Block designs for assembling interlocking structures

Yinan Zhang[1] and Devin Balkcom[2]

***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 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. Simulation results show planned layouts for shapes involving thousands of blocks, and shapes with several hundred blocks have been built by hand. As a proof of concept, a robot was used to assemble 16 blocks. The paper also describes a method for assembling blocks in parallel.**

I. Introduction

The goal of the work described in this paper is to enable robotic assembly of large structures from modular components. Structures like furniture, houses, and bridges are often built of small modular pieces such as wood planks, logs, and bricks that are easy to manufacture. This paper presents a modular design that allows the components to *interlock* in such a way that the constructed shape acts as a single body as long as a few blocks, called *keys*, are fixed in place.

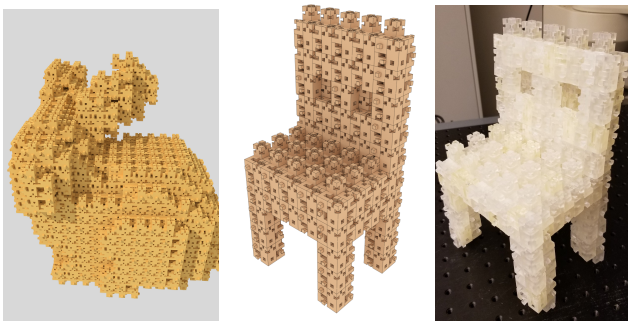
Kinematic interlock presents some advantages over traditional connection methods such as glue, cement, screws, nails, or friction locks. The interlocks may be structurally strong, allow simple assembly by robots, allow disassembly and re-use of the components, and may be suitable for underwater or other environments where adhesives are ineffective.

The algorithms described in this paper take 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: cubes of $1 \times 1 \times 1$, and posts of size $1 \times 1 \times 2$ sizes with connectors. The assembly requires only translation motions.

The basic approach of this paper is similar to that previously presented by the authors in [25]. The largest contribution of this work relative to the previous is the reduction of the types of blocks needed from nine to two; this significant reduction in complexity is enabled by a new block design and a new layout algorithm. 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.

[1]Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA. yinan.zhang.gr@dartmouth.edu

[2]Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA. devin.balkcom@dartmouth.edu



(a) Stanford Bunny assembled in simulation. (b) Chair model assembled in simulation. (c) Chair model assembled in real-world.

Fig. 1: Models assembled

II. Related work

Interlocking structures have a long history. Wood joints are commonly applied in architectures and furniture in ancient China and Japan [26]. Some people believe timber connections improve seismic characteristics [8] of an architecture. In the recent years, computer scientist have also shown interests in interlocking structures. Kong *et al.* applied curve matching techniques for finding solutions of 2D and 3D interlocking puzzles. Song *et al.* has several works on designing interlocking puzzles, interlocking furniture, and re-usable interlocking pieces [17], [18], [16], [5]. Yao *et al.* [22] proposed a method for interactively designing joinery structures and analyze the stability. The present work is closest in spirit to Song *et al.* [18] which considers the problem of designing reusable components to be assembled into different forms relying on geometric constraints.

A. Robotic construction

Robotic construction research in architecture dates back to the 1990s [1] when Andres *et al.* created a prototype, namely ROCCO, capable of gripping and laying bricks for a fault-tolerant assembly. The same robotic system was later applied on site assembly operations by Balaguer *et al.* [4]. DimRob, a system with an industrial robot arm mounted on a mobile platform, designed by Helm *et al.* [7] took the similar idea of having a mobile robot to perform construction tasks, including planning and laying out brick structures. This prototype was later developed into a mobile robot, In situ Fabricator, for construction at 1:1 scale, by Giftthaler *et al.* [6]

To overcome the fact that ground-based mobile robots are not able to reach high positions, some researchers proposed

to use aerial robots. Willmann *et al.* [21], for example, used autonomous flying vehicles to list and position small building elements. Similarly, 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: quadrocopters assembled a tensile structure out of ropes capable of holding several human beings.

WinSun Decoration Design Engineering 3D-printed a five-story build in Suzhou, Jiangsu, China [11]. Keating *et al* [9] built a large mobile 3D printer using a robot arm to extrude adhesive materials.

B. Modular robots

Instead of focusing on the robot control system for carrying building elements, some researchers have designed new building elements. Rus and Vona [14] developed Crystalline, a modular robot with a 3DOF 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.* [20] later introduced two three-dimensional stochastic modular robot systems that are self-reconfigurable and self-assemblable by successive bonding/releasing of free-floating unit. Romanishin *et al.* [12] proposed a momentum-driven modular robot. SamBot, a similar cube shaped modular robots with rotation mechanism was introduced by Wei *et al.* [19].

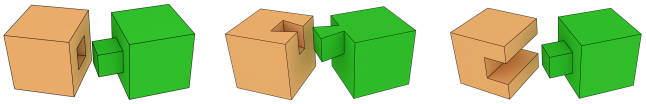
Inspired by LEGO, Schweikardt *et al.* [15] proposed a robotic construction kit, roBlocks, with programmable cubic blocks for eduction purpose. Kilobot, a swarm of 1000 crawling mobile robots, was introduced by Rubenstein *et al.* [13], along with algorithms for planning mechanisms allowing kilobots to form 2D shapes.

III. Overview

This section will first introduce some basic definitions. A structure is build layer-by-layer. Each *layer* is a set of blocks with the same z-coordinate. A layer is built such that if the first and the last blocks are immobilized with respect to their relative positions, the whole layer is immobilized. Layers are built out of two-unit wide strips called *segments*. Each segment is also an interlocked structure with the last piece being the key. Based on the order of laying out segments, each segment is designed to have a key that can be constrained by the next segment, which will be adjacent.

We use two kinds of blocks, one of $1 \times 1 \times 1$ unit size and one of $1 \times 1 \times 2$ unit size. The taller blocks allow a lower layer to be connected with its upper layer, making the two layers inseparable. The construction does not introduce any new keys to the structure unless a layer that has no adjacent upper layer is built.

The next five sections describe the construction process, starting with the simplest interlocked shapes and moving to a method for constructing voxelized 3D structures.



(a) Normal joints. (b) Tangential joints. (c) Two-way joints

Fig. 2: Three different joint pairs.

IV. Blocks

The intuitive way of filling a voxelized shape is by laying out unit-sized cubes in each voxel. Zhang and Balkcom ([25]) used this method. Many joints in their design were added to blocks for the connection between layers. We propose a different way to connect layers by using blocks of two units height. With the lower half sitting on one layer, the upper half is naturally connected withe the upper layer. This choice greatly reduces the kinds of blocks required for assembling an interlocking structure.

In this paper, a block is a $1 \times 1 \times 1$ (Figure 3) cube or $1 \times 1 \times 2$ (Figure 4) post with particular male and female joints. This section first introduces three kinds of joint pairs and then our design of the two kinds of blocks with their notations for assembly.

A. Joint pairs

A joint pair is a structure consisting of a male and a female connectors in two individual blocks constraining motions in multiple directions either normal to the contact face for along one of four coordinate directions tangential to the contact face.

The first kind of joint pair has a normal male joint and a normal female joint, which allows block disassembly motion in the non-penetrating normal direction of the contact face. See Figure 2a.

The second kind of joint pair has a tangent male joint and a tangent female joint, which allows block disassembly motion in the non-penetrating tangential direction of the contact face. See Figure 2b.

The third kind of joint pair allows block disassembly motion in both normal and one tangential direction of the contact face. See Figure 2c.

B. Block design

We designed two types of blocks: one unit-cube sized block for filling empty space in a layer and lock with existing blocks, and one two-unit high block for connecting blocks between current and upper layers. See Figure 3 and 4

Type I block has two male joints on two opposite sides that can connect, in tangential direction, with female joints in type II blocks allowing motion only in the assembly direction. Figure 7a and 7b shows how type I side male joints connect with type II female joints in two different directions. Type I also has a male joint on the bottom that connects, in normal direction, either with other type I blocks' top female joint or type II blocks female joints to prevent motion other than in the assembly direction. The female joints on the opposite sides of type I block allows type II blocks' male joint to

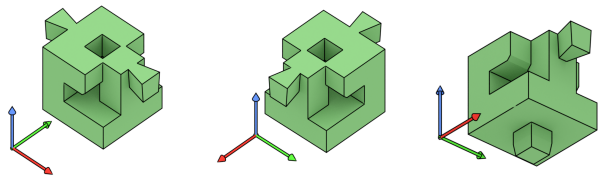


Fig. 3: Different sides of Block I

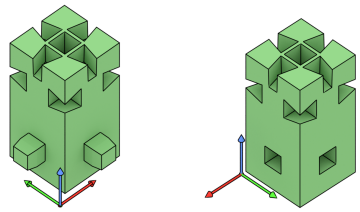


Fig. 4: Different sides of Block II

fall and slide to connect, which allows the type II block to disassemble only in two directions, one perpendicular to the side and one alone the female joint.

Type II block is tow-unit high. The lower half of a type II block connects with type I blocks in the same layer, while the upper half of the block connects with type I blocks in the upper layer, thus two layers are connected. With type II blocks, if the motion of the last assembled block in the lower layer is constrained by blocks in the upper layer, the whole structure below the layers without upper layer are constrained and stable.

Based on the way a block is assembled, we use 8 notations to represent assembly direction and block orientation. Figure 5 lists all cases of type I block layout. "B1DW" means "Type I block assembled by moving DOWN with open female joints facing the WEST". "B1SN" means "Type I block assembled by moving from SIDE with bottom male joint facing the NORTH". Similarly, type II blocks have 8 notations (Figure 6), where "B2W1" means "Type II block assembled by sliding toward WEST". Although type II block's male joins when connected with a female joint in a type I block allows two directions of disassembly motions, when both male joints are connected with other blocks, the block can only move in one direction. So in the flat representation, one side of male joint is drawn as it can move in only one direction.

V. Squares

A square is the smallest interlocking structure made of 4 blocks in a 2x2 pattern. To assemble a square, we first make sure a type II block exists, then connect two type I blocks in diagonal positions of a 2x2 grid adjacent to the top half of the pre-existing block. Finally, another type II block is assembled in the diagonal position of the first block to constraint motions of both type I blocks relative to the first type II block. Here, the pre-existing type II block is called a *post*. The z-coordinate of a square is defined by the z-coordinate of its type I blocks. Figure 7 shows the process of assembling one kind of square. The last assembled

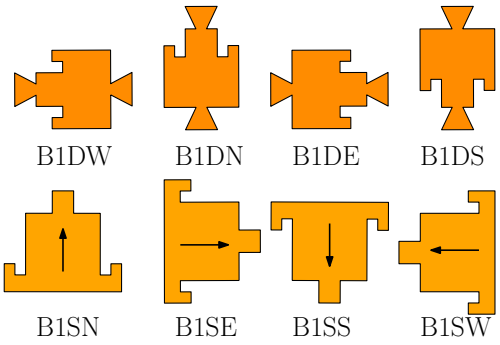


Fig. 5: Different ways to assembly type I block and corresponding notations.

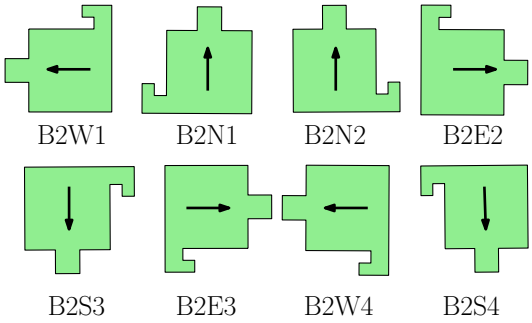


Fig. 6: Different ways to assembly type II block and corresponding notations.

piece is the key of this structure. They key's top half is left unconnected to any blocks, which makes it a post block for a square on the top. Figure 8a is corresponding simplified representation of blocks.

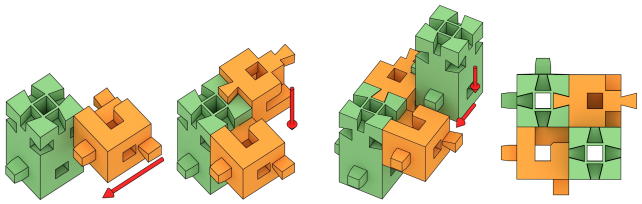
In every square, one type I block has a male normal joint able to connect and constraint motions of another existing block in the same layer. The square assembled in Figure 8a constrains motion of a key block adjacent in $X-$ direction. By placing the male joint in a different position, a square can also constraint the motion of a key block adjacent in $Y+$ direction. (See Figure 8b.)

We notate the four pieces of a square s as $s(a)$, $s(b)$, $s(c)$ and $s(d)$, ordered as shown in Figure 8c. A square s whose pre-existing post is in the $s(a)$ position and an adjacent type I block constrains blocks on the $X-$ side. Figure 8a, is called a AX_{-} square. Similarly, Figure 8b is a AY_{+} square.

Table I lists the block types and assembly orders and directions of two squares.

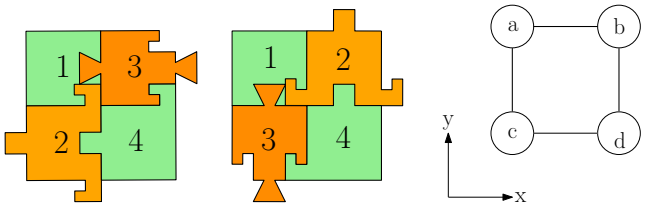
	Square 1	Square 2
s(a)	B2	B2
s(b)	B1DE	B1SN
s(c)	B1SW	B1DS
s(d)	B2W1	B2N1
order	a-c-b-d	a-b-c-d

TABLE I: Block types with assembly directions and assembly orders of two squares. $B2$ means a pre-existing type II block.



(a) Slide in a B1SW block. (b) Drop in a B1DE block. (c) Drop and slide in a B2W1 block. (d) Overhead view of a finished square.

Fig. 7: Assembly of a AX_- square



(a) Block types and assembly order of a AX_- square. (b) Block types and assembly order of a AY_+ square. (c) Positions of a square.

Fig. 8: Two kinds of squares

VI. Segments

In the previous section, we assembled a square as the smallest interlocking structure. We now introduce a method to joint squares into a longer interlocking structure, a segment. We will first assemble a simple segment built from left (or $X-$) to right (or $X+$), assuming posts are in the upper (or $Y+$) half of each line, then discuss the methods to build segments from different directions with posts in different positions.

A segment is a list of n squares in a $1\times n$ pattern, assuming a list of n posts exist in the same row or column. We denote a segment as $Y_{l+}X_+$ if its posts are in the left position of the $Y+$ half and it builds from $x-$ to $x+$ direction with the key block at the $x+$ end of the segment. Similarly we have $Y_{l+}X_-$, $Y_{r+}X_+$, $Y_{r+}X_-$, $Y_{l-}X_+$, $Y_{l-}X_+$, $Y_{r-}X_+$ and $Y_{r-}X_+$ segments. Segments that build along y-axis are treated as 90° rotation of segments build along x-axis. For simplicity, we will only discuss segments build along x-axis.

Algorithm 1 gives the pesudo-code of assembling a $Y_{l+}X_+$ segment (Figure 9) with n squares where $n \geq 2$. If $n = 1$, we only need to assemble a AY_+ square. Otherwise, we first connect, from left to right, a list of $n-1$ AX_- square and finally one AY_+ square. The key piece of the segment is the last assembled type II block capable of moving in $Y-$ direction. In section VII, we will discuss how to constrain the motion of the key of each square.

A. Structure mirrors

Knowing how to assemble $Y_{l+}X_+$ segments, one can layout an array of segments one-by-one and create some interlocking planar structures. However, these structures requires the key of every line to be in the $X+$ end and constrained by the next adjacent line. In order to build more

Algorithm 1 Assembling of a $Y_{l+}X_+$ segment

```
1: function BUILDY_{l+}X_+SEG(n, posts)
2:     if n = 1 then
3:         Assemble a AY_+ square using the (first) post.
4:     else
5:         for i ∈ [0 ... n − 2] do
6:             Assemble a AX_− square using posts[i]
7:         Assemble a AY_+ square using the last post.
```

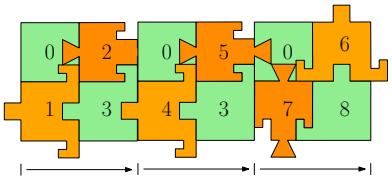


Fig. 9: Construction process of a $Y_{l+}X+$ segment.

complicated planar structures, we introduce the concept of *mirrors*.

Definition 1 (Mirror): Two objects are mirrors iff there exists a plane, p, such that the reflection through p of each point in one object is also a point in the other object.

Definition 2 (X-mirror): Object A is an x-mirror, $m_x(B)$, of another object B iff they are symmetric and their reflection plane is perpendicular to the x-axis.

Similarly, two objects can be *y-mirrors* of each other. Construction of a mirror structure follows the same order of the original structure with opposite directions along the same axis. Table II lists mirrors of each block in both directions.

block	x-mirror	y-mirror
B1DW	B1DE	B1DW
B1DN	B1DN	B1DS
B1DE	B1DW	B1DE
B1DS	B1DS	B1DN
B1SN	B1SN	B1SS
B1SE	B1SW	B1SE
B1SS	B1SS	B1SN
B1SW	B1SE	B1SW

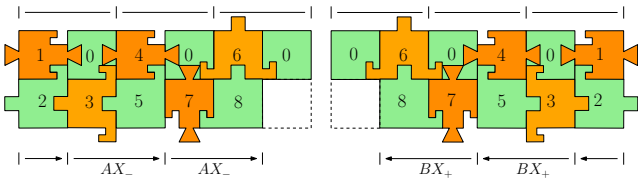
block	x-mirror	y-mirror
B2W1	B2E2	B2W4
B2N1	B2N2	B2S4
B2N2	B2N1	B2S3
B2E2	B2W1	B2E3
B2S3	B2S4	B2N2
B2E3	B2W4	B2E2
B2W4	B2E3	B2W1
B2S4	B2S3	B2N1

TABLE II: Mirrors of all blocks

With the definition of mirrors, we are now able to build a $Y_{r+}X_-$ segment. Given a $Y_{r+}X_-$ segment of n squares, we simply construct this structure by x-mirroring a $Y_{l+}X_+$ segment.

Two other types of segment we need to deal with are $Y_{r+}X+$ (Figure 10a) and its x-mirror $Y_{l+}X-$ (Figure 10b). Algorithm 2 gives the pseudo-code for assembling a $Y_{r+}X+$ segment. Given a $Y_{r+}X+$ segment with n squares, where $n \geq 2$. We firstly assembly two blocks ($B1DE$ and $B2W1$) in the left two positions. Then assembly a $Y_{l+}X+$ segment of $n-1$ squares. Although the bottom-right position is left empty, the segment is still interlocked and provides all posts for the upper layer.

So far, we have constructed segments with key pieces at either ends. The last type of segment we want to build



(a) Assembly of a $Y_{r+}X+$ segment.

(b) Assembly of a $Y_{l+}X-$ segment.

Fig. 10: Construction of two segments x-mirror to each other. Arrow indicates the construction direction. Numbers indicates orders.

Algorithm 2 Assembling of a $Y_{r+}X_+$ segment

```
function BUILDY_{r+}X_+SEG(n, posts)
    if n = 1 then
        Assemble a x-mirror of AY_+ square.
    else
        Put a B1DE block at left side of posts[0]
        Put a B2W1 block at botton-left of posts[0]
        BUILDY_{l+}X_+SEG(n − 1, posts[0 : n − 2])
```

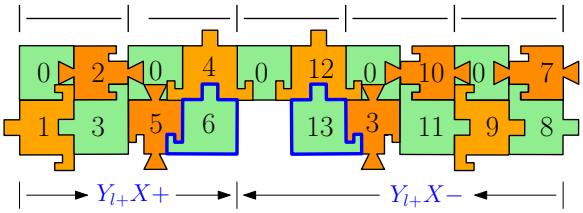


Fig. 11: Construction process of a $Y_{l+}X$ segment whose key pieces are in the middle. Blue pieces are keys.

has key pieces in the middle. A segment along x-axis of $n \geq 3$ squares whose posts are in the top-left positions is notated as a $Y_{l+}X$ segment if the key pieces is not at either end of the segment. See Figure 11. This kind of segments are constructed as one $Y_{l+}X+$ segment connected with a $Y_{l+}X-$ segment. This construction process, however, requires the next-built adjacent segment to have at least two adjacent squares constraining each key.

VII. Layers

We can build layers by laying out a set of connected segments. This process is non-trivial and requires careful ordering and segment type definition. This section introduces the process of constructing layers. 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 first introduce the ordering of segments in a component. Once ordered, segment types are ready to be selected and segments assembled. We later discuss some special cases caused by the nature of our block design and square structure, and techniques to ensure interlocking.

A. Layer key(s)

As the first step of any layer construction, we want to determine the key(s) of the layer. A layer is immobilized if key(s) is/are fixed with respect to the posts of a layer. Since every even layer has an upper layer with the exact same shape, type II blocks that connect the upper layer will be immobilized as long as the upper layer is interlocked preventing the horizontal motion of any posts. Here we only consider the odd layers.

For any odd layer component without adjacent upper layer blocks, we select a type II block at the $X-$ 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 type II block that is covered by an upper layer square.

Under this rule, every layer component constrains the key of its lower adjacent component. Any layer components that do not have an adjacent upper layer introduces a new key that will not be covered. The number of key pieces of the whole structure is the number of layer components without an adjacent upper layer.

B. Segment construction order

Assuming the layer's key square and all posts of squares are given, 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 a y-segment, where a y-segment is a segment parallel to y-axis and not contained by any other segment. We construct every even layer simply by assembling every segment as $X_{l-}Y+$, or $rot_{90}(Y_{r+}X-)$ types ordered from left to right, where rot_{90} is 90-degree clockwise rotation. An even layer component is not interlocked, because there can be many segment keys unconstrained. However, all square keys are posts in the upper layer, as long as the upper layer is interlocked, the two-layer structure is interlocked.

In odd layer components, each square is initially considered to have a post in the bottom-right position. This, however, could change after deciding the segment types. We firstly build a set of post lists where each list contains connected posts with the same y-coordinate. Each *list* represents a segment, the post locations of segments will be defined after defining the type of each segment associated with a list. 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 the $Y-$ side of l and the left end post of l is adjacent to l_n, S_l is a $Y_{r+}X-$ segment.
- If l_n is at the $Y-$ side of l and the right end post of l is adjacent to l_n, S_l is a $Y_{r+}X+$ segment.
- If l_n is at the $Y-$ side of l and neither ends of l is adjacent to l_n, S_l is a $Y_{r+}X$ segment.

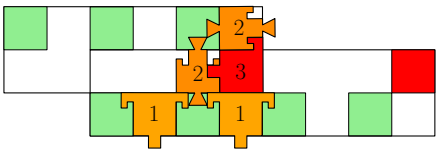


Fig. 12: A $Y_{l-}X_{+}$ segment assembled after a $Y_{l+}X_{+}$ segment. Green blocks are posts, and red blocks are keys of each segment. Numbers indicates the assembly order.

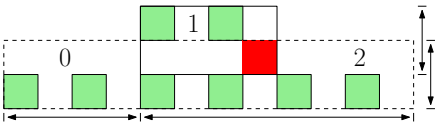


Fig. 13: A similar case as in Figure 12 while the both ends of the upper segment adjacent to the lower segment. We solve this case by breaking the lower segment into two segments.

- If l_n is at the $Y+$ side of l and the left end post of l is adjacent to l_n, S_l is a $Y_{r-}X-$ segment.
- If l_n is at the $Y+$ side of l and the right end post of l is adjacent to l_n, S_l is a $Y_{r-}X+$ segment.
- If l_n is at the $Y+$ side of l and neither ends of l is adjacent to l_n, S_l is a $Y_{r-}X$.

The key piece of $Y_{r+}X$ and $Y_{r-}X$ can be any type II block adjacent to l_n.

The segment associated with the last built list has been specified a key. It's type is therefore determined.

C. Special Cases

In each layer, we now know the type of each segment and the order of construction. Directly following the construction of each segment specified in Section VI, any set of connected segments in the same layer whose posts are all in $Y-$ or $Y+$ side is an interlocking structure. However, when two sets of connected segments whose posts are in different sides meet, the structure become tricky and requires additional attention.

Figure 12 is an example where a segment with posts in the $Y-$ side has to be assembled after a segment with posts in the $Y+$ side. Before working on the upper segment, we assemble B1SS blocks in the lower segment's type I block positions that adjacent to the upper segment, in case the lower segment needs to constrain any blocks in the $Y-$ side. We will modify the last square of the upper square and use two type I blocks capable of moving in $Z+$ direction and a B2W1 block as key (red). This square is not interlocked by now, but will be constrained by other squares in the lower segment.

A problem similar to Figure12 occurs when both ends of the upper segment is adjacent with the lower segment (Figure 13). We can solve this problem by dividing the lower segment into two segments, one contains no posts adjacent to the upper segment will be built before the upper segment, and one contains the rest posts will be build after the upper segment.

A more complicated case is shown in Figure 14 where a $Y_{r+}X$ segment and a $Y_{r-}X$ segment will be finished before the segment in the middle. We require the upper segment's

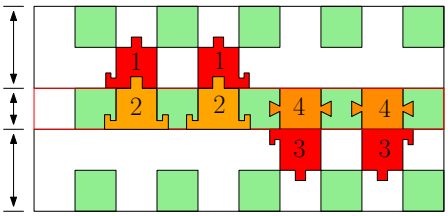


Fig. 14: Special cases where two segments with posts in different sides are finished before the segmetn in the middle. Red blocks are keys of two segments ($Y_{r+}X$ and $Y_{r-}X$ types). Numbers indicates the assembly order.

key(s) to be adjacent to the blocks of the middle segment that has neighbors in the $Y-$ side. The middle segment is the last finished segment in this layer. In this case we firstly finish the upper segment, then assemble B1SN blocks in the middle segment adjacent with the upper segment except for the positions adjacent to the keys of the lower segment. After the lower segment is assembled, we put B1DE block(s) to constrain the motion of the lower segment key(s). The last-assembled block(s) in the middle segment will be constrained by the upper layer. If the upper layer is not wide enough to cover the block(s), we will expand the upper layer.

VIII. Automatic Assembly

Algorithm 3 gives an overview of the construction process.

Algorithm 3 Algorithm overview

```
1:  function ConstructVoxelModel(M)
2:      M' ← split every voxel into 8 equal size cubes.
3:      for each layer L_i of M' from bottom to top do
4:          Layout any missing posts.
5:          if L_i is an even layer then
6:              Set all segment types to X_{l-}Y_{+}.
7:          else
8:              Determine the key of each layer component.
9:              Order segments in each layer component.
10:             Determine the key(s) of each segment.
11:             Determine the type of each segment.
12:             Find special cases.
13:             Modify L_i and L_{i+1} if necessary.
14:         Assemble blocks.
```

Our construction starts from the bottom layer to the top. For each layer, we first check if all required posts exist. If not, we layout these posts before any other 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 of 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 exists as mentioned in Section VII-C (Line 12). Since $Y_{r+}X$ and $_{r-}X$ segments requires at least two adjacent square, we need to modify current layer if the condition is not satisfied. Special case as in Figure 14 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.

IX. Parallel construction

Laying out blocks one-by-one could be time consuming when the structure contains a large number of blocks. In this section, we provide an algorithm that generates a parallel construction order to accelerate the process. We first consider the 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 layout 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 when assembling the new block is called *preliminaries* of the new block. Every block has a preliminary below it if an adjacent block exists in the lower layer. Consider a block at position (x, y) in any layer. Table III is a list of preliminaries of different types of blocks, excluding the lower layer preliminaries.

	Preliminaries
B1DW, B1DE	$(x-1, y)$, $(x+1, y)$
B1DN, B1DS	$(x, y-1)$, $(x, y+1)$
B1SN	$(x-1, y)$, $(x+1, y)$, $(x, y+1)$
B1SE	$(x, y+1)$, $(x, y-1)$, $(x+1, y)$
B1SS	$(x-1, y)$, $(x+1, y)$, $(x, y-1)$
B1SW	$(x, y+1)$, $(x, y-1)$, $(x-1, y)$
B2W1, B2N1	$(x-1, y)$, $(x, y+1)$
B2N2, B2E2	$(x, y+1)$, $(x+1, y)$
B2S3, B2E3	$(x+1, y)$, $(x, y-1)$
B2W4, B2S4	$(x-1, y-1)$, (x, y)

TABLE III: Preliminaries of each type of block.

Besides preliminaries listed above, another observation is that, inside each square, type I squares with normal joint connecting blocks in the same layer (B1SN, B1SE, B1SS, B1SW blocks) should be assembled before others (B1DW, B1DN, B1DE, B1DS).

With the preliminaries 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 preliminary of block j. The construction follows the order of removing nodes with in-degree of 0. Each construction agent will take a block whose preliminaries are satisfied, 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. With this observation, we have the following theorem.

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

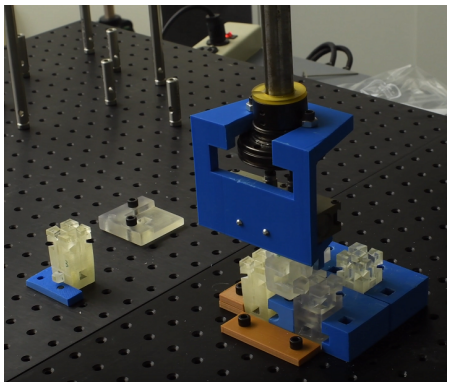


Fig. 15: Robotic assembly of a layer with 4 squares using a 4 DoF robot arm.

Proof: Consider the construction of a solid layer of $n \times n$ squares. For the sake of simplicity, we scale the width of each square as one. After assembling the square at the corner $(0, 0)$ position, two adjacent squares will be assembled at the next time step, then three at the next time step, then four, and so on. It takes k step to construct $k(k+1)/2$ squares. Let $k(k+1)/2 \geq n^2/2$, meaning half of the layer is constructed, $k \geq n$. So the construction of a layer takes at most $2n = O(n)$ steps. Similarly, for a solid cube of $2n$ layers with $n \times n$ squares in each layer; the parallel construction also takes $O(n) = O(\sqrt[3]{N})$ time. ■

X. Results and robotic assembly experiment

We ran simulations to assemble several models including a Stanford bunny and a chair model, as shown in Figure 1.

The Stanford bunny model has 7337 blocks while the chair model has 472 blocks. The assembly of both models are done in sequential order. The rendered animation of chair assembly can be found in [23]

We also 3D printed 406 blocks and assembled into a small chair based on the rendered animation. The assembled chair is a simplified version of the chair in simulation. Two layers are removed to save material and assembly time cost. Four legs of the chair are relatively loose compare with other parts, because every two layers are connected by only one post. In the real-world assembly, it is preferred to have multiple posts linking very two layers.

To show how a model can be assembled automatically, we used an Adept robot arm with 4 degrees of freedom to assemble a layer of 4 squares, as shown in Figure 15. The recorded assembly can be found in [24]. Although the goal is full automatic robotic assembly, because the robot precision is not enough, we slightly adjusted the positions of some blocks by hand to avoid collisions.

XI. Conclusions, limitations, and future work

In this work, we explored a design of blocks for assembling structures that utilize geometric constrains to guarantee the stability of the structure. We proposed a design of two blocks that can be used to assemble voxelized interlocking structures with the number of key pieces equal to the number

of layers without adjacent upper layers. Our work also automatically generates an assembly order and directions for each construction that forms the shape of any given voxelized model.

The primary contribution of this work is that we proved a relatively tight upper bound of the number of different blocks required to assemble any tightly interlocked structure with a small number of pieces. The assembly can be done with pure translations and requires no adhesive materials to constraint motions between pieces, making it ideal for robots to do such construction in environments difficult for human to stay, such as under water and outer space.

However, we still have many problems to address before the robotic construction become practical. One problem is the support of pieces without adjacent pieces in the lower layer. Like in 3D printing, blocks do not float in the air and must be supported by additional structure. Another problem is when some blocks or joints break after the whole construction. Under current construction scheme, broken parts can make the part in lower levels a non-interlocking structure. We would like to design structures with interlocking sub-structure so the whole structure is not endangered by a few fragile parts.

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] 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.

[6] 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.

[7] 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.

[8] Xu Minggang Qiu Hongxing. Analysis of seismic characteristics of chinese ancient timber structure.

[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] Tuan C. Nguyen. Yes, that 3d-printed mansion is safe to live in. Washington Post, 5 February, 2015. https://www.washingtonpost.com/news/innovations/wp/2015/02/05/yes-that-3d-printed-mansion-is-safe-to-live-in.

[12] 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.

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

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

[15] 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.

[16] 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*.

[17] Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. Recursive interlocking puzzles. *ACM Transactions on Graphics (TOG)*, 31(6):128, 2012.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Yinan Zhang. Chair assembly with two kinds of blocks. (iros 2018). https://youtu.be/4xcNXqkYKDw.

[24] Yinan Zhang. Robotic assembly experiment with two kinds of blocks. (2.5x speed). https://youtu.be/06qzBg5Oiig.

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

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