SMAC: Symbiotic Multi-Agent Construction

Caleb Wagner, Neel Dhanaraj, Trevor Rizzo, Josue Contreras, Hannan Liang, Gregory Lewin, and Carlo Pinciroli

***Abstract*—We present a novel concept of a heterogeneous, distributed platform for autonomous 3D construction. The platform is composed of two types of robots acting in a coordinated and complementary fashion: (i) A collection of communicating *smart construction blocks* behaving as a form of growable smart matter, and capable of planning and monitoring their own state and the construction progress; and (ii) A team of *inchworm-inspired builder robots* designed to navigate and modify the 3D structure, following the guidance of the smart blocks. We describe the design of the hardware and introduce algorithms for navigation and construction that support a wide class of 3D structures. We demonstrate the capabilities of our concept and characterize its performance through simulations and real-robot experiments.**

***Index Terms*—Building automation, distributed robot systems, multi-robot systems, robotics and automation in construction, swarm robotics.**

I. Introduction

MULTI-ROBOT systems promise solutions for construction at much larger scales than the robots themselves. With many robots working in parallel, construction can be completed in a fast and efficient manner. Decentralization may also enable robust solutions with no single point of failure, paving the way for uncrewed construction of colonies in adverse environments such as the Moon or Mars.

The design space of possible approaches to collective construction is vast, and it requires a careful co-design of hardware, software, and fabrication [1]. A particularly important open problem is how to coordinate the construction process autonomously. In this letter, we frame it as the study of *how intelligence should be distributed across the system* [2], [3].

To address this question, we explore a concept for multi-robot construction that comprises two types of robots. The first type is a *smart construction block*, where all of the blocks as a whole behave as a form of growing smart matter. By communicating with one another, the smart blocks monitor the state of the structure and its construction progress in a decentralized fashion. The second type of robot is the *builder*, unaware of the global target structure but able to manipulate and transport blocks in 3D thanks to its inchworm-shaped body. Builders perform localization and path planning by communicating with the smart blocks, which perform the necessary computation. The two types of robots constitute a *symbiotic* system in that their continuous collaboration is necessary for success. For this reason, we called our approach *SMAC — Symbiotic Multi-Agent Construction*.

Related work. The coordination of robotic construction through interaction with the environment is inspired by stigmergy in social insects [4]. Several approaches have been proposed in this context, both in simulation [2], [5]–[9] and with real robotic platforms [10]–[15]. In particular, AMAS [2], [14] is a smart block / inchworm concept where the building material is capable of storage, communication, and power transmission. With AMAS, the inchworms perform construction in a decentralized manner through a gradient-based algorithm where the blocks act as a shared 'blackboard'. SMAC is based on a different mechatronic design, where the blocks plan the construction rather than store information. SROCS [13] is another approach based on smart blocks used for local information storage. An extension of SROCS in which the smart blocks participate in planning construction is currently being developed [9]. However, the builder robot is a crane incapable of traversing the ongoing structure, making it impossible to build structures beyond the builder's own size. The problem of navigating and localizing within an ongoing structure has been the subject of several influential works. Localization is typically achieved by tracking the number of blocks covered since the entry point of the structure. In addition to the already cited AMAS [14], TERMES [16] pioneered a concept in which a wheg-equipped robot, co-designed with a convex block, constructs 2.5D structures by block deposition and removal. The robot follows a plan calculated by an optimization algorithm [17] with the ability to recover from placing errors. Bill-E [18] is another inchworm robot designed to navigate and build versatile 3D structures composed of special blocks that latch magnetically, rather than mechanically as in AMAS and SMAC. Planning is centralized in a dedicated machine [15], rather than being performed by the inchworms or the blocks.

Novelty and contributions. This work explores the co-design space of collective construction with symbiotic robot teams comprising growing active matter and mobile robots. In this domain, our letter provides four main contributions:

1) We propose the co-design of two types of robots, both of which present novel challenges and solutions;
2) We introduce planning algorithms that achieve construction and navigation of 3D structures;
3) We showcase a subclass of structures that are achievable through our system;

Manuscript received October 15, 2020; accepted February 9, 2021. Date of publication March 1, 2021; date of current version March 22, 2021. This letter was recommended for publication by Associate Editor S. Hauert and Editor M. A. Hsieh upon evaluation of the reviewers' comments. *(Caleb Wagner and Neel Dhanaraj contributed equally to this work.) (Corresponding author: Carlo Pinciroli.)*

The authors are with the Department of Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: cwagner@wpi.edu; dhanaraj@usc.edu; tarizzo@wpi.edu; jdcontrerasalbuj@wpi.edu; hliang2@wpi.edu; glewin@wpi.edu; cpinciroli@wpi.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3062812, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3062812

4) We demonstrate the effectiveness of our approach through simulations (particularly regarding scalability) and real-world experiments.

The letter is organized as follows. In Section II we list the main requirements. In Section III we describe the hardware design, and in Section IV we describe our algorithms for planning and navigation. We report the results of the evaluation of our approach in Section V and discuss conclusions in Section VI.

II. Problem Statement

SMAC is designed to explore novel questions regarding large-scale, multi-robot 3D construction. In particular, we aim to separate the planning/monitoring logic from the physical aspects of construction, such as navigation and block placement. This enables the study of research questions centered around the concept of *extended stigmergy* [3], in which simple, reactive robots build smart structures composed of modules capable of dynamic monitoring and (re)planning.

For the purposes of this work, we consider three high-level design requirements to achieve this vision:

1) We want our builders to navigate over challenging 3D structures. The robots must also be capable of manipulation/placement of blocks in any part of the structure.
2) We want our smart blocks to locally compute and communicate information pertinent to the ongoing construction task. Communication must occur both among blocks and between a block and an attached inchworm.
3) Finally, we aim to demonstrate the potential of this kind of distributed, spatial intelligence by devising algorithms that achieve construction efficiently and scalably.

III. Robot Platforms

A. Inchworm Builder and Smart Block Co-Design

We first define the parameters and decisions that dictate the inchworm and block co-design.

Smart block side length. The first design parameter is the side length L of the smart blocks, which we consider the 'unit' of the lattice constituting any target structure. A large side length would allow for better electronics and mechanical capabilities for a block, but it would also increase the size and weight of the inchworm. A short length, on the other hand, allows for a smaller, lighter inchworm, while limiting the capabilities of the smart block.

Inchworm-block docking interface. The docking interface between an inchworm and a block face dictates how the inchworm climbs a structure and manipulates a block, as well as how the blocks assemble to become a structure. We selected a 4-dowel pin-to-hole design for two reasons: *(i)* this type of interface can sustain sufficient shear loads; and *(ii)* it fosters alignment, which is beneficial for both inchworm locomotion and placing blocks onto the structure. An actuated screw is used to create a positive connection between two faces. The interface is shown in Fig. 2.

Inchworm-block communication. The communication interface between an inchworm and a block occurs between the inchworm's end effector and any female face of a block. This allows the inchworm to both communicate and localize itself on the structure without ambiguity. Localization of both the smart blocks and the inchworm is achieved through communicating the coordinates, axis, and direction of the face engaged in communication.

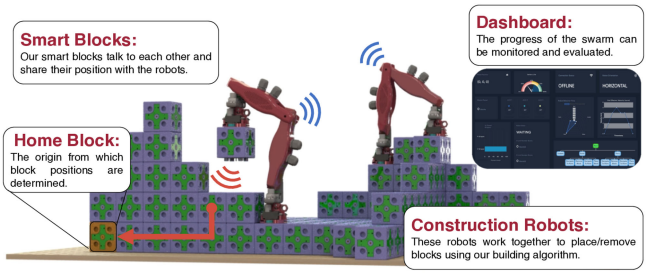

Fig. 1. Overview of proposed concept.

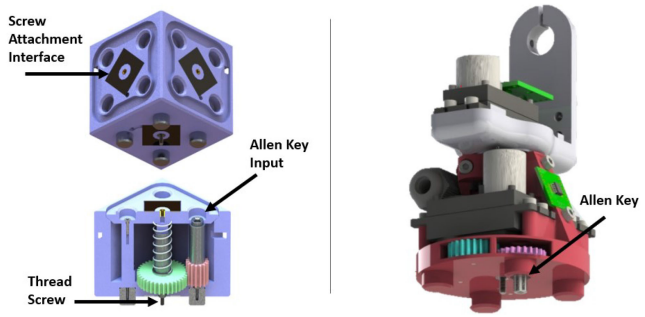

Fig. 2. CAD model of the smart block and inchworm end effector, showing the common interface mechanism used by both designs.

B. Inchworm Builders

In the literature, robots such as wheg-equipped [16], aerial [19], quadrupedal [20], and bipedal robots [14], [18] have demonstrated the traversal of simple 3D structures to place building material with varying levels of dexterity and control complexity [1]. In the quest for a versatile design with low control complexity, we identified an inchworm-inspired platform as a satisfactory solution [14], [18]. An inchworm can traverse over almost all block structures and can pick and place blocks onto structures at any position and orientation in their workspace. This capability does not compromise other important factors such as weight and cost.

Inchworm design summary. Our inchworm robot is a symmetric, bipedal, serial-linkage robot with 6 links and 5 rotational joints (see Fig. 3). Two of these joints are wrist joints located at each end effector, which enable the inchworm to traverse 3D structures and manipulate the blocks along 2 axes of rotation.

Link design and analysis. We derived a relationship between the required link lengths and the fundamental inchworm motion configurations to ensure versatile 3D navigation. Fig. 4 shows the fundamental motions that we considered. For this analysis, the side view of the inchworm simplifies the analysis by reducing the problem to a 4-link robot (labeled A through D in Fig. 4). The minimum link length is constrained by the robot's ability to navigate a convex corner without collision as shown on the right of Fig. 4. To determine a relationship between the inchworm link

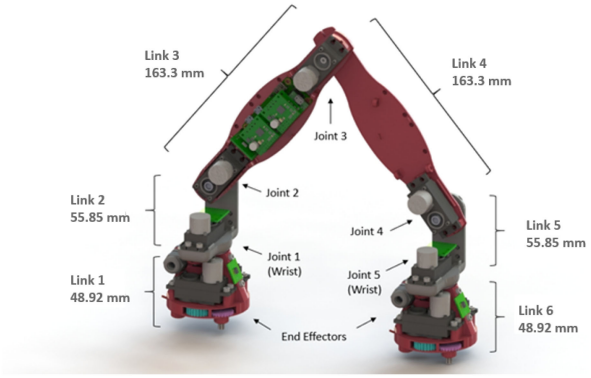

Fig. 3. CAD design of the inchworm builder.

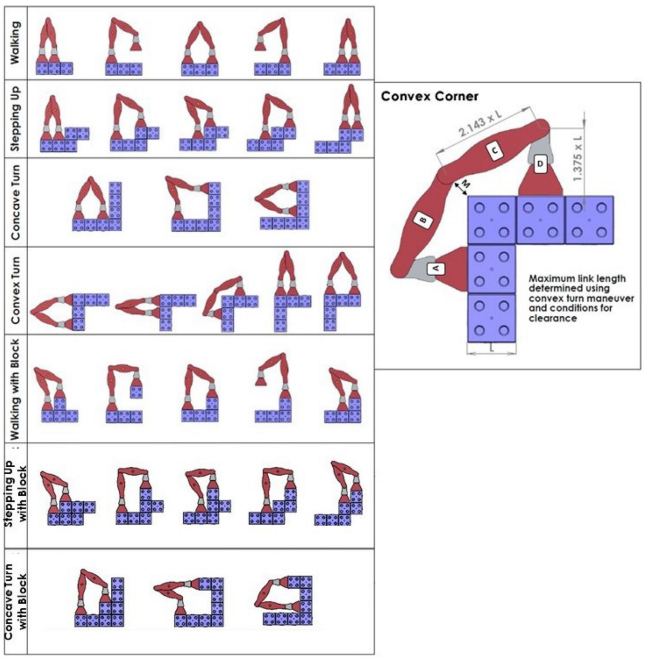

Fig. 4. Navigation capabilities of the inchworm.

length and block side length, L, we expressed the link lengths as vector loop equations [21]. To solve these equations in the convex corner configuration, we set the length of the A and D links to $1.375\,L$ and the clearance vector between the middle joint and the block corner to $0.25L$. By solving the vector loop equations, the relationship between the block side length, L, and the inchworm link lengths are expressed by $L_{\mathrm{A}} = L_{\mathrm{D}} = 1.375L$ and $L_{\mathrm{B}} = L_{\mathrm{C}} = 2.144L$. Furthermore, if we define a worst-case scenario as a fully extended, static robot that is perpendicular to the direction of gravity, the required joint torque increases quadratically with respect to L. Based on budget considerations and a review of available commercial off-the-shelf actuator options, the standard form factor RC-servo was the strongest candidate for joint actuators. We therefore defined the block side length as $L = 76.2\,\mathrm{mm}$ and the current inchworm prototype is realized using RC-servo motors. The link lengths of the final inchworm are reported in Fig. 3. The resulting inchworm is large enough to fully enclose the necessary electromechanical components.

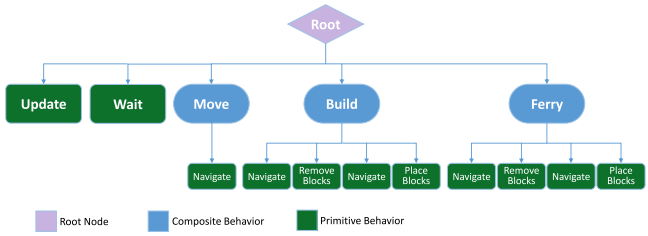

Fig. 5. Inchworm robot behavior tree.

Motor selection. To specify a suitable motor, we calculated the dynamic torques for the worst-case scenario where each joint has a positive instantaneous velocity and acceleration. For this calculation, we considered a joint speed of 30 °/s, while the maximum acceleration was determined from a trapezoidal velocity profile [22]. Using a point mass dynamic model of the robot, the worst-case dynamic joint torque resulted in 0.17 Nm. We chose a standard form factor JX PDI-HV5932 mg 30 Nm servo as it provides a safety factor of 1.5 based on the worst-case dynamic joint torque.

Gripper end effector design. The end effector design, depicted in Fig. 2, consists of a flat disk with four pins integrated into the surface, as seen on the right of Fig. 2. The pins are 12.7 mm in diameter and absorb the shear loads on the robot-to-structure interface and align the end effector with the block. We use a single, actuated 4–40 threaded rod to screw into the structure and absorb the normal load acting on the interface. The rod and pin ensure that the overall connection can sustain the loads required to be attached to the structure. Based on the dynamic model of the inchworm, the maximum torque between the inchworm and structure was calculated to be 25.7 Nm during fast movements. The maximum expected load on the screw is 67.6 N, which is well within the proof load of the gripper screw. To attach a block, the gripper has an actuated Allen key that is used to turn the block screwing mechanism. Lastly, each gripper has a Near-Field Communication (NFC) antenna that is used to communicate with the structure. NFC is discussed in Section III-C.

Electronic design. The inchworm uses a Teensy 3.6 microcontroller and Raspberry Pi Zero for low-level control and joint trajectory planning. We added 14-bit, absolute position encoders (AS5048B from AMS) to the joints to provide precision position feedback ($0.02\,^{\circ}$ resolution). The current prototype of the robot is powered by a $9\,V$ power supply instead of a battery for simplicity in conducting experiments.

Inchworm behaviors. We structured the inchworm behaviors into a behavior tree [23]. We found that this approach fits well with the reactive nature of the inchworm behavior, while being simple and extendable. The main subtrees involve *(i)* receiving updates from the structure; *(ii)* waiting; *(iii)* moving to a specified location; and *(iv)* building sections of the structure. Each of these subtrees is composed of simple behaviors such as picking up blocks, pathfinding, and placing blocks. With reference to Fig. 5, the behaviors are prioritized top to bottom, left to right. Individual behaviors are described in the Planning and Navigation section.

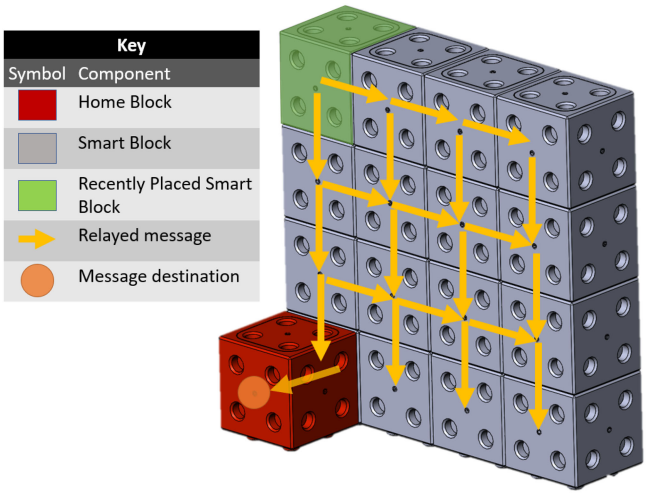

Fig. 6. Structure made of smart blocks showing the flooding communication strategy.

C. Smart Blocks

To address the second design goal in Section II, our smart blocks attach to and communicate with nearby blocks and inchworms, which allows the following functionalities: system awareness of structure configuration, detection of failed or detached blocks, and communication of location or structure status to inchworm.

Communication. For communication to be possible across all faces of a smart block, we considered several technologies, including Inter-Integrated Circuit (I^2C), Universal Asynchronous Receiver/Transmitter (UART) and NFC, the latter being favored because it eliminated the need to design inter-block electrical connections required for the other methods. The final implementation uses six NXP PN532 NFC controllers and antennas, one for each face, all controlled by an Arduino Mini microcontroller. For two adjacent block faces to communicate over NFC, one NFC controller must be in card emulation mode and one in card reader mode. For a block to receive messages, the NFC controller goes into card emulation mode and the microcontroller polls each block face for an incoming message. For a block to send a message, the NFC controller goes into card reader mode and the message is relayed to the receiving NFC face. The microcontroller keeps the NFC block faces in card reader mode until the message transmission has been confirmed. By default, each block face is set to card emulation mode and is only set to card reader mode when sending a message. This behaviour allows for synchronous communication between 2 blocks. As shown in Fig. 6, the blocks communicate through a flooding algorithm in which every block broadcasts messages on all its faces.

Inchworm-block connections. Each block has five female and one male connection face. The male connection face uses a 4–40 threaded screw, like the robot gripper, to attach to a structure. The other five female faces have threaded inserts for connections with male faces. The top female face (opposite the male face) has an Allen key input (see Fig. 2) to activate the attachment screw. The inchworm uses the screwing mechanism in the following sequence:

1) The inchworm actuates its end effector screw to connect to the block's top female face;
2) The inchworm actuates the Allen key to engage (attaching) or disengage (releasing) the attachment thread screw on the block;
3) After placing the block, the inchworm disconnects from the block's female face.

Power and LEDs. Each block is powered by a 600 mAh LiPo battery. Average battery life is about 4 hours. Each block face has programmable Neopixel LEDs, which are used to convey the state of the block to the user. For example, green means the block is incorporated into a structure, flashing yellow indicates a block is waiting to be added to the structure, and flashing red indicates that an adjacent block has been removed or failed.

Home blocks. A variation of the smart block, the *home block*, serves as structure seed and initial planner for the entire system. This block is equipped with a Raspberry Pi Zero.

D. Structure Perception Functionality

Three key functions leverage the symbiosis between the blocks and the inchworms:

1) **Structure Status.** The blocks update the structure's current configuration as blocks are dynamically added.
2) **Robot Position.** Each inchworm requests its own position and the position of other robots with respect to the structure, enabling localization in 3D space.
3) **Heartbeat Protocol.** Each block periodically communicates its status to neighboring blocks, allowing the structure to compare the current configuration to the previous one, thereby identifying missing or failed blocks.

IV. Planning and Navigation

We devised algorithms for planning and navigation that take advantage of the hardware design presented in Section III.

A. Planning

The planning algorithm (see Fig. 8) is responsible for calculating the sequence of operations that must be accomplished to build the target structure. These operations include retrieving and placing blocks in the locations indicated in the blueprint of the target structure. The algorithm is executed by the home block (or blocks, in case multiple home blocks are deployed).

Divisions. Because navigation is a time-consuming activity for the inchworms, the planning algorithm splits the structure into *divisions*. Each inchworm claims a specific division. Inside of a division, the inchworm receives instructions from and communicates to the planner through the smart blocks.

Ferrying. When inchworms are notified that a new block is available for placement, the inchworm in the nearest division picks the block up and either places it in the inchworm's own division, or carries it to the border of one of the adjacent divisions. The inchworm responsible for the adjacent division picks up the block and repeats the above procedure. We call *ferrying* the act of passing blocks across divisions until the target division is reached. Ferrying, akin to a bucket-brigade, is a form of task partitioning [24], [25] that lowers the navigation cost

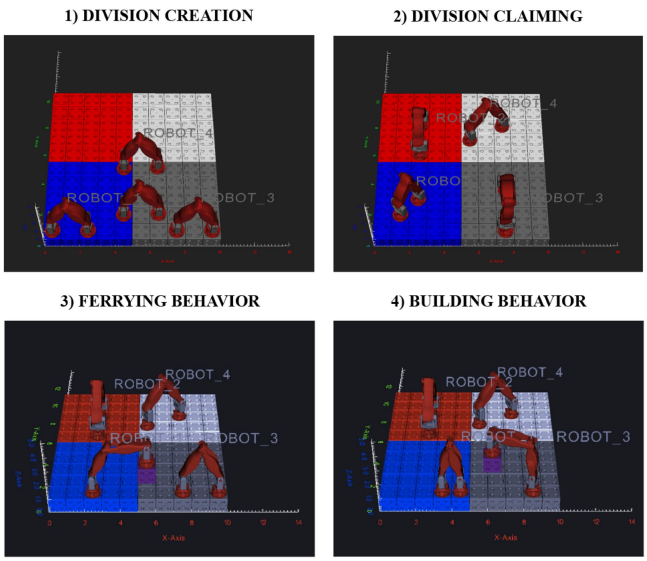

Fig. 7. Examples of steps of building algorithm.

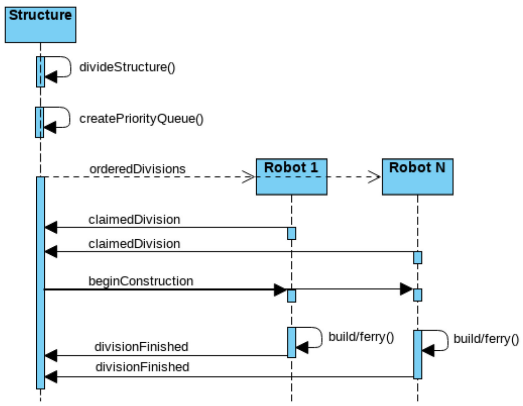

Fig. 8. Sequence diagram of the building algorithm executed for every layer of the target structure.

of the inchworms. The home block provides instructions to the inchworms by sending messages through the blocks.

Division creation. Divisions are created by subdividing a given structure into areas where an inchworm is meant to perform work, as shown in Fig. 7 (division creation). While the divisions extend to 3D shapes, only a layer of the structure, defined as a group of blocks 1 unit high, is considered at a time. The size and shape of divisions can vary, but ultimately they are meant to minimize the amount of inching the inchworms must perform to either ferry or build blocks. For the purposes of experimentation, the size and shape of divisions were directly chosen by a human operator and were homogeneous throughout the entire structure. Future improvements involve automating the selection of the size and shape of divisions and allowing heterogeneous divisions. The number of divisions created is dependent on the size of the structure and the size of the divisions. A partial ordering of the divisions is used to construct a priority queue, which determines the order in which divisions should be built. New blocks are introduced into the system at a designated feeding location. For simplicity, the demonstrations here use a single feeding location at the home block. Multiple feeding locations can be used, however, as the concept of a human ferrying blocks to the inchworms is similar to an inchworm ferrying blocks to another inchworm: blocks are placed onto the structure, which is notified of the added blocks, and the home block sends messages to the inchworms. The partial ordering is determined as the relative distance from the feeding location and calculated using the wavefront algorithm. Divisions that are closer to the feeding location are built first. Once these divisions are built, the inchworms traverse the divisions to ferry blocks to divisions farther away from the feeding location.

Division claiming. Once the partial ordering is established for a given layer of the structure, the inchworm robots are tasked with moving to the different divisions, as shown in Fig. 7 (division claiming). A distributed voting algorithm is used to construct a priority queue that ultimately assigns inchworms to divisions. Using the communication capabilities of the smart blocks, each inchworm calculates its distance from all the divisions. The inchworms then place votes on each of the divisions using the calculated distances as the vote value. In a greedy fashion, each inchworm gets the division with the closest distance. If multiple inchworms attempt to claim the same division, a conflict resolution strategy is applied wherein the inchworm with the larger distance (higher vote) claims the division. Because the building algorithm proceeds outwards from the feeding location, conflicts are most likely to arise when an inchworm is equidistant from multiple divisions. By awarding a contested division to the farthest robot, the distance for the 'losing' inchworm typically is not increased substantially, if at all.

Block placement. After the inchworms are distributed, the operator introduces new smart blocks into the system, which signal to the inchworms that new blocks are available. If an inchworm is in a division that has already been built, it ferries the block to another division, as shown in Fig. 7 (ferrying behavior). If the division has not been built, the inchworms query the planner for the blueprint of that division and place the blocks accordingly, as shown in Fig. 7 (building behavior). In the ideal scenario, the inchworms place blocks in a spiral fashion, starting from the center of the division and moving outwards. This method was chosen so that after the first block is placed at the center of the division, the inchworm can climb onto the block and act as a fixed-base robot arm, pivoting around the block to place additional blocks. If the shape and size of divisions prevent the inchworm from reaching each location, the robot will inch to place a block. In other situations, the inchworms are unable to move to the center of a division, as there is not a block underneath supporting it. In these scenarios, the inchworm will move to the closest block it is capable of standing on and build outwards from there. When ferrying, the inchworms place blocks towards the outer edges of their own division such that a robot in an adjacent division has to extend as little as possible into the initial division, thus minimizing the chances of collision or periods where an inchworm must wait for another inchworms to move.

Reactive replanning. Once the inchworms have finished building or ferrying a division, the inchworms will disperse, or select the next division from the building algorithm priority queue and move to that division. In the new division, they will repeat the process of either building or ferrying blocks. Selecting

another division from the priority queue while other inchworms are ferrying or building is used to reduce the time the inchworms spend waiting for other inchworms to finish. The procedure of dividing a layer into divisions, dispersing the inchworms, and then building or ferrying blocks through the divisions is repeated for each layer in the structure until the entire structure is built.

Achievable structures. Our algorithm can build any structure that does not present upside-down L-shape overhangs. It is possible to build simple cantilevers (up to the structural abilities of the blocks) by attaching a male face to an open, vertical (female) face. Because the algorithm builds from the bottom up, however, it is not possible to remove a block from underneath an existing block. It is thus impossible to use a temporary support block for the lower block of an inverted L-shape. Exploring these scenarios is left as future work.

B. Navigation

In order to reach any given location on the structure, the inchworm robots must be capable of finding a traversable path to the given location.

We developed a simple path planning algorithm, called *Face**, that takes advantage of the versatile mobility of the robots. Face* is an extension of the A* algorithm that treats the exposed faces of each block as nodes in a network, and uses the Manhattan distance to each node as the cost associated with reaching that node. Analogously to the A* algorithm, Face* selects the node with the lowest associated cost. For each node, an inverse kinematics simulation ensures no collisions with the structure are possible.

It is important to note that a path is only generated for the end effector closest to the goal, not for both end effectors. The second end effector does not require a separate path to be generated as it can follow in the footsteps of the first end effector. Also note that if an end effector is not engaged to the structure, Face* first considers if the already disengaged end effector is capable of reaching the given position, which allows the inchworm to swing in place like a fixed-base robot arm and reduce unnecessary inching. Finally, to avoid the possibility of two robots colliding while they are moving, the algorithm expands the area around (adds padding to) all other robots and excludes the expanded regions from its search. The amount of padding that is added is set to the length the robots are able to reach in all directions.

V. Experimental Evaluation

A. Simulated Experiments

We developed a custom simulation environment to visualize and collect statistics on the performance of the system. The simulation environment was built using the Visualization Tool Kit (https://vtk.org/). The experiments were run on a 2015 MacBook Pro. In our simulated experiments, the smart blocks could connect along all the faces, in contrast to the single male face connection available on the real smart blocks we built.

Metrics. The performance metrics we considered for the simulated experiments are *(i)* whether the construction process succeeded; and *(ii)* the speed at which the structure was constructed, measured in simulated time steps.

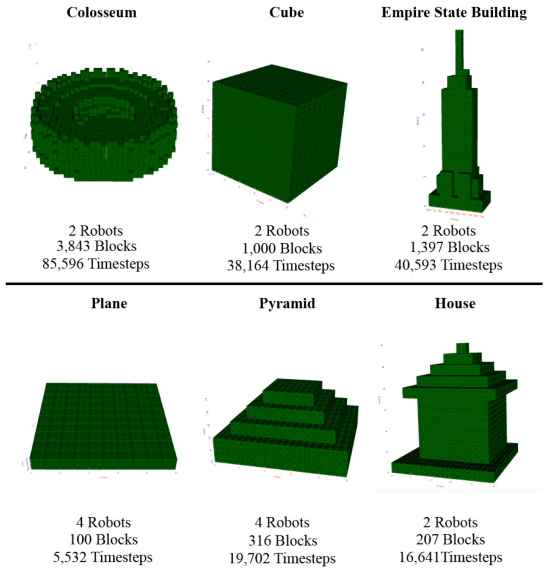

Fig. 9. Examples of structures achievable by our system.

Setup. We tested our approach with diverse structures including pyramids, temples, churches, castles, the Empire State Building, and the fictional Star Trek Reliant ship, among others (see Fig. 9). We compared the construction speed for different numbers of inchworms to investigate the effect of adding additional robots on construction time. In the simulations, inchworms were randomly placed on a surface composed of smart blocks. The inchworms were then tasked with distributing themselves according to the distribution algorithm explained in Section IV-A. After the inchworms had distributed themselves, a virtual operator sets smart blocks down in a corner of the layer, designated as the feeding location. Prompted by the introduction of a new block into the system, the inchworms reacted by moving the blocks throughout the system as explained in Section IV. For all simulations, the inchworms were able to place all blocks.

Scalability study. The results of two experiments are reported in Fig. 10. The experiments involve 1 to 4 robots tasked with building a 10×10 plane and a pyramid of 316 blocks. As shown in Fig. 10, the time decreases with additional robots, especially when comparing a single robot with four robots.

Task specialization. To analyze how collaboration between inchworms affects performance, we recorded the total amount of time spent in each behavior by the inchworms. The main behaviors tracked were receiving updates from other robots and the structure (*Update*), moving (*Move*), waiting (*Wait*), building (*Build*), and ferrying (*Ferry*). The results shown in Fig. 11 refer to the same structures considered in the scalability study. For the single inchworm experiment, the inchworm spends almost all its time either ferrying or building. Note that, in this experiment, the single inchworm ferries to itself; it ferries a block to the next division, and then moves to that division to ferry/build with that block. Almost no time is spent waiting, and a very small portion of time is spent receiving updates. This is expected, as there are no other inchworms with which it must coordinate.

For the experiments with multiple robots, we observed that the time spent across the different behaviors is very heterogeneous. Although capable of exhibiting all the behaviors, the inchworms

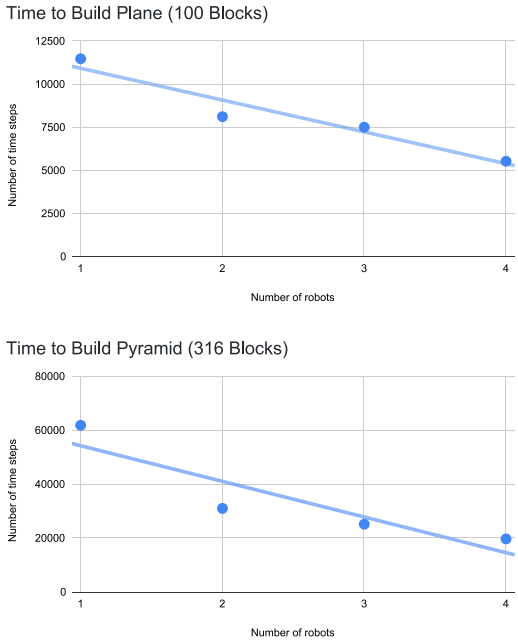

Fig. 10. Time required to build structures for different numbers of inchworms.

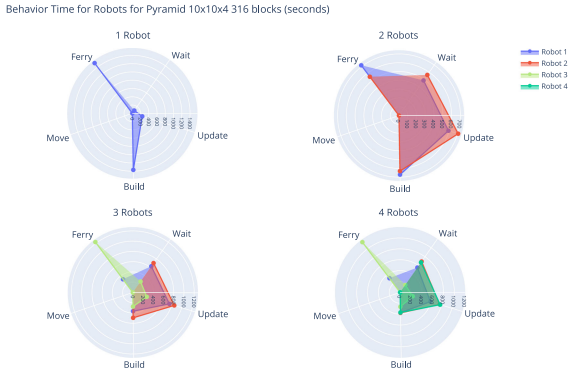

Fig. 11. Time spent performing behaviors for different numbers of robots.

specialized themselves into different groups, in which some of the inchworms focus more on ferrying while others focus more on building. In particular, we observed that one inchworm did most of the ferrying, while the other inchworms mainly focused on building. The appearance of task specialization depends on the type of structure built. We also observed that the waiting time and the time spent receiving updates of all inchworms is much higher compared to the single inchworm experiment. By increasing the number of inchworms, the need for increased coordination demands more communication between the inchworms and the smart blocks. Samples of these tests are reported at https://www.nestlab.net/doku.php/papers:smac.

B. Real-World Experiments

The goal of the real-world experiments was to evaluate the hardware and investigate whether some of the results found in the simulation would be consistent with the hardware. We constructed one inchworm and five smart blocks.

Unit testing. We performed unit tests to verify inchworm motions needed for construction. We also tested the smart blocks

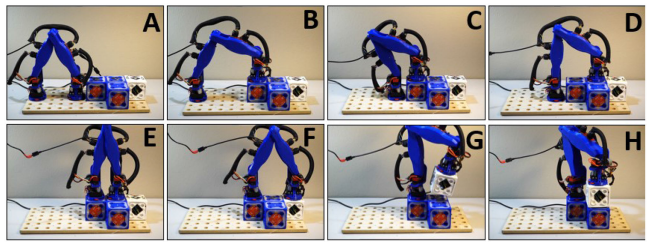

Fig. 12. In A-D robot steps onto structure, E-F robot walks on structure, G-H robot picks up and places block.

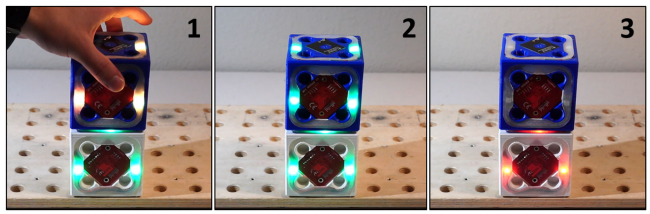

Fig. 13. 1) A block is added to the structure. 2) The top block turns green to indicate "all OK." 3) The bottom block changes color to red after it detects an error in the top block, which has gone offline.

to detect when new blocks were added or removed, or when a block went offline, which could be attributed to a fault, dead battery, or other error. The tests were all successful. Samples of these tests are reported at https://www.nestlab.net/doku.php/papers:smac.

Inching precision test. We tested the precision of the inching motion to estimate how often an inchworm may fail. The inchworm was made to take an inching step and retract 10 times. This experiment was also replicated while an inchworm was taking a step with a block. It was found that the inchworm was able to successfully step and grip 100% of the time and step while carrying a block 80% of the time, which, while not ideal, was sufficient for performing the demonstrations. Future experiments will focus in characterizing the reliability and robustness of the robot.

Inching and gripping test. We tested the capability of our inchworm to take three steps on a plane while carrying a block. We timed the inching speed to be 1 step approximately every 40 s. The majority of this time was spent engaging the end effector screw, with about 20 s on average for engaging with the structure and about 10 s for disengaging with it.

Inchworm system test. A complete system-level test was performed to demonstrate that the inchworm is able to inch, traverse a structure, manipulate a block, and place it on the structure. A series of snapshots from the demonstration are shown in Fig. 12, and video footage is available at https://www.nestlab.net/doku.php/papers:smac. Though designed to complete complex motions such as inching around convex and concave corners or climbing up a vertical wall, more comprehensive tests are left for future work.

Smart block communication. We performed a series of tests to validate that a new block is detected when it is added into the structure, and that the structure can detect when a block is removed from the structure. The images in Fig. 13 show that the blocks change the color of their LEDs to convey the change in the structure. Experimentally, the average time for

two smart blocks to exchange a 62-byte package was 500 ms, which corresponds to an average bit rate of 992 bits/s. We also estimated the impact of a large structure on communication latency. The number of hops between any block and the home block is given by the Manhattan distance between them. A message originating from block at (x, y, z) of a 3D structure must traverse $|x| + |y| + |z|$ blocks to reach a home block at (0, 0, 0). To put this in perspective, a message sent by a block placed in the corner of a $10 \times 10 \times 10$ cube would take 15 s to reach the opposite diagonal corner. This has important implications when building large structures, motivating the need for multiple home blocks to decrease this latency. We will study these aspects in future work.

VI. Conclusions and Future Work

This letter proposes SMAC, a robotic platform for collective construction composed of two types of robots: *(i)* a collection of smart blocks, forming a network that allows dynamic planning and monitoring; and *(ii)* an inchworm-inspired builder robot, able to navigate a 3D structure and deposit blocks under the guidance of the smart blocks. We presented the hardware design of our platform, along with a set of simulated and real-world experiments aimed at demonstrating its capabilities.

Our work is a step in the study of 'extended stigmergy' in multi-robot construction, a promising concept in which the structure being built is an active component of the process [3], simplifying the design of the builder robots. We envision our system as a technology for in-orbit or planetary construction. In addition, while the smart blocks could be used as actual construction material, they would also be suitable as *smart scaffolding* [26]. This would enable reusable, cost-effective technology for remote, large-scale, autonomous construction.

Future work will improve the capabilities of our hardware and explore planning algorithms with extended stigmergy that respect structural stability constraints. In particular, ongoing efforts are directed at a universal mating system to reduce the structural limitations resulting from the gendered connection and the unequally gendered faces.

Acknowledgment

We thank Prof. Raghvendra Cowlagi for guidance and fruitful conversations and Cameron Collins for his assistance in the manufacturing of the hardware of the platform.

References

[1] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, "A review of collective robotic construction," *Sci. Robot.*, vol. 4, no. 28, 2019, Art. no. eaau 8479.

[2] Y. Terada and S. Murata, "Modular stucture assembly using blackboard path planning systems," in *Proc. Int. Symp. Automat. Robot. Construction*, 2006, pp. 852–857.

[3] J. Werfel and R. Nagpal, "Extended stigmergy in collective construction," *IEEE Intell. Syst.*, vol. 21, no. 2, pp. 20–28, Mar./Apr. 2006.

[4] G. Theraulaz and E. Bonabeau, "A brief history of stigmergy," *Artif. Life*, vol. 5, no. 2, pp. 97–116, 1999.

[5] G. Theraulaz and E. Bonabeau, "Coordination in distributed building," *Science*, vol. 269, no. 5224, pp. 686–688, 1995.

[6] A. Grushin and J. A. Reggia, "Stigmergic self-assembly of prespecified artificial structures in a constrained and continuous environment," *Integr. Comput.-Aided Eng.*, vol. 13, no. 4, pp. 289–312, 2006.

[7] J. Werfel, D. Ingber, and R. Nagpal, "Collective construction of environmentally-adaptive structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2007, pp. 2345–2352.

[8] C. Pinciroli, M. S. Talamali, A. Reina, J. A. R. Marshall, and V. Trianni, "Simulating kilobots within argos: Models and experimental validation," in *Proc. 11th Int. Conf. Swarm Intell. (ANTS 2018), Ser. Lecture Notes Comput. Sci.* Berlin, Germany: Springer, Oct. 2018, *in Press*.

[9] Y. Zheng, M. Allwright, W. Zhu, M. Kassawat, Z. Han, and M. Dorigo, "Swarm construction coordinated through the building material," Université Libre de Bruxelles, Belgium, Tech. Rep., 2020.

[10] A. Martinoli, A. J. Ijspeert, and F. Mondada, "Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots," *Robot. Auton. Syst.*, vol. 29, no. 1, pp. 51–63, 1999.

[11] N. Napp and R. Nagpal, "Distributed amorphous ramp construction in unstructured environments," *Robotica*, vol. 32, no. 2, pp. 279–290, 2014.

[12] T. Soleymani, V. Trianni, M. Bonani, F. Mondada, and M. Dorigo, "Bio-inspired construction with mobile robots and compliant pockets," *Robot. Auton. Syst.*, vol. 74, pp. 340–350, 2015.

[13] M. Allwright, N. Bhalla, H. El-faham, A. Antoun, C. Pinciroli, and M. Dorigo, "SRoCS: Leveraging stigmergy on a multi-robot construction platform for unknown environments," in *Proc. Int. Conf. Swarm Intell*. Berlin, Germany: Springer, 2014, pp. 158–169.

[14] Y. Terada and S. Murata, "Automatic modular assembly system and its distributed control," *Int. J. Robot. Res.*, vol. 27, no. 3–4, pp. 445–462, 2008.

[15] B. Jenett, A. Abdel-Rahman, K. Cheung, and N. Gershenfeld, "Material-robot system for assembly of discrete cellular structures," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4019–4026, Oct. 2019.

[16] K. H. Petersen, R. Nagpal, and J. K. Werfel, "TERMES: An autonomous robotic system for three-dimensional collective construction," *Robot.: Sci. Syst. VII*, 2011.

[17] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "Scalable compiler for the TERMES distributed assembly system," *Distributed Autonomous Robotic Systems*. Berlin, Germany: Springer, 2019, pp. 125–138.

[18] B. Jenett and K. Cheung, "Bill-e: Robotic platform for locomotion and manipulation of lightweight space structures," in *Proc. 25th AIAA/AHS Adaptive Structures Conf.*, 2017, Art no. 1876.

[19] R. Stuart-Smith, "Behavioural production: Autonomous swarm-constructed architecture," *Arc. Des.*, vol. 86, no. 2, pp. 54–59, 2016.

[20] R. Volpe, "The lemur robots," 2019. [Online]. Available: https://www-robotics.jpl.nasa.gov/systems/system.cfm?System=5

[21] D. H. Myszka, *Machines and Mechanisms*. Englewood Cliffs, NJ, USA: Prentice Hall, 2004.

[22] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006.

[23] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Boca Raton, FL, USA: CRC Press, 2018.

[24] F. L. Ratnieks and C. Anderson, "Task partitioning in insect societies," *Insectes sociaux*, vol. 46, no. 2, pp. 95–108, 1999.

[25] G. Pini, A. Brutschy, M. Frison, A. Roli, M. Dorigo, and M. Birattari, "Task partitioning in swarms of robots: An adaptive method for strategy selection," *Swarm Int.*, vol. 5, no. 3/4, pp. 283–304, 2011.

[26] E. Komendera, D. Reishus, and N. Correll, "Assembly by intelligent scaffolding," Dept. Comput. Sci., Univ. Colorado at Boulder, Tech. Rep., 2011.