Auto(mated)nomous Assembly

Yuxi Liu[1], Boris Belousov[2], Niklas Funk[2], Georgia Chalvatzaki[2], Jan Peters[2], and Oliver Tessmann[1](✉)

[1] TU Darmstadt, Digital Design Unit, Darmstadt, Germany
{liu,tessmann}@dg.tu-darmstadt.de
[2] TU Darmstadt, Intelligent Autonomous Systems, Darmstadt, Germany
{boris,niklas,georgia,jan}@robot-learning.de

Abstract. The paper presents research on a hierarchical, computational design approach for the aggregation of dry-joint, interlocking building blocks and their autonomous assembly by robots. The elements are based on the SL Block system developed by Shen-Guan Shih. The work proposes strategies to assemble multiple SL blocks to form larger aggregations which subsequently turn into building elements on another scale. This approach allows reconsidering the resolution of architectural constructions. Building elements that have previously been considered as solid and monolithic can now be aggregated by many small SL-Blocks. Those dry-joint aggregations allow for easy disassembly and reassembly into different configurations and therefore contribute to a circular reuse of building elements. In order to facilitate such a permanent transformation, the research also includes first steps towards the autonomous assembly of building blocks through a robot including the planning for how to optimally place the parts, as well as ensuring feasible execution by the robot. The goal is a fully autonomous pipeline that takes as input a user-defined, desired shape, and the available building blocks, and directly maps to actions that are executable by the robot. As a result, the desired shape should be optimally resembled through the robot's autonomous actions. The research therefore addresses handling the combinatorial search space regarding the possibilities to combine the available parts, incorporate the constraints of the robot, creating a feasible plan that ensures the stability of the structure at any point in the construction process, avoiding collisions between the robot and the structure, and in the case of SL-Blocks, trying to ensure that the overall structure is interlocking.

Keywords: Hierarchical assembly · SL-Blocks · 3D Polyomino · Dry-joint construction · Autonomous assembly · Reinforcement learning

1 Introduction

1.1 Intelligent Construction and Automation

The construction industry that urgently needs an increase in efficiency and productivity is often meant to be disrupted by borrowing concepts such as vertical integration and prefabrication from other sectors (Daniel 2021).

In contrast, Danial Hall et al. list a set of specific topics for the fragmented AEC network of very diverse stakeholders ranging from small craft producers to large scale construction companies and system providers that need to be developed. Amongst others the authors see a need for "new technical systems that better support manufacturing and assembly activities, capture […] experience and knowledge for continuous improvement" (Hall et al. 2022). Hall et al. furthermore regard the complete move away from on-site production as a misconception of industrializing construction. Assembling building parts will always stay an on-site activity. Historically it had large repercussions on the design of the parts themselves. Bricks for example are dimensioned for manual assembly, including methods for compensating tolerances and dealing with changing material properties (Berge 2007).

Today architecture and the construction industry are confronted with new demands and challenges for the assembly of parts. A lack of skilled labor and the need for affordable housing requires automation of assembly procedures. Resource scarcity and the impact of construction on climate change ask for the reuse of building elements. Thus, dry-joint, reversible constructions and combinatorial versatility for building parts are necessary, so that they exceed the lifespan of a single building in order to be reused in future constructions. At the same time the diverse set of materials within a construction with its various functions such as load bearing, climate regulating, thermal boundary generating etc. and the different producers of such parts require a common interface to form a coherent system.

In this research we address the above listed challenges for construction through the use of the interlocking SL-Block system that is assembled by autonomous robots using machine learning for task and motion planning and tactile sensing for contact-rich assembly procedures and the identification of different materials through the robot. The authors propose to reconsider the scale of building elements and achieve a more circular material use in architecture through small-scale elements aggregated with hierarchical design strategies (Fig. 1).

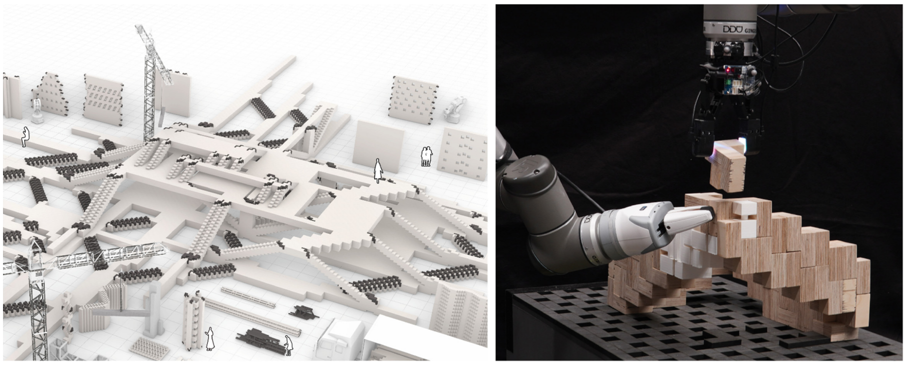


Fig. 1. Left: Construction typologies based on dry-jointed elements. Right: Robotic assembly with visuo-tactile sensors integrated into the gripper (Image: DDU)

2 State of the Art

The concept of dry-joint construction can be traced back to the ancient Inca structures, in which all elements are immobilized by their geometric interface, preventing the assembly from falling apart (Protzen 1985). Contemporary research, involving computational techniques (combinatorial, parametric, generative, algorithmic) and digital fabrication in architecture empowers architects to rethink the entire process chain of the construction system and various possibilities of designing interlocking elements. Notably, the concept of digital materials (Gershenfeld 2012) describes materials as compositions of a set of finite discrete elements with discrete connection details whereby the building components can be reversibly constructed into various configurations responding to architectural needs.

Within such discrete systems, topological interlocking (Dyskin et al. 2001) is a design principle by which building blocks of special shape can form a freely spanning, non-monolithic, reversible load-bearing system only through their geometry and the topology of their jointing. Estrin et al. studied multi-layer interlocking elements while exploring the use of hybrid materials (Estrin et al. 2011). Meanwhile, Mather et al. developed the finite element method (FEM) to investigate the mechanical properties of such systems (Mather et al. 2012). Tessmann presented parametrically designed interlocking elements for rebuilding planar and curved surfaces (Tessmann 2012). Weizmann et al. investigate the correlation between block geometry and structural behavior in flat assemblies of convex interlocking blocks (Weizmann et al. 2021). To enable more flexibility in the design phase of dry-joint elements, researchers from Nanyang Technological University and Tel Aviv University developed a recursive, computational approach for designing interlocking 3D puzzles (Song et al. 2012). In 2017, the same team further developed a computational solution to support the design of a set of interlocking joints that can be connected in different ways to create reconfigurable interlocking furniture (Song et al. 2017). In 2016, Shen-Guan Shih discovered a specific type of polycube called SL-Block. It is an octacube formed by S-and L-shaped tetracubes. Two SL-Blocks can be joined into six different conjugate pairs called SL-Engagements (Fig. 2), each represented with a string (h, a, d, s, t, y) and based on a corresponding geometric transformation (Fig. 3).

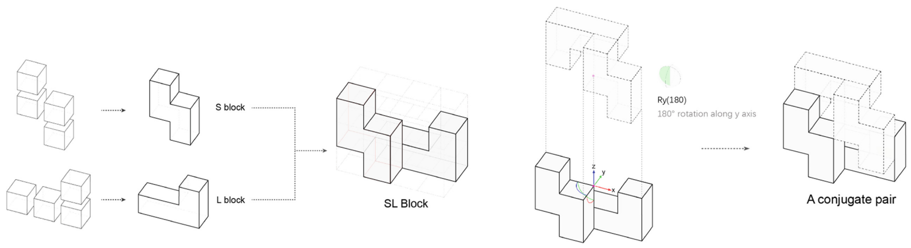


Fig. 2. SL Block and its conjugate pair.

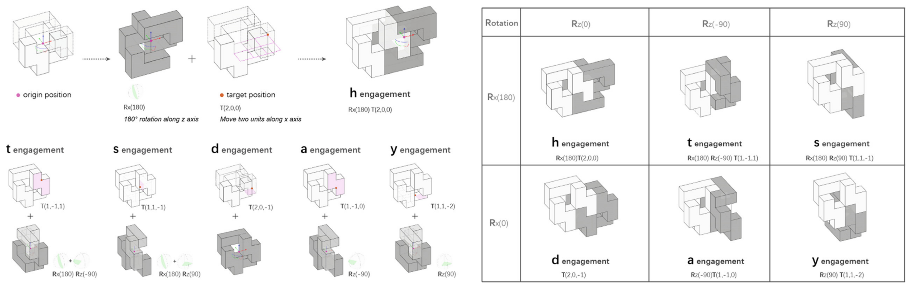


Fig. 3. Assembly Rules for SL Blocks (Image: Yuxi Liu, DDU, 2021)

SL-Engagements can be aggregated into a wide variety of new 3D geometries called SL-Strands. A mathematical representation using polynomial expressions for combinations of SL-Blocks simplifies the notification of the hierarchical construction process of multiple SL-Strands (Shih 2018). Wibranek et al. configured various interlocking building elements (e.g., column, wall, and slab) on the basis of multiple SL-Strands and used reference geometry, such as curves, to sequentially align SL-Blocks along them (Wibranek et al. 2022).

SL-Blocks introduce the potential of 3D polyomino (space filling 3D puzzles) in dry-joint architectural construction. 3D polyomino puzzles are based on initially 2D polyomino. Introduced by Golomb in 1954, a polyomino is a planar geometric figure formed by joining one or more squares edge to edge. The goal of a polyomino tiling problem (Golomb. 1966) is to cover a finite surface using a set of repetitive elements without any overlaps or blanks. 3D polyomino seek to achieve the same in 3d space. They are known to be an NP-complete combinatorial optimization problem. Human designers need to find optimal solutions from randomly generated samples, the number of which has increased exponentially. To help tackle this problem, various probabilistic and approximated algorithms have been explored such as the Parallel Algorithm (Takefuji and Lee 1990), Transfer-Matrix Algorithm (Clisby and Jensen 2012), Quantum Annealing (Eagle et al. 2019). However, tiling a planar figure with a fixed number of polyominoes is a typical combinatorial problem in mathematics. The generated solutions do not directly perform as interlocking architectural structures. To explore the design possibilities of 3D polyominoes for interlocking assemblies, concepts like 3D Polyomino Puzzle (Lo et al. 2009), Recursive Interlocking Puzzles (Song et al. 2012), Reconfigurable Interlocking Furniture (Song et al. 2017) develop top-down computational approaches to design interlocking parts for assembly.

Besides the structural and combinatorial difficulties, fully autonomous robotic assembly additionally requires the automation of both task and motion planning, i.e., determining the order of assembly actions and the execution trajectories, respectively. Assembling architectural structures is a long-horizon manipulation task consisting of several stages of decisions and sub-task executions. Classical approaches for task-and-motion planning (TAMP) typically rely on a complete problem description, including kinematics and dynamics (Toussaint 2015; Kaelbling and Lozano-Pérez 2010). As these fully-optimization based approaches are usually computational expensive, lately, methods combining classical TAMP and learned heuristics have been proposed (Garrett et al.

2016, Driess et al. 2021). Notably, Hartmann et al. developed a multi-agent TAMP framework for handling the problem of constructing a large pavilion from many geometrically unique building elements while using multiple robots (Hartmann et al. 2020). Garret et al. present a formulation for 3D robotic spatial extrusion taking into account structural stiffness and potential collisions between the robot and the new structure (Garrett et al. 2020). Contrary to the previous approaches, there also exist many works that approach the problem through end-end learning, thereby directly mapping from problem definition to low-level actions without requiring any additional modeling. Instead, a simulation is required to investigate the action's effects for steering the training process. Exemplarily, (Bapst et al. 2019) train policies for 2D construction problems using graph representations and reinforcement learning. Wibranek et al. use Reinforcement Learning (RL) to assemble a sequence of SL-Blocks into an interlocking structure (Wibranek et al. 2021). Most recently, Funk et al. presented a first method that jointly tackles the problem of combinatorial optimization for block placement and robotic TAMP through combining graph-based reinforcement learning and Monte Carlo Tree Search. Therefore, this method is capable of autonomously reassembling arbitrary structures (predefined by a human), with a robot, given a set of available polycubes (Funk et al. 2022a).

3 SL Blocks for Architectural Design and Construction

In our research we are developing dry-joint constructions in which aggregations of elements unfold structural behavior through interlocking. Elements are constrained in their rotational and translational freedom by neighboring elements. Interlocking is different from merely stacking or layering elements in that the connections between the elements lock various degrees of freedom (DoF) allowing for not only compression-active constructions. Dry-jointed interlocking elements can be easily assembled, disassembled and reassembled for a circular use in construction. SL Blocks are of particular interest as they can be assembled in many different configurations. We study the six SL-Engagements through physical models with SL-Blocks made from different materials and explore the dry-joint construction process of assembling SL Block into a branching column that can be used as a load-bearing structure for a roof. (Fig. 4) Through one-directional scaling we design elements that become interfaces between the SL-Block system and monolithic building elements (Fig. 5).

Subsequently, we explore two main challenges for architectural design with SL Blocks with a computational approach: i) Automating the currently manual and tedious string input to generate interlocking SL-Strands that follow the compositional rules. ii) Develop hierarchical design strategies to systematically aggregate those SL-Strands into architectural constructions.

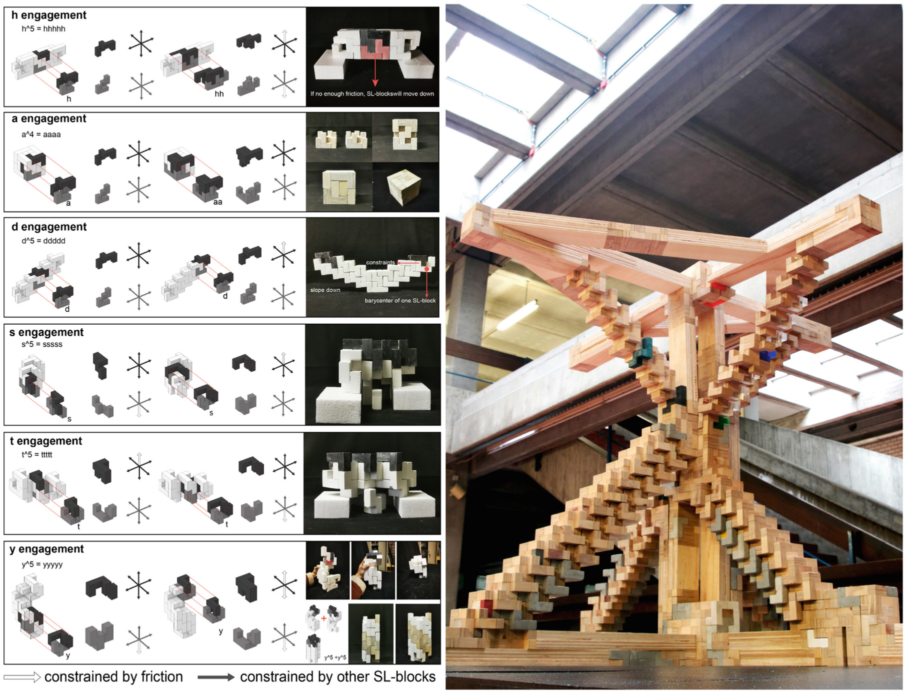


Fig. 4. Left: Constraint analysis and physical models of repetitively assembled SL-Engagements. Right: A branching column assembled with SL-Blocks made from different materials.

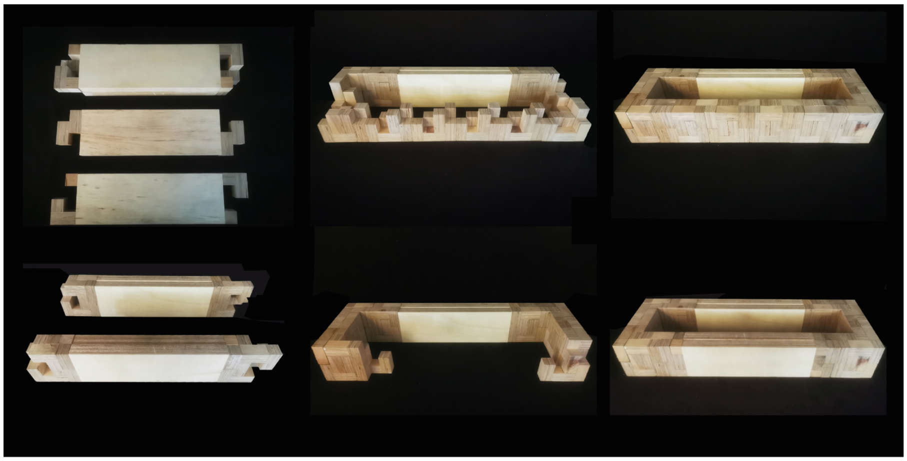


Fig. 5. Left: Interfaces for connecting SL-Blocks. Right: Process of assembling interface and SL-Blocks into closed-loop SL-Strands

3.1 Hierarchical Design Strategy

Hierarchical classification is an effective approach to understanding interdependencies between elements of the interlocking system and bringing order to the design space. Shih first introduced the strand hierarchy where the lowest level of interlocking SL Strands is built upon the conjugate pair (Shih 2016). In addition to Shih's hierarchy, we reveal the geometrical hierarchy within the conjugate pair of SL Block: Cube, three Tetracubes, and three Octacubes. As shown in Fig. 6, three types of Tetracube can recursively composite the same form of a conjugate pair. Furthermore, we add two more levels that will be defined below: SL-Strand Merge, and SL-Strand Aggregation (Fig. 7).

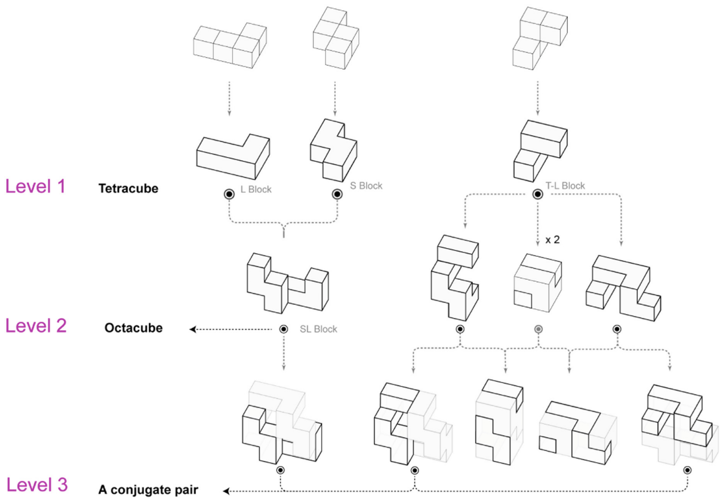


Fig. 6. A hierarchical classification of the conjugate pair of SL Block

Level 3–4 shows the above-described SL-Block engagement strategy and their representation as a transformation matrix.

Level 4 shows SL-Strands forming shapes that resemble building components such as beams, columns, frames, and bracings. However, those elements are not monolithic but assembled from a multitude of dry-joint SL-Blocks that can be easily disassembled and reassembled into other shapes.

Level 5 shows SL-Strand Aggregations: SL-Strands from Level 4 assembled into larger geometric and tectonic systems. SL-Strand Aggregations no longer rely on the intricate interlocking logic of SL-Blocks and therefore offer an interface to more conventional constructions. Three connections are developed: Joinery connections links to traditional construction, nested and borromean connection are rather mathematical concepts (D Auckly 2020) that we are aiming to investigate in terms of applicability within combinatorial assemblies.

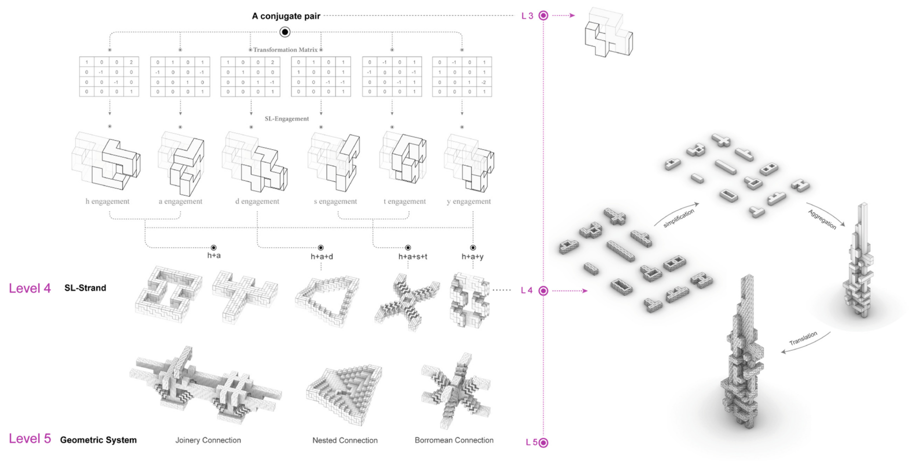


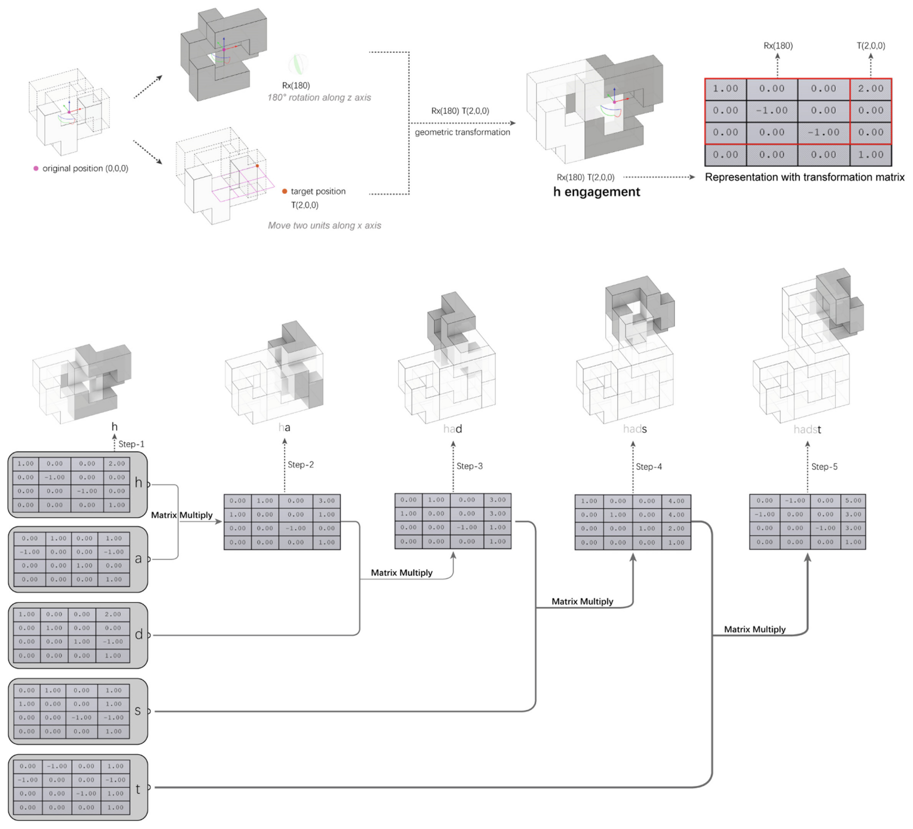
Fig. 7. SL Strand Merge and SL Strand Aggregation

3.2 Automate Generation of SL-Strands

The generation of SL-Strands is based on the sequential assembly of six SL-Engagements, each standing for a type of geometric transformation that can be represented by a transformation matrix. The transformation matrix defines the position and orientation of the elements within the assembly. Since the assembly process starts with one determined initial engagement and adds other engagements one by one, we can consider the connection rule of each step as a new transformation matrix that results from the multiplication between the matrix of the currently used SL-engagement and the matrix of the previous step. Therefore, the string sequence for generating SL-Strands is equal to a sequence of the transformation matrix. (Fig. 8).

Defining the number of steps and the type of SL-Engagements, we use brute-force search to find all possible assemblies. However, not all resulting SL-Strands are applicable to architectural design. We chose those SL-Strands that form a closed system, in which all elements are kinematically constrained by each other. Thus, the last SL-Engagement must interlock with the first SL-Engagement to close the loop. Closing the loop of a SL-Strands is important to ensure that the resulting element is compatible with more conventional construction elements by interfacing with planar surfaces (see Fig. 4: Level 4 shapes)..

This problem can be seen as how to find the Hamilton Cycle(closed-loop) within a directed graph (Fig. 9). We extract the center points of the first-placed and last-placed SL-Engagements from all possibilities and choose only those results in which their coordinates are the same. Although successful in automatically finding the closed SL-Strands from SL-Engagements, this approach is a unidirectional search algorithm that only allows finding the optimal results assembled by less SL-Engagements. In a Unidirectional search algorithm, if we want to find closed SL-Strands from the assembly of n conjugate Pairs using h and a SL-Engagements. First, we need to generate all the possibilities, the number of which is 2^n. The code was written in python. Even if we



Representation with transformation matrix:

1.00	0.00	0.00	2.00
0.00	-1.00	0.00	0.00
0.00	0.00	-1.00	0.00
0.00	0.00	0.00	1.00

h

1.00	0.00	0.00	2.00
0.00	-1.00	0.00	0.00
0.00	0.00	-1.00	0.00
0.00	0.00	0.00	1.00

a

0.00	1.00	0.00	1.00
-1.00	0.00	0.00	-1.00
0.00	0.00	1.00	0.00
0.00	0.00	0.00	1.00

d

1.00	0.00	0.00	2.00
0.00	1.00	0.00	0.00
0.00	0.00	1.00	-1.00
0.00	0.00	0.00	1.00

s

0.00	1.00	0.00	1.00
1.00	0.00	0.00	1.00
0.00	0.00	-1.00	-1.00
0.00	0.00	0.00	1.00

t

0.00	-1.00	0.00	1.00
-1.00	0.00	0.00	-1.00
0.00	0.00	-1.00	1.00
0.00	0.00	0.00	1.00

Step-2

0.00	1.00	0.00	3.00
1.00	0.00	0.00	1.00
0.00	0.00	-1.00	0.00
0.00	0.00	0.00	1.00

Step-3

0.00	1.00	0.00	3.00
1.00	0.00	0.00	3.00
0.00	0.00	-1.00	1.00
0.00	0.00	0.00	1.00

Step-4

1.00	0.00	0.00	4.00
0.00	1.00	0.00	4.00
0.00	0.00	1.00	2.00
0.00	0.00	0.00	1.00

Step-5

0.00	-1.00	0.00	5.00
-1.00	0.00	0.00	3.00
0.00	0.00	-1.00	3.00
0.00	0.00	0.00	1.00

Fig. 8. Transformation matrix and SL-Engagements in the process of being assembled.

implement the code in C++, which would increase the computational speed, the combinatorial search space of 2^n still remains a severely limiting factor (i.e., considering 100 SL blocks there are 2^{100} possibilities that have to be explored).

In order to increase the calculation efficiency and the number of assembled blocks, we proposed a bidirectional search approach and extended the initial six SL-Engagements to twelve types through the inverse of the transformation matrix. SL-Strands can be generated by simultaneously assembling SL-Engagements along with clockwise and anti-clockwise directions (Fig. 10).

Compared with the unidirectional search approach, bidirectional search replaces a single search graph with two smaller sub graphs. In each direction, we only have to explore a combinatorial search space of $2^{(n/2)}$ and can then afterwards efficiently compare whether the end-points of both directions match, which significantly improves the computational speed and the tractability of the algorithm. The use of bidirectional significantly improves the computational efficiency of generating SL-Strands (Fig. 10).

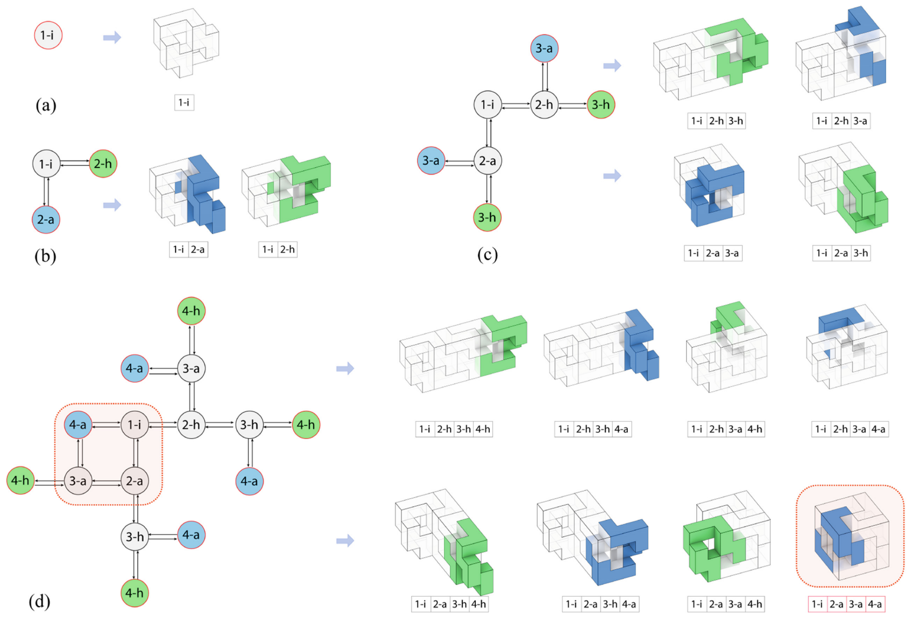
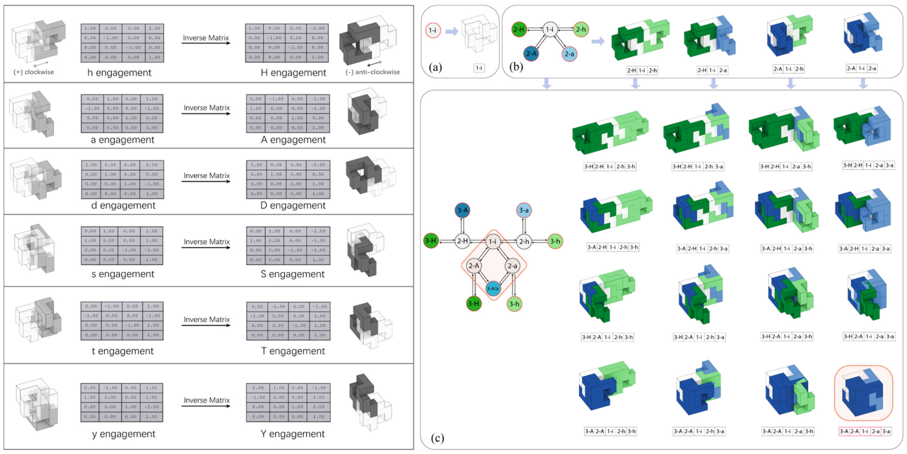


Fig. 9. Unidirectional example graphs showing the process of finding the closed SL-Strands from all possible results of assembling four conjugate pairs with h and a SL-Engagements. (the node of the graph represents the transformation matrix of the current assembled SL-Engagement, the edge of the graph represents the interlocking relation between SL-Engagements).



h engagement

1.00	0.00	0.00	2.00
0.00	-1.00	0.00	0.00
0.00	0.00	-1.00	0.00
0.00	0.00	0.00	1.00

H engagement

1.00	0.00	0.00	-2.00
0.00	-1.00	0.00	0.00
0.00	0.00	-1.00	0.00
0.00	0.00	0.00	1.00

a engagement

0.00	1.00	0.00	1.00
-1.00	0.00	0.00	-1.00
0.00	0.00	1.00	0.00
0.00	0.00	0.00	1.00

A engagement

0.00	-1.00	0.00	-1.00
1.00	0.00	0.00	-1.00
0.00	0.00	1.00	0.00
0.00	0.00	0.00	1.00

d engagement

1.00	0.00	0.00	2.00
0.00	1.00	0.00	0.00
0.00	0.00	1.00	-1.00
0.00	0.00	0.00	1.00

D engagement

1.00	0.00	0.00	-2.00
0.00	1.00	0.00	0.00
0.00	0.00	1.00	1.00
0.00	0.00	0.00	1.00

s engagement

0.00	1.00	0.00	1.00
1.00	0.00	0.00	1.00
0.00	0.00	-1.00	-1.00
0.00	0.00	0.00	1.00

S engagement

0.00	1.00	0.00	-1.00
1.00	0.00	0.00	-1.00
0.00	0.00	-1.00	-1.00
0.00	0.00	0.00	1.00

t engagement

0.00	-1.00	0.00	1.00
-1.00	0.00	0.00	-1.00
0.00	0.00	-1.00	1.00
0.00	0.00	0.00	1.00

T engagement

0.00	-1.00	0.00	-1.00
-1.00	0.00	0.00	1.00
0.00	0.00	-1.00	1.00
0.00	0.00	0.00	1.00

y engagement

0.00	-1.00	0.00	1.00
1.00	0.00	0.00	1.00
0.00	0.00	1.00	-2.00
0.00	0.00	0.00	1.00

Y engagement

0.00	1.00	0.00	-1.00
-1.00	0.00	0.00	1.00
0.00	0.00	1.00	2.00
0.00	0.00	0.00	1.00

Fig. 10. Left: The twelve types of SL-Engagements and their corresponding transformation matrix. Right: Bidirectional example graphs showing the process of finding the closed SL-Strands assembled by four conjugate pairs with H, h, A and a SL-Engagements.

3.3 Towards Fully Autonomous Construction – from Shape Input to Robotic Assembly

Parallel to the exploration of design strategies for discrete, dry-joint elements we aim at automating the assembly of those elements with autonomous robots. The above shown methods still depend on hand-crafted heuristics and do not consider the robot-in-the-loop, i.e., the robotic execution of the part placements. Yet, the ultimate goal of autonomous assembly would be a fully autonomous pipeline, that takes as input a user-defined, desired shape, and the available building blocks, and directly maps to actions that are executable by the robot. As a result, the desired shape should be optimally resembled through the autonomous actions by the robot. To achieve this, we need to come up with an approach that is capable of i) handling the combinatorial search space regarding the possibilities to combine the available parts, ii) taking the constraints of the robot into account, such as for instance its limited workspace, iii) creating a feasible plan that ensures the stability of the structure at any point in the construction process, iv) avoiding collisions between the robot and the structure, and v) in the case of SL-Blocks, trying to ensure that the overall structure is interlocking.

To handle all these different levels of complexity altogether, we propose an approach based on model-free reinforcement learning, in particular, Q-learning (Mnih et al. 2013). Thereby, we avoid having to create an analytical model that would be capable of capturing all the desires, but instead directly try to predict the quality of all the actions that are currently available, and choose to execute the best one. Nevertheless, this necessitates a simulation environment (cf. Figure 11), as initially, the quality of the actions are unknown, and have to be learned during the so-called training process. Exemplarily, the quality of action a, given current state s, is given by $Q(s,a) = r(s,a) + \max(a')$ $Q(s',a')$, where r(s,a) is the immediate reward of executing action a in state s (i.e., assigning a positive reward if the filling of the desired shape increased and the action was successful, and a negative one otherwise), and $\max(a')Q(s',a')$ represents the value of the best available action in the state s' that we end up in, thereby, taking the taks's long horizon into account, as we can only judge at the end whether all the actions optimally resemble the desired target shape. For predicting the quality of the actions, we make use of a graph-based neural network (cf. Figure 12) (Scarselli et al. 2008). As shown in the figure, we discretize the desired structure that is to be built, and also represent all the available building blocks through the voxels that they are made up of. This allows us to define a graph, with nodes, representing the centers of the respective voxels. To predict the action values, we first exchange information between all the nodes in the graph to get a better understanding of the assembly scene, and finally predict all the values for moving one of the voxels of an unplaced block, to a position in the grid that is still unoccupied. Despite the fact that it is straightforward to represent the assembly scene as a graph, graph-based representations come at the additional advantage of being invariant to the problem size as the exchange of information is defined locally between nodes. Thus, the absolute number of nodes is irrelevant. Therefore, we can use the same learned graph-representation to handle problems with different sets of available building blocks and desired target shapes.

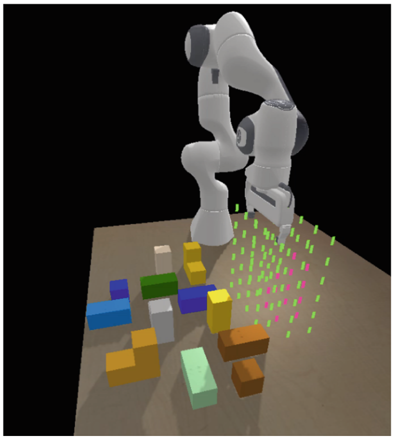


Fig. 11. Illustration of the robotic assembly simulation environment, with the set of available building blocks on the left hand side, a robotic manipulator (Franka Panda), and the voxelized target shape. The pink voxels should be occupied during the construction process, while the green voxels should be left empty. Figure 13 illustrates the sequential robotic assembly using the trained policy.

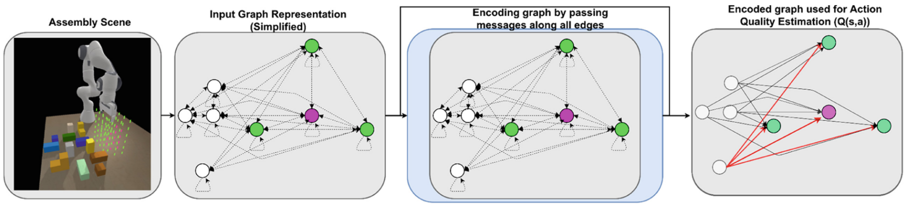

Assembly Scene
Input Graph Representation (Simplified)
Encoding graph by passing messages along all edges
Encoded graph used for Action Quality Estimation (Q(s,a))

Fig. 12. Depicting the process of decision making using the graph neural network. Given the assembly scene, we create a graph based representation of it. Note that we only display a simplified version of the assembly scene in which there are 4 voxels (3 green ones that should remain empty, and 1 pink one that should be filled), and two building blocks (an L-shaped block, visualized by the 3 interconnected white nodes in the middle left, and a single block, visualized by the single node in the bottom left). Next follow some rounds of message passing in which the nodes exchange information along the edges to create a more global understanding of the assembly scene. In the last step, the encoded graph is obtained for action selection. With the red arrows we exemplarily visualize predicting the values for all the actions of placing the single block at all the unoccupied spaces. Note, we also estimate the values for all the other white nodes that correspond to voxels of building blocks that are not placed yet. Lastly, we execute the action with the highest value and then repeat the process.

After having learned this graph-based representation, Fig. 13 illustrates one rollout of the learned policy in which a desired structure is built from the rectilinear poly-cubes. Yet, in its current form, the goal of the overall structure to be interlocking is not taken into account. Note however that the algorithm is fully autonomous, i.e., it only receives as the input the desired target shape and the available building blocks, and the rest, i.e., the part placement and robotic actions are exclusively handled by the previously presented algorithm. For more details on graph-based reinforcement learning for autonomous assembly, please see (Funk et al. 2022b).

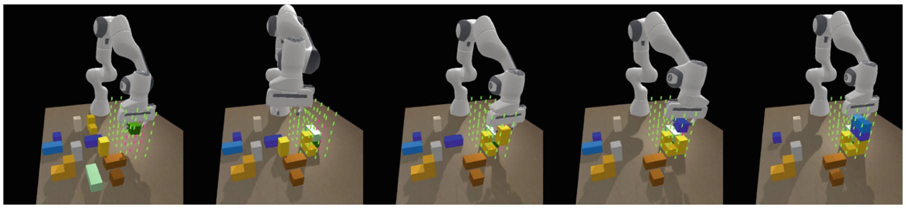


Fig. 13. Illustration of the step-wise assembly process of the structure shown in Fig. 11. As shown, the autonomous assembly agent successfully combines 6 building blocks to optimally recreate the desired, predefined structure.

4 Conclusion and Future Work

In this work, we have presented hierarchical design approaches enabling the use of SL blocks in architectural design and construction processes. The developed tools aim at identifying how to combine multiple SL blocks together to form larger structures. As presented in the paper, this allows the building of larger installations entirely from SL-blocks, as well as the seamless integration of elements made from SL blocks with conventionally manufactured ones, which is highly desirable as these interlocking elements can be disassembled and reassembled to something new, thereby making construction more resource efficient and going towards the concept of a circular economy.

Future research is required to develop these tools further. These investigations should explore the hierarchical structural systems built with SL Blocks on different scales. The complex combinatorics of aggregating SL Blocks into stable building structures are currently under investigation to be automated with recursive searching and learning algorithms. The structural behavior and analysis of such aggregations will be examined with the Finite Element Method and Discrete Element Method. The computational simulation enables us to understand the forces and movement of individual SL blocks within a discrete structural system, rationalizing the material distribution of SL blocks.

Finally, we have also presented a method that is capable of handling all the inherent difficulties of fully autonomous construction, thereby handling both the planning for how to optimally place the parts, as well as ensuring feasible execution by the robot. In the future, this method has to be extended to allow for building more complex structures consisting of more parts, and in particular, SL-Blocks, as well as being capable of reasoning about the interlocking of the entire structure by incorporating insights from recursive interlocking puzzles.

Acknowledgment. This research is funded by the Bundesinstitut für Bau-, Stadt- und Raumforschung on behalf of the Bundesministeriums des Innern, für Bau und Heimat with funds from the Zukunft Bau research grant programme. Niklas Funk acknowledges the support from the Artificial Intelligence in Construction (AICO) grant by the Nexplore/Hochtief Collaboration Lab at TU Darmstadt.

References

Auckly, D.: Folklore, the Borromean rings, the icosahedron, and three dimensions. Math. Mag. **93**(5), 330–342 (2020)

Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., Hamrick, J.: Structured agents for physical construction. In: International Conference on Machine Learning (2019)

Berge, B.: Ecology of Building Materials. Routledge (2007). https://doi.org/10.4324/9780080504988

Clisby, N., Jensen, I.: A new transfer-matrix algorithm for exact enumerations: self-avoiding polygons on the square lattice. J. Phys. A: Math. Theor. **45**(11), 115202 (2012)

Daniel, D.: Katerra's $2 billion legacy. Architect Magazine. https://www.architectmagazine.com/technology/katerras-2-billion-legacy_o (2021). Last Accessed 01 May 2022

Driess, D., Ha, J.S., Toussaint, M.: Learning to solve sequential physical reasoning problems from a scene image. Int. J. Robot. Res. **40**(12–14), 1435–1466 (2021)

Dyskin, A.V., Estrin, Y., Kanel-Belov, A.J., Pasternak, E.: Toughening by fragmentation—how topology helps. Adv. Eng. Mater. **3**(11), 885–888 (2001)

Eagle, A., Kato, T., Minato, Y.: Solving tiling puzzles with quantum annealing. In: arXiv preprint (2019)

Estrin, Y., Dyskin, A.V., Pasternak, E.: Topological interlocking as a material design concept. Mater. Sci. Eng., C **31**(6), 1189–1194 (2011)

Funk, N., Chalvatzaki, G., Belousov, B., Peters, J.: Learn2Assemble with Structured Representations and Search for robotic architectural construction. In: Conference on Robot Learning (2022a)

Funk, N., Menzenbach, S., Chalvatzaki, G., Peters, J.: Graph-based reinforcement learning meets mixed integer programs: an application to 3D robot assembly discovery. In: arXiv preprint (2022b)

Garrett, C., Huang, Y., Lozano-Perez, T., Mueller, C.: Scalable and probabilistically complete planning for robotic spatial extrusion. In: Proceedings of Robotics: Science and Systems (2020)

Garrett, C., Kaelbling, L., Lozano-Pérez, T.: Learning to rank for synthesizing planning heuristics. In: IJCAI (2016)

Gershenfeld, N.: How to make almost anything: the digital fabrication revolution. Foreign Aff. **91**, 43 (2012)

Golomb, S.W.: Tiling with polyominoes. J. Comb. Theory **1**(2), 280–296 (1966)

Hall, D.M., Lessing, J., Whyte, J.: New business models for industrialized construction. In: Bolpagni, M., Gavina, R., Ribeiro, D. (eds.) Industry 4.0 for the Built Environment. SI, vol. 20, pp. 297–314. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-82430-3_13

Hartmann, V., Oguz, O., Driess, D., Toussaint, M., Menges, A.: Robust task and motion planning for long-horizon architectural construction planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020)

Kaelbling, L., Lozano-Pérez, T.: Hierarchical planning in the now. In: Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)

Mather, A., Cipra, R., Siegmund, T.: Structural integrity during remanufacture of a topologically interlocked material. Int. J. Struct. Integr. **3**(1), 61 (2012)

Mnih, V., et al.: Playing atari with deep reinforcement learning. In: arXiv preprint (2013)

Protzen, J.P.: Inca quarrying and stonecutting. J. Soc. Archit. Hist. **44**(2), 161–182 (1985)

Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)

Shih, S.G.: On the hierarchical construction of sl blocks–a generative system that builds selfinterlocking structures. Advances in Architectural Geometry, 3778–4 (2016)

Shih, S.G.: The art and mathematics of self-interlocking SL blocks. In: Proceedings of Bridges 2018: Mathematics, Art, Music, Architecture, Education, Culture (2018)

Song, P., Fu, C.W., Cohen-Or, D.: Recursive interlocking puzzles. ACM Trans. Graph. (TOG) **31**(6), 1–10 (2012)

Song, P., et al.: Reconfigurable interlocking furniture. ACM Transactions on Graphics (TOG) **36**(6), 1–14 (2017)

Takefuji, Y., Lee, Y.C.: A parallel algorithm for tiling problems. IEEE Trans. Neural Netw. **1**(1), 143–145 (1990)

Tessmann, O., Rossi, A.: Geometry as interface: parametric and combinatorial topological interlocking assemblies. J. Appl. Mech. **86**(11) (2019)

Tessmann, O.: Topological interlocking assemblies. In: Physical Digitality—Proceedings of the 30th International Conference on Education and Research in Computer Aided Architectural Design in Europe (2012)

Toussaint, M.: Logic-geometric programming: an optimization-based approach to combined task and motion planning. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)

Weizmann, M., Amir, O., Grobman, Y.J.: The effect of block geometry on structural behavior of topological interlocking assemblies. Autom. Constr. **128**, 103717 (2021)

Wibranek, B., Liu, Y., Funk, N., Belousov, B., Peters, J., Tessmann, O.: Reinforcement learning for sequential assembly of SL-blocks-self-interlocking combinatorial design based on machine learning. In: Proceedings of the 39th eCAADe Conference (2021)

Zhanat, K., Corrales, J.A., Perdereau, V.: Tactile sensing in dexterous robot hands. Robot. Auton. Syst. **74**, 195–220 (2015)