2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Generating Robust Assembly Plans in Constrained Environments

Frederik W. Heger

Abstract— **In the future, teams of robots will construct outposts on Mars and orbital structures in space. Such tasks will require assembly of a large number of components into structures. Automatic generation of assembly sequences is a difficult and well-studied problem in structured factory environments that are specifically engineered for the assembly task at hand, but it is much less understood in less constrained settings. Instead of representing the problem in the space of the many degrees of freedom of the robots and components involved in the assembly, we approach the problem in the space of valid configurations of the structure to be assembled. We use a graph-based framework to describe valid assembly configurations and feasible assembly steps. In addition to reasoning about kinematic feasibility of assembly steps, we consider the quality of potential configurations with respect to actions for the mobile robots. This method automatically repositions the structure in the workspace so that components to be assembled are most approachable. That is, the sequence of assembly and the position of the structure as it is assembled is chosen so as to maximize the area in which mobile robots can operate to perform their tasks. We present simulation results from a simple five-componetn assembly with and without the constraints of a narrow confined environment. Results show that our method allows over twice as much space available for robots during assembly. In addition, the plans preserve most of their free-space flexibility in tight workspaces where other planning approaches are left with only a few candidate solutions.**

I. Introduction

Robots of the future will go beyond the passive motion capabilities that characterize today's machines – they will directly manipulate their surroundings by clearing obstacles if necessary and performing complex tasks such as assembly or disassembly. Capable and dextrous manipulation is essential for useful robots to accomplish such tasks. Manipulation of the environment and objects within it will enable robotic systems to expand their mission capabilities beyond what is possible today. Such capabilities will be especially useful in the construction of space habitats, planetary outposts and orbital structures (Fig. 1) – places where human workers cannot or do not want to go.

Assembly in our context means arranging components relative to one another so that low-level docking controllers can then guide mobile robots to establish the required connections. This task has the interesting property of being inherently a discrete step-by-step process, but one that is carried out by robots moving through a continuous workspace. For the kinds of assembly scenarios we consider here, speed of operation is not the primary measure of quality of a plan. While efficiency is an important aspect, robust and reliable operation is more important. We argue that currently available approaches can solve specific parts of the whole problem, but they fall short of a comprehensive solution.

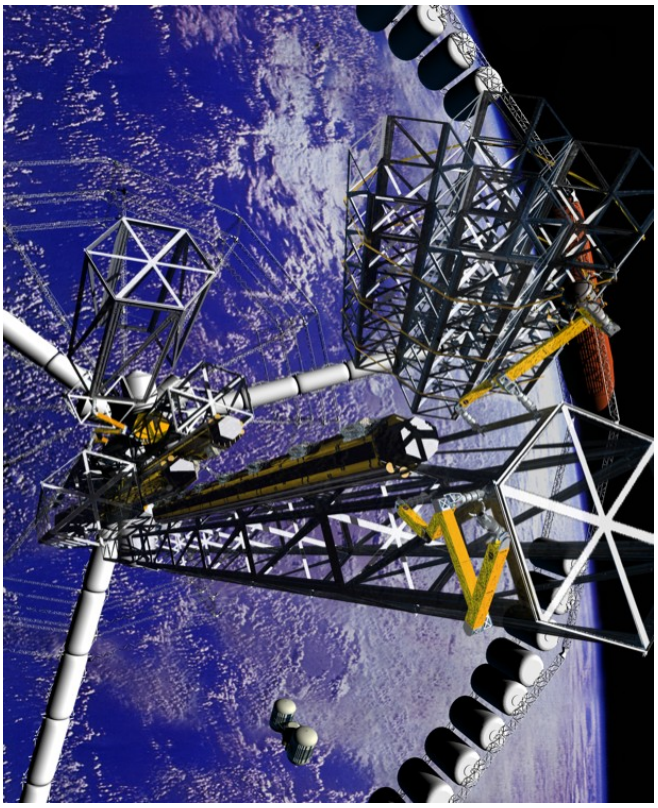

Fig. 1. An artist's rendition of a team of construction robots assembling a truss as the base frame for a large orbital structure. The robots' motions are constrained to walking along the truss. Taking advantage of working as a team, subassemblies can be constructed and then brought to the main structure for final assembly.

Today, large-scale mission planning is often a tedious process of manually scripting long sequences of procedures and contingency plans. An autonomous planner should be smart about potential problems and difficult or critical steps along the assembly sequence and actively plan to avoid predictable, expected or likely trouble where possible. Execution of a plan should not start unless the planner can provide a reasonable guarantee that it is nominally feasible and likely to succeed based on what the system knows from its past performance.

The high dimensionality of the motion planning aspect of assembly makes it infeasible to tackle the problem from that direction, the number of alternatives to consider is simply too large. Notice, however, that the structure to be assembled is fairly constrained, both by the workspace and by ordering constraints among the components. We propose a hybrid approach that leverages constraints in the symbolic assembly

Frederik W. Heger is a Ph.D. student at the Robotics Institute at Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA. fwh@cs.cmu.edu

sequence to focus the (expensive) motion planning efforts to assembly steps that are promising at the abstract level.

Our approach plans from the point of view of the task to be accomplished. Since the task is more constrained than the robots that will perform it, we can more easily avoid dead-ends, and we can reason about it symbolically. However, any plan is only as good as it is feasible to be executed by real robots. Therefore, we reason about moving the (partial) structure to good locations in the workspace and verify potential (symbolic) plan segments using (continuous) motion planning techniques. For an assembly step to be valid, we require a motion plan with sufficient clearance. We also score nominally feasible steps according to metrics such as mean clearance along the path and accessibility of the docking locations. Finally, we evaluate the robustness of a plan by the number of alternative valid steps remaining after taking workspace constraints into account.

In this paper, we present a new planning system for effective autonomous planning of complex assembly tasks for mobile robots. We first review relevant related work in different areas of robotics research and show how available approaches only solve parts of the problem we are considering. Motivated by the cases where existing methods fall short, we describe our approach that uses a graph-based representation and mixes discrete state transitions with continuous constraint representation that can overcome these issues. We show simulation results in which our planner produced significantly more robust plans when considering structure moves than when not. Finally, we conclude and present future directions and extensions of this work.

II. Related Work

Previous work relevant to assembly planning falls into two main categories: approaches that tackle the problem as a sequencing problem on an abstract symbolic level, and ones that consider fine-grained motions of the robots involved. We see both as integral aspects of a larger problem that cannot be solved well with either one method alone.

Symbolic planning methods profit from abstracting the problem into simple operators that require preconditions and produce effects. A sequence of operators that transform a given initial condition into a desired final configuration describes a plan for the scenario [1][2][3]. Such approaches efficiently take advantage of the step-by-step nature of many problems and abstract away difficult to compute constraints into simple heuristics. This abstraction, however, limits the reasoning about the real world to simple yes/no queries (e.g., Is this a valid action? Can the robot get there?). Information about what to do to resolve an impasse in a constructive manner is generally not available (i.e., How can it get there? How much do certain parameters need to change?). As a result, symbolic planners for real-world systems are generally either conservative or wrong.

For systems with multiple robots available to perform a task, scheduling systems consider task and resource allocations of symbolic action sequences to available agents as well as deadlines that need to be met by the system [4][5][6]. As in symbolic planning, the physical reality of the problem can only be approximated as it cannot be easily expressed in a set of heuristics. Infeasibility of a schedule often cannot be detected until the robots, during execution, come to a dead end caused by a workspace constraint unknown to the scheduler (e.g., the robots cannot reach their positions for the next task). Plan verification systems can help reduce such problems by verifying a symbolic plan step by step either after it is completed or while it is being planned [7]. Since in our case validity depends on the solution of a (high-dimensional) motion planning problem, verification is a fairly expensive operation.

Due to its step-by-step nature, traditional assembly planning is a prime setting for symbolic planning approaches. The main concerns of existing systems is on assembly feasibility and serviceability (e.g., access to certain components without having to disassemble the entire assembly) of products or assemblies [8][7]. The robot motions considered in such applications are generally limited to horizontal and vertical part insertions by a robot in a work cell. The focus is on optimal plans to maximize efficiency of the assembly/production process. Once a plan has been found, the assumption is that it can and will be executed thousands of times in exactly the same way.

A common feature of the approaches mentioned thus far is that the robots are restricted to reach into a workspace from the outside. The problem of navigating through a changing and constrained workspace is not considered in these formulations. Homem de Mello developed a representation for describing mechanical assembly sequences based on AND/OR graphs [8][9] similar to our representation. Using this graph structure, he presented a complete and correct algorithm for generating assembly sequences of a desired configuration by planning the disassembly of the goal [10].

Alternatively, the problem can be approached from a motion planning perspective where workspace constraints and physical reality of the environment are inherently taken into account. Koditschek et al. tackle the problem of endogenous (robot and parts live in the same space) assembly using a navigation function approach [11][12]. They describe the problem as a non-cooperative game and use a feedback-based event-driven approach to generate robot motion. This work implicitly assumes that all states between the initial and final configurations are valid, which is not the case in assembly scenarios where only a few states represent stable configurations. There is no guarantee that extrema in the navigation functions coincide with valid assembly states. Lengyel et al. present a grid-based motion planning approach to solve the piano mover's problem using graphics hardware to achieve real-time performance [13]. Klavins describes self-assembly using graph grammars to encode local interactions agents may engage in. The resulting global process results in an organization behavior that brings individual parts into an assembled configuration [14].

Since assembly scenarios have a distinct underlying step-by-step structure, pure motion planning approaches do not produce the results we are looking for. Stilman et al.'s

navigation among movable obstacles [15] plans first in an abstract graph of configuration space segments and then uses motion planning techniques to evaluate paths suggested by graph edges. Yang and Brock's work on decomposition-based motion planning [16] focuses on important points along a trajectory that are most constrained to focus their high-resolution planning efforts and then plan intermediate motions between those key points at lower resolution.

In addition to our own work in multi-robot assembly [17] we are aware of one other group where real robots cooperate to assemble a (simple) structure [18]. Both efforts thus far focus on the execution part of the problem and operate according to a simple script written by hand that is followed by the robots. The planning system we describe here will replace manual scripting of assembly actions and will generate a task tree that can be executed by robots.

III. Motivation

As assembly robots are required to operate beyond the structured environment of work cells and attempt to perform useful tasks in more general settings, purely symbolic approaches and ones based on motion planning alone will quickly reach limits of their performance. Environmental (size, reachability, etc.) constraints become increasingly important as the extends of the workspace shrink relative to the size of the structure to be assembled.

Symbolic planners, unable to consider such constraints, rely on abstractions that in general are either (overly) conservative or even wrong. As a result, they will inevitably produce many plans that cannot be executed. Similarly, as the structures to be assembled become more and more complex, motion planning approaches will require very high-dimensional representations that become intractable. More importantly, motion planners are lacking an understanding of certain discrete states being valid (stable sub-assemblies, etc.) whereas others are not (components brought part way from their storage location to the place where they are to be attached to the structure being assembled). With fewer manipulation agents than parts to be manipulated, components have to be moved in a certain order.

We propose a hybrid planning approach that exploits the symbolic step-by-step characteristics of assembly problems to find promising assembly sequences and then uses motion planning methods to validate those steps in the context of workspace and resource constraints. We use the fact that the structure to be assembled is more constrained than individual components to focus high-resolution motion planning techniques to simpler sub-problems. In addition, we reason about constraints on the structure as a whole at a lower resolution to determine good initial conditions for the individual assembly operations. Combining both planning modalities, we are able to produce plans that are feasible and remain flexible in case execution-time problems (e.g., drift in the state of the structure, occlusions, etc.) require re-planning.

The chosen scenario to show the capabilities of our planner is the assembly of furniture from individual components in a small room (Fig. 2). Given the desired final layout of furniture in the room, a description of the room and knowledge about what pairwise connections between components are required for the different pieces of furniture, we want our planner to produce plans executable on mobile robots that will result in the room being furnished as desired.

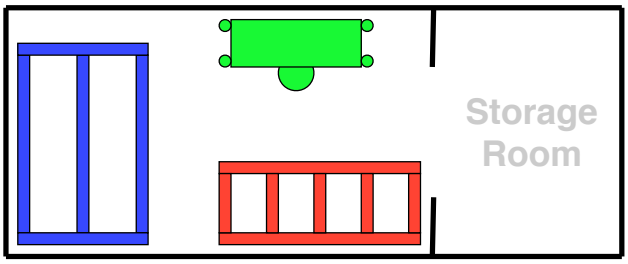

Fig. 2. Plan view of a room to be furnished with a bed on the left, a bookshelf along the bottom and a desk with chair along the top wall. Our approach produces robust plans that allow robots to assemble and arrange the furniture as desired from individual parts located in the storage room.

IV. Approach

We approach the assembly planning problem in a task-centric way. Instead of planning directly for the robots that will carry out specific subtasks, we plan (at first with abstractions) for the order and motions of the components being assembled, which in turn frames the robot motion planning problem. In this hybrid framework, we use both abstract symbolic planning and motion planning modalities in order to produce robust plans that are likely to be successful when executed by teams of mobile robots.

As is common in the assembly planning literature, we plan the disassembly of the goal structure and encode possible sequences in an assembly graph. We annotate that graph with to kinds of workspace information: freedom of motion for components through assembly steps described by edges of the graph, and approachability of components in assembly configurations marked by graph vertices throughout the workspace. Finally, we search the graph for good or best sequences according to the costs associated with assembly steps in response to the constraints on the problem. This search technique is valid for an initial plan as well as for new solutions should an execution-time error occur.

We consider monotone (components are assembled into their final positions relative to the structure, no intermediate placements) and binary (two subassemblies are combined at each assembly step) assemblies. For this paper, we are further limited to linear sequences (only individual components can be added to the structure). Consider the furniture room problem described in the previous section (Fig. 2). At the highest level, this task requires finding a sequence of furniture pieces to assemble (e.g., bed-desk-shelf, or shelf-bed-desk, etc.). A selected furniture sequence then gets decomposed into part motions required to assemble each piece of furniture. If for example there is no way (at the motion planning level) to assemble a required bed part in the (abstract) sequence desk-bed-shelf, the assembly graph is updated and the planner has to backtrack to a higher level to find another sequence.

Finally, the motion of a component further decomposes into several sub-tasks (acquire, carry, dock, etc.) that have to be considered before execution can begin.

A. Representation

We call the graph structure underlying our approach an assembly graph (see Fig. 3). Vertices represent valid assembly states, and edges encode nominally feasible assembly steps. Since we are currently only considering linear assemblies, the assembly graph is an augmented directed acyclic graph (DAG). As assemblies become more complex and we want to take into consideration the potential for parallel operations in separate sub-assemblies, AND/OR graphs [9] will be more suitable to the problem. Note, however, that the DAG representation we currently use is equivalent to the corresponding (simple) AND/OR graph.

The assembly graph respects all constraints imposed by the structure itself (i.e., ordering of parts, etc.), but it knows nothing of the particular workspace or agent resource situation in place. In a sense, the assembly graph assumes a free-space environment and self-mobile "flying" components. The graph is static for a given goal structure and encodes all possible assembly sequences. Any path following the directed edges from the empty assembly state to the goal state represents a valid sequence of actions (note the similarity to the output of a symbolic planner). In a free-space assembly setting, this plan is often sufficient to be instantiated and executed on robots. Thus, in a free-space setting, the additional computation necessary for our approach is unnecessary.

Consider now an assembly scenario of the same structure but in a fairly constrained environment and with real robots. In this case, the validity of edges (i.e., the component motions associated with these edges) are affected by environmental constraints. An assembly step is only valid if the component to be assembled can be brought into position in such a way that there is a reasonable expectation that the robots that will be moving the component have enough room to operate. Evaluating the validity of an edge of the assembly graph involves solving a motion planning problem associated with that edge. As graph edges are invalidated, so are graph vertices (assembly states) that are left without in- or out-edges. In constrained environments it is not unlikely that most if not all of the entire graph gets invalidated, leaving the problem with no solution.

In order to avoid this situation, we introduce a measure of "approachability" for each component that can be removed from a given assembly state. In its simplest form, approachability is a measure of distance from walls and obstacles – the further a component is from other objects in the environment, the more approachable it is. If we allow a partial structure to be moved around the workspace, there are better and worse poses of the sub-assembly for the removal of a particular component. Encoding such structure repositioning moves within the graph vertices (since the configuration does not change) enables us to preserve the validity of many assembly graph edges simply by repositioning the structure between assembly steps.

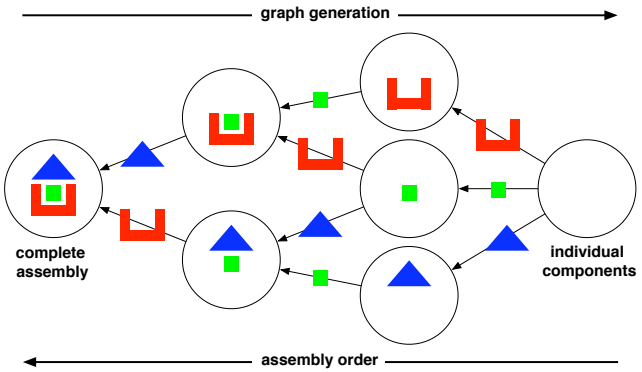

Fig. 3. An example assembly graph for a three-part assembly. The graph is constructed backwards from the complete assembly state by considering all components that can be removed (i.e., all components external to the structure). Once constructed, the graph contains all valid assembly sequences for the desired goal without workspace constraints.

B. Implementation

At this point, primarily for run-time reasons, we are only planning for single pieces of furniture (not the entire room). There is significant potential for efficiency improvement throughout the implementation which is part of ongoing work. Fundamentally, the approach will work also for larger and more complicated problems.

The assembly planner library is implemented in C++ using the boost graph library to represent the assembly graph. We use the MPK Motion Planning Kit from Stanford University [19][20] for the motion planning aspects of the planner. This choice was made primarily for ease of integration, but any motion planner that can plan motions for arbitrarily shaped components through arbitrary workspaces will work here.

During the assembly graph generation process, each state's approachability scores for all its components have to be calculated. This calculation is carried out at poses spaced every 20 cm in x and y and every 15° in θ. It is currently implemented as a flood-fill of the workspace from walls and obstacles inward toward the structure. As the flood hits the components, we record the flood value and then compute statistics on all hits for each component to estimate that component's approachability. Once a component is at a certain (large compared to a characteristic size of the robots available) distance from all walls and obstacles, its approachability is at its maximum value. We achieve this behavior by letting the flood plateau at a certain value. Each assembly state can report the structure pose for best approachability for a given component. Among all poses with equal approachability, we chose that with the shortest Euclidian distance from the goal pose.

In a second pass over the assembly graph, a motion planning problem is instantiated using MPK for each edge in the graph. If the planner finds a valid motion for the components that are involved in the particular assembly state transition from its best pose back to that its storage location, the plan is stored with the edge, and a cost value (length of motion plan, clearance along the path, etc.) is computed. Otherwise the edge is removed form the assembly graph.

With the assembly graph completely constructed and anno-

tated, we have to select a sequence of edges (i.e., an assembly sequence). For this paper we are considering two alternative selection criteria to select the next best edge. In both cases we start from the completely disassembled state and use a greedy method to select the immediate best next option to take. The first consideration is path length. This metric is a combination of the path length of the component moving along the edge of the assembly graph and the length of the structure motion path required, weighted by the number of components in the structure. The second choice is to select that component that is most approachable as the next step.

For visualization purposes, we have developed a GUI using the Qt graphics toolkit. Through it, the user can set planner parameters, trigger planning, and eventually view an animation of the selected assembly sequence.

V. Results

We conducted a series of experiments in simulation to evaluate the benefits of considering moving the structure being assembled during planning on the resulting plan's quality and robustness. Plan quality is high if there is sufficient room for components (and robots) to move and if component connections are easily accessible. A measure of robustness is the assembly graph's size after annotation. In three environments, we used our planner to find assembly sequences for an idealized planar "bookshelf": Case 1 (Fig. 4, top) in relative free-space, Case 2 (Fig. 4, center) with a constrained goal configuration, and Case 3 (Fig. 4, bottom), a narrow version of Case 1.

For each case, we planned assemblies with and without repositioning the structure. We evaluated two criteria for goodness of assembly steps when selecting the best assembly sequence: the length of (component and structure) motion along a particular assembly graph edge and the approachability of the component being added to the structure corresponding to a graph edge.

Table I shows a summary of our results. Since the numbers for both edge selection criteria (approachability and shortest motion) were very similar, we only present results for the most approachable component criterion. All columns in the table are indicative of plan quality and robustness: mean clearance is a measure of space available to the moving component as it is brought from its storage location to the structure, goodness is a score of reachability of the actual docking locations (see Fig. 5), and graph remaining shows the valid assembly states and transitions after planning (compare those to 26 states and 53 transitions in the original graph) as a measure of how many options remain to accommodate execution-time problems.

The first observation is to note that in Case 1 the algorithm has the same performance whether or not the structure is allowed to be moved. This is not surprising since there is plenty of free space around the goal configuration, and even with motion, only minimal structure repositioning was observed. Comparing goodness scores across the three cases, notice how Case 2 with motion is able to preserve most of the goodness of the freespace assembly, whereas without motion

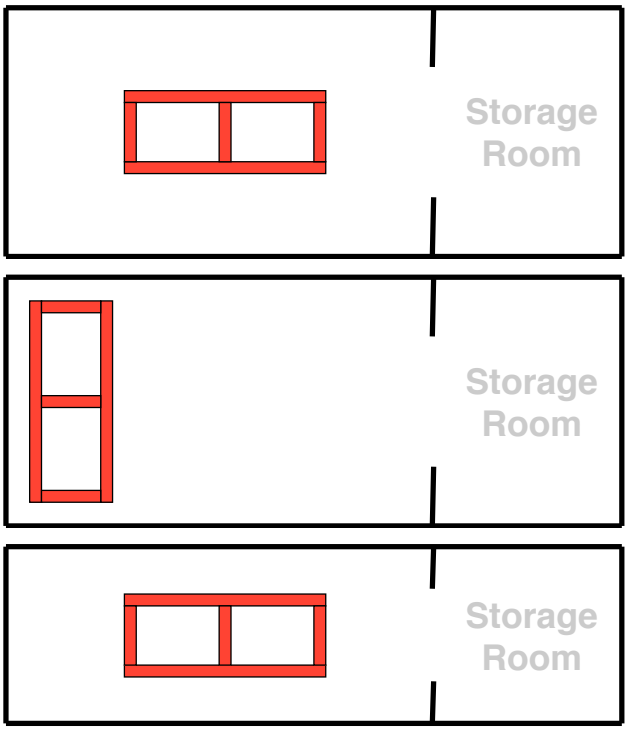

Fig. 4. Three goal configurations. **Case 1 (top):** relative freespace, reasoning about approachability not necessary. **Case 2 (center):** constrained goal configuration, reasoning about motions (especially structure repositioning) during planning increases the robustness of the plan by doubling the accessibility of part connection locations. **Case 3 (bottom):** tightly constrained workspace, planning with structure repositioning increases the plan robustness by doubling the clearance around moving parts as they are brought to the structure.

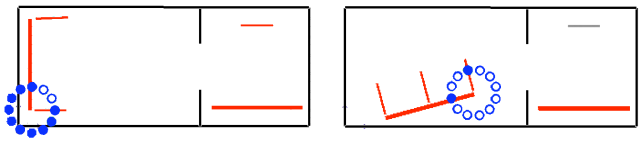

Fig. 5. Evaluation of the reachability of a docking location without (left) and with (right) repositioning the partial structure. Both frames show the evaluation of an end piece assembly (although at different points along the entire assembly sequence). Each circle marks a potential robot position (filled=blocked, empty=accessible). With motion, the accessibility is significantly higher than without.

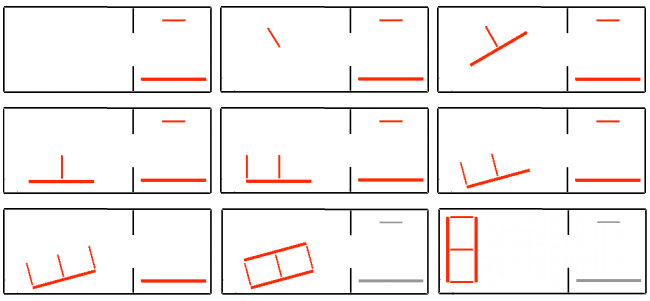

Fig. 6. A representative assembly sequence for case 2. The frames are ordered left-to-right, top-to-bottom. All component assembly and structure repositioning steps are shown. Note how the structure is assembled mostly horizontally and then rotated into position as the last step.

the reachability of docking locations drops to approximately half. In fact, during the simulation, the planner moved the structure into the center of the workspace, assembled the structure there, and finally rotated it into place (see Fig. 6).

	case	mean clearance	goodness	graph remaining
no move	1	9.49	0.71	25 / 48
no move	2	13.24	0.31	13 / 20
no move	3	3.88	0.44	18 / 30
move	1	11.30	0.70	26 / 53
move	2	14.75	0.68	22 / 41
move	3	7.48	0.44	22 / 40

TABLE I
EXPERIMENTAL RESULTS USING APPROACHABILITY CRITERION. MEAN CLEARANCE IS A MEASURE OF AREA AVAILABLE FOR MOTION, GOODNESS DESCRIBES ACCESSIBILITY OF DOCKING LOCATIONS

While Case 3 considers a very tight workspace with and without structure motion, including the repositioning in the plan resulted in twice the clearance around the moving component compared to the static case. Finally, not allowing the structure to be repositioned results in as many as half the potential assembly states and 62% of possible assembly steps being invalidated.

VI. DISCUSSION

In the still fairly simple workspaces used in this experiment, the two criteria of most approachable component and shortest motion to select the next component for assembly gave nearly identical results. In more cluttered workspaces and with more elaborate structures, we expect to see pronounced differences. In addition, we will include further criteria (such as clearance along the path, goodness at the goal location, etc.) and optimize a combined cost function.

In a freespace environment, there is no win over the simple assembly graph method. However, with constrained goal configurations and in overall tight workspaces, our method improves the quality and robustness of the resulting plan by producing plans of higher goodness to perform docking tasks, higher clearance along the path of motion for added components, and by maintaining the validity of a larger part of the original assembly graph. These plan characteristics provide a good starting point for more detailed planning toward the generation of full task trees that can be executed by mobile assembly robots.

VII. CONCLUSION

In this paper, we present assembly planning as an important problem to solve in order to increase robots' usefulness into the domain of active manipulation tasks. We show that the state of the art in autonomous robot planning can successfully solve important sub-problems of assembly, but that there is currently no system available that is able to plan entire scenarios for mobile robots performing construction tasks in constrained spaces.

We present a planner that plans in the space of the task itself while considering enough real-world constraints to make the resulting plans useful and realistically executable. We use assembly graphs to exploit the symbolic nature of assembly problems and augment them using motion planning approaches where necessary.

Our planner can generate more robust and reliable plans in a simulation environment than available approaches. There is still room for improvements to the runtime of the planning process, but fundamentally all necessary tools are in place to plan entire room furnishing missions. We have plans to integrate this planner into our robot testbed [17] and use it in combination with a powerful scheduling system [6] to be able to demonstrate its capabilities in a real-world robotic assembly scenario.

REFERENCES

[1] S. E. Fahlman, "A Planning System for Robot Construction Tasks," MIT AI Laboratory, Cambridge, MA, Tech. Rep., 1973.

[2] J. Hoffmann and B. Nebel, "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[3] H. L. S. Younes and R. G. Simmons, "VHPOP: Versatile Heuristic Partial Order Planner," *Journal of Artificial Intelligence Research*, vol. 20, pp. 405–430, 2003.

[4] X. Xia and G. A. Bekey, "SROMA: An Adaptive Scheduler for Robotic Assembly Systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1988.

[5] B. Fox and K. Kempf, "Opportunistic Scheduling for Robotics Assembly," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, March 1985.

[6] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran, "ASPEN - Automated Planning and Scheduling for Space Mission Operations," in *Proceedings of the International Symposium on Space Missions Operations and Ground Data Systems (SpaceOps)*, Toulouse, France, 2000.

[7] S. G. Kaufman, R. H. Wilson, R. E. Jones, T. L. Calton, and A. L. Ames, "The Archimedes 2 Mechanical Assembly Planning System," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, 1996.

[8] L. S. Homem de Mello and A. C. Sanderson, "Planning Repair Sequences Using the AND/OR Graph Representation of Assembly Plans," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 1988, pp. 1861–1862.

[9] L. S. Homem de Mello, "Task Sequence Planning for Robotic Assembly," Ph.D. dissertation, Carnegie Mellon University, May 1989.

[10] L. S. Homem de Mello and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 228–240, April 1991.

[11] H. I. Bozma and D. E. Koditschek, "Assembly as a Noncooperative Game of its Pieces: Analysis of 1D Sphere Assemblies," *Robotica*, vol. 19, pp. 93–108, 2001.

[12] C. S. Karagöz, H. I. Bozma, and D. E. Koditschek, "Feedback-Based Event-Driven Parts Moving," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1012–1018, December 2004.

[13] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, "Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware," *Computer Graphics*, vol. 24, no. 4, pp. 327–335, 1990.

[14] E. Klavins, "Self-Assembly From the Point of View of its Pieces," in *Proceedings of the American Control Conference (ACC)*, June 2006.

[15] M. Stilman and J. J. Kuffner, "Navigation Among Movable Obstacles: Real-Time Reasoning in Complex Environments," in *Proceedings of the International Conference on Humanoid Robotics*, 2004.

[16] Y. Yang and O. Brock, "Efficient Motion Planning Based on Disassembly," in *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, MA, June 2005.

[17] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated Multiagent Teams and Sliding Autonomy for Large-Scale Assembly," *Proceedings of the IEEE*, vol. 94, no. 7, July 2006.

[18] A. Stroupe, T. Huntsberger, A. Okon, and H. Aghazarian, "Precision Manipulation with Cooperative Robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata*, L. Parker, F. Schneider, and A. Schultz, Eds. Springer, 2005.

[19] G. Sanchez and J.-C. Latombe, "A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, Lorne, Victoria, Australia, November 2001.

[20] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive Dynamic Collision Checking for Single and Multiple Articulated Robots in Complex Environments," *IEEE Transactions on Robotics and Automation*.