Distributed Construction by Mobile Robots with Enhanced Building Blocks

Justin Werfel*† jkwerfel@mit.edu
Yaneer Bar-Yam†‡ yaneer@necsi.org
Daniela Rus* rus@csail.mit.edu
Radhika Nagpal‡ rad@eecs.harvard.edu

*MIT CSAIL
Cambridge, MA 02139

†New England Complex Systems Institute
Cambridge, MA 02138

‡Harvard University
Cambridge, MA 02138

***Abstract*— We describe a system in which autonomous robots assemble two-dimensional structures out of square building blocks. A fixed set of local control rules is sufficient for a group of robots to collectively build arbitrary solid structures. We present and compare four versions in which blocks are (1) inert and indistinguishable, (2) uniquely labeled, (3) able to be relabeled by robots, (4) capable of some computation and local communication. Added block capabilities increase the availability of nonlocal structural knowledge, thereby increasing robustness and significantly speeding construction. In this way we extend the principle of stigmergy (storing information in the environment) used by social insects, by increasing the capabilities of the blocks that represent that environmental information. Finally, we describe hardware experiments using a prototype capable of building arbitrary solid 2-D structures.**

I. INTRODUCTION

In this paper we present a system for automated construction, in which autonomous mobile robots transport modular building blocks to build a user-specified structure. We present simple, fixed, local control rules by which robots can collectively construct arbitrary two-dimensional structures without internal holes. This is achieved with no direct communication between robots, but rather by using the partially built structure for implicit coordination. We describe significant advantages to be gained from enhancing the blocks' ability to store and process information. If blocks are given some ability for computation and communication with their neighbors in the structure, then robot capabilities can be less, robustness is improved, and the speed of building can be much greater. For applications not permitting blocks of this complexity, allowing blocks to store state (e.g., using passive, low-cost RFID tags) can still substantially improve speed and robustness.

Automating construction could facilitate endeavors such as the production of low-cost housing, and alleviate problems such as high accident rates reported with traditional construction [1]. Automation would be particularly useful in settings where human presence is dangerous or problematic. For instance, robots could be sent to build habitats in extraterrestrial environments, to await later human travelers. Similarly, robot construction systems might be especially well-suited to underwater settings, where human building activity is difficult, but the environment has advantages such as effective weightlessness and mobility in three dimensions; conceivably their use could one day even open up the oceans for colonization.

Swarm approaches, involving larger numbers of simpler robots rather than one or a few with more sophisticated capabilities, can have advantages in such endeavors, in particular with respect to parallelism, decentralization, and robustness. Such systems are suited to work on many subtasks of a project simultaneously. They further can typically absorb the loss of many components without a significant impact on task completion; similarly, they tolerate components acting in no exact order, which is useful because of the difficulty of preplanning robot behavior in detail in uncertain environments.

Swarms of building robots can draw inspiration from social insects such as ants and bees. These insects use stigmergy to guide their building activities: e.g., termites deposit materials in ways influenced by their immediate surroundings, and in turn influence those surroundings by depositing material. In this way the insects communicate implicitly through manipulation of the environment.

Stigmergy is a powerful and simple tool, with limitations: while it can be used to produce structures with given qualitative characteristics [2], it does not easily allow the consistent production of specific structures; and no general principle has been described for taking a particular desired structure and extracting a set of low-level behaviors that building agents can follow to produce it. Here we address both these limitations, describing fixed sets of low-level control rules that can be applied to reliably generate particular structures according to a high-level user-specified design. Further, we discuss benefits to be gained from *extended stigmergy*, increasing the ability of the building materials to embody and process environmental information. In a hardware prototype where a mobile robot builds arbitrary solid structures out of self-aligning blocks, we implement this extended stigmergy using writable RFID tags.

II. RELATED WORK

Our work is complementary to previous research on distributed approaches to automating construction. In some systems, where building elements are moved by robots [3]–[5] or ambient fluid forces [6] to form a given structure, the sequence of element placements is planned by hand in advance. Some researchers have studied other issues related to construction, such as inter-robot communication [3], [7] or debris cleanup [8]. These studies have not focused on the problem of building a particular desired structure given a high-level description.

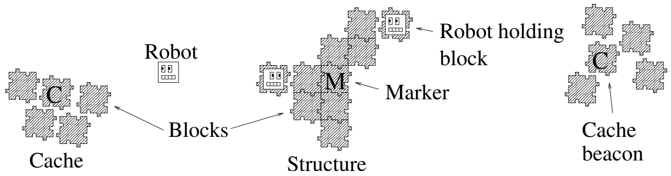

Fig. 1. Overview sketch of the system. As the structure is assembled, it forms a grid with an implicit coordinate system, of which the marker can be taken to be the origin.

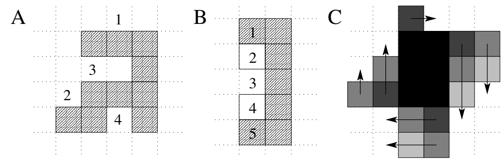

Fig. 2. (A) Examples of valid and invalid prospective attachment sites for a sample structure. Shaded squares represent grid sites already occupied by blocks. A new block can be attached at sites 1 or 2; 3 and 4 are too spatially constrained to allow a block to be maneuvered into position.
(B) Two separated blocks (sites 1, 5) may not be attached in the same row if all sites between them are meant to be occupied. Otherwise, attaching blocks at the intermediate sites (2, 4) will eventually result in an unfillable gap (3).
(C) Building up a structure by layers, always starting new rows from the end and extending them clockwise, as shown by the arrows. Lighter blocks have been attached more recently.

Self-reconfigurable modular robots [9], [10] face the related task of needing to rearrange modules from one configuration to another, subject to some characteristic set of movement constraints, including the requirement that all modules remain connected. In our system we separate the elements into mobile robots and nonmobile blocks, since when building a static structure, blocks need not move again once attached. In this way robots can be reused for other structures, and blocks can be optimized for better structural properties and lower cost. However, some modular robotics researchers have addressed the similar goal of achieving user-specified arrangements of blocks in a decentralized manner [11], [12]; and some features common in modular robot research are useful in a system like that described here, e.g., self-aligning connectors and communication links between physically attached modules.

When blocks are capable of communication and computation, the construction problem is related to that of programmed self-assembly [13], [14]. Such work considers tiles which can change state and binding properties, and investigates how to program them so that they aggregate into desired shapes when mixed. These studies offer alternate approaches for generating rules to produce desired structures from certain classes. However, they do not explicitly consider the geometry of tiles, and constraints it imposes on attachment order.

III. The Collective Construction Problem

In the scenario we consider, mobile robots and caches of building blocks are deployed at random into an obstacle-free workspace. A marker indicates the location for the start of construction. The goal is for the robots to collect blocks from the caches and arrange them into a desired solid structure (i.e., no internal holes), starting at the marker (Fig. 1). The marker and caches broadcast distinct long-range signals which robots can use to navigate to them. The marker is an object with the same geometry and attachment mechanisms as a block, and serves as a 'seed' to which blocks can initially be attached.

We assume that robots have the following capabilities: (a) Robots can move in any direction in the plane, alone or while holding a block, and avoid collisions. (b) They can follow beacons to get to block caches and to the building site. (c) They can follow the perimeter of the partially built structure and recognize when they turn corners. (d) They can take blocks from caches and attach them to the structure. We do not assume that robots can communicate with one another. While robots can be designed to communicate wirelessly, two-way communication in a network with dynamic topology can be unreliable, may scale poorly, and requires ad-hoc multi-hop routing which is non-trivial for mobile entities. We will explore the use of explicit inter-robot communication in future work.

The first three of the robot capabilities above have repeatedly been demonstrated in a variety of autonomous robotic systems [15]. Manipulation of the environment, however, is typically much more difficult. We thus impose two measures for the sake of capability (d). First, we make the conservative assumption that if there are blocks on opposite sides of a potential attachment site, then the site is physically too constrained for a robot to maneuver a new block into (Fig. 2A). This constraint will simplify the task of mechanical design. Moreover, by preventing gaps like that at site 4, situations where a robot needs to maneuver a block down a longer 'tunnel' (like that around site 3) will also be prevented. Secondly, we specify that while robots must be able to bring a block to a desired location, they are not solely responsible for precision alignment. Instead the blocks have self-aligning connectors (active or passive), so that it is sufficient for the robot to get the block close to the attachment site. In §VI we present a hardware prototype that implements many of the robot tasks mentioned above, including self-aligning blocks.

We require that any concavities in the desired shape be wide enough for two perimeter-following robots to pass each other unimpeded. The extent to which this constraint limits possible structures will depend on the hardware implementation.

IV. Distributed Construction Algorithms

We represent the shape to be constructed using a lattice, with a coordinate system whose origin is the marker. Each site in this lattice specifies whether or not a block should be attached at the corresponding site in the structure. We will call this lattice the *shape map*. Building a prespecified structure then means attaching blocks exactly at the sites specified by the shape map.

If a robot with the shape map can determine its position in the structure's coordinate system, then it can trivially determine whether or not a site should be occupied. Because of limitations of position estimation and odometry, establishing position is a very difficult task in general. However, robots can use the structure as a reference to determine their location. Additionally, block edges and other features can be used to correct the effects of odometry errors.

Algorithm 1 Pseudocode procedures for assembly of a structure of inert blocks. An 'end-of-row' site is one where the robot is either about to turn a corner to the left, or the site directly ahead is not supposed to have a block according to the structure design. Below, variant methods for establishing position, for identical, distinct, or writable blocks.

```
    while structure not complete do
      get block from cache
      go to structure
      Establish-Position {see below}
5:    seen-row-start ← false
      while still holding block do
        if (site should have a block) and
            ((site just ahead has a block) or
            (seen-row-start and (at end-of-row)) then
10:         attach block here
        else
          if at end-of-row then
            seen-row-start ← true
          follow perimeter counterclockwise
```

```
1A-Establish-Position: {Identical blocks}
    while not adjacent to labeled side of marker do
      follow perimeter counterclockwise
```

```
1B-Establish-Position: {Distinct blocks}
    while not adjacent to previously known label do
      record label in temporary map
      follow perimeter counterclockwise
    fill in label map from temporary map
```

```
1C-Establish-Position: {Writable blocks}
    read position from neighboring label
```

The order of attachment must be partially constrained, to avoid unwanted gaps caused by intermediate configurations which prevent robots from reaching desired sites. Two separated blocks must not be attached in the same row if all sites between them are meant to be occupied. If this condition is violated, then further addition of blocks will eventually lead to an unfillable gap (Fig. 2B). One strategy is to build up the structure by layers, adding new blocks in rows along the edge of the existing structure (Fig. 2C).

Building the structure in this way—effectively growing layers outwards from the seed, to fill all sites the design specifies should be occupied, while not allowing unwanted gaps to form—will ultimately produce any desired shape.

We now describe four algorithms that follow this approach, using different levels of capabilities in the robots and blocks. In each case we describe the algorithm for a single robot, and then discuss considerations for many operating simultaneously.

A. Identical, inert blocks

In the first algorithm we assume that blocks are indistinguishable, with no capacity for communication. In this case the robots are responsible for knowing the shape map and avoiding unwanted gaps. Alg. 1A sketches the following approach.

Establishing Location. For any fixed perceptual distance, a robot will be able to distinguish only a limited number of distinct local configurations. Solely local observations are then insufficient if the system is to be capable of building arbitrary structures; a robot needs additional state to infer its location. One simple method to do this is to make one edge of the marker distinct, and to position that marker edge along an edge of the shape map. The marker then serves as a landmark, which will be found by any robot following the perimeter of the structure at any stage of construction.[1] Upon reaching the structure, then, a robot follows the perimeter until reaching that landmark. It then knows its position and orientation, which it updates thereafter by keeping track of the number of blocks it passes and turns it makes. In this way the partial structure provides a source of odometry for the robot.

Avoiding Gaps. Alg. 1 ensures that no separated blocks will be attached in the same row if all sites between them should be occupied, as follows. (a) Line 8 allows a block to be attached at a site that has two occupied neighbors, as with site 2 in Fig. 2A. Such attachments correspond to adding new blocks to existing rows, and cannot result in violation of the rule against separated blocks. (b) Line 9 specifies that a new row can be started only if the robot has verified on its tour that no block has been placed earlier in that row, and there are no sites further along in the row where a block might already have been placed.

Unfillable gaps are thus avoided with this algorithm. It will also result in blocks filling the whole area specified by the structure design, as is straightforward to prove by contradiction. Thus any solid structure will reliably be built.

Multiple Robots. Each robot following Alg. 1A acts in such a way that further consistent actions will lead to the successful completion of the structure. It makes no difference whether those further actions are taken by the same robot, or by other robots. Fig. 3 shows a group of ten robots, operating independently in this way with no explicit cooperation through coordination, nevertheless building an arbitrary prespecified shape. The partially built structure provides cues for implicit coordination; robots sense how much has already been built and act appropriately. Extensions may be necessary to prevent robots interfering with one another's movement; but the algorithm works equally for a group as for an individual.

B. Distinct, inert blocks

Alg. 1A requires robots to find the marker before they can attach a block, and to keep track of their movement along the structure, with unfavorable implications for construction speed

[1] It is possible to establish location even if the marker is indistinguishable from all other blocks. Using the grid formed by the blocks as a guide, a robot can circle the structure, keeping track of its movement, and recognize when it has returned to a previously visited site. It then knows the shape of the partial structure it has circled. If all robots use the same convention to register this partial structure with the shape map, they can reliably determine their position in a common coordinate system (even if the partial shape they observe is out of date because other robots have meanwhile been attaching blocks). However, this approach is slower than that where the marker is distinct, and requires more dynamic memory from the robots. Thus we explore it no further here.

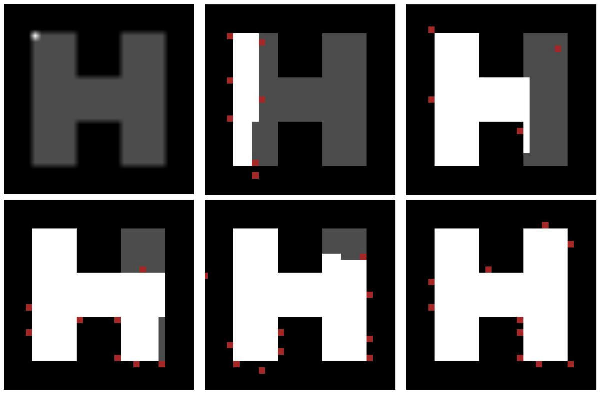

Fig. 3. Simulated construction of a sample rectangular structure of inert blocks, showing successive snapshots during the process of construction by ten robots. White: blocks; brown: robots carrying blocks; gray: empty cells where blocks should be attached; black: empty cells that should be left empty. The marker is in the upper left corner.

and behavioral robustness (§V). An alternative is to make all blocks distinct, so that each becomes a potential landmark. Reliably distinguishing an arbitrary number of blocks could be very difficult for robots if it is to be accomplished, e.g., visually. However, it could be done simply and reliably, e.g., by labeling each block with an RFID tag.[2]

The simplest labels are static, as with read-only RFID tags: every block is distinct, but there is no advance information regarding where blocks might end up attached to the structure. A robot must then maintain a dynamic *label map*, storing the labels and locations of all blocks in the structure. The initial label map has the marker at the origin.

When the robot reaches the structure perimeter, it can establish its position by reference to a block label. Disambiguating orientation may be done in a number of ways: the robot may have its own compass; it can go on to find a second known block (or, initially, a distinct edge of the marker as before); or the four sides of each block can be distinct, with robots storing in their label maps the orientation of each block as well. Finding a legal attachment site, so as to avoid unwanted gaps, can then be done as before.

Construction will proceed faster with this algorithm than with identical blocks. A robot can establish its location more readily, not needing to follow the perimeter all the way to the marker to do so. The marker can be located in the middle of the desired structure, rather than along an edge, allowing construction to proceed on all sides.

Multiple Robots. With many robots adding blocks in parallel, each can encounter blocks at the structure whose labels it has not previously seen. Thus upon reaching the structure, a robot follows the perimeter, keeping track of block labels along the way. It can add those labels to its label map once it encounters a block whose position it knows. As it goes on to search for a legal attachment site, it can continue to update its label map with any other unknown labels encountered along the way. Alg. 1B summarizes this approach.

With the marker located in the middle of the structure rather than along an edge, it is possible for a robot to return to the structure to find no blocks it recognizes along the entire perimeter. In such a case, this robot may not be able to contribute any further to construction. However, it is impossible for all robots to be lost in this way, since some robot(s) must have placed the blocks that form the perimeter; thus completion of the structure is not at risk. This form of robot loss can be avoided by requiring the marker to be along an edge of the desired structure, or by having robots directly communicate map information to one another when they meet.

C. Writable, inert blocks

Robots may be able to change the state of block labels. For instance, some RFID tags are writable, with on the order of 1 kb memory that any transceiver can write to and any other can read from. In this case every block can store its coordinates explicitly, and thus act as an unambiguous landmark. A robot can quickly establish its position upon reaching the structure, disambiguate orientation as with static labels, and proceed directly to find a legal attachment site, writing coordinates to the new block when it attaches it (Alg. 1C).

Robots need not maintain the (potentially extensive) dynamic memory for a label map; the blocks collectively do that, embodying that information where it is needed. The cost of dynamic rather than static labeling is that it represents a more complicated capability for robots and blocks. However, depending on the implementation (again, as with RFID tags), that cost may be small or negligible.

Multiple Robots. As with Alg. 1A, actions taken by one robot will not conflict with those taken by others. Because every block label stores position information, robots need not travel any distance along the perimeter to find the marker (Alg. 1A) or a known block (Alg. 1B) by which to establish their position, so that construction will be still faster. Similarly, there is no risk of robot loss through not recognizing any labels around the perimeter. Further, if blocks are rearranged or replaced during the course of construction (as may occur, e.g., if error correction becomes necessary [16]), robots operating with statically-labeled blocks could encounter conflicts between their label maps and observations, whereas dynamically-labeled blocks will simply be rewritten.

D. Communicating blocks

We can extend the capabilities of blocks to store, process, and communicate information by embedding processors in them. Blocks have a direct physical connection to each other once they are connected to the structure; that link can be the basis for reliable, unambiguous, rapid communication. The structure then becomes a distributed network with nearest-neighbor connectivity. Sensor nodes, such as the Berkeley motes, have demonstrated low-power, low-cost computing devices [17], while modular robot research has

[2]Passive RFID (Radio-Frequency IDentification) tags are circuits that can store information without a power source. When an external transceiver focuses a RF beam at a tag, the current induced in the tag's antenna enables it to transmit a response—e.g., a unique ID code.

Algorithm 2 Pseudocode procedure for assembly of a solid structure of communicating blocks.

A: Blocks

loop
  for all sides S **do**
    if a robot asks to attach a block to S **then**
      if (design specifies a block there) and (no blocks are yet attached in that row) **then**
        get confirmation from other blocks in this row that they will not allow attachment in that one
        allow attachment
      else if (design specifies a block there) and (a block is attached in that row adjacent to that site) **then**
        allow attachment
      else
        forbid attachment

B: Robots

while structure not complete **do**
  get block from cache
  go to structure
  while still holding block **do**
    ask any adjacent structure blocks if block can be attached here
    if all structure blocks answer yes **then**
      attach block here
    else
      follow perimeter counterclockwise

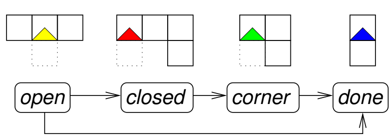

Fig. 4. Examples of block faces in each of the four possible states, with the finite state machine for a single face. Dotted lines show the potential attachment site associated with a face.

demonstrated designs for reliable communication between physically-connected devices [9], [10].

The responsibility for determining whether or not a site is available for block attachment can then be shifted from the robots to the structure itself. Robots become responsible only for transporting blocks to available sites.

Alg. 2 outlines control rules for robots and blocks for building a solid structure. Blocks along the structure perimeter must be able to communicate with robots traversing the perimeter, albeit at short range and with low bandwidth. Robots simply circle until they find a site that gives them permission to attach. Blocks in the structure have the shape map, and give permission for attachment at a site only if the same two criteria are satisfied: (1) the shape map specifies a block there; (2) there are no separated blocks in the same row, unless the shape map specifies a deliberate gap between them.

Block Algorithm. A sufficient decentralized algorithm for blocks is as follows. Each block stores the shape map, its own location in the shared coordinate system, and a state for each of its faces. This latter state is associated with where blocks have already been attached in the row the face borders. There are four possible states (Fig. 4): *open*, if no blocks have yet been attached in the adjoining row, so that attaching at that face would start a new row; *closed*, if attaching at that face would put two separated blocks into the same row; *corner*, if an adjacent block means that the site in question has two neighbors; *done*, if a block has been attached to that face. Faces bordering sites that the shape map specifies should be left empty are always *closed*.

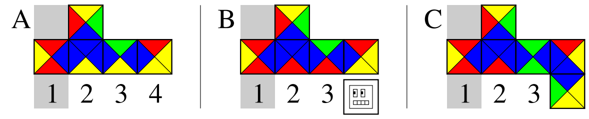

Fig. 5. Example of block algorithm. The shape map specifies that blocks are to be attached everywhere except at the two shaded sites. See text for details.

New blocks can be attached to *open* or *corner* faces. When a robot is given permission to attach a new block to an *open* face, the structure block sends a message along that row in both directions; each recipient sets its corresponding face, formerly *open*, to *closed*, and passes the message on, thus locking out the rest of the row from attachment. Once a block attachment is completed, the original structure block sets its face to *done*, and passes a message to neighbors on both sides to set their face to *corner*.

Fig. 5 illustrates an example, focusing on the south faces of the blocks in the lower row. Initially (A) the leftmost is *closed*, because no block is meant to be attached at site 1; the other three are *open*, since no block has yet been attached in the row of sites 2–4. When a robot is given permission to attach at site 4 (B), the adjacent block sends a message along its row, and the blocks bordering sites 2 and 3 set their south faces to *closed*. After attachment (C), the block to which the new one was attached sets its south face to *done*, and sends a message to its neighbor to set its south face to *corner*.

A newly attached block obtains from its neighbors the shape map and its coordinates, and sets the state of its faces as follows. Define a function $f : f(\textit{done}) = \textit{corner}, f(\textit{open}) = \textit{open}, f(\{\textit{corner, closed}\}) = \textit{closed}$. Any faces x directly attached to the structure are set to *done*. Any faces y adjacent to those faces x are set to $f(y')$, where y' is the face of the block attached to x that is the same face as y (i.e., {north, east, south, west}). The remaining face of the newly attached block, if any, is set to *open*. This strategy allows new blocks to set their states based on messages only from immediately attached blocks.

In Fig. 5C, the new block sets its north face to *done*, because of the attachment there; its west face to *corner* ($= f(\textit{done})$), because there is a block to the north whose west face is *done*; its east face to *open* ($= f(\textit{open})$), because there is a block to the north whose east face is *open*; and its south face to *open*, because no blocks are attached to its east or west faces.

Communication over distances further than a single block is only necessary when the first block is being attached in a new (*open*) row. Thereafter, permission for further attachment

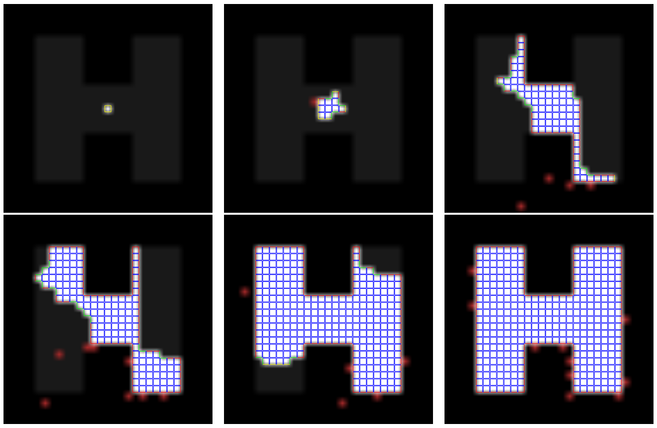

Fig. 6. Simulated construction of a sample rectangular structure of communicating blocks, showing successive snapshots during the process of construction by ten robots. White: blocks; brown: robots carrying blocks; gray: empty cells where blocks should be attached; black: empty cells that should be left empty. Block side states are shown in the same colors as in Figs. 4 and 5. The marker is in the center.

in that row can be given or denied without communication, based on local state only; and communication when further blocks are attached need not involve messages passed beyond immediate neighbors. As Fig. 6 suggests, long *open* rows are rare and communication beyond immediate neighbors is seldom needed in practice. The number of inter-block messages experimentally scales linearly with the number of blocks.

Block power could be supplied centrally from the marker, or self-contained. Power on the order of 20 mW per block would be sufficient, using, e.g., Chipcon CC1000 chips for communication with perimeter-following robots and serial connections for inter-block communication.

Multiple Robots. Finite message propagation speed means that before a robot is given permission to attach a block in an *open* row, the structure must ensure that no other robot is being given permission at the same time elsewhere along the row. Thus when a robot requests permission to attach at an *open* site, communication along the structure row is necessary, to notify the rest of the row of the intent to attach, and to obtain confirmation that permission will not be granted elsewhere. Collisions between messages corresponding to simultaneous attachment requests must be resolved. Only then does the structure block give the robot permission to proceed, and send the message down the row to set faces to *closed*.

V. Discussion and Comparison

The four approaches we describe for construction have many similarities. The robots use identical, simple control rules that are the same irrespective of the shape being built. There are no special leader robots that need to be elected or maintained. The task is executed by many robots running the same control rules asynchronously and in parallel. The robots act independently of one another; a robot does not maintain any state regarding other robots in the system. Robots do not need to maintain any explicit multi-hop communication structure; instead, the partially built structure provides a coordination mechanism. As a result of these properties, the algorithms automatically adapt to unexpected delays and to varying numbers of robots, which can be removed or added during the course of construction. However, they also differ in robustness, performance, and cost.

Robustness. With inert, identical blocks, robots must keep track of their movement after passing the single landmark the marker represents, over potentially long distances for large structures. The grid of blocks they move along is a crucial reference in this task. However, if a robot does miscount its movements somehow, it will be misaligned with the structure coordinate system and may attach a block at a site meant to be left empty. If a robot drifts away from the perimeter, then after regaining it, it will need to travel all the way back around to the marker to ensure knowing its location correctly.

By contrast, with labeled or communicating blocks, location references are available throughout the structure, and can be used to correct such errors. Moreover, with communicating blocks, robots need not be able to count blocks as they pass them nor recognize geometric features of the structure; this greater simplicity means fewer failure points in the robots. Conversely, there are additional failure points in the more complex blocks, though failure may be less likely than in robots because no additional mobility or actuation is involved.

Performance. Construction with Alg. 1B will always build a given structure as fast or faster than with Alg. 1A; Alg. 1C will always be as fast or faster than Alg. 1B, and Alg. 2 faster than Alg. 1C. Robots are eligible to attach blocks sooner upon reaching the structure, and need not travel as far along the perimeter—with identical blocks, they must first reach the marker; with static labels, they must reach a known block; with writable labels, any block will do; with communicating blocks, robots are eligible to attach immediately upon reaching the structure, without needing to survey a full row first to ensure that no distant blocks have been attached. With any but identical, inert blocks, the marker can be located in the middle of the desired structure, so that construction can proceed on all sides. With communicating blocks, new rows may be started anywhere, not just at one end, and can be extended in both directions. These factors will also tend to result in more sites where block attachment is permitted at one time, so that the parallelism of the group of robots can be better exploited.

The extent of these advantages can be explored with simulations. Experiments for each approach compared (1) the total number of steps taken by a group of ten robots along the structure perimeter, and (2) the maximum number of sites where block attachment would be permitted, during construction of square structures of varying side length.

Fig. 7 shows the results, averaged over ten runs for each structure size. Robots using identical, inert blocks travel an order of magnitude further along the structure than those using communicating blocks, with the labeled approaches falling in between. With identical, inert blocks, robots must start from the marker to find an attachment site, forcing construction to proceed in a highly stereotyped way; only a few sites are ever eligible for attachment at once. Labeled blocks allow many more simultaneous attachment sites, with no significant

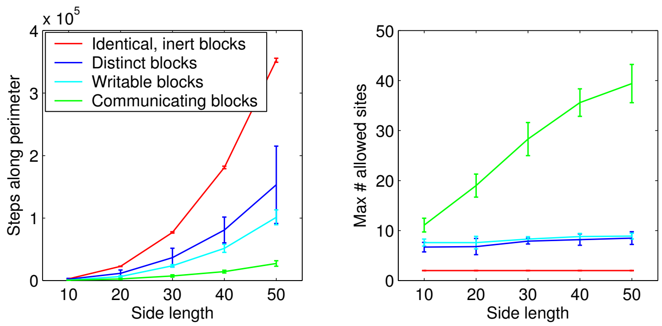

Fig. 7. Left: Number of steps taken along structure perimeter, as a function of structure side length. Right: Maximum number of sites during construction where block attachment would be allowed at any one time.

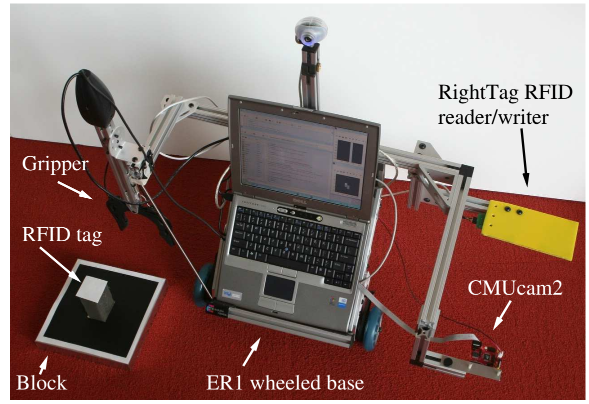

Fig. 8. Hardware (robot and block).

difference between the static and writable variants. With communicating blocks, not only is the maximum number of sites simultaneously available for attachment much greater still, so that robots can find one more readily and more robots can attach at once; but this number also grows with the size of the structure, contributing to the shorter travel distances observed.

Cost. Among inert blocks, labeled ones will be more expensive in terms of materials and fabrication, and robots will require the additional capabilities necessary to interact with the labels. Depending on the implementation, e.g., using RFIDs, these costs need not be great: at present, the cost of RFID tags is on the order of $0.50, and of readers, $100. Blocks capable of communication will in turn be more expensive than inert ones. To what extent will depend in part on the application; if our approach is applied to structures assembled out of high-level prefabricated units, the additional cost of communication capabilities will be more marginal than for structures assembled out of bricks. Overall, work on pervasive computation [17], modular robots [9], and RFID technology for ubiquitous labeling [18] is reducing the costs of components that could be used for systems like those described here. Already significant progress has been made, and costs will decrease further as use of these technologies becomes more widespread.

VI. Implementation

While the approaches we describe are high-level algorithmic ones, we have taken care to ground them on simple basic capabilities that can be made robust and self-correcting, crucial to a hardware realization. We have implemented a prototype system that demonstrates these key capabilities and can build arbitrary solid structures using any of the three algorithms for inert blocks (communicating blocks have not yet been implemented physically).

Fig. 8 shows the hardware: a laptop controller drives an ER1 (Evolution Robotics) wheeled base and gripper, and obtains visual feedback from a CMUcam2 mounted to one side and in front, pointing downward. The camera is configured to register white areas. In our environment, it is sufficient to outline blocks with white borders; in a more complicated environment, a more complex approach or the use of different sensors could improve robustness. A RightTag RFID read/write board and antenna is mounted to the left of the robot, so that it will pass over block centers as the robot follows the perimeter. Blocks are $8.5'' \times 8.5'' \times 1.5''$, made from sheet metal, with foam handles affixed to the top surface for the gripper to grasp, and RFID tags mounted atop the handles. For self-alignment, we use neodymium magnets, mounted in alternating-polarity pairs on each side so that two blocks brought into proximity will be drawn together for any pair of faces.

The gripper has no vertical degree of freedom. It is mounted such that a gripped block is held above the ground with about $2''$ of clearance. When the robot finds a site where a block should be attached, it maneuvers so that the held block is approximately above that site, and drops it; the magnets bring it into alignment. The cache is implemented as a $2''$ pedestal, on which blocks are placed by hand to await pickup (Fig. 9). A line on the floor nearby acts as a visual reference to guide the robot into position to pick up the block.

The robot demonstrates the key elements of our approaches with inert blocks (Fig. 9): the ability to maneuver to a cache, pick up a block, and bring it to the structure; perimeter-following; recognition of grid points and sites where block attachment is valid; reading and writing block labels; and attachment of blocks to form a desired final structure. The additional key elements necessary for communicating blocks have been demonstrated in work on modular robots [9], [10].

To evaluate the reliability of the basic behaviors, we had the robot travel around the perimeter of a 2x2 square structure, measuring its progress by visual reference to the marked block edges, and checking this discrete position estimate for accuracy against the coordinates stored in read/write RFID tags. In addition, each time it passed a block, it wrote two random bytes to the tag and read them back. In approximately 30 minutes, it recognized 122 block boundaries to measure its progress by, turned 41 corners, located 82 block tags, read 328 bytes from and wrote 164 bytes to those tags, without any errors in any of these actions or in its position estimate.

The robot successfully found and retrieved blocks from the cache in 10 of 10 trials. There are three kinds of sites where a block might need to be attached: in the middle of a wall (Fig. 2A, site 1), just before turning a corner to the left (the site to the left of site 1), or at a site with two neighbors (site 2). In

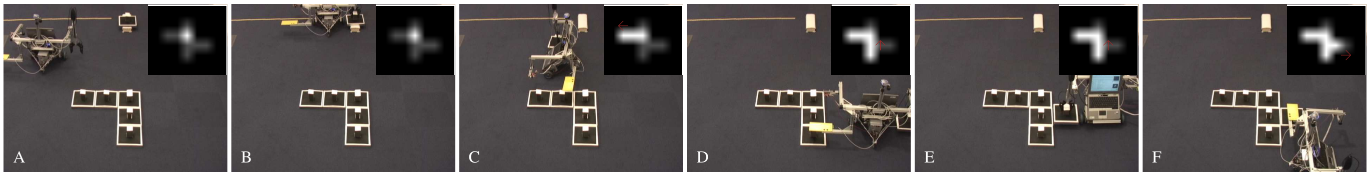

Fig. 9. Process of adding one block to the structure. The cache is at top, structure in progress at bottom, with the marker at its upper right. Inset: the robot's knowledge about the structure's progress and its own position: desired structure in gray, existing blocks in white, robot location (if known) shown as arrow.
A: The robot, traveling toward the cache from the vicinity of the structure, initially knows only that the marker must be present.
B: Using the line on the floor as a reference, it maneuvers to and picks up a block from the cache.
C: Once at the structure, it can use its RFID reader to determine its position, and its camera to follow the perimeter. Existing blocks are added to the robot's map as it observes them.
D: Eventually the robot reaches an empty site where a block is desired, and where one may be attached according to Alg. 1.
E: It maneuvers to attach its block at that site, dropping it into place...
F: ...and writes the block's new coordinates to its tag.

10 trials for each of these three classes, the robot successfully attached blocks, and wrote their new coordinates to their RFID tags, respectively 9, 10, and 9 times.

As expected, the hardest robot tasks were those involving physical manipulation. This difficulty was compounded by the fact that with the current hardware configuration, the camera is the only source of feedback for position evaluation, and is more than 75 cm away from the gripper. When attaching a block, the robot adjusts its position using the camera at the attachment site, and then maneuvers blindly to move the gripper into place, relying on the not infallible precision of the ER1 wheelbase. We believe that adding sensors at the gripper in the next hardware revision, to allow adjustment after the gripper is approximately in position, will greatly improve the reliability of block attachment.

VII. Conclusions

In this paper we have outlined a system for automated construction by a group of robots. We have described simple algorithms by which robots could assemble arbitrary solid structures out of blocks, compared variants using blocks with increasing degrees of sophistication, and presented a hardware prototype that demonstrates our approach. Since particular robots are never assigned particular critical tasks, nor do the algorithms depend on actions being executed in any particular order, the system is highly robust to the temporary or permanent loss of robots, so long as some robots remain. Individual blocks, similarly, are not crucial for task completion. With communicating blocks, it is undesirable for them to fail after their attachment to the structure; but an error correction procedure can recover from that eventuality [16].

While building structures from inert, indistinguishable blocks is possible, incorporating communication abilities into the blocks brings considerable benefits in speed and robustness. It may be, depending on the application and implementation, that the cost or complexity associated with communicating blocks makes such an approach prohibitive. In such a case, the use of passive labels on the blocks can achieve the robustness advantage and a significant degree of the speed improvement, more easily and inexpensively.

Various systems for automated construction are presently in early stages of design, specifying inert (but specialized) building materials [4], [5]. Our results suggest that incorporating a capability for communication into those materials, or at least some heterogeneity using labels, could be of considerable utility. Additionally, for automated construction systems based on either inert or active [6] materials, our approach gives a principled way to partially order assembly to generate desired structures.

Acknowledgments

We thank Q. Zondervan, I. Vasilescu, C. Detweiler, K. Kotay, and M. Vona for useful discussions. This work was supported by NSF grants EIA-0130391, IIS-0426838, IIS-0513755, and ARO award W911NF-0510219.

References

[1] B. Khoshnevis and G. Bekey, "Automated construction using contour crafting—applications on earth and beyond," *Journal of Rapid Prototyping*, vol. 9, pp. 1–8, 2003.
[2] G. Théraulaz and E. Bonabeau, "Coordination in distributed building," *Science*, vol. 269, pp. 686–688, 1995.
[3] C. Jones and M. Matarić, "Automatic synthesis of communication-based coordinated multi-robot systems," in *Proc. IROS 2004*, Sendai, Japan, 2004, pp. 381–387.
[4] J. Everist, *et al.*, "A system for in-space assembly," in *Proc. IROS 2004*, Sendai, Japan, 2004, pp. 2356–2361.
[5] Y. Terada and S. Murata, "Automatic assembly system for a large-scale modular structure: hardware design of module and assembler robot," in *Proc. IROS 2004*, Sendai, Japan, 2004, pp. 2349–2355.
[6] P. White, *et al.*, "Three dimensional stochastic reconfiguration of modular robots," in *Proc. RSS I*, Cambridge, MA, 2005.
[7] J. Wawerla, G. Sukhatme, and M. Matarić, "Collective construction with multiple robots," in *Proc. IROS 2002*, Lausanne, Switzerland, 2002.
[8] C. Parker, H. Zhang, and R. Kube, "Blind bulldozing: multiple robot nest construction," in *Proc. IROS 2003*, Las Vegas, USA, 2003.
[9] D. Rus and G. Chirikjian, Eds., *Autonomous Robots*, vol. 10, Jan. 2001.
[10] M. Yim, Y. Zhang, and D. Duff, "Modular robots," *IEEE Spectrum*, vol. 39, pp. 30–34, 2002.
[11] K. Støy and R. Nagpal, "Self-reconfiguration using directed growth," in *Proc. DARS 2004*, Toulouse, France, 2004.
[12] K. Kotay and D. Rus, "Generic distributed assembly and repair algorithms for self-reconfiguring robots," in *Proc. IROS 2004*, 2004.
[13] C. Jones and M. Matarić, "From local to global behavior in intelligent self-assembly," in *Proc. ICRA 2003*, Taipei, Taiwan, 2003, pp. 721–726.
[14] J. Bishop, *et al.*, "Self-organizing programmable parts," in *Proc. IROS 2005*, Edmonton, Canada, 2005.
[15] R. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.
[16] J. Werfel, Y. Bar-Yam, and R. Nagpal, "Building patterned structures with robot swarms," in *Proc. IJCAI'05*, Edinburgh, Scotland, 2005.
[17] J. Hill, *et al.*, "System architecture directions for networked sensors," in *ASPLOS-IX*, Cambridge, MA, 2000.
[18] "RFID: The UPC of the 21st century," White paper, Manhattan Associates, 2003.