Interlocking block assembly

Yinan Zhang and Devin Balkcom

Dartmouth College

Abstract. This paper presents a design for interlocking blocks and an algorithm that allows these blocks to be assembled into desired shapes. During and after assembly, the structure is kinematically interlocked if a small number of blocks are immobilized relative to other blocks. There are two types of blocks: cubes and double-height posts, each with a particular set of male and female joints. Layouts for shapes involving thousands of blocks have been planned automatically, and shapes with several hundred blocks have been built by hand. As a proof of concept, a robot was used to assemble sixteen blocks. The paper also describes a method for assembling blocks in parallel.

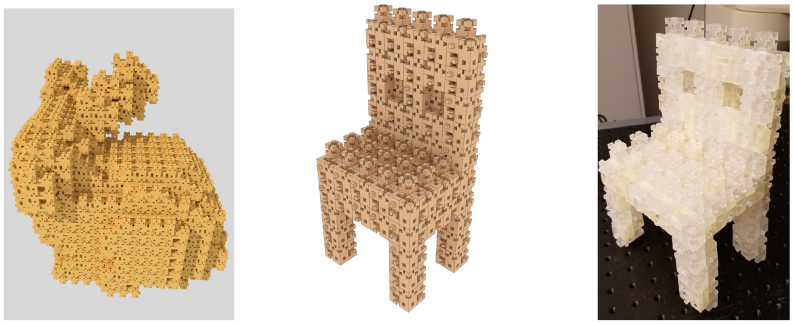
1 Introduction

The goal of the work described in this paper is to enable robotic assembly of large structures from blocks that interlock without the need for glue, cement, screws, or other connectors. Figure 1 shows two models for which layouts and assembly plans were generated automatically by the presented algorithm. The motion of blocks are constrained by joints; later blocks reinforce and immobilize prior blocks. Each structure has a few blocks that can move, called keys; the bunny has two keys, in the ears, and the chair has one, at the top of the back of the chair. If the keys are immobilized, the structure is rigidly interlocked.

Kinematic interlock presents some advantages over traditional connection methods such as glue, cement, screws, nails, or friction locks. The interlocks may be structurally strong, allow simple assembly by robots, allow disassembly and re-use of the components, and may be suitable for underwater or other environments where adhesives are ineffective. Relative to 3D printing, fabricating in parts may present some advantages. Individual components may be fabricated efficiently, packed for storage and transport, repaired or replaced as needed, and allow design changes.

The algorithm described in this paper takes a voxelized 3D model as input, and finds an assembly plan such that the interlocked structure covers the specified voxels. There are two types of block: $1 \times 1 \times 1$ *cubes*, and $1 \times 1 \times 2$ *posts*, with connectors arranged in a particular way. The assembly requires only translation motions.

The concept of interlocking block assembly was previously presented by the authors in [28]. However, the technical work in the current paper is effectively entirely new. New block designs and layout algorithms enabled the reduction of the types of blocks needed from nine to two, and have allowed structures that appear to be more robust and easier to assemble. The paper also explores construction of physical structures much larger than previously built (406 pieces compared to 64), as well as a more convincing demonstration of robotic assembly. New theoretical contributions include an analysis of how blocks may be assembled in parallel, speeding up assembly.



(a) Stanford Bunny assembled in simulation.

(b) Chair model after automatic layout.

(c) Chair assembled by hand.

Fig. 1: Models assembled

2 Limitations

This paper focuses almost entirely on the geometry of a particular design of blocks, and associated layout algorithms. As such, many critical issues that would need to be solved for a practical system have been neglected. Chief among these is the need for analysis of the rigidity and robustness of the final structures. The physical experiments conducted use 3D printed blocks, and are no more than a proof of concept, using a very small number of blocks.

Although the algorithm presented can build essentially arbitrary voxelized structures, overhanging components of layers are not interlocked until a second identical layer is placed above. This means that some external (though temporary) means of support is needed during construction, just as in 3D printing. We believe that modifications to the algorithm and block designs would allow overhanging layers to be build to be stably interlocked during construction, but have not made these modifications.

We would also like to gain a better understanding of how to design and analyze block types and layout algorithms. Effectively, the block types designed are the result of trial-and-error and creative thought; we do not present significant mathematical or mental tools to find other block designs.

3 Related work

Interlocking structures have a long history. Wood joints such as the dovetail and mortise and tenon are used in carpentry around the world; in China and Japan, complex interlocking designs have permitted the construction of wooden buildings with no screws or nails [29]; In the paleontology community, evidence has recently been presented that supports a hypothesis that the backbones of theropod dinosaurs interlocked to provide support for the extremely large body mass [23].

The present work is closest in spirit to Song *et al.* [16, 17, 15, 6, 19] which consider the problem of designing reusable components to be assembled into different forms relying on geometric constraints; the primary contribution of the current work is a universal block design and layout algorithm that allows construction of arbitrary geometries.

Yao *et al.* [24] proposed a method for interactively designing joints for structures and analyzing the stability. Kong *et al.* applied curve matching techniques for finding solutions for assembly of 2D and 3D interlocking puzzles; the layout algorithms considered in the current paper generate assembly motions together with the design.

Robotic construction research dates back to the 1990s [1] when Andres *et al.* created a prototype, ROCCO, capable of gripping and laying bricks. The same robotic system was later applied to site assembly operations by Balaguer *et al.* [4]. More recent works include DimRob, a system with an industrial robot arm mounted on a mobile platform (Helm *et al.* [8]) used for construction tasks. This prototype was later developed into a mobile robot, In situ Fabricator, for construction at 1:1 scale (Giftthaler *et al.* [7]).

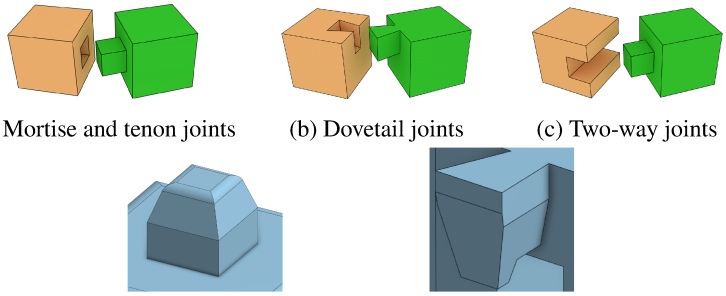
Willmann *et al.* [22], for example, used autonomous flying vehicles to lift and position small building elements. Augugliaro *et al.* [3] demonstrated a system of multiple quadrocopters precisely laying out foam blocks forming a desired shape. Lindsey *et al.* [10] built cubic structures using quadrocopters. Augugliaro *et al.* [2] explored another approach of construction: quadcopters assembled a rope bridge capable of supporting people. Keating *et al* [9] built a large mobile 3D printer using a robot arm to extrude adhesive materials.

Instead of focusing on the robot control system to carry building elements, some researchers designed new building elements. Rus and Vona [13] developed Crystalline, a modular robot with a 3-DoF actuation mechanism allowing it to make and break connections with other identical units; a set of such robots form a self-reconfigurable robot system. White *et al.* [21] introduced two three-dimensional stochastic modular robot systems that are self reconfigurable and self assemble-able by successive bonding and release of free-floating units. Romanishin *et al.* [11] proposed a momentum-driven modular robot. SamBot, a cube shaped modular robots with rotation mechanism was introduced by Wei *et al.* [20]. Daudelin *et al.* [5] present a self reconfigurable system that integrates perception, mission planning, and modular robot hardware. Tosun *et al.* [18] created a design framework for rapid creation and verification of modular robots.

Inspired by LEGO, Schweikardt *et al.* [14] proposed a robotic construction kit, roBlocks, with programmable cubic blocks for educational purpose. Kilobot, a swarm of 1000 crawling mobile robots, was introduced by Rubenstein *et al.* [12], along with algorithms for planning mechanisms allowing kilobots to form 2D shapes.

4 Interlocking blocks and constraint graph

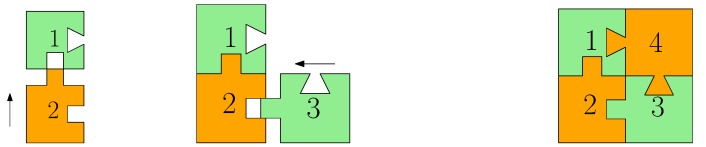
A block is a rigid body that has male and female joints allowing assembly with other blocks, typically by sliding one block against the other using a simple translation. Figure 2 shows three different joint pairs that may connect blocks in the system we describe in the paper. A *mortise and tenon* joint pair (Figure 2a) allows blocks to be disassembled only in the non-penetrating normal direction of the contact surface. A *dovetail* joint pair (Figure 2b) allows block motion only in a particular tangential direction. The third joint pair we use in the current design is a *two-way joint* (Figure 2c), which allows motions of associated blocks in both normal and tangential directions. The dovetail and mortise and tenon joints fully constrain rotational motion, but the two-way joint permits one rotational degree of freedom. We manufactured the joints so the front part in the in-



(a) Mortise and tenon joints (b) Dovetail joints (c) Two-way joints

(d) Tenon joint design. (e) Dovetail joint design.

Fig. 2: Three different joint pairs and detailed design.



(a) Tenon joint. (b) Tenon joint. (c) Dovetail joint assembled from top down.

Fig. 3: Assembling an interlocking 4-block square. Arrow indicates the assembly direction. Block 4 is the key.

sertion direction is thiner thus reduce surface contacts, providing better error tolerance. See Figure 2d and 2e.

Blocks are assembled into a structure in order, and the last block assembled can be removed by reversing the most recent translation assembly motion. Therefore, the last block assembled must be *attached* to the structure using glue, friction, a screw, or some other external method; we call such a block a *key*.

Figure 3 shows a 2D projection of an interlocking structure assembled using blocks with dovetail and mortise and tenon joints. First, block 2 is assembled to block 1, using a tenon joint on the top of the blocks, and moving block 2 in the positive y direction, assuming a coordinate frame aligned with the page. Block 3 then slides in and connect with block 2 with another tenon joint. The final block is assembled from top down, connecting block 1 and 3 using two dovetail joints and limiting block 2 and 3 to move in y negative or x positive directions.

4.1 The constraint graph

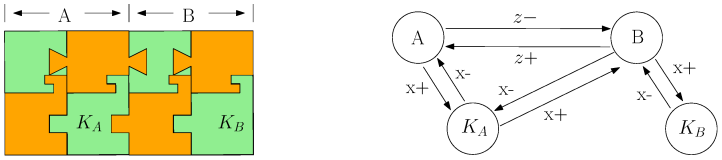
To better understand how joints constrain motions, we represent a structure using a directed graph. Each vertex in the graph represents a block. A pair of directed edges is added between vertices corresponding to blocks that are in contact; $w(e_{i,j})$ denotes the set of permitted motions of j relative to i.

Consider a partition of the graph into some non-overlapping subsets of vertices. A partition is *separable* if there exists a motion that satisfies constraints by all in-edges along the boundary of the subset of vertices. In this particular work, the block and joint design limit motions of every block to translations directly along axes, simplifying the

(a) A 2×2 interlocking structure. Numbers indicate the order. Block 4 is the key.

(b) Graph representation. Each edge allows some motions for associated blocks.

Fig. 4: A four-block interlocking structure and its graph representation.



(a) Two interlocking substructures connected by a dovetail joint and a mortise & tenon joint.

(b) Forming a larger interlocked structure from two interlocked structures.

Fig. 5: Any interlocking substructure can be viewed as two nodes in the graph, which simplifies the graph representation.

analysis. (The two-way joint is used only as an auxiliary connection for blocks whose motions are already constrained to pure axis-aligned translations by other joints.)

Figure 4b shows an example of a constraint graph for the previous example of a four-block interlocking structure. Consider partitioning the structure into two parts $\{1,2\}$ and $\{3,4\}$. These parts are inseparable, since $w(e_{2,3}) \cap w(e_{1,4}) = \{x+\} \cap \{z+\} = \emptyset$. By checking more partitions of the structure, we find only $\{1,2,3\}$ and $\{4\}$ are separable. If block 4 is attached to either of its neighbors, the structure is rigid.

We call a structure *k-interlocked*, if when k keys are attached to neighbors, no partition is separable. The example structure is 1-interlocked with block 4 as the key.

Analyzing a large structure gets difficult when there are a large number of blocks, because the number of possible partitions on the graph increases exponentially. Fortunately, proof that a complete structure is interlocked can be accomplished in a hierarchical fashion, by first showing that smaller components are interlocked, and then using those components to build larger interlocking structures.

Figure 5 shows an example of how a larger interlocking structure can be built from smaller interlocking substructures. Interlocking substructures A and B are similar to those shown in Figure 4a); the careful eye may note some additional geometry on each block representing dovetail joints attached from the side; these joints provide some redundant constraints that add rigidity to the final structure.

The keys of the substructures are K_A and K_B, and are not considered to be part of A and B. To show that the entire structure is 1-interlocked by K_B, it is sufficient to consider only partitions that separate K_A from A or K_B from B, since A and B act as rigid bodies if their keys are not separated. Figure 5b shows the graph representation.

4.2 Overview of the layout algorithm

The example above suggests an approach to constructing large interlocked structures. We can build 4-block interlocked squares, and use a second interlocked square to build an 8-block rectangle (Figure 5a). Inductively, we can extend the rectangle as far as we like by adding additional squares to the end; we call such a structure a *segment*.

Intuitively, some additional connections might be added to connect segments to form a flat structure that we will call a *layer*. 3D volumes may then be constructed from stacks of layers. Figure 6 shows a conceptual picture, with the single key block of each new larger structure shown in red.

The remainder of the paper addresses the details needed to allow implementation of this process. How should segments interconnect to form a layer? How should layers interconnect? How should joints be arranged on blocks to allow creating segments and layers from only a few types of block? How should layers be automatically shaped to allow construction of geometries more interesting than large cubical volumes?

Algorithm 1 presents an overview of layout algorithm; the details will be discussed in later sections of the paper, as indicated by the section numbering indicated in the algorithm; the reader may wish to only skim the algorithm on first reading. For now, it is worth noting that the input to the algorithm is a *voxelized* model describing the desired output shape. Each voxel is further subdivided into eight subvoxels; each subvoxel will be effectively be instantiated by a block.

The blocks are labelled by layer and segment. Layer and segment labels allow assignment of joint types that must connect adjacent blocks. Once the joint types have been assigned, blocks providing these joint types can be selected. The output is a sequence of block assembly orders that constructs an interlocking structure shaped as the input model. Since our model is built layer-by-layer, the final structure will have k keys, where k is the number of layers that do not have another layer on their top.

One critical observation is that joint types for a pair of blocks are selected by the layout algorithm based on the location of those blocks in the segment and layer. Since there are three joint types, each male or female, and six faces on a cube, this suggests that there might be $6^6 = 46656$ different types of block to construct. Fortunately, patterns in the segments and layers mean that not all of these block types occur. Further tricks allow reduction of the number of block types to two. As an example, consider the blocks in Figure 5a. Adding a mortise joint on the right side of block K_B makes it a copy of K_A, reducing the number of types of blocks.

It is also worth pointing out the approach we have taken to connecting adjacent layers. To provide a firm connection, we use a block of height two as a connector between layers; we call this block a *post*.

5 Blocks and squares

The layout algorithms makes use of two types of blocks: a *cube* is a unit-cube sized block for filling empty space in a layer and locking with existing blocks, and a *post* is a two-unit high block. See Figure 7. The lower half of a post block connects with cube blocks in the same layer, while the upper half of the block connects with cube blocks in the upper layer. The post blocks also act as key blocks of substructures.

Algorithm 1 Algorithm overview

```
1:  function ConstructVoxelModel(M)
2:      M' ← split every voxel into eight dimension-1 cubes.
3:      for each layer L_i of M' from bottom to top do
4:          Lay out any missing posts.
5:          if L_i is an even layer then
6:              Set all segment types to X_l_Y_+. (Section 6)
7:          else
8:              Determine the key to each layer component. (Section 7.1)
9:              Order segments in each layer component. (Section 7.2)
10:             Determine the key(s) to each segment. (Section 7.2)
11:             Determine the type of each segment. (Section 7.2)
12:             Find special cases. (Section 7.3)
13:             Modify L_i and L_{i+1} if necessary. (Section 7.3)
14:         Assemble blocks, potentially in parallel. (Section 8.1)
```

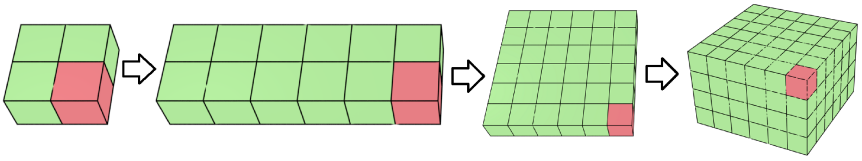


Fig. 6: Building an interlocking 3D structure. Blocks interlock to form a square; squares form an interlocking segment; segments form a layer; layers interlock to form the structure. The keys are marked red.

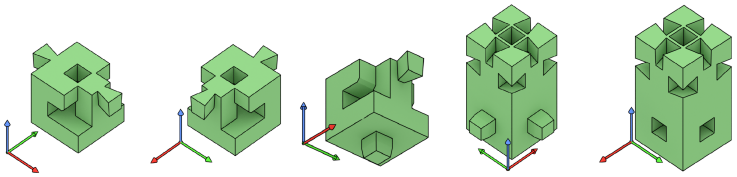
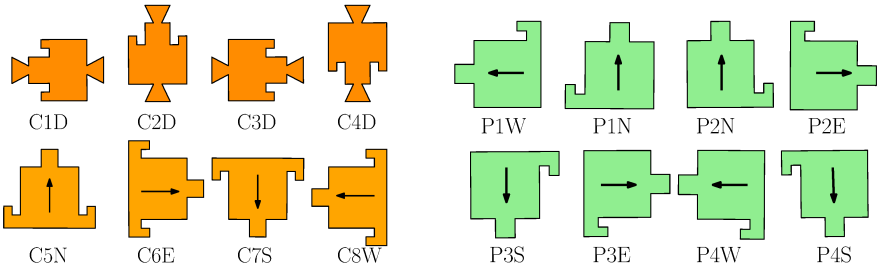


Fig. 7: Different views of cube and post blocks.

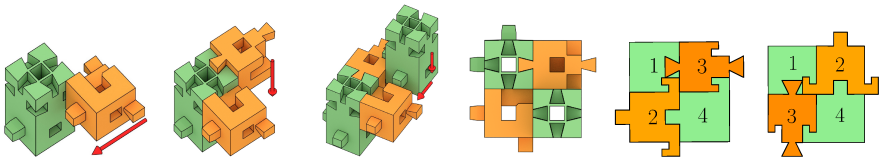
We carefully analyzed segments and layers to determine how joints might be arranged on cubes and posts. A *cube* block has two dovetail male joints on two opposite sides that can connect, in a tangential direction, with female joints in post blocks allowing motion only in the assembly direction. Figure 9a and 9b show how a cube's side male joints connect with a post's female joints in two different directions. A cube also has a male joint on the bottom that connects, in the normal direction, either with the top of a cube or post. The female joints on the opposite sides of the cube block allow post blocks' male joint to drop and slide to connect, which allows the post block to disassemble only in two directions.

To describe a layout and an assembly process, some notation is helpful. For each block, we use a triplet of characters indicating block type, orientation of the block, and assembly direction. Figure 8 shows all of the triplets used in assembly of structures in this paper. For example, *C1D* means "**C**ube in orientation 1, assembled by moving



(a) Cube orientations and assembly directions. (b) Post orientations and assembly directions.

Fig. 8: Different ways to assembly cube and post blocks and corresponding notations.



(a) Slide in a C8W block. (b) Drop in a C3D block. (c) Assemble a P1W block. (d) Top view of the square. (e) S_a square. (f) S_b square.

Fig. 9: Assembly process of a square, and two designs for a square.

down". Figure 8b shows all of the notation triplets used in the current approach. Not all axis-aligned orientations of cubes and posts are needed to construct structures; for example, posts only occur in the four orientations generated by rotating the post in Figure 7 around the z axis in ninety-degree increments.

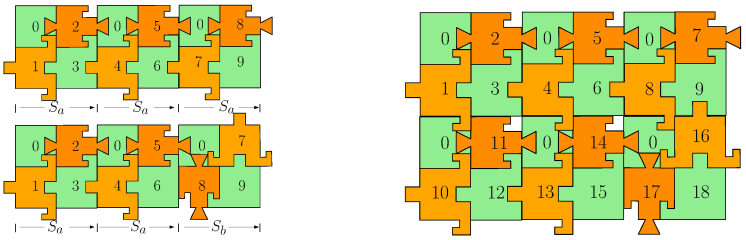
Block designs are crafted to allow design of squares, segments, and layers. A *square* is the smallest interlocking structure we consider, composed of four blocks: two posts and two cubes. By using posts and cubes in different orientations, different squares may be constructed, as shown in Figure 9e and 9f as S_a and S_b. Different squares will be used in Section 7 to constrain key block motions of other adjacent segments in the same layer, allowing interlock of the layer.

Figure 9 shows the process of assembling one kind of square. The first piece, a post, may connect to a layer below the current one. Two cubes are added to the top of the post. The second post acts as a key, and the top half of this post extends above the square to provide a connection to a square that may be later built above the current one.

6 Segments

We now introduce a method to link squares into a longer interlocking structure, a *segment*. A segment is composed of n squares in a $1 \times n$ pattern. To build a segment, we assume n posts have already be pre-placed in the prior layer, such that the top of each post appears in the same position in each square; these posts allow the segment to interlock with the prior layer.

We will discuss how to assemble a simple segment built from left to right in the direction of the x axis, assuming posts are in the upper (or $y+$) half of each line; other segments are symmetric and will not be discussed in detail. We denote a segment as



(a) Two $Y_{l+}X_+$ segments. (b) A layer built by two $Y_{l+}X_+$ segments.

Fig. 10: A simple segment and an example layer built by connecting two segments.

$Y_{l+}X_+$ if the posts are in the left position of the y positive half and the segment is built towards the x positive direction with the key block at the end.

Figure 10a visualizes the process of assembling a $Y_{l+}X_+$ segment of 3 squares. We connect, from left to right, $n-1$ S_a squares. The final square of a sub-segment can be of type S_a or S_b. The key piece of the segment is the last assembled post block. A sub-segment with a S_b final square is not interlocking, but when connected with previous segment(s), the S_b square prevents the adjacent block in the y positive direction from moving and interlocks the structure. Building another $Y_{l+}X_+$ segment on the y negative side will create an interlocking layer (Figure 10b). In Section 7, we will discuss how to constrain the motion of the key in different types of segments.

6.1 Structure mirrors

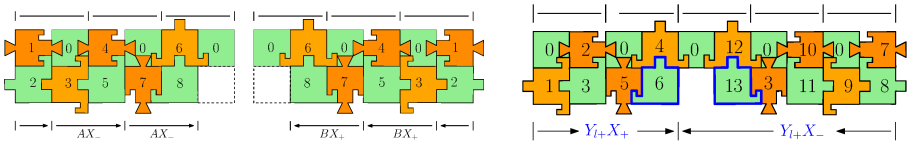
Knowing how to assemble $Y_{l+}X_+$ segments, one can lay out an array of segments one-by-one and create interlocking planar structures as in Figure 10b. However, these structures require the key to every segment to be in the x positive end and constrained by the next adjacent segment. In order to build more complicated planar structures, we introduce the concept of *mirrors*.

Definition 1 ***(x-mirror).*** *Object A is an x-mirror, $m_x(B)$, of another object B if one is a reflection of the other with reflection plane perpendicular to the x-axis.*

We define an analogous y-mirror operation. Cube and post designs are symmetric in such a way that x and y mirror operations can be accomplished by simple rotation of the block. Construction of a mirrored structure follows the same order of the original structure with opposite directions along the same axis; for example, we may build a $Y_{r+}X_-$ segment by x-mirroring a $Y_{l+}X_+$ segment.

Two other types of segments we will need for layer construction are $Y_{r+}X_+$ (Figure 11a) and its x-mirror $Y_{l+}X_-$ (Figure 11b). To build a $Y_{r+}X_+$ segment with n squares, where $n \geq 2$, we first assemble two blocks ($C3D$ and $P1W$) in the left two positions. Then assemble a $Y_{l+}X_+$ segment of $n-1$ squares. When all pre-existing posts are prevented from moving along z axis, the segment is interlocked.

For many input geometries, it may turn out that neither end of a segment is adjacent to the next segment, causing the key to be exposed. In this case, we may replace a single segment by two segments grown from the ends, effectively allowing placement of a pair of keys at an arbitrary position in the middle, as shown in Figure 11c. These keys may then be immobilized by later segments.



(a) A $Y_{r+}X_{+}$ segment. (b) A $Y_{l+}X_{-}$ segment. (c) A pair of segments with keys in the middle.

Fig. 11: Construction of two x-mirrored segments. Arrows indicate construction direction. Numbers indicate assembly order.

7 Layers

Now that we know how to build different kinds of segments, we can connect a set of segments on the same plane to create complicated interlocking 3D structures, by careful assignment of subvoxels from the original model into layers, segments, squares, and blocks.

A *layer* is a set of squares with the same z-coordinate. A set of connected squares with the same z-coordinate is a *layer component*. We assume all layer posts are provided by the prior layer. This is a fundamental limitation of our approach – it does not allow overhanging structures to be generated without building additional supports.

The section first introduces the ordering of segments in a component. Once ordered, segments are ready to be assigned square types and assembled. Then we discuss some special cases caused by the nature of our block design and square structure, and techniques to ensure interlock.

7.1 Layer key(s)

As the first step of building any interlocking structure, we determine the key(s) of the layer. A layer is immobilized if the key(s) is fixed with respect to its neighbors. Since every even layer has an upper layer with the exact same shape, based on the division of voxels into subvoxels, post blocks that connect the upper layer will be immobilized as long as the upper layer is interlocked, preventing the horizontal motion of any posts. Therefore, we only consider the odd layers in this section.

For any odd layer component without adjacent upper layer blocks, we select a post block at the x negative end of a boundary segment as the key, where a boundary segment is a segment with adjacent neighbors on only one side. If the odd layer component has an adjacent upper layer, the key can be any post block covered by an upper layer square.

Under this rule, every layer component constrains the key to its lower component. Any layer components that do not have an immediate upper layer introduce a new key that will not be covered. The number of key pieces of the whole structure is thus the number of layer components without an immediate upper layer. This introduces an interesting effect of the orientation of the object to be constructed. For example, the chair in Figure 1b has a single key, but if the chair were built upside-down, then there would be four keys: one in each leg.

7.2 Segment construction order

Once a layer's key square and all starting posts of squares are known, the second step of assembling a layer is to determine the order and type of each segment.

In the preprocessing step, every voxel is broken into two squares, making every layer of voxels two layers in the assembly. The bottom layer has an even z-coordinate value, while the up layer has an odd z-coordinate. Every segment in an even layer is constructed along y-axis directions. We simply assemble every segment as $X_{l-}Y_{+}$, or 90° clockwise rotation of a $Y_{r+}X_{-}$ segment, from left to right. An even layer component is not necessarily interlocked, because there can be many segment keys unconstrained and able to move in the x positive direction. However, all square keys are posts in the upper layer, and as long as the upper layer is interlocked, or all posts are prevented from moving in x positive direction, the two-layer structure is interlocked.

Each square in an odd layer component is initially assumed to have a post in the bottom-right position. This, however, could change after the segment types have been assigned. We first build a set of post lists where each list contains posts with the same y-coordinate, and two adjacent posts are 2 units away. Each *list* will be built into a segment. Two posts are considered adjacent if their x or y-coordinates have a difference of 2. Two lists are considered adjacent they have adjacent posts. Lists are ordered by their shortest distances to the final list that contains the post of the key square, where the distance between two adjacent lists is 1.

Given a list l and the next-built adjacent list l_n, the type of the segment S_l associated with l is determined as described below:

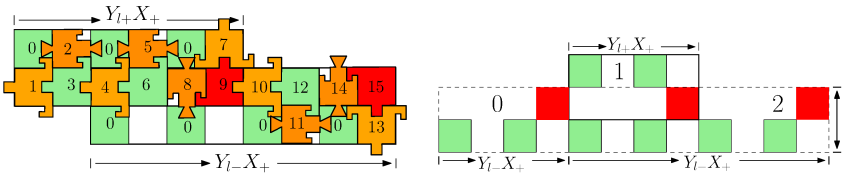
- If l_n is at $y-$ side of l & the left end post of l is adjacent to l_n, S_l is $Y_{r+}X_{-}$.
- If l_n is at $y-$ side of l & the right end post of l is adjacent to l_n, S_l is $Y_{r+}X_{+}$.
- If l_n is at $y-$ side of l & neither ends of l is adjacent to l_n, S_l is broken into a $Y_{r+}X_{-}$ and a $Y_{r+}X_{+}$ segment.
- If l_n is at y positive side of l & the left end post of l is adjacent to l_n, S_l is $Y_{r-}X_{-}$.
- If l_n is at y positive side of l & the right end post of l is adjacent to l_n, S_l is $Y_{r-}X_{+}$.
- If l_n is at y positive side of l & neither ends of l is adjacent to l_n, S_l is broken into a $Y_{r-}X_{-}$ and a $Y_{r-}X_{+}$ segment.

The segment associated with the last built list has been specified a key (line 8 of Algorithm 1). Its type is thus determined.

7.3 Special Cases

At this point, the type of each segment and the order of construction in each layer has been selected. Many interlocking layer structures can be assembled by directly following the construction of each segment as specified in Section 6. However, depends on the successor segments, some small modifications might be applied to insure the interlocking of adjacent segments.

Consider a segment with key(s) in the y negative side, for example $Y_{l+}X_{+}$. Its successor can be (1) a segment whose key will be constrained by further segments in y negative side, (2) a segment with key being constrained in y positive side, or (3) a segment whose key will be constrained by the upper layer. We now list all possible cases that need modifications.



(a) A $Y_{l-}X_+$ segment built after a $Y_{l+}X_+$ segment. Some positions are left empty.

(b) A longer lower segment. The lower segment is broken into two segments.

Fig. 12: Two special cases of building adjacent segments. Green blocks are posts, and red blocks are keys of each segment. Numbers indicates the assembly order.

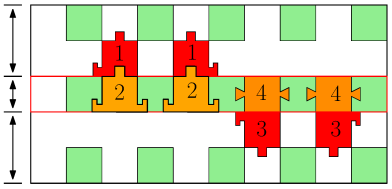


Fig. 13: Special cases where two segments with posts in different sides are finished before the segment in the middle. Red blocks are keys of two segments ($Y_{r+}X$ and $Y_{r-}X$ types). Numbers indicates the assembly order.

Case (1): A $Y_{l+}X_+$ segment followed by another $Y_{l+}X_+$ segment. We use a S_b squares at the later built segment to prevent the segment's key from moving. See Figure 10b. Otherwise, a $Y_{l+}X_+$ segment always uses a S_a end square.

Case (2) contains four subcases where the current segment has one or both ends adjacent to its successor whose key is in the x positive or negative side. Figure 12a shows one subcase. The first $Y_{l+}X_+$ segment is still assembled as usual. We leave some positions adjacent to the first segment unfilled, and assembled the rest part. Figure12b is a similar subcase where both ends of the segment are adjacent to the successor. We divide the lower segment into two segments, one containing no posts adjacent to the upper segment will be built first, the other containing the rest posts will be built after the upper segment. In the other subcases, the successor has a key in the x negative direction, we change the upper segment to $Y_{l+}X-$ and create an x-mirror of the previous case.

Case (3) is shown in Figure 13 where a $Y_{r+}X$ segment and a $Y_{r-}X$ segment are assembled before the segment in the middle. We require the upper and lower segments' keys to be in different x positions. To ensure interlocking, we firstly finish the upper segment, then assemble two C5N blocks in the middle segment. After the lower segment is assembled, we put in C3D block(s) in the middle to constrain the motion of the lower segment key(s). The last assembled blocks (keys) in the middle will be constrained by its upper layer. If the upper layer is not wide enough to cover the keys, we must expand the upper layer (Line 13 in Algorithm 1).

8 Automatic assembly and parallel construction

Algorithm 1 gives an overview of the construction process. Our construction starts from the bottom layer to the top. For each layer, we first check if all required posts exist.

If not, we lay out these posts before starting the assembly (Line 4). Even layers are constructed using $X_{l-}Y_{+}$ segments (Line 5, 6). Odd layers need to find the keys first (Line 8). Based on the key to each layer component, we order segments (Line 9) then determine segment keys and segment types (Line 10, 11). Before assembling, we check if any special cases exist as mentioned in Section 7.3 (Line 12). Since $Y_{r+}X$ and $Y_{r-}X$ segments require at least two adjacent square, we need to modify the current layer if the condition is not satisfied. The special case as in Figure 13 can also require the upper layer to expand and cover lower layer keys (Line 13). We then finally assemble blocks based on block types and special cases.

8.1 Parallel construction

Laying out blocks one-by-one is time-consuming when a structure has a large number of blocks. This section provides an algorithm that generates a parallel construction order to accelerate the process. We first consider preliminaries blocks of assembling each new block, and build a graph between blocks. By querying the graph for blocks whose preliminaries are satisfied, we can have multiple agents to lay out the blocks.

Consider a block b to be assembled in a layer. Any adjacent block(s) to be assembled later should not be prevented by the male joint(s) of b, meaning the joints of a block connect to only the pre-existing blocks. Along the assembly direction of b, the male joints of b should not be able to touch any blocks. The blocks that must be assembled before a new block to prevent collision are called *predecessors* of the new block. Every block has a predecessor below it if an adjacent block exists in the lower layer. Consider a block at position (x,y) in any layer. Table 1 is a list of predecessors of different types of blocks in the same layer.

Block type	Predecessors	Block type	Predecessors
C1D, C3D	$(x-1,y)$, $(x+1,y)$	C8W	$(x,y+1)$, $(x,y-1)$, $(x-1,y)$
C2D, C4D	$(x,y-1)$, $(x,y+1)$	P1W, P1N	$(x-1,y)$, $(x,y+1)$
C5N	$(x-1,y)$, $(x+1,y)$, $(x,y+1)$	P2N, P2E	$(x,y+1)$, $(x+1,y)$
C6E	$(x,y+1)$, $(x,y-1)$, $(x+1,y)$	P3S, P3E	$(x+1,y)$, $(x,y-1)$
C7S	$(x-1,y)$, $(x+1,y)$, $(x,y-1)$	P4W, P4S	$(x-1,y-1)$, (x,y)

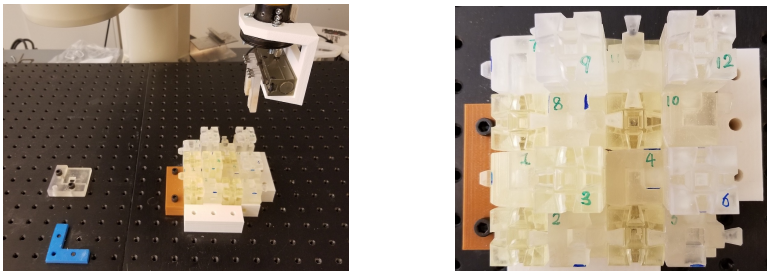
Table 1: Predecessors of each type of block.

Besides predecessors listed above, inside each square, cube blocks with mortise joints connecting blocks in the same layer (C5N, C6E, C7S or C8W blocks) must be assembled before others (C1D, C2D, C3D or C4D blocks).

With the predecessors of each block, we then construct a directed graph $G = \{V,E\}$, where V is the set of blocks, and directed edge $e_{i,j} \in E$ indicates block i being a predecessor of block j. The construction follows the order of removing nodes with in-degree of 0. Each construction agent/thread will take a block whose predecessors have been placed, and remove the node from the graph when the block assembly is finished.

A simple observation with the parallel construction is, after the construction of one square s, all the adjacent squares to be assembled after s in the sequential order are ready to assemble. We therefore have the following theorem:

Theorem 1. *Parallel construction of a solid cube of N squares takes* $O(\sqrt[3]{N})$ *time.*



(a) Robotic assembly of a layer with 4 squares using a 4-DoF robot arm.

(b) Overhead view of the assembled structure. Numbers indicate assembly order.

Fig. 14: Assembling 16 blocks using a 4-DoF robot arm. No sensors are used. Blocks are initially placed in correct orientation for the arm to pick up.

Proof. First consider constructing a solid layer of $n \times n$ squares. For simplicity, we scale the width of each square to one. After assembling the square at the corner $(0,0)$, two adjacent squares in x and y positive directions will be assembled at the next time step, then three, four, and so on. It takes k steps to construct $k(k+1)/2$ squares. When $k = n$, over $n^2/2$ squares are constructed. So constructing a layer takes at most $2n$ steps. In a cube, since finishing every square allows all adjacent squares in x, y and z positive directions to assemble. When the last square of the bottom layer is done, it takes one more step to finish the upper layer. So $2n-1$ more steps will finish all upper layers. Therefore a solid cube of $2n \times n \times n$ squares takes $O(n) = O(\sqrt[3]{N})$ time to assemble.

9 Results and robotic assembly

We algorithmically designed plans to assemble several models including a Stanford bunny and a chair model, and animated the results in software. Figure 1 shows these examples. The Stanford bunny model has 7337 blocks while the chair model has 472 blocks. The assemblies of both models are done in sequential order. The rendered animation of chair assembly can be found in [26]

We also 3D printed 406 blocks and assembled into a similar chair based on the rendered animation. The assembled chair is a simplified version of the chair in the simulation; two layers were omitted to save material and assembly time. Four legs of the chair are relatively loose compare to other parts, because each pair of layers in the legs are connected by only one post and a mortise and tenon joint.

For robotic assembly, we used a 4-DoF Adept robot arm to assemble both a one-layer structure with 4 squares (Figure 14a) and a two-layer structure with 6 squares. A closer look at the one-layer structure is shown in Figure 14b. This interlocking layer has two $Y_{l-}X_{+}$ segments. Assembly orders are marked with numbers. The recorded assembly can be found in [25] and [27]. The assembly shown in Figure 14a was recorded without human interruption. The joint designs show good tolerance during construction.

10 Acknowledgements

This project received seed funding from NSF IIS-1813043 and the Dubai Future Foundation. The authors are grateful to Haopeng Zhang and Geoffrey Hsuan-Chieh Huang, who helped build 3d models of blocks, built robot grippers and recorded videos. The authors also thank Jeremy Betz for useful insights on the geometry of joints. Thanks also to Emily Whiting, as well as members of the Dartmouth robotics lab, for useful feedback and insights throughout.

References

[1] Jürgen Andres, Thomas Bock, Friedrich Gebhart, and Werner Steck. First results of the development of the masonry robot system rocco: a fault tolerant assembly tool. In *Automation and Robotics in Construction XI*, pages 87–93. Elsevier, 1994.

[2] Federico Augugliaro, Ammar Mirjan, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. Building tensile structures with flying machines. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3487–3492. IEEE, 2013.

[3] Frederico Augugliaro, Sergei Lupashin, Michael Hamer, Cason Male, Markus Hehn, Mark W Mueller, Jan Sebastian Willmann, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems*, 34(4):46–64, 2014.

[4] C Balaguer, E Gambao, A Barrientos, EA Puente, and R Aracil. Site assembly in construction industry by means of a large range advanced robot. In *Proc. 13th Int. Symp. Automat. Robotics in Construction (ISARC'96)*, pages 65–72, 1996.

[5] Jonathan Daudelin, Gangyuan Jing, Tarik Tosun, Mark Yim, Hadas Kress-Gazit, and Mark Campbell. An integrated system for perception-driven autonomy with modular robots. *arXiv preprint arXiv:1709.05435*, 2017.

[6] Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. Computational interlocking furniture assembly. *ACM Transactions on Graphics (TOG)*, 34(4):91, 2015.

[7] Markus Giftthaler, Timothy Sandy, Kathrin Dörfler, Ian Brooks, Mark Buckingham, Gonzalo Rey, Matthias Kohler, Fabio Gramazio, and Jonas Buchli. Mobile robotic fabrication at 1: 1 scale: the in situ fabricator. *Construction Robotics*, 1(1-4):3–14, 2017.

[8] Volker Helm, Selen Ercan, Fabio Gramazio, and Matthias Kohler. Mobile robotic fabrication on construction sites: Dimrob. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4335–4341. IEEE, 2012.

[9] Steven J Keating, Julian C Leland, Levi Cai, and Neri Oxman. Toward site-specific and self-sufficient robotic fabrication on architectural scales. *Science Robotics*, 2(5):eaam8986, 2017.

[10] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. *Proc. Robotics: Science & Systems VII*, 2011.

[11] John W Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4288–4295. IEEE, 2013.

[12] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[13] Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.

[14] Eric Schweikardt and Mark D Gross. roblocks: a robotic construction kit for mathematics and science education. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 72–75. ACM, 2006.

[15] Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. Cofifab: coarse-to-fine fabrication of large 3d objects. *ACM Transactions on Graphics (TOG)*, 35(4):45, 2016.

[16] Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. Recursive interlocking puzzles. *ACM Transactions on Graphics (SIGGRAPH Asia 2012)*, 31(6):128:1–128:10, December 2012.

[17] Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. Reconfigurable interlocking furniture. *ACM Transactions on Graphics (TOG)*, 36(6):174, 2017.

[18] Tarik Tosun, Gangyuan Jing, Hadas Kress-Gazit, and Mark Yim. Computer-aided compositional design and verification for modular robots. In *Robotics Research*, pages 237–252. Springer, 2018.

[19] Ziqi Wang, Peng Song, and Mark Pauly. DESIA: A general framework for designing interlocking assemblies. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6), 2018. Article No. 191.

[20] Hongxing Wei, Youdong Chen, Jindong Tan, and Tianmiao Wang. Sambot: A self-assembly modular robot system. *IEEE/ASME Transactions on Mechatronics*, 16(4):745–757, 2011.

[21] Paul White, Viktor Zykov, Josh C Bongard, and Hod Lipson. Three dimensional stochastic reconfiguration of modular robots. In *Robotics: Science and Systems*, pages 161–168. Cambridge, 2005.

[22] Jan Willmann, Federico Augugliaro, Thomas Cadalbert, Raffaello D'Andrea, Fabio Gramazio, and Matthias Kohler. Aerial robotic construction towards a new field of architectural research. *International journal of architectural computing*, 10(3):439–459, 2012.

[23] JP Wilson, DC Woodruff, JD Gardner, HM Flora, JR Horner, and CL Organ. Vertebral adaptations to large body size in theropod dinosaurs. *PLoS ONE*, 11(7), 2016.

[24] Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. Interactive design and stability analysis of decorative joinery for furniture. *ACM Trans. Graph.*, 36(2):20:1–20:16, March 2017.

[25] Yinan Zhang. One-layer structure robotic assembly experiment with two kinds of blocks. (2.5x speed). `https://youtu.be/1_lbVyPcLOI`.

[26] Yinan Zhang. Robotic assembly of interlocking blocks - wafr 2018. `https://youtu.be/lV2xIA_Q8SI`.

[27] Yinan Zhang. Two-layer structure robotic assembly experiment with two kinds of blocks. (2.5x speed). `https://youtu.be/ZjFFZzrl69s`.

[28] Yinan Zhang and Devin Balkcom. Interlocking structure assembly with voxels. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016.

[29] K. Zwerger and V. Olgiati. *Wood and Wood Joints: Building Traditions of Europe, Japan and China*. Birkhäuser, 2012.