The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Experiments in Decentralized Robot Construction with Tool Delivery and Assembly Robots

Adrienne Bolger, Matt Faulkner, David Stein, Lauren White, Seung-kook Yun and Daniela Rus

Computer Science and Arti cial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
enneirda@mit.edu, mfaulk@mit.edu, stein@mit.edu, llwhite@mit.edu, yunsk@mit.edu, rus@csail.mit.edu

***Abstract*— Our prior work [1] presented a decentralized algorithm for coordinating the construction of truss shaped objects out of multiple components (rods and connectors). In this paper, we consider how to transfer the theory to practice, implementing the algorithm to create a decentralized multi robot construction system. The system is composed of mobile manipulators and smarts parts with an embedded communication device. We discuss the delivery and assembly algorithms that comprise this system and the assumptions behind them. We present data from extensive hardware experiments with 4 robots coordinating an assembly task.**

I. Introduction

Robot assembly control is a fundamental problem for many robotics applications ranging from construction, to manufacturing, and to search and rescue operations. We are interested in robot group control strategies for assembly operations that are (1) fully decentralized and distributed on the group, (2) adaptive to changes in the environment and the group, (3) provably convergent, and (4) experimentally feasible. In our previous work [1], [2] we introduced a controller for coordinated assembly that meets the first three desiderata. In this paper we discuss the algorithmic implications of implementing this controller on a physical platform consisting of arbitrarily large groups of robots specialized as tool delivery and assembly robots, and we present results of experiments with 4 robots. Specifically, we consider a distributed algorithm that causes a network of robots to coordinate the delivery of parts for a desired assembly, and the activity required to create the assembly.

Specifically, we describe a distributed algorithm that takes as input the specifications of an object to be assembled from rods and connectors, causes the robots (1) to identify the subassemblies that can be created in parallel, (2) deliver parts to each subassembly team so that the subassemblies get created in approximately the same amount of time, and (3) place the parts in the required sequence to construct the desired object. Our implemented solutions to these problems rely on using smart parts for the assembly. The smarts come from embedded two-way communication systems that allow the parts to transmit their location (in the form of a beacon) as well as their geometric and mass properties to the robots. The robots use communication-enhanced grippers to locate, identify and grasp the objects. Our solutions to problems (1) and (2) are general with respect to this grasping modality. Our solution to problem (3) applies to planar objects and illustrates the correct position of the parts. The actual assembly to create a rigid object is not yet solved.

The robot system for construction is composed of 4 mobile robots with a 4-dof manipulator and two kinds of components (truss and connector) with embedded IR beacon for communication with the robots. Each robot is also equipped with communication devices for localization, inter-robot communication, and robot-part communication. The theoretical algorithms in [1], [2] guarantee stable and convergent controllers, but moving from theory to hardware implementation requires changing the original assumptions and the algorithmic details that rely on them. We discuss the differences between the theoretical and the practical algorithms and present data from extensive subassembly partitioning and tool delivery experiments. We also discuss data from a preliminary planar implementation of the assembly algorithm that places the parts in the correct sequence.

A. Related work

Our work builds on prior research on robotic construction, which includes several construction robots such as SM^2 which is a truss-walking inspection robot developed for space station trusses [3], and Skyworker for truss-like assembly tasks [4]. Werfel et al. [5] described a 3D construction algorithm for modular blocks in a distributed setting. Stochastic algorithms for robotic construction with dependency of raw materials were analyzed in [6]. Our previous work on robotic construction includes Shady3D [7], [8], [9] utilizing a passive bar and an optimal algorithm for reconfiguration of a given truss structure to a target structure [10].

II. Problem Formulation: Coordinated Robotic Construction

We are given a team of robots, n of which are specialized as assembly robots and the rest are specialized as part delivering robots in Euclidean space $Q \subset \mathbb{R}^N (N = 2, 3)$. The robots can communicate locally with other robots within their communication range. The robots are given a target shape represented as a target density function $\phi_t : Q \rightarrow \mathbb{R}$. ϕ_t represents the goal shape geometry by specifying the intended density of construction material in space. For example, in Figure 1 the yellow region has high density (many materials) while the white region has low density.

This formulation applies to any construction components. To simplify exposition and better illustrated the connection to the implemented system we focus on truss structures built with two types of components: connectors and links in order to simplify exposition and figures. To represent truss structures, ϕ_t is defined point-wise on the grid that corresponds to the truss. The point density is proportional to the number of possible truss connection at the point. We assume that the robots move *freely* in an Euclidean space (2D and 3D) which of course is tackled in our experiments.

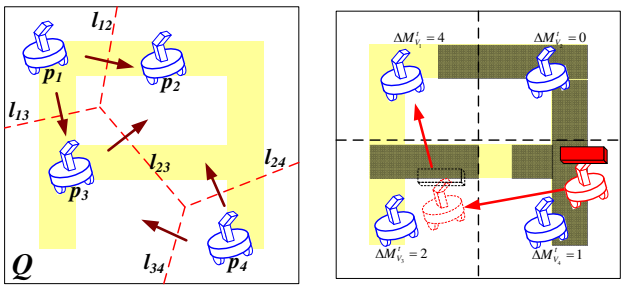
We developed a decentralized algorithm that coordinates the robot team to deliver parts so that the goal assembly can be completed with maximum parallelism [1]. Algorithm 1 and Figure 1 show the main flow of construction in a *centralized* view. In the first phase (Figure 1(a)), assembly robots locate themselves using a distributed coverage controller which assigns to each robot areas of the target structure that have approximately the same assembly complexity. In the second phase (Figure 1(b)) the delivering robots move back and forth to carry source components to the assembly robots. They deliver their components to the assembly robot with maximum *demanding mass* ΔM_V. The demanding mass is defined as the amount of a source component required for an assembly robot to complete its substructure. After an assembly robot obtains a component from a delivering robot, it determines the optimal placement for this component in the overall assembly and moves there to assemble the component. The assembly phase continues until there are no source component left or the assembly structure has been completed. In this paper, we implement the second phase.

Algorithm 1 Construction Algorithm

1: Deploy the assembly robots in Q
2: Place the assembly robots at optimal task locations in Q
3: **repeat**
4: **delivering robots:** carry source components to the assembly robots
5: **assembly robots:** assemble the delivered components
6: **until** task completed *or* out of parts

A. Delivery and Assembly Algorithms

Once the assembly robots are in place, construction may begin. During construction we distribute the source components (truss elements and connectors) to the assembly robots in a balanced way. Global balance, which is defined as balance of delivery to all the assembly robots, is asymptotically achieved by a probabilistic target selection of delivering robots that uses ϕ_t as a probability density function. For local balance defined for only neighboring robots, the delivering robots are driven by the gradient of demanding mass defined as the remaining structure to be assembled by the robot. Robots with more work to do get parts before robots with less work. Each assembly robot waits for a new truss element or connector and assembles it to the most demanding location in *its Voronoi region*. Therefore, construction is purely driven by the density function regardless of the amount of the source components. We ensure all control processes are distributed and robot communication is restricted to direct neighbors. Details of the control algorithms are explained in [1].



(a) (b)

Fig. 1. Example of the equal-mass partitioning and delivery by the gradient of the demanding mass. 4 mobile manipulators (assembly robots) are displayed in a convex region Q that includes the A-shaped target structure. The yellow region has high density ϕ_t. The mass of a robot is the size of the total yellow region in its partition (Voronoi region.) $p_i (i = 1, 2, 3)$ denotes the position of the assembly robots and the red-dotted lines l_{ij} are shared boundaries of the partitions between two robots. $\Delta M^t_{V_i}$ is the demanding mass.

III. Experimental System

In this paper, we focus on delivery and assembly experiments. Experimenting equal-mass partitioning is left for future work. Similar algorithms have been implemented before in our previous work [11].

A. Experimental Testbed

Our hardware system consists of a team of mobile manipulators, smart parts each with an embedded communication device, and a motion capture system. The robots operate on a square area, and a source cache is located at the end of the workspace (The blue half-circle plate in Figure 13). Trusses and connectors are manually supplied to the cache during experiments. In order to help grasping, each 3D-printed smart part contains a custom IR chip and a battery designed to talk to the robots. The robots localize using data from the motion capture system broadcast over a mesh network.

1) Mobile manipulator: The robot consists of a commercially available iCreate mobile platform and a CrustCrawler robotic arm with a custom chassis as shown in Figure 2. Specifications of each component are in Table I. The gripper of the arm has been replaced by an instrumented gripper which contains an infrared communication transceiver and is contoured to align a grasped part as the gripper closes. The special design allows the gripper to reliably grasp parts despite centimeter-scale uncertainty in a position of the parts, by passively aligning the grasp point into a unique orientation as the gripper closes. The robot has three communication protocols: IR, UDP and xBee, which are used for communication with the smart parts, other robots and motion capture system, respectively. We equipped each robot with a small Dell Inspiron Mini 10s netbook which runs a Java-based controller.

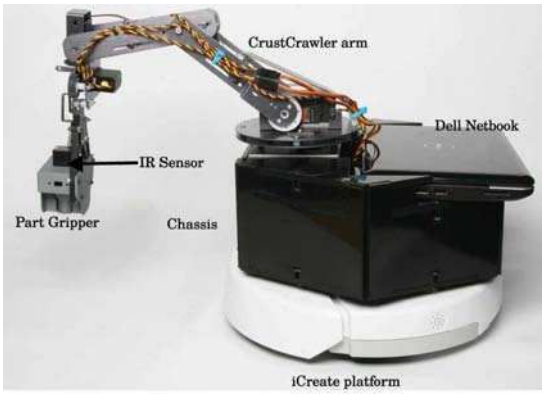


Fig. 2. Side view of robot hardware with the Crustcrawler arm. From a fixed base, the arm allows for grasping an object on the ground in a half-arc in front of it with a depth of about 20cm.

Mobile		iRobot iCreate
Arm	Model	CrustCrawler SG5-UT
	DoF	4
	Reach	0.5 m
	Payload	0.6 kg
Communication		IR, UDP, xBee

TABLE I
SPECIFICATIONS OF THE ROBOT

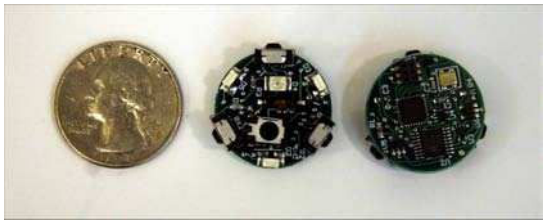


Fig. 4. The small IR communication modules on a PCB that can be embedded in parts to create a smart environment for the robots to sense. Figure reproduced with permission [12]

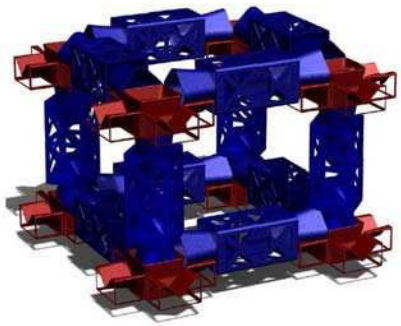


Fig. 5. This 3D-rendered image of a cube is constructed from 8 junctions, and 12 struts. Picture reproduced with permission [12].

2) Smart parts: Instrumented trusses and connectors: Smart parts enable grasping for robotic delivery and assembly via communication. We explore the use of communication as an alternative to using computer vision for part identification and grasping. IR communication devices are instrumented as shown in Figure 4 on the robots and within each parts. A part can guide a robot to its location and tell the robot its part type.

Figure 3 shows two types of the smart parts: truss and connector. The connector is capable of connecting 6 trusses in the North, South, East, West, Up, and Down directions. Figure 5 shows a cube built from 8 connectors and 12 trusses. With a rechargeable 3.7v 210mAh lithium polymer battery, the parts weigh 60 grams. The truss is 18 cm long.

B. Infrastructure for localization and communication

For delivery and assembly, the robots receive precise location information from a Vicon motion capture system

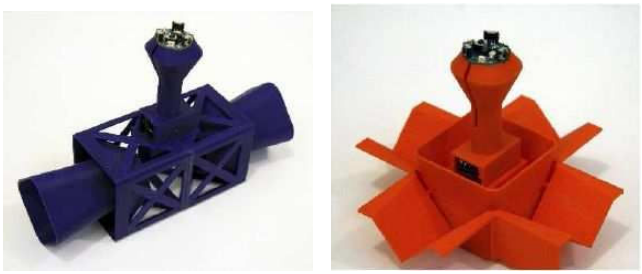


Fig. 3. Smarts parts to be delivered: (LEFT) a red connector (RIGHT) a blue truss

providing the 2D positions and the rotational heading with accuracy to the millimeter and milli-radian respectively at 10 Hz using a commercial xBee RF (radio frequency) wireless mesh network. Between the robots, a UDP multicast channel on the local network is implemented with a singe WLAN router. The UDP packets contain a logical time-stamp, a robot ID number, their current positions, and their current target robot. The robots also broadcast their states such as whether or not they are currently carrying or dropping off a part, which part type they are carrying, where they are carrying this payload, and the knowledge of any other known placed parts.

C. Software architecture

The software architecture is structured hierarchically. The highest level planner can be swapped while using the same underlying modules. We use this modularity to create *assembly* and *delivery* planners, either of which can control the robot functions as shown in Figure 6.

Each software module is implemented in Java and runs in its own Java thread. The planner thread controls manipulation and motion of a robot. The planner gives the robot only an end destination and information on any obstacles, such as moving robots or parts on the ground. The planner waits for the motion to finish before trying to manipulate the arm, and gives the robot arm two commands: *pick up the part* or *put down the part*. The planner makes the decisions on where and when to move and manipulate parts by updating with the information received by the communication module. The communication module contains the most up to date information for the planner, which the planner uses to

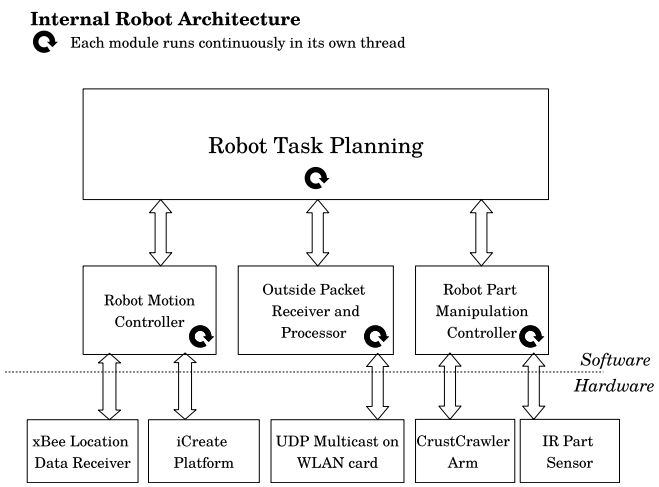


Fig. 6. The hierarchical software architecture of the robot platform.

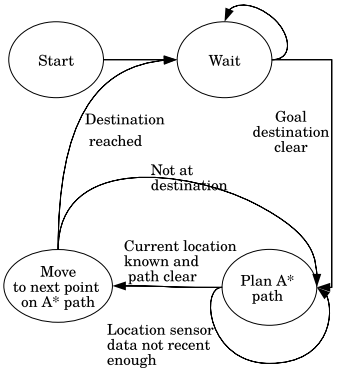


Fig. 7. The motion planning FSM of the robot software.

determine where to move next. The planner is responsible for navigation, manipulation and communication modules commands , and these three modules handle low level control for the mobile, the arm, the manipulating IR sensors, and the communication messaging hardware.

IV. From Theory to Practice

Implementing Algorithm 1 on the robot system requires revisiting its assumptions with respect to what can be measured, implemented, and computed efficiently, and making corresponding changes to control loops. The main differences between the theory and the practice are listed in Table II. The most important components are manipulation and navigation, used both for assembly and delivery.

A. Navigation

The theory assumes point-sized robots that move through each other and already built structures, and we extend the algorithm to deal with physically moving around other robots and parts by passing more information in our communications messages. The robot motion and navigation software module, shown in Figure 7 takes commands from the high level planner and moves the robot as close to a desired position as possible. The 5m×5m sized map is divided into the equally sized grids each of which has 0.1m side length. Location data completely rely on the external motion capture system, and A* navigation algorithm which updates every second has the delivery robot navigate to approach a destination location. Using a simple proportional motion controller appropriate for the iCreate platform. Along the way, it checks for collision avoidance, and will not move to a location blocked by an obstacle or other robots. The algorithm has a failure mode which stops the robots when the location sensor data has not been updated for more than 3 seconds.

B. Manipulation

Once a robot is docked at the supply station, or parked near enough to a part, the task planner uses the arm module to find and pick up the part. Algorithm 2 is implementation of the search-and-pick motions based on the robot-part communication.

Algorithm 2 Arm Manipulation Part Search Algorithm

```
repeat
    Open gripper for wide FOV
    while IR sensor does not see part do
        Arc scan back and forth π radians
    end while
    startTheta = current arm position
    while IR sensor still sees part do
        Radial scan forward.
    end while
    endTheta = current arm position
    Narrow gripper field of view
    while IR sensor does not see part do
        Move arm in and out along radius while arc-scanning
        between startTheta and endTheta radians
    end while
    Open gripper wide.
    Lower arm on top of part
    Close gripper
until Arm closed over part
```

The field of view of the IR sensor attached to the inside of the arms end is widened and narrowed by physically widening and narrowing the gripper on the end of the arm, and the arm finds parts by iteratively scanning smaller and smaller areas for an IR signal. Snapshots of grasping is shown in Figure 10.

C. Communication

The communication module of the robot runs constantly in its own thread to provide the latest whole-system state to the task planner. The module maintains the latest state of every other robot broadcasting in the signal range. It stores the most recent message (determined by packet timestamps implemented using distributed logical time) received from each robot, and broadcasts out its own state on the same

Experiment	Controller from [1]
• Nonholonomic robot dynamics arises position errors and turning delays	• Holonomic robot
• Noisy measurement of global position	• Knowledge of exact global position
• Robots with volume and dynamics, path planning required	• Robots are point masses
• Collision avoidance algorithm required	• Robots pass through the environment
• The next part to be delivered is dependent of the current structure	• No dependency between trusses and connectors
• Pickup causes a bottleneck	• Picking up parts from supply cache takes very short time
• IR beacons for communication between robots and materials	• Pin-point knowledge of types and locations of materials
• UDP messaging system using acknowledgements and logical time to recover packet loss	• Synchronous communication for complete information about surroundings
• Asynchronous propagation of information	• Immediate update of information from neighbors
• Hardware failure causes part to be dropped	• Parts never lost or dropped on map

TABLE II

CONTROLLER FROM [1] VS. EXPERIMENT

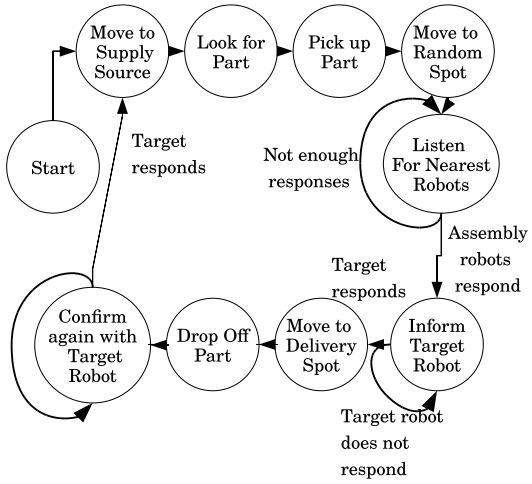


Fig. 8. The task planning event loop for the delivery robots. The main loop pauses and loops back on itself at points where continuing requires asynchronous communication from other robots.

Algorithm 3 Delivery Robot Part Delivery Algorithm

```
repeat
    Move to supply source
    Pick up part
    Move to random location on map
    repeat
        Listen for demanding mass from nearby assembly robots
    until Sufficient network time passes.
    Target assembly robot with highest demanding mass.
    repeat
        Inform target robot of our intent to deliver a part
    until We receive a response from target
    Move to delivery location
    Put down part
    repeat
        Inform target that part has been delivered
    until We receive a response from target
until No more assembly robots asking for parts.
```

channel. The communication module also keeps track of parts that other robots have reported putting down on the field of construction already so the robot knows to avoid them while navigating the environment. Finally, in an effort to provide a handshake mechanism between two robots, the communication module keeps track of parts expected by an assembly robot, whether or not a delivery robot has delivered them yet, and whether or not the target assembly robot has acknowledged the delivery.

D. Delivery

The delivery algorithm, constructed as a finite state machine in Figure 8, follows the theory and takes steps to account for the real world challenges of multiple robot systems such as collision avoidance, asynchronous communication, and part dependencies. The robots have theoretical access to perfect information about the locations and demanding mass value of the surrounding robots, which we replace with a fault tolerant, asynchronous communication protocol to allow robots to learn about the surrounding parts and robots. Finally, the original algorithm assumes that the delivery order of parts will have no affect on the assembly of the structure. The practical delivery algorithm replaces the notion of parts as simple blocks with a model of parts as part of a blueprint, where the order in which parts are delivered can be factored into demanding mass calculated at any given time. These extensions to the algorithm allow it to be carried out on the physical system.

The system follows Algorithm 3 to complete its task, with the sub-modules taking over much of the error-handling. The navigation module, as discussed earlier, handles possible collisions while moving to the source and to robots, and it waits for the source to be clear of other robots before docking. The delivery robot acquires a specialized part from the supply source as noted in Algorithm 2. Asynchronous communication takes the place of actual gradient following when picking a location to deliver a part.

The model of the system as a blueprint of parts, chained together with dependencies, allows the assembly robots to look at the map, determine which parts are still needed at a given time, and request that number of parts to the delivery robots. This implementation does not change the delivery algorithm and helps prevent bottlenecks in spots

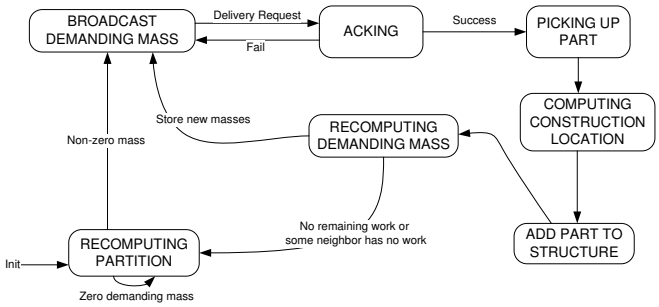


Fig. 9. The task planning event loop for the assembly robots.

where the demanding mass for the completed structure and the demanding mass at that moment are different.

E. Assembly

The assembly algorithm, demonstrated as a finite state machine in Figure 9, adds to the original algorithm similar systems as in the the delivery algorithm, including collision avoidance and awareness of the local structure. We also completely replace the computation of the optimal edge to place next, and change the delivery mechanism from a direct handoff to a passing of parts within the general vicinity of the assembly robot. In the original algorithm we compute the least connected edge in our structure and add a part, and also as the model does not consider collision it assumes there is always space for multiple robots to perform a handoff. In our implementation we take advantage of a blueprint, and only allow the placement of parts that both depend on no other parts to hold them up and that do not prevent a robot from reaching the location of an unplaced part. Among these parts, the optimal part is the one that most increases the number of placeable parts in the partition. We also determine handoff points rather that requiring the delivery robot to directly access the assembly robot inside the structure.

A structure is now represented as a blueprint of inter-dependent parts, where each part maps to a node on both a directed graph representing the physical dependencies of parts (with an edge from any part to any part that directly requires it to be placed) and an undirected graph of the part's proximity to other parts (with an edge between any two parts within a robot's radius of each other). We define a part p as active if it has no parents on the directed graph and that a path exists from every part the robot is responsible for to the edge of the map which does not pass through p. By assuming that the density of parts is bounded, we can provably recompute the set of parts which is active in sublinear time using discrete gradients. As the only parts which can be placed without adding imposable constraints to the task are active ones, we only use active parts when computing demanding mass, meaning the total mass of a partition can both increase or decrease significantly after each placement. We uniquely weight the contribution of a part on the blueprint to the demanding mass by the net change it would have on the size of the set of active parts and break ties by assigning more weight to parts which would remove more constraints from inactive parts, breaking further ties by preferring the centroid of the robot's Voronoi partition. The optimal part placement is determined by the active part with the greatest weight, which means robots place parts in such a way as to allow more parts to be placed, if possible.

V. Experimental Results

For testing platform, we use 2 assembly robots (labeled with robot 4 and 5) and 2 delivery robots (labeled with robot 2 and 3) in a 5x5 meter rectangle. The testing platform also involved a motion capture system to provide robot localization information and a GUI that gathers all the activities with communication and displayed them. Below we discuss the behavior of our robots over the course of these runs in terms of both our algorithm and practical considerations. Note that the system is decentralized except for the locational information from the Vicon motion capture system.

A. Delivery

1) Test Scenario: For evaluation, a single blueprint is chosen demonstrating different features of a real system and the number and locations of the assembly robots change for different runs. We specialize the delivery robots further: one picks up truss parts only and one picks up connector parts only. The supply dock for parts is located at position (0,0), however the parts at the supply dock are moved around to test the robots' ability to pick up reachable parts. The robots could sense the different types of parts 100% of the time by communicating with them over IR.

2) Robot Adaptation: In an ideal setup and execution, the delivery robots alternate between the two assembly robots. To test adaptivity, we also run a variation in which one robot stops demanding parts halfway through the test. This failure of the assembly robots causes the delivery robots to adapt, delivering parts only to the remaining robot.

We ran the scenario with two assembly robots on the platform 12 times. All runs produced the correct alternating delivery behavior. Both the joint delivery robot and the truss delivery robot alternated targets and delivered to both assembly robots, seen in Figure 13. The delivery robots alternate targets in response to the demanding mass reported to them by the assembly robots, shown in Figure 11.

We ran the same scenario as before 3 times with a simulated failure, in which one of the assembly robots was taken off the map. Even when the assembly robot removed had a higher demanding for parts, its failure resulted in the delivery robots delivering to the remaining robot. In all cases, the communication between delivery and assembly robots confirmed the deliveries and changed the demanding masses of the assembly robots. Over all 12 test scenario runs, the 2 delivery robots completed 45/48 delivery attempts. Three failed deliveries were the result of arm hardware failure on a single robot. A summary of test runs can be seen in Table III.

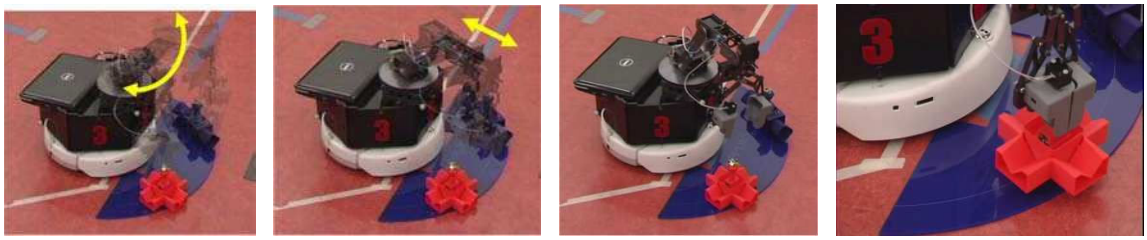


Fig. 10. Snapshots of grasping. The arm moves along an arc to find a rough position of a part and does fine search by radial motion. Grasping is done after confirming the part.

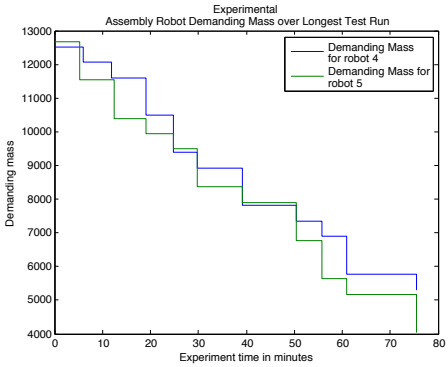


Fig. 11. The demanding mass of assembly robots, named robots 4 and 5, drops whenever a part delivery occurs. Delivery robots changed targets to whichever robot had the highest demanding mass at the time. The unit of demaning mass is undimensional and proportional to amount of the partitions.

Trial	Runtime (MM:SS)	Avg. Runtime	Success	Failure
1	06:05	06:05	1/1	
2	07:36	07:36	1/1	
3	07:20	07:20	1/1	
4	13:58	06:59	2/2	
5	37:33	06:16	6/6	
6	21:40	07:13	3/3	
7	14:18	04:46	3/3	
8	23:04	04:37	5/5	
9	41:28	06:55	6/6	
10	15:49	05:16	1/3	gripper weakened
11	71:05	05:55	11/12	dropped a part
12	23:17	04:39	5/5	
Total	04:43:13	06:54	45/48	

TABLE III
SUMMARY OF ROBOT DELIVERY TEST RUNS

3) Run Time Empirical Analysis: Each delivery robot averaged 7 minutes for a round trip delivery, spending much of its time dealing with the supply dock rather than the other robots in the system. The summary is in Table III. The robots spent a significant amount of time parked in the supply dock, searching for parts: the robotic arm requires an average of 2.75 minutes (32% total time)to search for and pick up the correct type of part. This large amount of time caused a backup in the system: for all test runs in which both delivery robots ran at once, each delivery robot spent an average time of 2.57 minutes *per delivery* waiting for the other delivery robot to move out of the way.

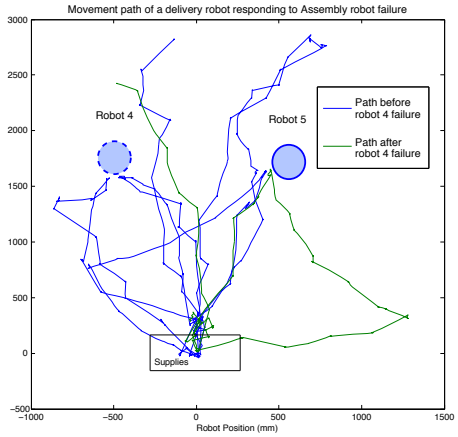


Fig. 12. Adaptive behavior of the system: a delivery robot begins by delivering parts fairly to robot 4 and robot 5. When robot 4 (on the left) fails in the middle of the test, the delivery robot begins delivering only to robot 5.

B. Preliminary assembly

Preliminary tests of the assembly system includes handoffs of a part and putting down the part at the designated location We tested 7 trials of handoffs and 2 trials of put-down with 100% success.

Note that actual assembly is not implemented yet. We have are currently conducting additional tests to ensure that the algorithms in this paper are, in fact, robust.

C. Communication

1) UDP among robots: Over 12 delivery test runs, on each round trip delivery, the delivery robot required 4.8 message packets total to deliver 2 messages to the target assembly robot, meaning each message from a delivery robot had to be resent at least once on average before a response was received from a target assembly robot. The assembly robots spend part of each delivery robot's delivery round asking for parts at a rate of 1Hz, meaning that during the average delivery, each assembly robot sent out an average of 180.5 messages before a delivery robot could pick up a part and respond to a needy assembly robot.

2) IR between a robot and a part: The smart parts used in this experiment broadcast data about what they are and their relative orientation to receivers mounted on the robot

Fig. 13. Snapshots of a test run of the even demanding mass delivery scenario. Assembly robots begin positioned at 2 different points of highest demand for parts. As the red connector parts are delivered, the maximum demanding mass for the entire map changes, causing the delivery robot to change delivery targets, first to robot 5, then to robot 4.

grippers. The parts also allow transmitters to modify a message portion of the data they broadcast. In over 1000 tests, robots were able to autonomously locate and grasp a part and modify its message with the part randomly placed in a semi-circular region with a 33cm with a 99.3% success rate.

VI. Conclusion

This paper describes our experience with transition of a complex decentralized algorithm from theory to practice The coordinated assembly by a multi robot system consists of four mobile manipulators and smart parts with the IR beacons to help communication between a robot and a part. In order to make the system demonstrate the desired algorithmic behavior, we combined the high-level algorithms controlling the actions of the robots with lower level controllers for viable communication channels, stable robot localization and navigation, collision avoidance, and part manipulation. The resulting system demonstrated a use for distributed robotics in industry that involved distributed control, autonomous and mobile robots, and an active ability to change their environment. Our next steps are focused in improving the capability of the assembly system, to demonstrate the use of the system for building of truss-like objects such as boxes and bookshelves.

VII. Acknowledgements

This project has been supported in part by The Boeing Company, the U.S. National Science Foundation, NSF grant numbers IIS-0426838, Emerging Frontiers in Research and Innovation (EFRI) grant #0735953, MURI SMARTS grant #N0014-09-1051, MURI ANTIDOTE grant #138802, and MURI SWARMS grant #544252. Seung-kook Yun is supported in part by Samsung Scholarship. We are grateful for this support.

References

[1] S. kook Yun, M. Schwager, and D. Rus, "Coordinating construction of truss structures using distributed equal-mass partitioning," in *Proc. of the 14th International Symposium on Robotics Research*, Lucern, Switzerland, August 2009.

[2] Seung-kookYun and D. Rus, "Adaptation to robot failures and shape change in decentralized construction," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2010.

[3] M. Nechyba and Y. Xu, "Human-robot cooperation in space: SM^2 for new spacestation structure," *Robotics & Automation Magazine, IEEE*, vol. 2, no. 4, pp. 4–11, 1995.

[4] S. Skaff, P. Staritz, and W. Whittaker, "Skyworker: Robotics for space assembly, inspection and maintenance," *Space Studies Institute Conference*, 2001.

[5] J. Werfel and R. Nagpal, "International journal of robotics research," *Three-dimensional construction with mobile robots and modular blocks*, vol. 3-4, no. 27, pp. 463–479, 2008.

[6] L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system." in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1953–1958.

[7] S. kook Yun and D. Rus, "Optimal distributed planning for self assembly of modular manipulators," in *Proc. of IEEE/RSJ IEEE International Conference on Intelligent Robots and Systems*, Nice, France, Sep 2008, pp. 1346–1352.

[8] S. kook Yun and D. Rus, "Self assembly of modular manipulators with active and passive modules," in *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, May 2008, pp. 1477–1482.

[9] Carrick Detweiler, Marsette Vona, Yeoreum Yoon, Seung-kook Yun, and Daniela Rus, "Self-assembling mobile linkages," *IEEE Robotics and Automation Magazine*, vol. 14(4), pp. 45–55, 2007.

[10] S. kook Yun, D. A. Hjelle, H. Lipson, and D. Rus, "Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements," in *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[11] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus, "From theory to practice: Distributed coverage control experiments with groups of robots," in *Proceedings of International Symposium on Experimental Robotics*, Athens, Greece, July 2008.

[12] M. Faulkner, "Instrumented tools and objects: Design,algorithms, and applications to assembly tasks," Master's Thesis, Massachusetts Institute of Technology, CSAIL Distributed Robotics Laboratory, June-Aug. 2009.