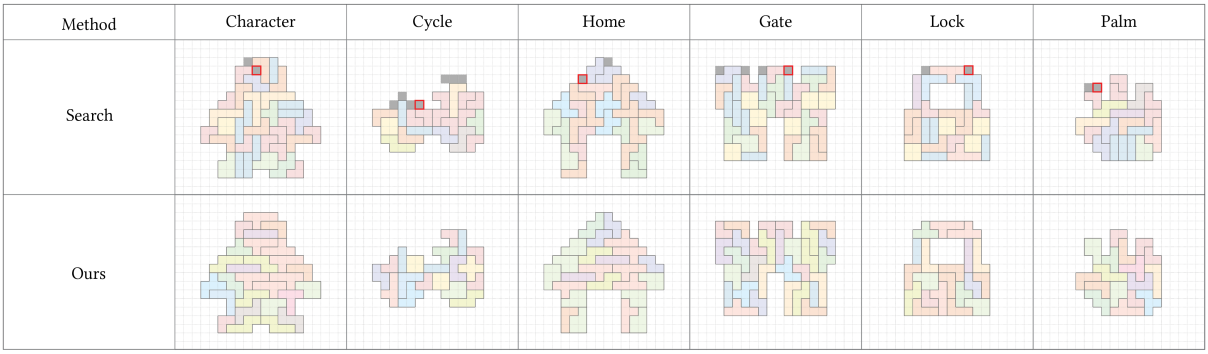


Fig. 7. We compare the tiling results generated by beam search with stability constraints and our stability-aware tiling policy. Our policy more effectively yields exact cover designs, whereas beam search often leads to a small number of residual pixels. Residual pixels are shown in grey, and red bounding boxes highlight the anchors of the terminal steps.

the existence of a valid final disassembly sequence. However, violating it immediately renders the final design infeasible. The proof is straightforward: if the final design possesses a valid disassembly sequence, that sequence can be directly mapped to a sequence for disassembling any modified intermediate design by simply omitting disassembly actions involving tiles that have not yet been placed.

Verifying this proposition requires extending the sequencing model π_{seq} to evaluate the disassemblability of modified tiling designs. Since graspability is considered only through the action mask, the input of the sequencing policy remains unchanged. Consequently, adapting π_{seq} only requires augmenting the training dataset with designs in which the remaining piece is fixed.

During training of the assembly-aware tiling policy, the sequencing policy π_{seq} remains frozen. At each tiling step, we evaluate both the structural stability of the current design and the disassemblability of the corresponding modified design. We found that explicitly verifying disassemblability by searching for a feasible sequence remains computationally expensive, even with the sequencing policy π_{seq}. We therefore use the state value function $V_{\pi_{\text{seq}}}$ to estimate disassemblability. Let $\tilde{s}'$ denote the corresponding modified tiling design. The reward function r is then defined as follows:

$$r = \begin{cases} V_{\pi_{\text{seq}}}(\tilde{s}')\alpha(P) & \text{valid action and } s' \text{ is stable} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Similarly, during MCTS, we employ the approximated reward

$$\hat{r} = V_{\pi_{\text{seq}}}(\tilde{s}')g(s')\alpha(P) \tag{3}$$

for a valid action P, where $g(s')$ denotes the predicted stability probability of the tiling design s', as introduced in Section 3.1. This reward formulation enables us to fine-tune the pretrained stability-aware tiling policy from Section 3.1, with π_{seq} kept fixed, to obtain an assembly-aware tiling policy.

4 Results

Our framework is implemented in Python, leveraging the mctx library [DeepMind et al. 2020] for GPU-accelerated Monte Carlo Tree Search. We perform $N_{\text{mcts}} = 256$ MCTS simulations with a maximum search depth of 30 at each step. The policy network is built using the JAX and Haiku [Hennigan et al. 2020] ecosystems, while the CuRobo library [Sundaralingam et al. 2023] handles collision-aware inverse kinematics and motion planning. Structural stability is verified via the Newton physics engine [The Newton Contributors 2025]. We set the friction coefficient to 0.1; additional simulator parameters are provided in the supplementary material. We conducted training on a computer equipped with an NVIDIA RTX 5880 GPU, 32 Intel CPU cores, and 48 GB of system memory. Our code is available at https://github.com/haonanomo/AssemblyAwareTiling.

4.1 Stability-aware Tiling

Our dataset comprises 200 target images, which are divided into a train set of 100 and a test set of 100. We have an inventory of 88 polyominoes, accounting for all unique transformed variants. We train a unified tiling policy with a batch size of 72 images. During training, we maintain a replay buffer containing up to 200,000 explored samples. Discarded samples are stored in a separate dataset and later used to train the sequencing and stability surrogate models. The policy is updated every episode with a learning rate of 5×10^{-5}, while the stability surrogate is updated every $N_g = 200$ episodes with a learning rate of 2×10^{-4}. The accuracy of our stability surrogate on the test set is over 87%. At a classification threshold of 0.5, the surrogate achieves a false-positive rate of 29.7% and a false-negative rate of 1.62%. The total training time spanned 120 hours over 1,000 episodes. To compare the efficacy of our surrogate against a purely simulator-based MCTS, we conducted runtime profiling. Each MCTS iteration with the surrogate is approximately 100 times faster than simulator-based MCTS.

Table 1 reports the performance of our stability-aware tiling policy. Specifically, we evaluate three metrics: average coverage area, the success rate of exactly covering, and inference time. We report results of both the baseline and our policy by selecting the best stable tiling among all generated designs for each target image. We employ Monte Carlo Tree Search to select actions. Our method achieves significant improvements over the baseline approaches across all evaluation metrics. A comparison between our method and the baseline search method is shown in Figure 7.

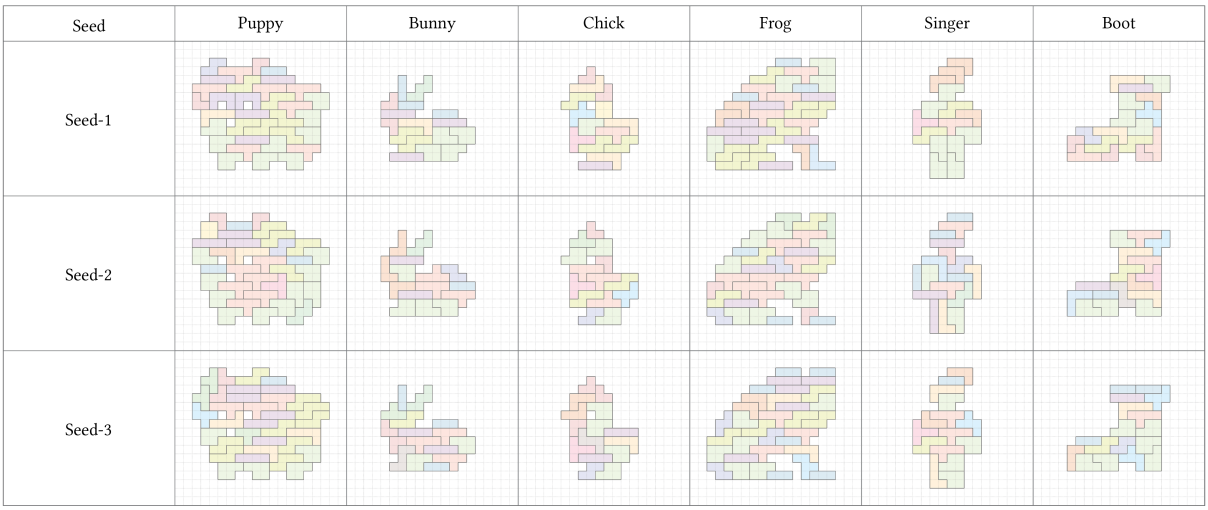


Fig. 8. We demonstrate that, by varying the initial inference seed, our assembly-aware tiling model can generate diverse tiling designs.

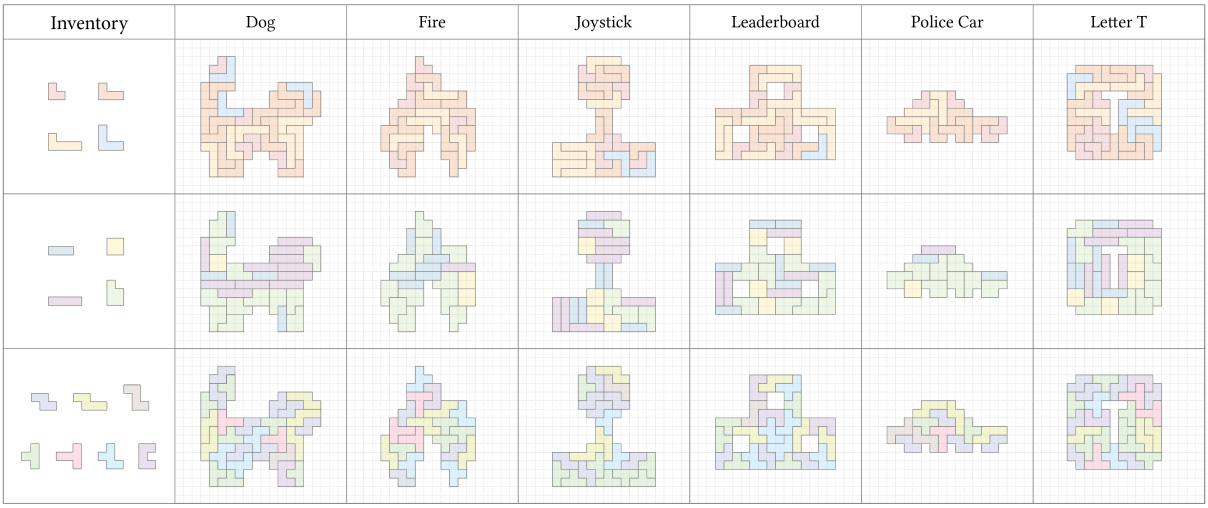


Fig. 9. We demonstrate that, by varying the initial inventory, our assembly-aware tiling model can generate diverse tiling designs.

4.2 Robotic Disassembly Planning

Our experimental setup consists of two Franka Research 3 robots spaced 1.7m apart, with a workspace resolution of 5cm per voxel. Each robot is equipped with a two-finger parallel gripper with an 8cm jaw width, and its maximum reachability is 1.05m. We enforce a constraint that prohibits a single robot from grasping more than one polyomino part. To support efficient graspability evaluation, we pre-sample 24 candidate grasp poses per voxel and compute, offline, a corresponding database of collision-free inverse kinematics solutions for each robot, enabling fast lookup during training.

The stability-aware tiling process generates a dataset of 10,000 intermediate states. We employ a beam search ($k = 64$) to filter these states, conservatively classifying any design that lacks a valid disassembly sequence as non-disassemblable. Our final dataset for training our sequencing policy consists of 1,000 verified unique tiling designs. This dataset is partitioned into 800 training and 200 testing samples. We train a unified sequencing policy π_{seq} using a batch size of 512. A replay buffer of 200,000 is maintained during training. The policy is updated every episode, and a hold-aware stability surrogate is updated every $N_g = 200$ episodes, respectively, using a learning rate of 10^{-4} and 5×10^{-5}. The accuracy of our hold-aware stability surrogate on the test set reaches above 85%. Using the same classification threshold of 0.5, the surrogate achieves a false-positive rate of 36.3% and a false-negative rate of 2.74%. Training spanned 72 hours over 1,000 episodes. The model achieved a training success rate of 87.7% and a test accuracy of 70.8%.

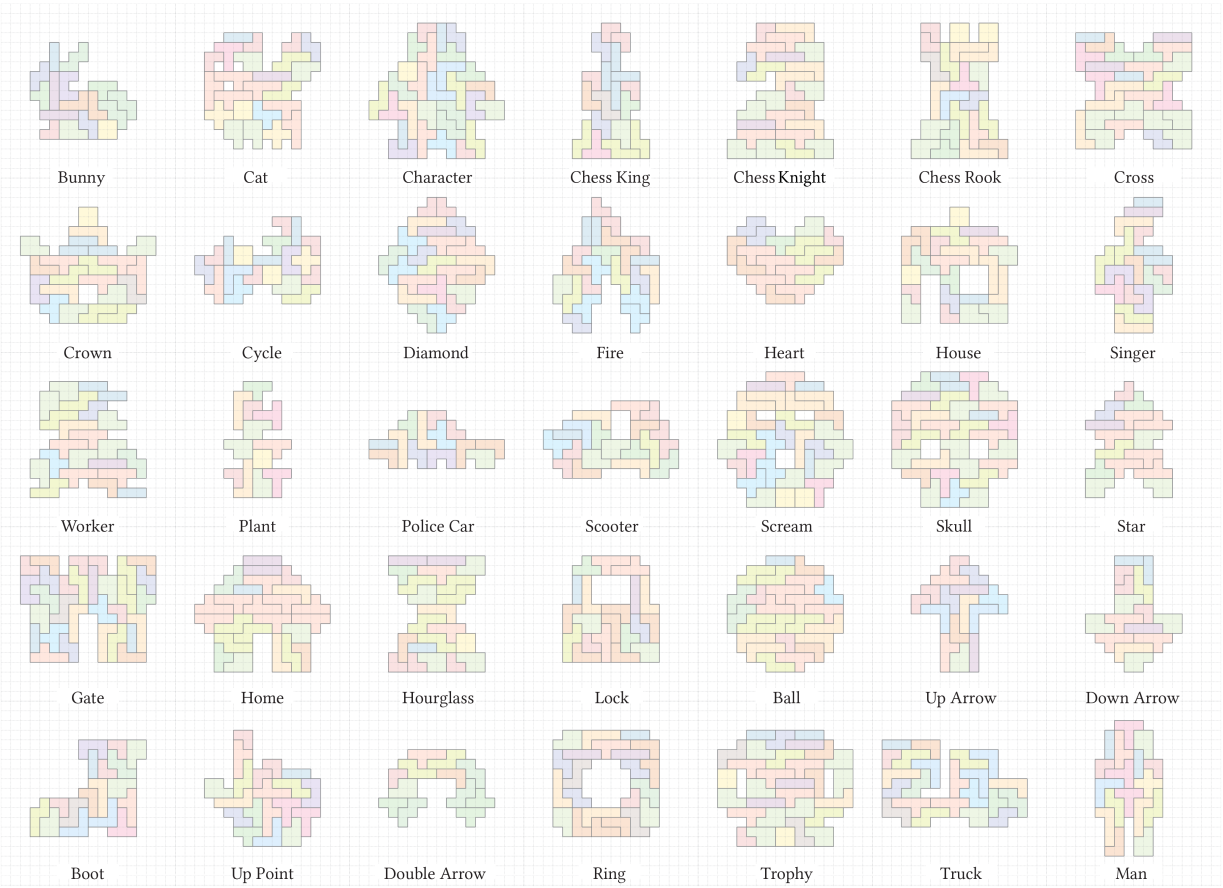


Fig. 10. A gallery showcasing our assembly-aware tiling designs.

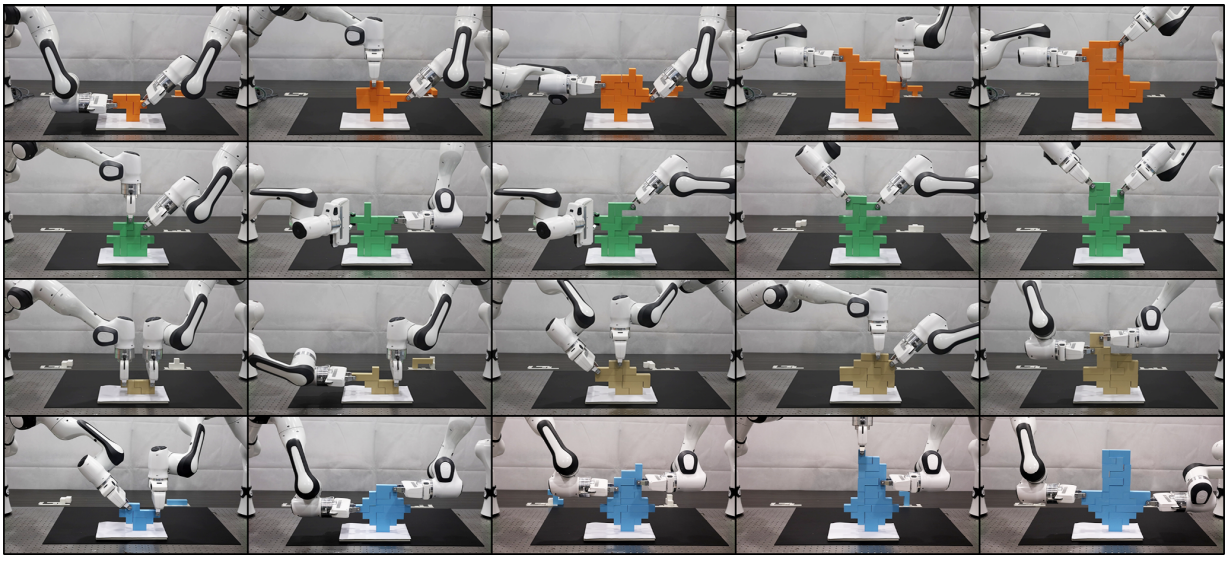


Fig. 11. Snapshots of the robotic assembly process for four tiling designs. The complete assembly videos are provided in the supplementary material.

Table 1. Performance statistics for stability-aware tiling. *Beam* denotes beam search with geometry-only constraints; *Beam-st* denotes beam search with additional stability constraints; *RL-st* denotes a single inference run of our stability-aware policy; and *RL-st10* denotes the best result among 10 inference runs of our stability-aware policy. The reported runtime excludes stability verification, which takes approximately 1.2s per design on average.

Method	Avg. Coverage		Exact Cover		Time(s)
	Train	Test	Train	Test	
Beam	0.169	0.231	0.130	0.210	16.1
Beam-st	0.926	0.935	0.500	0.570	126.2
RL-st	0.954	0.916	0.810	0.700	1.6
RL-st10	**0.991**	**0.979**	**0.970**	**0.850**	4.3

Table 2. Performance statistics for assembly-aware tiling. Method names follow Table 1, with the suffix *asm* denoting the assembly-aware policy. Assemblability verification time is excluded from all reported runtimes; this verification takes approximately 100s per design on average.

Method	Avg. Coverage		Exact Cover		Time(s)
	Train	Test	Train	Test	
Beam	0.070	0.140	0.070	0.140	17.0
Beam-st	0.508	0.543	0.320	0.380	130.4
RL-st	0.569	0.545	0.500	0.390	1.7
RL-st10	0.597	0.591	0.580	0.510	4.5
RL-asm	0.754	0.728	0.640	0.550	2.0
RL-asm10	**0.965**	**0.972**	**0.900**	**0.860**	7.1

4.3 Assembly-aware Tiling

We fine-tune the pre-trained stability-aware tiling policy in 72 hours on the same dataset. Table 2 reports the performance of our assembly-aware tiling policy. To verify the generated results, we employ a beam-search-based sequencing algorithm to determine the assemblability of each design. The inference times reported in Table 2 do not include this verification. Note that beam search typically produces a large number of candidate designs, resulting in substantially higher verification time.

Since stability is enforced at every tiling step, all generated designs satisfy the stability constraint. Among all generated designs, the fraction verified as assemblable increases from 58% with the stability-aware policy to 79% with the assembly-aware policy. Our assembly-aware tiling results demonstrate substantial performance gains, highlighting the importance of integrating assembly constraints into the tiling policy training.

To demonstrate that our policy does not overfit, Figure 8 illustrates how varying the inference seed generates a diverse set of designs for the same target image. Furthermore, Figure 9 demonstrates the policy's adaptability; by varying the input inventory and individual piece counts, we can produce diverse designs that accommodate different piece distributions. A gallery of assembly-aware tilings is shown in Figure 10.

To validate our method in real-world settings, we conduct physical assembly experiments. Following the framework of [Wang et al. 2025], we employ a depth-first search procedure guided by our pre-trained sequencing policy to identify valid disassembly sequences. We then adopt the trajectory generation method proposed in [Wang et al. 2025] to produce smooth, collision-free disassembly motions while maintaining structural stability throughout the process. Finally, we reverse the resulting trajectories to obtain feasible assembly motions. Figure 11 presents snapshots of our dual-arm robotic system assembling four representative polyomino designs.

5 Limitations & Future Work

First, we focus on learning an assembly-aware tiling policy for 2D standing polyominoes. Extending the framework to general 3D voxelized structures is a natural next step. Although the underlying architecture is designed to support 3D tiling, scaling to a large voxelized target shape will introduce substantial computational challenges due to the significantly increased state and action spaces. In addition, our framework could potentially be adapted for interlocking puzzle design [Song et al. 2012] by replacing the stability checker with a global interlocking checker. Second, the current sequencing model enforces assembly feasibility mainly through inverse kinematics checks at each assembly step. For more complex 3D assemblies, full trajectory-level feasibility, including collision-free motion planning, should also be incorporated. Moreover, the current framework assumes a fixed robot configuration and does not generalize across different robot placements or hardware setups. Extending the method to handle varying robotic platforms and workspace configurations remains an important direction for future research. Third, our physical experiments reveal several sim-to-real gaps. The Newton stability simulator occasionally predicts unstable designs as stable, particularly in borderline cases. In addition, hardware precision limitations introduce placement deviations during assembly, while insertion operations can cause unintended collisions with neighboring components, resulting in assembly errors or task failures. To address these issues, future work will explore integrating more advanced robot learning approaches, such as Vision-Language-Action (VLA) models [Intelligence et al. 2025], to improve manipulation precision and enable autonomous failure detection and recovery.

6 Conclusion

We introduced an assembly-aware framework for generating standing polyominoes, a challenging class of 2D equilibrium puzzles that serves as a benchmark for bimanual robotic assembly. By explicitly incorporating physical stability and dual-arm graspability into the tiling process, our method goes beyond conventional geometry-driven approaches to produce designs that are not only visually coherent but also physically realizable.

Integrating assembly constraints into the tiling process is particularly challenging because both assembly sequencing and tiling are computationally demanding problems, rendering joint optimization generally intractable. To address this challenge, we jointly train efficient tiling and sequencing policies in a tightly coupled framework,

enabling the generation process to account for assembly feasibility throughout design synthesis.

Overall, our framework bridges geometric design and robotic execution, enabling the large-scale generation of physically grounded assembly instances. We believe this work opens new opportunities for exploring the interplay among geometric design, structural stability, and manufacturability, while also providing a foundation for extending assembly-aware design to more complex 3D structures and multi-robot systems.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and valuable suggestions. This work was supported in part by the BYD-HKUST Joint Lab Research Project (BYD26IS02), the Shenzhen Loop Area Institute (FPF10120250006), the National Natural Science Foundation of China (Grant No. 62302422), and CORE, a joint research center for ocean research between Laoshan Laboratory and The Hong Kong University of Science and Technology.

References

Aleksandra Anna Apolinarska, Matteo Pacher, Hui Li, Nicholas Cote, Rafael Pastrana, Fabio Gramazio, and Matthias Kohler. 2021. Robotic assembly of timber joints using reinforcement learning. *Automation in Construction* 125 (2021), 103569.

Matan Atad, Jianxiang Feng, Ismael Rodríguez, Maximilian Durner, and Rudolph Triebel. 2023. Efficient and feasible robotic assembly sequence planning via graph representation learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8262–8269.

Rulin Chen, Xuyang Ma, Praveer Tewari, Chi-Wing Fu, and Peng Song. 2025. Inverse Tiling of 2D Finite Domains. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. 1–11.

Rulin Chen, Ziqi Wang, Peng Song, and Bernd Bickel. 2022. Computational design of high-level interlocking puzzles. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–15.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. 2023. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987* (2023).

Hyunsoo Chung, Jungtaek Kim, Boris Knyazev, Jinhwi Lee, Graham W Taylor, Jaesik Park, and Minsu Cho. 2021. Brick-by-brick: Combinatorial construction with deep reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 5745–5757.

Michael F Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. 2003. Wang tiles for image and texture generation. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 287–294.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. 2020. *The DeepMind JAX Ecosystem*. http://github.com/deepmind

Niklas Funk, Georgia Chalvatzaki, Boris Belousov, and Jan Peters. 2022. Learn2assemble with structured representations and search for robotic architectural construction. In *Conference on Robot Learning*. PMLR, 1401–1411.

Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, and Marco Hutter. 2017. Autonomous robotic stone stacking with online next best object target pose planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2350–2356.

Marcus Garvie and John Burkardt. 2020. A new mathematical model for tiling finite regions of the plane with polyominoes. *Contributions to Discrete Mathematics* 15, 2 (2020), 95–131.

Marcus R Garvie and John Burkardt. 2022. A parallelizable integer linear programming approach for tiling finite regions of the plane with polyominoes. *Algorithms* 15, 5 (2022), 164.

Seyed Kamyar Seyed Ghasemipour, Satoshi Kataoka, Byron David, Daniel Freeman, Shixiang Shane Gu, and Igor Mordatch. 2022. Blocks assemble! learning to assemble with large-scale structured reinforcement learning. In *International Conference on Machine Learning*. PMLR, 7435–7469.

Andrew Goldberg, Kavish Kondap, Tianshuang Qiu, Zehan Ma, Letian Fu, Justin Kerr, Huang Huang, Kaiyuan Chen, Kuan Fang, and Ken Goldberg. 2025. Blox-Net: Generative Design-for-Robot-Assembly using VLM supervision, Physics, Simulation, and a Robot with Reset. In *2025 International Conference on Robotics and Automation (ICRA)*. IEEE.

Solomon W Golomb. 1996. *Polyominoes: puzzles, patterns, problems, and packings*. Vol. 16. Princeton University Press.

Branko Grünbaum and Geoffrey Colin Shephard. 1987. *Tilings and patterns*. Courier Dover Publications.

Dan Halperin, Jean-Claude Latombe, and Randall H Wilson. 1998. A general framework for assembly planning: The motion space approach. In *Proceedings of the fourteenth annual symposium on Computational geometry*. 9–18.

Tom Hennigan, Trevor Cai, Tamara Norman, Lena Martens, and Igor Babuschkin. 2020. *Haiku: Sonnet for JAX*. http://github.com/deepmind/dm-haiku

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. 2025. pi0.5: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054* (2025).

Ryan Luke Johns, Martin Wermelinger, Ruben Mascaro, Dominic Jud, Ilmar Hurkxkens, Lauren Vasey, Margarita Chli, Fabio Gramazio, Matthias Kohler, and Marco Hutter. 2023. A framework for robotic excavation and dry stone construction using on-site materials. *Science Robotics* 8, 84 (2023), eabp9758.

Craig S Kaplan. 2009. *Introductory tiling theory for computer graphics*. Morgan & Claypool Publishers.

Craig S Kaplan and David H Salesin. 2000. Escherization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 499–510.

Satoshi Kataoka, Youngseog Chung, Seyed Kamyar Seyed Ghasemipour, Pannag Sanketi, Shixiang Shane Gu, and Igor Mordatch. 2023. Bi-manual block assembly via sim-to-real reinforcement learning. *arXiv preprint arXiv:2303.14870* (2023).

Naoki Kita. 2023. Dissection puzzles composed of multicolor polyominoes. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, e14968.

Naoki Kita and Kazunori Miyata. 2021. Computational design of polyomino puzzles. *The Visual Computer* 37, 4 (2021), 777–787. doi:10.1007/s00371-020-01968-5

Donald E Knuth. 2000. Dancing links. *arXiv preprint cs/0011047* (2000).

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. 2020. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4051–4058.

Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 2009. 3D polyomino puzzle. In *ACM SIGGRAPH Asia 2009 papers*. 1–8.

Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. 2022. Planning assembly sequence with graph transformer. *arXiv preprint arXiv:2210.05236* (2022).

André Menezes, Pedro Vicente, Alexandre Bernardino, and Rodrigo Ventura. 2021. From rocks to walls: a model-free reinforcement learning approach to dry stacking with irregular rocks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2057–2065.

Jaehong Oh, Seungjun Jung, and Sawoong Kim. 2025. Learning to Assemble the Soma Cube with Legal-Action Masked DQN and Safe ZYZ Regrasp on a Doosan M0609. *arXiv preprint arXiv:2508.21272* (2025).

Chi-Han Peng, Caigui Jiang, Peter Wonka, and Helmut Pottmann. 2019. Checkerboard patterns with black rectangles. (2019).

Chi-Han Peng, Helmut Pottmann, and Peter Wonka. 2018. Designing patterns using triangle-quad hybrid meshes. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. 2025. Generating Physically Stable and Buildable Brick Structures from Text. In *ICCV*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. 2012. Recursive interlocking puzzles. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

Luzhe Sun, Takuma Yoneda, Samuel W Wheeler, Tianchong Jiang, and Matthew R Walter. 2025. STACKGEN: Generating Stable Structures from Silhouettes via Diffusion. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 6235–6242.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. 2023. cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation. arXiv:2310.17274 [cs.RO]

The Newton Contributors. 2025. *Newton: GPU-accelerated physics simulation for robotics and simulation research.* https://github.com/newton-physics/newton

Yunsheng Tian, Joshua Jacob, Yijiang Huang, Jialiang Zhao, Edward Gu, Pingchuan Ma, Annan Zhang, Farhad Javid, Branden Romero, Sachin Chitta, Shinjiro Sueda, Hui Li, and Wojciech Matusik. 2025. Fabrica: Dual-Arm Assembly of General Multi-Part Objects via Integrated Planning and Learning. arXiv:2506.05168 [cs.RO] https://arxiv.org/abs/2506.05168

Yunsheng Tian, Karl DD Willis, Bassel Al Omari, Jieliang Luo, Pingchuan Ma, Yichen Li, Farhad Javid, Edward Gu, Joshua Jacob, Shinjiro Sueda, et al. 2024. Asap: Automated sequence planning for complex robotic assembly with physical feasibility. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4380–4386.

Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl DD Willis, and Wojciech Matusik. 2022. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–11.

Jingwen Wang, Wenjun Liu, Gene Ting-Chun Kao, Ioanna Mitropoulou, Francesco Ranaudo, Philippe Block, and Benjamin Dillenburger. 2023. *Multi-Robotic Assembly of Discrete Shell Structures.* 261–274.

Ziqi Wang, Wenjun Liu, Jingwen Wang, Gabriel Vallat, Fan Shi, Stefana Parascho, and Maryam Kamgarpour. 2025. Learning To Assemble With Alternative Plans. *ACM Transactions on Graphics (SIGGRAPH 2025)* 44, 4 (2025). doi:10.1145/3730824

Ziqi Wang, Peng Song, and Mark Pauly. 2018. DESIA: A general framework for designing interlocking assemblies. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Ziqi Wang, Peng Song, and Mark Pauly. 2021. State of the art on computational design of assemblies with rigid parts. In *Computer graphics forum*, Vol. 40. Wiley Online Library, 633–657.

Randall H Wilson. 1992. *On geometric assembly planning.* stanford university.

Hao Xu, Ka Hei Hui, Chi-Wing Fu, and Hao Zhang. 2020. TilinGNN: Learning to tile with self-supervised graph neural network. *arXiv preprint arXiv:2007.02278* (2020).

Yinan Zhang, Yotto Koga, and Devin Balkcom. 2021. Interlocking block assembly with robots. *IEEE Transactions on Automation Science and Engineering* 18, 3 (2021), 902–916.

Yahan Zhou, Shinjiro Sueda, Wojciech Matusik, and Ariel Shamir. 2014. Boxelization: Folding 3D objects into boxes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–8.