Huddle: Parallel Shape Assembly using Decentralized, Minimalistic Robots

Khai Yi Chin[0000−0002−8720−2451], Tingwei Meng[0000−0001−7467−601X], Zhe Chen[0000−0002−0425−3131], Daniel Bassett, and Yuri Ivanov[0009−0004−3105−4789]

Amazon Robotics, 300 Riverpark Dr, North Reading, MA 01864, USA
{khaiyic,tingweim,zhecm,basjdani,yuriivan}@amazon.com

Abstract. We propose a novel algorithm for forming arbitrarily shaped assemblies using decentralized robots. By relying on local interactions, the algorithm ensures there are no unreachable states or gaps in the assembly, which are global properties. The in-assembly robots attract passing-by robots into expanding the assembly via a simple implementation of signaling and alignment. Our approach is minimalistic, requiring only communication between attached, immediate neighbors. It is motion-agnostic and requires no pose localization, enabling asynchronous and order-independent assembly. We prove the algorithm's correctness and demonstrate its effectiveness in forming a 107-robot assembly.

1 Introduction

Multi-robot systems offer speed, scalability, and robustness through parallelized execution in applications such as transporting arbitrarily shaped objects [19] and assembling structures on demand [12]. This is further enhanced when the system is decentralized, as its centralized counterpart is associated with higher communication and computational burden from synchronizing robot activities. However, local robot interactions present a significant challenge, as they limit the range of conceivable emergent behaviors. The onus thus falls on designing correct local behaviors that lead to the desired global outcome.

In this work, we explore the self-assembly of decentralized, minimalistic robots into arbitrary 2-D shapes. To that end, we propose *Huddle*, a parallel shape assembly algorithm. Huddle relies on simple signaling to enable homogeneous robots to self-assemble without specialized motion control strategies or pose localization. Starting with a single robot, the assembly gradually takes shape as in-assembly robots signal at specific openings to attract passing-by robots. Attachments to the assembly are anonymous and order-independent. Inter-robot communication is required only between in-assembly robots that are immediate neighbors.

The key novelty of Huddle is its minimalism, owing in part to its rule-based formulation, but also to its agnosticism towards motion control strategies. Inter-robot information exchange is equally modest, requiring only two integers shared at each time step. Moreover, Huddle only needs the perimeter of the desired

shape to work, as we prove in our theoretical analysis. We also demonstrate the minimalism of Huddle using a physics-based simulated experiment where 107 robots assemble into a highly non-convex shape.

2 Related Work

Research on multi-robot self-assembly ranges from hardware design to algorithmic development [1]. Our work contributes to the algorithmic side: Huddle is a motion-agnostic, rule-based algorithm that enables self-assembly using local interactions, providing global shape guarantees without localization requirements.

Due to their minimalism, rule-based methods are common in assembly planning. Notable studies include the self-assembly of Kilobots (hop count propagation with connectivity constraints) [3, 14] and s-bots (hand-designed pattern extension rules) [4, 11], as well as the reconfiguration of SMORES-EP robots (centralized graph matching) [8, 9]. Saldaña *et al.* used an assembly tree to guide square robots into specific docking sequences and locations [15]. Using *assembler* robots, Werfel *et al.* leveraged stigmergy through construction blocks that store and communicate assembly information [20]. With 3-D self-assembly blocks, Stoy grew structures to match CAD models using a cellular automaton [16], while Feshbach and Sung proposed a sequential method for assemblies to reshape themselves [2]. Tolley and Lipson sampled attachment sites approved by a centralized algorithm to address attachment stochasticity [18]. Besides sharing features with such works—*e.g.,* promotion of attachment sites, signaling and self-alignment mechanisms—Huddle offers decentralized, parallel shape assembly without connectivity, motion, or identification requirements.

Another popular approach employs optimization techniques. Matthey *et al.* synthesized robot controllers using the Chemical Reaction Network framework for the assembly of heterogeneous parts [10]. Sun *et al.* used the mean-shift algorithm in their shape assembly strategy, which optimizes density functions online [17]. Methods built upon artificial potential fields are widespread as well, particularly in swarm robotics. This includes assembly of robots based on specifications from 2-D point clouds [7] or from analytic functions [13,21]. Our approach offers simplicity both in computational demands and in shape specification, requiring only perimeter coordinates.

3 Problem Statement

This work addresses the question: *How do we create an arbitrarily shaped, hole-free assembly using decentralized, minimalistic robots without specific motion control strategies?* By being motion-agnostic, we can be adaptable to application constraints. For example, centralized waypoint computation may be feasible in a closed warehouse environment, but less so in a large, outdoor setting where decentralized methods scale better.

Our robot has a hexagonal shape, permitting up to 6 neighbor attachments, with a signal emitter-receiver pair on each wall (hexagon side) for signaling and

communication. The signaling medium is kept abstract to maintain generality; practical implementations could involve optical or sonic devices. Each wall also has a passive attachment mechanism (*e.g.,* magnets [15]), a proximity sensor for obstacle avoidance, and the ability to identify wall occupancy.

Our approach is to use in-assembly robots to attract passing-by robots, starting with a root robot. The finalized assembly must match the target shape S exactly—provided that the shape is decomposable into hexagonal components. We assume passing-by robots can detect and follow the signals transmitted by in-assembly robots.

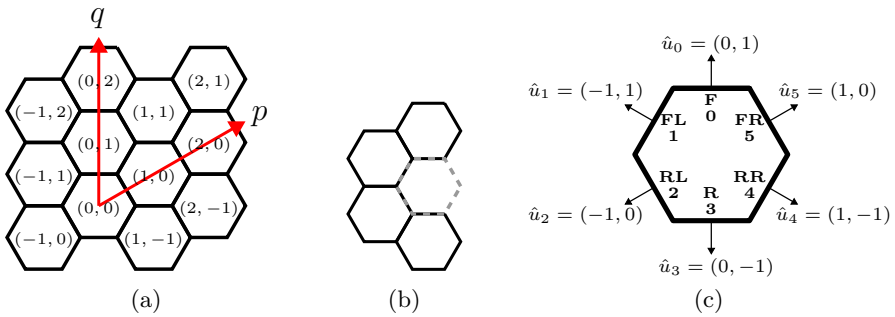


Fig. 1: (a) Axial coordinate system for the 2-D hexagonal space $\mathcal{H}$ that Huddle uses, where p and q are the column and row coordinates, respectively. (b) An unreachable position shown by the dashed hexagon. (c) Layout of wall indices and their corresponding directional unit vectors $\hat{u}_w$.

Formally, we represent a robot assembly at time t using an undirected, planar graph, $G(t) = (V(t), E(t))$. Each in-assembly robot i is represented by vertex $v_i \in V(t)$ and its attachment to neighbor j is represented by edge $e_{ij} = e_{ji} \in E(t)$. Without loss of generality, we use the axial convention for the 2-D hexagonal coordinate frame $\mathcal{H} = \mathbb{Z}^2$ (Fig. 1a). Each in-assembly robot occupies position $x_i \coloneqq (p_i, q_i) \in X(t) \subseteq \mathcal{H}$, where p_i and q_i are the column and row coordinates of robot i, respectively, and $X(t)$ is the set of occupied positions in $G(t)$. Only robots that share an edge can communicate.

To build $G(t)$, we assume an input of the target shape, given as a set of hexagonal coordinates ($S \subseteq \mathcal{H}$) that does not contain holes. As such, S can be summarily represented by its perimeter X_b, a set of boundary coordinates—coordinates of hexagons in S that have at least one unattached side. We assume that each robot has access to this set, which is invariant and can be provided offline or locally propagated from the root robot.

Passing-by robots attach only to an opening in the assembly that is formed by at least one (connectivity) and at most three robots (feasibility). We deem such an opening to be *reachable*. Fig. 1b shows an *unreachable* opening.

Definition 1 (Reachability). *Let $\mathcal{N}(x)$ be the adjacent coordinates of position x and $\deg(v)$ be the degree of vertex v. At t, we have $Y(t) = \{y \in S \backslash X(t) : \mathcal{N}(y) \cap$*

$X(t) \neq \emptyset\}$ *as the set of unoccupied positions adjacent to* $X(t)$*. A position* $y \in Y(t)$ *is reachable if adding a vertex* v_j *with coordinates* $x_j = y$ *would yield* $1 \leq \deg(v_j) \leq 3$*.* $G(t)$ *maintains reachability if all positions in* $Y(t)$ *are reachable.*

For the target shape to form without holes, the assembly must not contain enclosed empty regions at any time.

Definition 2 (Hole-Free Assembly). $G(t)$ *is hole-free if no simple cycle* C *in* $G(t)$ *bounds an interior region containing unoccupied positions from* S*.*

Let t_f be the time S is completed by $G(t)$. The problem we address in this work is as follows.

Problem 1. Given a hole-free shape S represented by perimeter coordinates X_b, determine a decentralized method for signal activation such that:

1. $G(t)$ maintains reachability $\forall t < t_f$ (Definition 1), and
2. $G(t)$ is hole-free $\forall t$ (Definition 2).

4 Methodology

We present Huddle, a parallel and decentralized shape assembly algorithm. The assembly grows in a concurrent, staggered manner, prioritizing longitudinal growth (intra-column) without blocking lateral expansion (inter-column). A designated *root* robot initiates the assembly by remaining stationary at a target location. As the first in-assembly robot, it attracts passing-by robots to join by beaming signals; newly joined robots in turn emit their own signals, progressively expanding the assembly until S is achieved. This growth pattern induces a *spanning tree* over the assembly's column segments, where *each segment is a node* and *edges connect segments of adjacent columns.*

Huddle accomplishes this via two phases, *Initialization* and *Signal Activation* (Fig. 2). The *Initialization* phase occurs once, at the instant a passing-by robot docks into the assembly. Then, this robot enters the *Signal Activation* phase at each time step to attract other passing-by robots.

4.1 Preliminaries

Wall type A robot wall can be one of two types.

- Null wall: A wall on an in-assembly robot *not* intended for attachment.
- Connection wall: A wall on an in-assembly robot intended for attachment.

Each wall also has a *status*: a null wall is always "null", while a connection wall can be "free" or "occupied" (Fig. 2c).

Wall direction As shown in Fig. 1c, we label the robot walls in a counterclockwise manner where $\hat{u}_w \in \mathcal{H}$ is the unit vector pointing in the direction of wall w from the robot center. When we refer to "left flank", walls **FL** and **RL** are included; likewise for "right flank". We use the term "fore-aft" when referring to the front (**F**) and rear (**R**) walls.

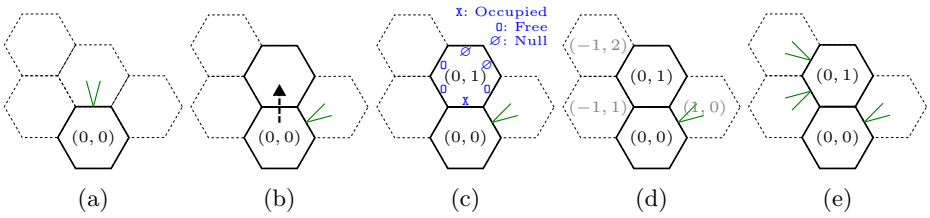


Fig. 2: Example of a 5-robot assembly. Dashed hexagons are target positions in S. (a) A root robot $(0, 0)$ in its *Signal Activation* phase, signaling (green) on its **F** wall to attract passing-by robots. (b) A passing-by robot detects and follows the signal to attach to the root robot. Upon connection, the root robot communicates its position $(0, 0)$ and the unit vector of the admitting wall $\hat{u}_0$ to the newly attached robot. As a right nucleus—determined prior to its entering the *Signal Activation* phase—the root robot now signals on its **FR** wall. (c) The newly attached robot enters the *Initialization* phase, computing its coordinate $(0, 1)$ and identifying its wall statuses. (d) The $(0, 1)$ robot determines its role as a left nucleus and its growth direction as $g_i = -1$, as it shares a row with the (rounded down) midpoint $(-1, 1)$ of the left column (Algorithms 1–4). (e) The $(0, 1)$ robot enters the *Signal Activation* phase, signaling its left flank (Algorithms 5 and 6). The robots remain in that phase until their connection walls are occupied.

Robot roles Each robot has one of two fixed roles, determined during the *Initialization* phase.

- Nucleus robot: An in-assembly robot that *seeds* the expansion of an adjacent column segment (*i.e.,* spawns a child node in the spanning tree); it signals on at least one flank (one or both walls).
- Regular robot: An in-assembly robot that *does not seed* any adjacent column segment; it signals on a flank wall only under a special condition.

We describe the special condition in Remark 1. Crucially, nucleus robots' ability to seed adjacent columns drives the assembly's *lateral expansion*, as we detail in Section 4.3. Both roles can signal on **F** and **R** walls for *longitudinal expansion*.

4.2 Initialization

When a passing-by robot i attaches to the assembly, it processes information received from its newly connected, in-assembly neighbor j. This information consists of robot j's position x_j, wall status vector $s_j(t) = (s_{j,0}(t), ..., s_{j,5}(t))$, where $s_{j,w}(t) \in \{\text{null}, \text{free}, \text{occupied}\}$, and the unit vector $\hat{u}_{j,w}$ of robot j's admitting wall w. For brevity, we drop the robot index in the unit vector $(\hat{u}_w)$ when clear from context, and omit time t in pseudocode.

Wall Type Identification Robot i first determines its position $x_i = x_j + \hat{u}_{j,w}$, then identifies its wall types and statuses to update $s_i(t)$. To do this, robot i

computes $\mathcal{X}_c$, the inclusion-minimal cycles among the simple cycles of the graph induced by X_b, enumerating them with Johnson's algorithm [6]. It then checks neighboring positions that are either part of X_b or enclosed by a cycle in $\mathcal{X}_c$ (using a ray-casting algorithm [5]). Walls facing these positions are connection walls; otherwise they are null walls.

Algorithm 1 Determine robot role

Input: Self-wall status vector s_i, perimeter X_b, cycles $\mathcal{X}_c$
Output: Nucleation requirement $\alpha_{\text{left}}, \alpha_{\text{right}}$
Initialize $\tilde{\alpha}_{\text{left}} \leftarrow$ undefined, $\tilde{\alpha}_{\text{right}} \leftarrow$ undefined
Evaluate $\tilde{\alpha}_{\text{left}}, \tilde{\alpha}_{\text{right}}$ with Algorithm 2
if $\tilde{\alpha}_{\text{left}} =$ undefined **then**
    Evaluate $\tilde{\alpha}_{\text{left}}$ with Algorithm 3
if $\tilde{\alpha}_{\text{right}} =$ undefined **then**
    Evaluate $\tilde{\alpha}_{\text{right}}$ with Algorithm 3
$\alpha_{\text{left}} \leftarrow \tilde{\alpha}_{\text{left}}$, $\alpha_{\text{right}} \leftarrow \tilde{\alpha}_{\text{right}}$
return $\alpha_{\text{left}}, \alpha_{\text{right}}$ ▷ $\alpha_{left}, \alpha_{right} \in \{false, true\}$

Role Determination Next, robot i evaluates its role using Algorithm 1, which relies on two subroutines, Algorithm 2 and Algorithm 3. We infer robot i's role from the output of Algorithm 1, which are boolean flags indicating if nucleation is required for a flank.

Algorithm 2 Determine robot role using self-wall status

Input: Self-wall status vector s_i
Output: Nucleation requirement $\beta_{\text{left}}, \beta_{\text{right}}$
Initialize $\beta_{\text{left}} \leftarrow$ false, $\beta_{\text{right}} \leftarrow$ false
for $F \in \{\text{left}, \text{right}\}$ **do**
    if F flank has at least 1 free wall **then**
        if free F flank wall is between 2 null walls **then**
            $\beta_F \leftarrow$ true
        else
            $\beta_F \leftarrow$ undefined
return $\beta_{\text{left}}, \beta_{\text{right}}$ ▷ $\beta_{left}, \beta_{right} \in \{false, true, undefined\}$

The first subroutine, Algorithm 2, considers only robot i's wall status vector s_i in deciding nucleation requirements. It checks for connection flank walls that are situated between two null walls; if such walls exist, robot i nucleates on the flanks with said wall arrangement.

However, $s_i(t)$ alone may be insufficient for identifying nucleation requirements. Thus, a second subroutine, Algorithm 3, does so by using adjacent target

columns relative to robot i's position $x_i = (p_i, q_i)$ as illustrated in Fig. 3a:

$$\mathcal{A}_{p_i+a} = \{Q^{(1)}_{p_i+a}, ..., Q^{(N_{p_i+a})}_{p_i+a}\}$$

where $a \in \{-1, 0, 1\}$ is the column direction (left, current, and right) and $p_i + a$ is the column index. $\mathcal{A}_{p_i+a}$ is a target column containing N_{p_i+a} contiguous segments, denoted $Q^{(n)}_{p_i+a}$. When only one segment (out of many) is being discussed, we drop the superscript $^{(n)}$ and identify it descriptively.

Algorithm 3 checks whether x_i is on the same row as midpoint m of a target adjacent segment Q_{p_i+a}. If so, robot i is designated as a nucleus. Otherwise, the algorithm verifies whether m has a valid connection in Q_{p_i}, and if not, finds the next closest point $m^\star$—making robot i the nucleus if $m^\star$ is on its row. Only one nucleus is responsible for seeding each adjacent segment (proved in Section 5).

Algorithm 3 Determine robot role using target adjacent column

Input: Perimeter X_b, cycles $\mathcal{X}_c$, self-coordinate x_i, adjacent direction $a \in \{-1, 1\}$
Output: Nucleation requirement γ
Find target segments in $\mathcal{A}_{p_i+a}$ and their midpoints M using X_b, $\mathcal{X}_c$, and x_i
if $\mathcal{A}_{p_i+a} = \emptyset$ **then** ▷ *no target segments in* $\mathcal{A}_{p_i+a}$
    $\gamma \leftarrow$ false
else
    Identify target segment $Q_{p_i+a} \in \mathcal{A}_{p_i+a}$ and its midpoint $m = (p_i + a, q_m)$
    if $q_i = q_m$ **then** ▷ *m is on the same row as x_i*
        $\gamma \leftarrow$ true
    else ▷ *m is not on the same row as x_i*
        Find self-target segment Q_{p_i} ▷ *target segment containing x_i*
        if $(p_i, q_m) \in Q_{p_i}$ **then** ▷ *m is on the same row as another point in Q_{p_i}*
            $\gamma \leftarrow$ false
        else ▷ *m does not have a valid attachment point in Q_{p_i}*
            Find $m^\star = (p_i + a, q^\star) \in Q_{p_i+a}$ nearest to m that connects to Q_{p_i}
            if $q_i = q^\star$ **then** ▷ *$m^\star$ is on the same row as x_i*
                $\gamma \leftarrow$ true
            else ▷ *$m^\star$ is not on the same row as x_i*
                $\gamma \leftarrow$ false
return γ ▷ $\gamma \in \{false, true\}$

Growth Direction Determination At this point, robot i is either a nucleus robot (nucleating on the left-, right-, or both flanks) or a regular robot (non-nucleating). Subsequently, robot i determines its growth direction, g_i using Algorithm 4. For a nucleus robot, g_i indicates where it will initiate lateral expansion; for a regular robot, g_i indicates the expansion direction it supports.

By the end of the *Initialization* phase, robot i has acquired information about its connection walls, role, and growth direction. Note that, as a boundary

Algorithm 4 Determine growth direction

Input: Nucleation requirement $\alpha_{\text{left}}, \alpha_{\text{right}}$, self-wall status vector s_i
Output: Growth direction, g_i
if α_{left} **and** α_{right} = true **then**
    $g_i \leftarrow 0$ ▷ *neutral (nucleus)*
else if ($s_{i,1}$ **or** $s_{i,2}$ = occupied) **or** α_{right} = true **then**
    $g_i \leftarrow 1$ ▷ *right*
else if ($s_{i,4}$ **or** $s_{i,5}$ = occupied) **or** α_{left} = true **then**
    $g_i \leftarrow -1$ ▷ *left*
else
    $g_i \leftarrow 0$ ▷ *neutral (regular)*
return g_i ▷ $g_i \in \{-1, 0, 1\}$

condition for our approach, the root robot assumes the origin $x_i = (0, 0)$ and skips the step where it receives information from a neighbor.

4.3 Signal Activation

In this phase, robot i determines which walls require signal activation. We assume the "signal" is a transmission of an IR beam. It need not contain any encoded information; we only require that a recipient robot can identify and follow it to the source. This phase runs iteratively until the robot's connection walls are occupied.[1] At the start of this phase, robot i updates and communicates its wall status vector $s_i(t)$ with its neighbors.

Signaling Wall Identification Then, using intrinsic information ($s_i(t)$, α_{left}, α_{right}), robot i identifies nominal walls W_i that require signal activation with Algorithm 5. These are connection walls found in the prior phase, with priority given to fore-aft walls. This ensures that each column is allowed to grow longitudinally (in the fore-aft directions) before nucleation happens laterally (in the flank directions), as premature lateral expansion could induce unreachable openings. Effectively, Algorithm 5 ensures the occupancy of fore-aft walls first—if they are connection walls—before flank walls can signal for both robot roles, prioritizing each spanning tree node's internal growth before it branches.

Remark 1. The special condition mentioned in Section 4.1 refers to the logic in lines 11–13 of Algorithm 5. Added to improve passing-by robots' signal detection probability, it can be removed without affecting the assembly process.

Delayed Wall Identification Next, using extrinsic information (neighbor j's wall status vector $s_j(t)$), robot i uses Algorithm 6 to delay certain fore-aft signals

[1] We do not consider robot rejoining here; if such a capability is desired, the *Signal Activation* phase can operate perpetually with departure-handling mechanisms.

Algorithm 5 Identify walls to activate signals

Input: Self-wall status vector s_i, nucleation requirement α_{left}, α_{right}
Output: Set of walls W_i requiring signals
Initialize $W_i \leftarrow \emptyset$
if all s_i = occupied **then** ▷ *no signaling needed*
  return W_i
if $s_{i,0}$ **or** $s_{i,3}$ = free **then** ▷ *prioritize fore-aft walls*
  for fore-aft wall $w \in \{0, 3\}$ **do**
    if $s_{i,w}$ = free **then**
      $W_i \leftarrow W_i \cup \{w\}$
  return W_i ▷ *ignore flank walls because of free fore-aft walls*
for flank wall $w \in \{1, 2, 4, 5\}$ with $s_{i,w}$ = free **do** ▷ *special flank signal condition*
  if w is between (2 occupied **or** (1 null fore-aft **and** 1 occupied)) walls **then**
    $W_i \leftarrow W_i \cup \{w\}$
if α_{left} = true **then** ▷ *left nucleus flank walls*
  $W_i \leftarrow W_i \cup \{\text{free left flank wall(s)}\}$
if α_{right} = true **then** ▷ *right nucleus flank walls*
  $W_i \leftarrow W_i \cup \{\text{free right flank wall(s)}\}$
return W_i

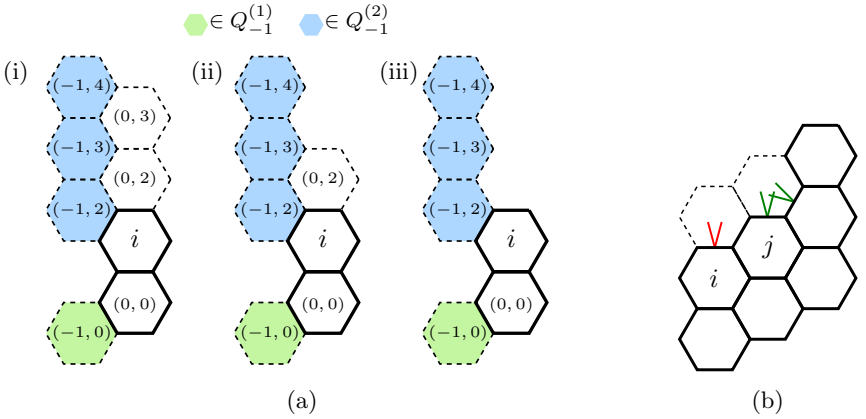


Fig. 3: (a) Notable situations during robot i's determination of its role, where it lies on a different row from the midpoint $m = (-1, 3)$ of $Q_{-1}^{(2)}$. Dashed hexagons are target positions for S, with color overlays indicating membership to target segments. (i) Located on the midpoint's row, a robot occupying $(0, 3)$—in the future—will be the nucleus (Algorithm 3, lines 12–13). (ii) Without a target $(0, 3)$ position, the new attachment point is $m^\star = (-1, 2)$, thus $(0, 2)$ is the nucleus position (Algorithm 3, lines 18–19). (iii) With its **FL** wall between two null walls, robot i is solely responsible for nucleating to the left (Algorithm 2). (b) Suppose $g_i = g_j = -1$ and robot i does not delay its signal (red) while robot j's front wall is free. An unreachable position will occur if robot i's front wall is occupied *before* robot j's.

when a neighbor j's fore-aft walls are still free. This prevents robot i from attracting attachments before j, which could create an unreachable position (Fig. 3b). As a result, robot i obtains $\tilde{W}_i$, the set of walls to delay signal activation.

By the end of the *Signal Activation* phase, robot i activates signals on the walls in $W_i \backslash \tilde{W}_i$ for the current time step. As long as any of its walls remain free, robot i stays in the *Signal Activation* phase and repeats Algorithms 5 and 6.

Algorithm 6 Identify walls to delay signal activation

1: **Input:** Self-wall status vector s_i, set of i's neighbors' wall status vectors (indexed by i's walls) $\{s_{(w)} : w \in \{0, ..., 5\}\}$, growth direction g_i
2: **Output:** Set of walls requiring delayed activation $\tilde{W}_i$
3: Initialize $\tilde{W}_i \leftarrow \emptyset$
4: **if** $g_i = -1$ **then**
5: ▷ ***FR*** *neighbor has free* ***F*** *wall*
6: $\tilde{W}_i \leftarrow \tilde{W}_i \cup \{0\}$ ▷ *delay* ***F*** *wall*
7: **if** $s_{i,4}$ = occupied **and** $s_{(4),3}$ = free **then** ▷ ***RR*** *neighbor has free* ***R*** *wall*
8: $\tilde{W}_i \leftarrow \tilde{W}_i \cup \{3\}$ ▷ *delay* ***R*** *wall*
9: **else if** $g_i = 1$ **then**
10: **if** $s_{i,1}$ = occupied **and** $s_{(1),0}$ = free **then** ▷ ***FL*** *neighbor has free* ***F*** *wall*
11: $\tilde{W}_i \leftarrow \tilde{W}_i \cup \{0\}$ ▷ *delay* ***F*** *wall*
12: **if** $s_{i,2}$ = occupied **and** $s_{(2),3}$ = free **then** ▷ ***RL*** *neighbor has free* ***R*** *wall*
13: $\tilde{W}_i \leftarrow \tilde{W}_i \cup \{3\}$ ▷ *delay* ***R*** *wall*
14: **return** $\tilde{W}_i$

5 Theoretical Results

Here, we show that Huddle solves Problem 1 by proving two properties separately: that $G(t)$ is reachable and always hole-free. Due to space constraints, we state the following lemmas without proof.

Lemma 1 (Unique nucleation). *Algorithms 2 and 3 ensure that any contiguous column segment Q_{k+a} is connected to only one nucleus robot from an adjacent column k (i.e., each node in the spanning tree has a unique parent).*

Corollary 1. *By Lemma 1, a segment Q_{k+a} expands only from a single origin—a seeding robot j, $x_j = (k+a, q_j) \in Q_{k+a}$, attached to a nucleus robot in column k. Separate segments of column $k+a$—i.e., each having a seeding robot attached to a different nucleus robot in column k—will never expand to join one another.*

Let a robot i that has at least one free fore-aft wall be defined as a longitudinal frontier robot (LFR): free $\in \{s_{i,0}(t), s_{i,3}(t)\}$. Both robots i and j in Fig. 3b are LFRs. As such, an LFR exits Algorithm 5 at line 10 and can only attract attachments to its free fore-aft wall(s), upon which it ceases to be an LFR.

Lemma 2 (Controlled longitudinal expansion). *Suppose a robot i with growth direction $g_i \neq 0$ in segment Q_{p_i} has a neighbor j belonging to segment $Q_{p_i - g_i}$, i.e., $|p_i - p_j| = 1$. If both are LFRs and share at least one free fore-aft wall index $w \in \{0, 3\}$ (i.e., $s_{i,w} = s_{j,w} = free$), Algorithm 6 ensures a passing-by robot k joins robot j's wall w before robot i's. That is, $p_k = p_j$ and $|q_k - q_j| = 1$.*

Corollary 2. *The controlled expansion guaranteed by Lemma 2 ensures that robots attaching to LFRs do not cause $G(t)$ to be unreachable (Fig. 3b) or contain holes.*

We now establish the main guarantees for Problem 1.

Theorem 1 (Reachability). *Following the algorithms in Section 4, the assembly satisfies Definition 1.*

Proof. By Lemma 1, let segment Q_{p_i} be the child of Q_{p_j} in the spanning tree, *i.e.,* seeded by one nucleus robot j in Q_{p_j}. We prove reachability separately in longitudinal and lateral expansion.

1. *Longitudinal*: From Corollary 2, the longitudinal growth of segment Q_{p_i} will not outpace its parent segment Q_{p_j}. The robot roles of the LFRs do not matter—without satisfying free fore-aft walls, flank signals will never be activated (Algorithm 5, lines 6–10). Thus, longitudinal expansion will result in reachable positions.
2. *Lateral*: A nucleus robot can only seed one segment per flank (Lemma 1). Suppose there exists a target segment Q_{p_i} where $|Q_{p_i}| > 1$ and let robot i be the first passing-by robot attracted to a flank wall of nucleus robot j, forming segment Q_{p_i}. Because robot i is the first of its segment Q_{p_i}, it does not create unreachable positions. Alternatively, a regular robot signals on a flank wall only when it is *not* an LFR and is adjacent to an LFR (Algorithm 5, lines 11–13). The presence of the adjacent LFR implies the longitudinal expansion case (for said LFR's segment), which is always reachable.

□

Theorem 2 (Hole-free). *Following the algorithms in Section 4, the assembly satisfies Definition 2.*

Proof. We show this using proof-by-contradiction. Recall that S is assumed to be hole-free. From Corollary 1, a column segment cannot be created from combining segments within its column—its longitudinal expansion must start from a seeding robot attached to a nucleus robot in the adjacent column.

Suppose that $G(t)$ contains a hole—enclosed unoccupied positions in S—within column k. The hole is effectively a gap between two segments within the column; the segments must have originated from two seeding robots, each attached to a different nucleus robot in the adjacent column (Lemma 1). But this is a contradiction: S is hole-free, so column k must be seeded by only one nucleus robot in an adjacent column. Therefore, $G(t)$ is always hole-free. □

6 Experimental Results

We evaluated our approach with Monte Carlo experiments (correctness) and a physics-based experiment in NVIDIA Isaac Sim (feasibility).

For our Monte Carlo experiments (Fig. 4), robot motion is abstracted: passing-by robots instantaneously populate random openings signaled by in-assembly robots. We varied the number of simultaneous robot attachments from 1 to 4 and tested our approach on 40,000 randomized swarm shapes, with the largest consisting of more than 250 robots. Huddle successfully formed the desired assembly for all 160,000 trials.

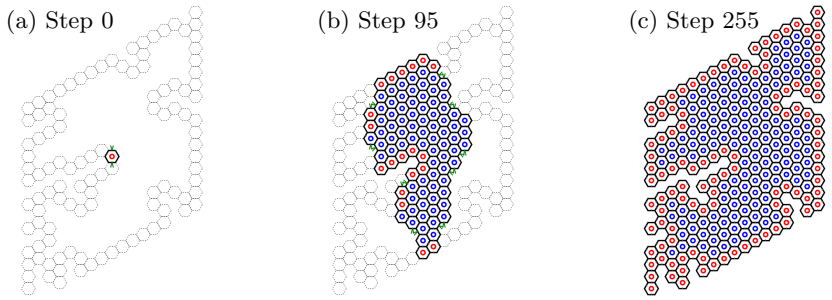


Fig. 4: 256-robot assembly example from the Monte Carlo experiments (one robot attachment per step). Dashed hexagons represent X_b, which are eventually populated by robots marked with red circles; hexagons marked with blue circles are interior robots. Note that our approach does not distinguish between perimeter and interior robots.

In the physics-based validation, our hexagonal robot (0.18 m sides) uses IR transceivers for both signaling (up to 3 m) and neighbor communication, with simulated magnetic attachment. As Huddle is motion-agnostic, our robots are omnidirectional, moving at a top speed of 0.5 m/s in a diffusive motion (Fig. 5a).

Once a robot joins the assembly by attaching to ≥ 1 in-assembly robot, it exchanges information with its neighbor(s) at every time step. Each robot i transmits two elements: (1) its wall status vector $s_i(t)$ (encoded as a single 16-bit unsigned integer), and (2) its hexagonal coordinates $x_i = (p_i, q_i)$ combined with the unit vector $\hat{u}_{i,w}$ of the transmitting wall w (jointly encoded as a 32-bit unsigned integer). Communication only occurs between in-assembly robots (as noted in earlier sections).

Due to continuous-space motion, the orientation of a joining robot is unlikely to match the attracting robot's. To that end, joining robots transform their wall directions to a *north-front* orientation (Fig. 1c) using the attracting robot's $\hat{u}_w$, ensuring Huddle works as described in Section 4.

We conducted an experiment with 107 robots in a square arena of 179 m^2 to form a highly non-convex shape: the calligraphic letter "$\mathcal{H}$" (Fig. 6). Over

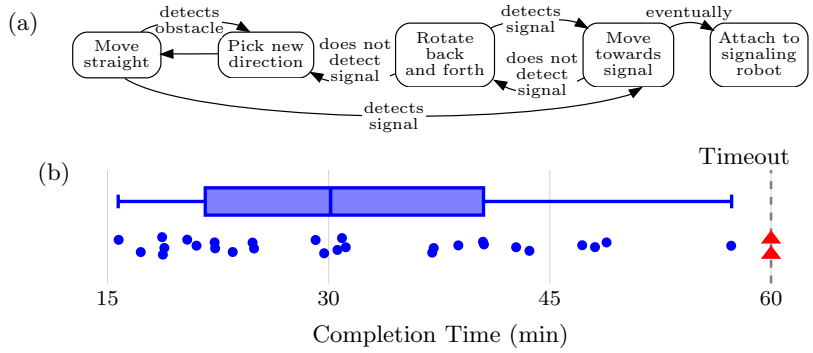
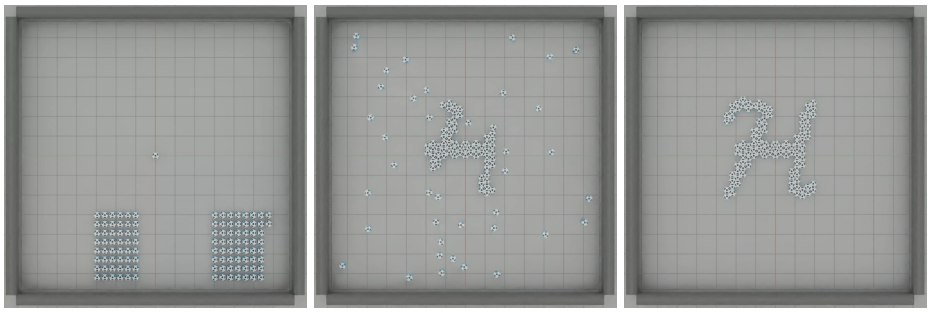


Fig. 5: (a) Robot diffusion motion state diagram. (b) Completion time of a 107-robot assembly forming "$\mathcal{H}$", where 2/30 trials timed out (1 robot remaining).

30 trials, only two did not complete the shape within a simulated hour, timing out with 106 in-assembly robots (Fig. 5b). While this shows a high success rate ($> 99\%$ of the shape is assembled in all trials), more efficient motion strategies can accelerate the assembly's completion, as we discuss in Section 7. We include a video demonstrating complete shape assembly in the supplementary materials.



(a) (b) (c)

Fig. 6: 107 robots forming "$\mathcal{H}$". (a) We place the root robot at the world origin, while the rest of the robots start in two clusters. (b) The robots diffuse within an arena bounded by four walls, joining the assembly as attracted by signals. (c) Each robot eventually finds an opening to attach, completing the assembly.

7 Discussion

Using only the shape perimeter, Huddle enables the parallel, asynchronous assembly of arbitrary shapes by robots with continuous motion. The robots do

not need to be identified or require pre-determined starting locations or goal assignments. We make no assumptions about the robots' motion, allowing order-independent assembly as long as they can encounter signals—intentionally or by chance. Once part of the assembly, each robot communicates limited information only with their nearest neighbors.

While pose localization is not required, the assembly completion rate can improve by using robots with more efficient motion control strategies that may rely on localization. This allows for robot navigation that can either take a decentralized form—such as circling the assembly (similar to [20])—or a more centralized one—such as clustering locations (near assembly openings) computed and disseminated by a server.

That said, there are practical limitations to using Huddle. In specifying the shape, the designer must be careful to avoid shapes that can cause signal occlusion during assembly; shapes containing one-robot-wide gaps tend to cause this—especially if those gaps are long continuous channels (Fig. 4c). Huddle also does not account for physical effects of signals which, depending on the medium, may be significant. For instance, unlike LED signals, two IR signals targeting the same opening may interfere (Fig. 3b).

Though Huddle is designed with hexagonal units in mind, the approach extends to square units, another shape that admits monohedral tessellation (tiling by congruent copies). To do this, we adjust Definition 1 to allow for at most 2 simultaneous connections. The four-wall structure simplifies flank signaling logic, as each lateral direction corresponds to a single wall rather than two. However, Huddle does not extend trivially to triangular units, the remaining regular polygon admitting monohedral tessellation.

Our formulation using hexagons lends itself to circular robots as well, since they can each have at most 6 neighbors (2-D kissing number). If rules about robot orientation can be enforced, Huddle can be extended to shape formation applications, like in [7,14,21], where a robot's circular footprint can either be its physical shape or safety boundary.

8 Conclusion

We presented Huddle, a parallel shape assembly algorithm that enables decentralized, minimalistic robots to form arbitrary shapes. With only the perimeter as input, Huddle guides assembly expansion without bespoke motion strategies, relying instead on signaling mechanisms alone. We proved that Huddle activates signals in a controlled manner such that the assembly will never contain unreachable openings or gaps. We also demonstrated its effectiveness with a high success rate in forming a highly non-convex shape despite using robots with unsophisticated motion. As future work, we aim to validate Huddle on real robotic platforms and explore avenues to reconfigure the assembly.

References

1. Bray, E., Groß, R.: Recent Developments in Self-Assembling Multi-Robot Systems. Current Robotics Reports **4**(4), 101–116 (Nov 2023). https://doi.org/10.1007/s43154-023-00106-y
2. Feshbach, D.A., Sung, C.: Reconfiguring Non-Convex Holes in Pivoting Modular Cube Robots. IEEE Robotics and Automation Letters **6**(4), 6701–6708 (Oct 2021). https://doi.org/10.1109/LRA.2021.3095030
3. Gauci, M., Nagpal, R., Rubenstein, M.: Programmable Self-disassembly for Shape Formation in Large-Scale Robot Collectives. In: Groß, R., Kolling, A., Berman, S., Frazzoli, E., Martinoli, A., Matsuno, F., Gauci, M. (eds.) Distributed Autonomous Robotic Systems, vol. 6, pp. 573–586. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-73008-0_40
4. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous Self-Assembly in Swarm-Bots. IEEE Transactions on Robotics **22**(6), 1115–1130 (Dec 2006). https://doi.org/10.1109/TRO.2006.882919
5. Heckbert, P.S.: Graphics Gems. Elsevier (Nov 2013)
6. Johnson, D.B.: Finding All the Elementary Circuits of a Directed Graph. SIAM Journal on Computing **4**(1), 77–84 (Mar 1975). https://doi.org/10.1137/0204007
7. Li, G., St-Onge, D., Pinciroli, C., Gasparri, A., Garone, E., Beltrame, G.: Decentralized progressive shape formation with robot swarms. Autonomous Robots **43**(6), 1505–1521 (Aug 2019). https://doi.org/10.1007/s10514-018-9807-5
8. Liu, C., Lin, Q., Kim, H., Yim, M.: SMORES-EP, a Modular Robot with Parallel Self-assembly. Autonomous Robots **47**(2), 211–228 (Feb 2023). https://doi.org/10.1007/s10514-022-10078-1
9. Liu, C., Whitzer, M., Yim, M.: A Distributed Reconfiguration Planning Algorithm for Modular Robots. IEEE Robotics and Automation Letters **4**(4), 4231–4238 (Oct 2019). https://doi.org/10.1109/LRA.2019.2930432
10. Matthey, L., Berman, S., Kumar, V.: Stochastic strategies for a swarm robotic assembly system. In: 2009 IEEE International Conference on Robotics and Automation. pp. 1953–1958 (May 2009). https://doi.org/10.1109/ROBOT.2009.5152457
11. O'Grady, R., Christensen, A., Dorigo, M.: SWARMORPH: Multirobot Morphogenesis Using Directional Self-Assembly. IEEE Transactions on Robotics **25**(3), 738–743 (Jun 2009). https://doi.org/10.1109/TRO.2008.2012341
12. O'Hara, I., Paulos, J., Davey, J., Eckenstein, N., Doshi, N., Tosun, T., Greco, J., Seo, J., Turpin, M., Kumar, V., Yim, M.: Self-assembly of a swarm of autonomous boats into floating structures. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 1234–1240 (May 2014). https://doi.org/10.1109/ICRA.2014.6907011
13. Pinciroli, C., Birattari, M., Tuci, E., Dorigo, M., Zapatero, M.D.R., Vinko, T., Izzo, D.: Self-Organizing and Scalable Shape Formation for a Swarm of Pico Satellites. In: 2008 NASA/ESA Conference on Adaptive Hardware and Systems. pp. 57–61. IEEE, Noordwijk, The Netherlands (Jun 2008). https://doi.org/10.1109/AHS.2008.41
14. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (Aug 2014). https://doi.org/10.1126/science.1254295
15. Saldaña, D., Gabrich, B., Whitzer, M., Prorok, A., Campos, M.F.M., Yim, M., Kumar, V.: A decentralized algorithm for assembling structures with modular robots. In: 2017 IEEE/RSJ International Conference on Intelligent Robots

and Systems (IROS). pp. 2736–2743. IEEE, Vancouver, BC (Sep 2017). https://doi.org/10.1109/IROS.2017.8206101
16. Stoy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems **54**(2), 135–141 (Feb 2006). https://doi.org/10.1016/j.robot.2005.09.017
17. Sun, G., Zhou, R., Ma, Z., Li, Y., Groß, R., Chen, Z., Zhao, S.: Mean-shift exploration in shape assembly of robot swarms. Nature Communications **14**(1), 3476 (Jun 2023). https://doi.org/10.1038/s41467-023-39251-5
18. Tolley, M., Lipson, H.: On-line assembly planning for stochastically reconfigurable systems. I. J. Robotic Res. **30**, 1566–1584 (Oct 2011). https://doi.org/10.1177/0278364911398160
19. Tuci, E., Alkilabi, M.H.M., Akanyeti, O.: Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art. Frontiers in Robotics and AI **5**, 59 (May 2018). https://doi.org/10.3389/frobt.2018.00059
20. Werfel, J., Bar-Yam, Y., Rus, D., Nagpal, R.: Distributed construction by mobile robots with enhanced building blocks. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. pp. 2787–2794 (May 2006). https://doi.org/10.1109/ROBOT.2006.1642123
21. Yang, H.a., Li, Y., Duan, X., Shen, G., Zhang, S.: A parallel shape formation method for swarm robotics. Robotics and Autonomous Systems **151**, 104043 (May 2022). https://doi.org/10.1016/j.robot.2022.104043