3-Dimensional Tiling for Distributed Assembly by Robot Teams

James Worcester, Rolf Lakaemper, and Mong-ying Ani Hsieh

Abstract. We consider the assembly of a three dimensional (3D) structure by a team of heterogeneous robots capable of online sensing and error correction during the assembly process. The automated assembly problem is posed as a general 3D tiling problem where the assembly components/tiles consist of various shapes and sizes. For a desired 3D structure, we first compute the partition of the assembly strategy into N_c sub-components that can be executed in parallel by a team of N_c assembly robots. To enable online error detection and correction during the assembly process, mobile robots equipped with visual depth sensors are tasked to scan, identify, and track the state of the structure. The objective is to enable online detection of missing assembly components and reassignment of these components to the team of assembly robots. We present the development of the planning, sensing, and control strategies employed and report on the experimental validation of these strategies using our multi-robot testbed.

1 Introduction

Distributed autonomous assembly of general two (2D) and three dimensional (3D) structures is a complex task requiring robots to have the ability to: 1) sense and manipulate assembly components; 2) interact with the desired structure at all stages

James Worcester · Mong-ying Ani Hsieh
Drexel University,
Mechanical Engineering & Mechanics,
Philadelphia, PA, 19104
e-mail: {jbw68,mhsieh1}@drexel.edu

Rolf Lakaemper
Temple University,
Computer & Information Sciences,
Philadelphia, PA 19122
e-mail: lakamper@temple.edu

of the assembly process; 3) satisfy a variety of precedence constraints to ensure assembly correctness; and 4) ensure the stability and structural integrity of the desired structure throughout the assembly process. While the distributed assembly problem represents a class of tightly-coupled tasks that is of much interest in multi-robot systems [1], it is also highly relevant to the development of next generation intelligent, flexible, and adaptive manufacturing and automation.

The execution of tightly-coupled tasks by multi-robot teams has mostly focused on cooperative grasping and manipulation [2, 3]. These works, however, do not address the challenges imposed by the need to satisfy specific precedence constraints during assembly to ensure correctness and stability of the desired structure. Existing approaches to distributed assembly can be broadly classified as micro/nano-scale or self-assembly and macro-scale assembly. In self-assembly, the objective is to devise local rules with global guarantees on assembly of stochastically interacting components [4, 5, 6]. Macro-scale assembly approaches include [7, 8]. In [7], assembly is achieved through a combination of robots with limited sensing and actuation capabilities and assembly components capable of storing and communicating location information with the robots. The focus of this work is on designing a set of consistent local attachment rules that ensure completeness and correctness of the assembly. In [9, 10], a workload partitioning strategy is presented to enable a team of robots to achieve parallel construction at the macro scale. The approach maintains a Voronoi decomposition of the structure based on the assembly robots' locations by minimizing the total difference in the masses of the assembly components in each cell.

In this work, we pose the 3D assembly problem as a three dimensional tiling problem where the team of robots is given a description of the desired structure. The structure is obtained by tiling, or connecting, a collection of assembly components of varying shapes and sizes. The assembly components attached to each based on predefined attachment sites and may differ depending on the geometry and size of the components. Given a desired 3D structure, we build on our previous work [11] to determine an allocation of the assembly task into subcomponents to enable parallel assembly by a team of autonomous robots. The objective is to determine the appropriate partition of the assembly task such that local attachment constraints, specified by the geometry of adjacent assembly tiles/components, and global precedence constraints, specified by structural stability requirements can be satisfied while minimizing workload imbalance among the team. While we have shown that the proposed partitioning strategy ensures the correctness of the distributed assembly strategy, the allocation is performed *a priori* and thus is unable to cope with execution time assembly errors, *e.g.*, incorrect and/or missed placements. To enable online error detection and correction of the assembly process, we consider the addition of a small number of mobile scanning robots capable of providing real-time visual feedback of the state of the structure during the assembly process. The objective is to enable the mobile scanning robots the ability to inform the assembly robots, in real-time, when an assembly tile/component has been incorrectly or not placed. Our main contribution is to experimentally show that a heterogeneous team of assembly and scanning robots can improve the robustness and enable the online adaptation of any given assembly strategy.

The paper is organized as follows: We describe our methodology in Section 2. The experimental setup and results are presented in Section 3. The experimental insights and lessons learned are reported in Section 4. We conclude with directions for future work in Section 5.

2 Methodology

Let S_d denote the desired 3-D structure and M denote the number of distinct assembly components/tiles/blocks where t_i denote a component/tile/block of type i. We will assume that each tile of type i can be described as a general polytope and that the robots know the geometries of the different tile types a priori. Furthermore, every tile of type i will have a fixed number of attachment sites. These attachment sites are locations where tiles can mate and lock onto other tiles.

To assemble the structure S_d, we assume a team consisting of N_a assembly robots, and N_s scanning robots, equipped with visual depth sensors. The scanning robots will be tasked to sense the state of structure during the assembly process. The assembly robots will use the information provided by the scanning robots to ensure correct placement of their respective tiles t_i.

2.1 Task Partitioning

Given S_d and N_a assembly robots, we employ the approach described in [11] to determine an appropriate partitioning of the assembly of S_d into N_a tasks that can be executed in parallel. The objective is to arrive at a partition that maximizes parallel execution of the assembly while minimizing workload imbalance between the robots without violating any of the placement precedence constraints between the assembly components. The approach uses Dijkstra's algorithm with multiple starting nodes to generate a set of assembly tasks for each robot. This results in a partitioning of components of S_d such that each robot's task is composed of tiles that are closest to the starting node. The starting nodes are chosen to be equally spaced along the exterior. This initial allocation strategies is then improved with a second phase of node trading to yield a more balanced workload among the robots. The last step of this approach is the generation of an assembly sequence for each robot that minimizes the time a robot must wait for the placement of supporting tile by another robot. This is achieved by maximizing the time between a placement and the placements of any supporting tiles.

It is important to note that the approach described in [11] is a partitioning strategy that is executed *a priori* and generates a distributed assembly strategy for a team N_c robots given S_d, and $\{t_1,\dots,t_M\}$.

2.2 Visual Feedback

To provide information to the robots about the current state of the physical structure S_p as it is being assembled, we implement a feedback system using visual depth sensors. The objective is to use online sensing to compare S_p with the robot's internal model of the currently assembled structure S_a, and to provide control data to the building process, based on differences between S_p and S_a. To keep an updated representation of the state of S_p, we add a sensing robot to the system, which is equipped with a depth sensor K_p, in our case the Microsoft Kinect sensor. The robot constitutes the system for visual inspection (VI). The VI-robot is independent of construction robots. It runs a prioritized exploration algorithm, which aims to map and update the dynamically changing physical structure with priority on currently targeted building regions. The input to this system is the internal structure S_a (Figure 1(b)), which models what the system is expected to see from the physical structure S_p (Figure 1(a)), and the raw visual sensor data (3D point cloud, Figure 1(c)), the output is a state for every block t_i of the internal structure S_a, denoting if the tile is present, missing, or occluded (currently no visual information about the tile is available).

Before an assembly robot adds a part t_i to the physical structure S_p, it queries the VI-system, if the targeted region data is updated and S_p matches the expected state of S_a. For this comparison, we simulate a robot internal system containing S_a and a virtual Kinect sensor K_v. Using ray-tracing, we simulate a Kinect scan of S_a. The outcome of the simulated ray tracing is compared with the real scan of the physical Kinect K_p to compute the state for each tile $t_i \in S_a$. The following sections will explain the VI system in more detail.

2.2.1 Coordinate System Matching

To compare the outcome of the physical scan and the virtual scan, we must find P_v, the pose of K_v in the virtual system. If we let P_p denote the pose of K_p in the physical system, then P_v has to equal P_p. The positioning is performed in multiple steps consisting of an overhead localization system, a floor based correction, and an Iterative Closest Point (ICP) alignment. In the following we use a right handed coordinate system. The horizontal plane is described by (x,z), height is described by the y-axis.

First, an overhead localization system gives an estimate of the horizontal (x,z) position of K_p. This includes the (x,z) coordinates as well as the yaw α, i.e. the rotation angle around the y-axis. The overhead localization system is provided by a network of cameras with errors in (x,y) below 5 cm and angular errors in α of ¡10 degrees.

To complete P_p, the missing pose-parameters y (the Kinect's height) and β,γ (pitch and roll, i.e. rotation around x and z axis respectively) are determined by a floor-based correction. We perform floor detection in the point cloud C_p resulting from the physical scan. Since the floor in the physical system defines the $x-z$ plane, a transformation T_f, which aligns the floor's normal with the y axis of the virtual system completes the estimate of P_p. We compute T_f by regression of the floor points to their projections in the $x-z$ plane (point to plane correspondence). As such, we note that T_f has no y-rotation component, i.e. the Kinect's yaw, as previously determined by the overhead system, is not altered by an otherwise ambiguous rotation. In addition, we re-compute the translational part of T_f in the $x-z$ plane, such that only the vertical position of K_v is affected.

While there are errors in the localization and ground plan position estimates and noise in C_p, they provide a sufficiently good starting point for an Iterative Closest Point (ICP) alignment [12]. We use P_p as a starting estimate for P_v, therewith we also transform C_p: we set $C_p \leftarrow T_f C_p$. We perform a 6D (3 location parameters, 3 directional parameters) point to plane ICP, with the goal to align C_p to S_a. ICP is a well known technique in robotics and computer vision, successfully applied to align (3D) point clouds, especially for robot mapping [13]. Given two point clouds C_1 and C_2, it finds, in an iterative way, a (locally) optimal transformation T_c that minimizes the squared sum of distances between points in C_1 and their iteratively re-determined closest neighbors in $T_c C_2$. ICP is known robust and fast as long as a good starting estimate of the point-poses is provided. In practice, the previously described steps to compute P_p proved to be sufficient as a starting point.

We perform a fast point-to-plane ICP version: $C_1 \subset C_p$ originates from the physical scan C_p, and consists of a subset of points, being candidates for points belonging to S_p. C_2 is iteratively generated as the projection points of C_1 onto the virtual structure S_a. Internally, S_a is represented as a set of planar patches, describing the geometry of the tiles t_i. Storing the tiles of S_a together with a hierarchy of axis aligned bounding boxes (AABB) allows for fast computation of the projections of C_1 onto S_a. The hierarchy is given naturally: we store an AABB for the structure S_a, for each tile $t_i \in S_a$ and each planar patch $p \in t_i$. Using this hierarchy of bounding boxes, C_1 results in a relatively small subset of C_p. In addition, we omit points that belong to the floor, as determined by the floor detection step. A single Kinect scan in hi-res (640 x 480) contains about 300000 points, the typical point cloud of candidates describing reflections from the structure S_p, after filtering, typically reduces the number of points to less than 10000. We limit our ICP to a maximum of 10 iterations. ICP results in T_{ICP}, an accumulated rotation and translation to align C_1 to S_a. When we apply T_{ICP} to P_p, this reduces pose errors from the former computation. We set $P_v = T_{ICP} P_p$. See Figure 1(d) for the result of this step.

ICP not only provides the pose P_v of K_v in the virtual system, but also the projection points $\bar{C_1}$ of C_1 onto the structure S_a. We therefore compute a connection between the physical point cloud and the virtual structure. In fact, for each tile t_i in S_a, we can determine how many projected points, called physical support points $s_i \subset \bar{C_1}$ of t_i are projected on t_i. The set of support points tells us, if a tile t_i of the virtual structure S_a is seen in the physical world. A tile t_i with a sufficient number of support points is present. However, the converse argument is not valid, since a tile without support could be physically present, but occluded. The next step, ray tracing, solves this problem.

2.2.2 Ray Tracing

This step determines the set of reflection points of a scan of the virtual Kinect K_v with pose P_v of the virtual building S_a. We position the virtual Kinect at pose P_v and simulate a ray-tracing using the Kinect's optical properties (resolution, view angles). Again, since the virtual building S_a is stored using planar polygons and a hierarchy of axis aligned bounding boxes, the ray intersection can be performed very efficiently. For each ray, we compute the closest intersection with a tile t_i from the Kinect, resulting in a virtual point cloud C_v. For each point in C_v, we know the supported tile t_i (i.e. the tile the generating ray intersected with). Ray tracing determines the support sets in the virtual system, that is, the support that we *should* see under the condition $S_a = S_p$. In contrast, $\bar{C_1}$ determines the real support, i.e. the support we *do* see. See Figure 1(e) for the result of the ray tracing step.

2.2.3 Tile Classification

Differences in support from C_v and $\bar{C_1}$ respectively determine if a tile is classified as present, missing, or occluded.

For every tile t_i, denote the number of physical and virtual support points by p_i and v_i respectively. Define r as the minimum ratio between p_i and v_i, $r = \min(\frac{v_i}{p_i}, \frac{p_i}{v_i})$, t_r is a threshold value for this ratio, set to 0.7. For our purpose, it proved to be sufficient to only compare the number of support points of each tile, *i.e.*, we are not explicitly using any geometric differences. Support below a threshold of 100 points is set to 0.

The state of a tile t_i of S_a reflects its presence in the physical structure S_p. We determine this state as follows:

- $v_i = 0 \Rightarrow$ the tile is occluded.
- $v_i \neq 0$ and $p_i = 0 \Rightarrow$ the tile is missing.
- $v_i \neq 0$ and $p_i \neq 0$ and $r \leq t_r \Rightarrow$ the tile is missing. This case implicitly tests geometric differences.
- $v_i \neq 0$ and $p_i \neq 0$ and $r > t_r \Rightarrow$ the tile is present.

If the state of a tile t_i is "missing", the building robots have to adjust. "Present" signals, that $t_i \in S_a$ and $t_i \in S_p$ at the expected position, the building process can continue. If a tile is in state "occluded', the VI-robot has to re-scan the building from a different position before the building process can proceed. See Figure 1(f) for an example.

2.3 Online Error Correction

The VI-robot(s) is responsible for assigning the replacement of any missing tiles it discovers. It does this by managing an auction for each block that should have been placed but is absent. Each assembly robot sends a message to the VI-robot(s) after

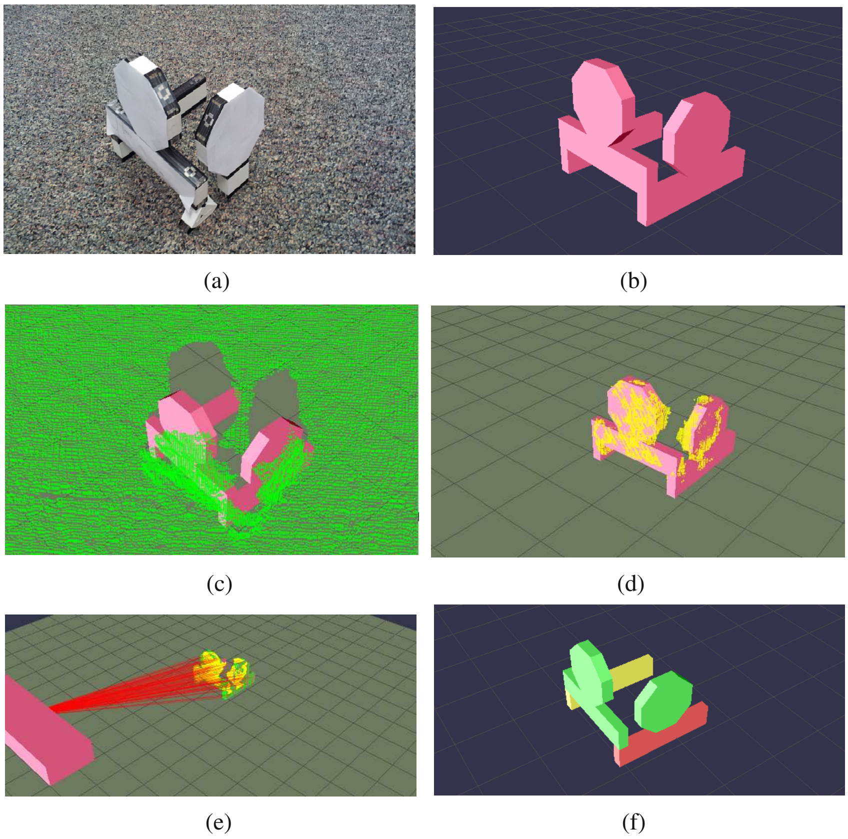

Fig. 1 Vision Feedback System. (a) Physical structure S_p (b) Virtual structure S_a. Note that S_p and S_a differ in this example: the long rectangle (front right) in the virtual structure is not present in S_p, it is replaced by a small cube. In the building process, this is an example of a missing/incorrect tile. (c) Green points show raw input data C_p from the physical Kinect sensor K_p. The Kinect's pose P_v in the virtual system was determined by the overhead positioning system. This figure shows the coordinate matching before floor based correction and ICP (d) After floor-based correction, ICP and candidate filtering: the yellow dots show the pose-corrected raw kinect data C_1, aligned to the virtual building. Points of the original raw data which were unlikely to support the structure were removed (floor- and bounding box based filtering). (e) Ray tracing: the red lines show some rays of the simulated Kinect K_v scan to determine the visibility of tiles $t_i \in S_a$. Yellow dots show the aligned real data C_1, green dots the result of the virtual scan C_v. The difference in support for each tile from yellow and green dots (real/virtual support points) is used to determine the state of each tile. (f) Result: Green tiles: present in S_a and S_v. Yellow tile: occluded (please note that this tile is occluded from view point P_v, as seen in (e). Here we rotated the view to make it visible). Red tile: Missing in S_p. The vision system correctly identified the front right rectangle as missing.

placing a tile. The VI-robot monitors these messages to maintain a state vector q, where q_i is 1 if the block has been placed and 0 otherwise. After each placement, the VI-robot reports a sensing vector q^s_j, where q^s_j is 1 if the block is definitely present, -1 if it is missing, and 0 if the presence or absence of the block cannot be determined. Then, if $q_i * q^s_j = -1$, a block that a robot claims to have placed is determined to be missing. Once the error has been detected, the scanning robot sends a message to inform the assembly robots that the block is missing and asks for bids to determine which robot will replace the missing block. Each robot then constructs a bid based on the following criteria:

$$b_i = w_i - A * c_i + B * d_{ij}, \tag{1}$$

where b_i is the bid of the i^{th} robot, w_i is the remaining workload of the i^{th} robot, c_i is the number of blocks still to be placed that are directly supported by the missing block, and d_{ij} is the distance between the missing block and the i^{th} robot's cache. The constants A and B are weights that can be optimized experimentally.

3 Experimental Validation

3.1 Setup

To evaluate the performance of the proposed online error detection and correction strategy, we implemented the proposed distributed assembly strategy on our multi-robot assembly testbed. The testbed consists of two mini-mobile manipulators (M3 robots), or $N_c = 2$, shown in Figure 2, each equipped with an iRobot Create base, a Crustcrawler 5 DOF arm, 802.11b wireless communication, and a Hokuyo URG laser range finder (LRF). The LRF was used by the assembly robots to detect, pick, and place the tiles during the assembly process. In addition to the two M3 robots, the testbed included one scanning robot equipped with a iRobot Create base and a Microsoft Kinect visual depth sensor. Overhead localization for the robots was provided using two visual cameras.

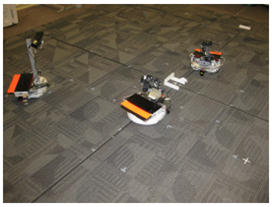

Fig. 2 Team of two assembly robots and one VI-robot with a partially completed structure

Each robot was given the global position of the structure's center and the positions of their respective parts cache. The assembly parts were plastic tiles of various shapes and sizes (side lengths from $4-17$ *cm*), each with a given set of magnetic attachment sites (see Figure 3(a)). Each robot was assigned their respective assembly plans determined by [11]. The assembly plans consisted of a list of tile identifiers in the computed assembly order. Distributed implementation of the plan was achieved by encoding the immediate supports for each component in the plan to ensure robots wait for the placement of a missing support tile by another robot before placing their parts.

The assembly tiles were grouped by type and placed in predefined locations in the workspace. The idea is to have a separate parts cache for each tile type. In our experiments, we considered the distributed assembly of 3D structures composed of 14 tiles with 5 distinct tile types. Figure 3(b) shows the desired structure for the experiment. To simulate missed placements, random assembly tiles were removed at various times during the assembly process.

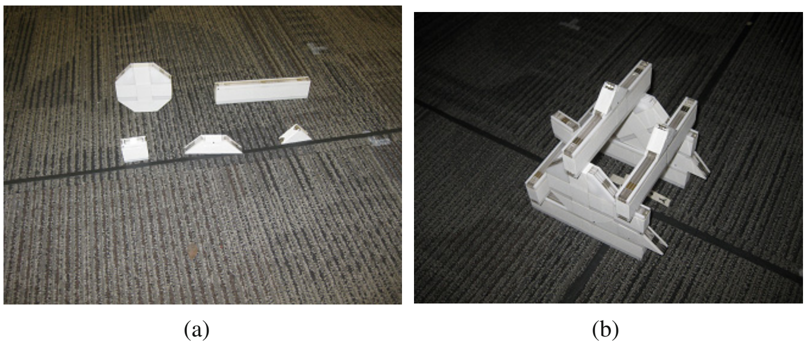

Fig. 3 (a) Sample assembly tiles. (b) Desired structure to be assembled.

3.2 Results

Fourteen experimental trials were run on the scanning robot for the desired structure shown in Figure 3(b). During each trial, one or more random assembly tiles were removed at different parts of the assembly process. Figure 4 shows the results of one of the experimental trials where the missing tile was successfully detected by the scanning robot. Out of twenty-two removed blocks, the scanning robot was able to successfully detect twelve of the missing tiles and reported undetermined for the other ten. There were no false positives during these trials, and only one false negative where a tile was reported as missing when it was actually present. The smaller tiles (square and triangle) were always reported as undetermined, while the larger tiles were always detected as missing after they had been removed in these trials.

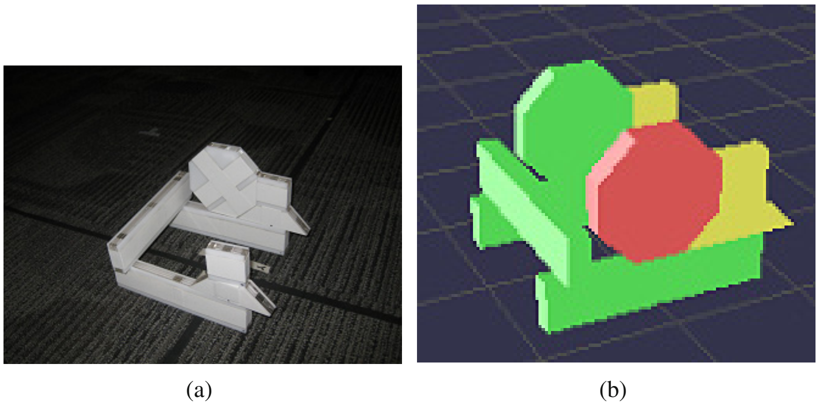

Fig. 4 (a) Tile removed. (b) Missing tile reported by the scanning tile.

Table 1 summarizes the assembly partition obtained at the start of an experimental trial for each robot. The tiles allocated to each robot are shown in the order in which they are supposed to be placed. Table 2 shows the updated assembly allocation as tiles are removed during the experiment, including the workload reallocation after the detection of errors.

Table 1 Initial Allocation for the 3D Structure in Fig. 3(b)

Robot 1	Tile ID	**Robot 2**	Tile ID
Long Rectangle	3	Trapezoid	7
Trapezoid	6	Octagon	5
Octagon	4	Square	8
Square	9	Square	10
Long Rectangle	11	Long Rectangle	12
Triangle	13	Triangle	14

4 Experimental Insights and Lessons Learned

The execution of complex tasks by a team of heterogeneous robots in a complex and dynamic environment with limited resources poses significant challenges. Most existing assembly strategies do not explicitly address the impact of sensing and actuation noise on the performance of a team of autonomous robots tasked to assemble complex three dimensional structures in an actual physical space. In our work, we consider the real-time on-board sensing requirements necessary for online adaptation of any distributed assembly strategy.

Table 2 Allocation After Detection of a Missing Tile

Robot 1	Tile ID	**Robot 2**	Tile ID
Long Rectangle	3	Trapezoid	7
		Removed tile	7
Trapezoid	6		
Octagon	4	Trapezoid	7
Removed tile	4		
Square	9	Octagon	5
		Removed tile	5
Octagon	4	Square	8
		Removed tile	8
		Octagon	5
Long Rectangle	11	Square	10
Triangle	13	Long Rectangle	12
Removed tile	13		
Square	8	Triangle	14
		Triangle	13

In our experimental setup, we considered two types of real-time on-board sensing: 1) the ability to localize the individual assembly tiles for pick-up and placement by the assembly robots, and 2) the ability to determine the state of the assembly structure during the entire assembly process. In both cases, the relative small size of the assembly tiles in relation to the sensing and actuation precision of the actuators and sensors used in the system posed significant engineering challenges. However, the ability to overcome these limitations at the small scale suggests that one can be more confident in the performance of the algorithms when employed on larger full scale systems.

5 Future Work

In this work, we presented a distributed 3D assembly strategy with online visual feedback to enable realtime error detection and correction. Our approach enables the online verification and adaptation of general 3D assembly strategies. An immediate direction for future work is to improve the visual feedback system to provide more detailed assessment of the state of the assembly structure. In particular, the reduction of false negatives by visually inspecting the structure via different viewpoints. A second direction for future work is to extend the visual feedback system to enable identification of incorrect assembly placements as well as missing tiles. Finally, we would like to enable online adaptation of the assembly strategy in the presence of incorrect tile placements. This, in conjunction with the visual feedback system, can significantly increase the robustness and adaptability of the system.

Acknowledgements. James Worcester and M. Ani Hsieh gratefully acknowledge the support of NSF grant CNS-1143941. Dr. Lakaemper is sponsored by the grant ARRA-NIST-10D012 of the National Institute of Standards and Technology (NIST).

References

1. Chaimowicz, L., Sugar, T., Kumar, V., Campos, M.F.M.: An Architecture for Tightly Coupled Multi-Robot Cooperation. In: Proc. IEEE Int. Conf. on Rob. & Autom., Seoul, Korea, pp. 2292–2297 (May 2001)
2. Mataric, M.J., Nilsson, M., Simsarian, K.: Cooperative Multi-Robot Box-Pushing. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1995), Pittsburgh, Pennsylvania, pp. 556–561 (August 1995)
3. Fink, J., Hsieh, M.A., Kumar, V.: Multi-robot manipulation via caging in environments with obstacles. In: Proc. IEEE International Conference on Robotics and Automation (ICRA 2008), Pasadena, CA, pp. 1471–1476 (May 2008)
4. Klavins, E.: Programmable Self-Assembly. Control Systems Magazine 24(4), 43–56 (2007)
5. Evans, W.C., Mermoud, G., Martinoli, A.: Comparing and modeling distributed control strategies for miniature self-assembling robots. In: Proc. of the 2010 Int. Conf. on Robotics and Automation (ICRA 2010), Anchorage, AK, pp. 1438–1445 (May 2010)
6. Matthey, L., Berman, S., Kumar, V.: Stochastic Strategies for a Swarm Robotic Assembly System. In: Proc. 2009 IEEE International Conference on Robotics and Automation (ICRA 2009), Kobe, Japan, pp. 1953–1958 (May 2009)
7. Werfel, J., Nagpal, R.: Three-dimensional construction with mobile robots and modular blocks. International Journal of Robotics Research 27(3-4), 463–479 (2008)
8. Heger, F., Singh, S.: Robust robotic assembly through contingencies, plan repair and re-planning. In: Proceedings of ICRA 2010 (May 2010)
9. Yun, S.K., Schwager, M., Rus, D.: Coordinating construction of truss structures using distributed equal-mass partitioning. In: Proc. of the 14th International Symposium on Robotics Research, Lucerne, Switzerland (August-September 2009)
10. Yun, S.K., Rus, D.: Adaptation to robot failures and shape change in decentralized construction. In: Proc. of the Int. Conf. on Robotics & Automation (ICRA 2010), Anchorage, AK USA, pp. 2451–2458 (May 2010)
11. Worcester, J., Rogoff, J., Hsieh, M.A.: Constrained task partitioning for distributed assembly. In: Proc. 2011 Int. Conf. on Intelligent Robots and Systems, IROS 2011 (September 2011)
12. Besl, P., McKay, N.: A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 239–256 (1992)
13. Nüchter, A., Lingemann, K., Hertzberg, J., Surmann, H.: Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698, pp. 304–319. Springer, Heidelberg (2005)