Multi-Step Planning for Robotic Manipulation

Max Pflueger and Gaurav S. Sukhatme

***Abstract*— Most current systems capable of robotic object manipulation involve ad hoc assumptions about the order of operations necessary to achieve a task, and usually have no mechanism to predict how earlier decisions will affect the quality of the solution later. Solving this problem is sometimes referred to as combined task and motion planning. We propose that multi-step planning, a technique previously applied in some other domains, is an effective way to address the question of combined task and motion planning. We demonstrate the technique on a complex motion planning problem involving a two-armed robot (PR2) and an articulated object (folding chair) where our planner naturally discovers extra steps that are necessary to satisfy kinematic constraints of the problem. We also propose some further extensions to our algorithm that we believe will make it an extremely powerful technique in this domain.**

I. Introduction

Robots with the ability to modify their environment in unstructured settings form one of the most exciting new frontiers in robotics. For the past two decades, an explosion of low cost sensing has led to advances in robotics in mapping, navigation, and perception. This, in turn, raises the possibility of practical problems of autonomous robots physically interacting with their environment. This is the problem of robotic manipulation in unstructured settings.

When thinking about practical robotic manipulation we are almost always dealing with 6 or 7 degree-of-freedom robot arms, mounted on a mobile base. These arms have complex kinematics that make analytic solutions to collision checking problems near impossible. Consequently, sampling-based planning algorithms are used to develop collision free trajectories. These planning algorithms have advanced to a point where 'off the shelf' algorithms exist that can produce practical, collision free, motion plans for many arms in diverse circumstances [2]. However, they leave something to be desired when we think about the objectives we actually care about in robotic manipulation problems.

Traditional robotic motion planning problems focus on the state of the robot. A more useful (though more difficult) problem formulation would specify goal conditions in terms of the state of the environment. To wit, in many useful problems, the state of the robot is not intrinsically important. If we wish to move a bowl on a table by using a robot, we care that the bowl gets to where we want it. Usually (as long as the robot doesn't get in our way or run into anything) we don't care where the robot goes to do achieve the goal. Combined task and motion planning refers to planning through a combination of the task space (the state of the environment) and the motion of the robot.

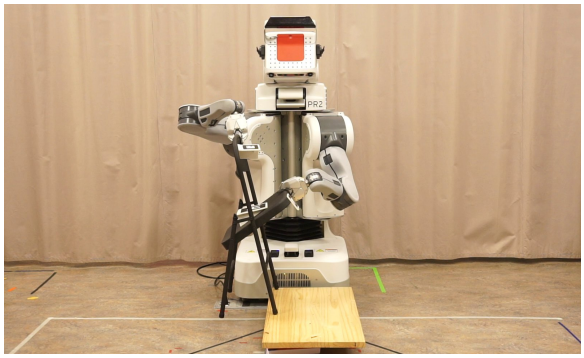

Fig. 1. PR2 robot in the process of folding a chair.

Although it may seem like a simple matter of adding more degrees of freedom, combined task and motion planning problems are significantly different from the problems usually encountered in robotic motion planning. Holonomic robots are those that are capable of instantaneously moving in any free direction (such as a ground robot with omni-wheels, or a robot arm in joint space). Planning is easier for holonomic robots because it is easy to develop a local algorithm to move the robot from one waypoint to another. Planning for non-holonomic robots is more difficult because of restrictions in how waypoints can be connected (requiring a more sophisticated local planner).

This problem is particularly acute for the types of spaces encountered in combined task and motion planning. In most traditional non-holonomic spaces it is possible to get to any nearby position in a small amount of time. For example a differential drive robot can use a forward and back motion to slide to the side a small distance in a relatively small amount of time.

In contrast, combined task and motion planning problems do not have this property, as the state of the objects in the environment cannot be changed until the robot is in a proper position to interact with them. Indeed, if we wish to move a bowl a very small distance on a table, but our robot is on the other side of the room, the start and goal states may be very close to each other, but the plan to go from start to goal may be very long.

Some previous approaches to combined task and motion planning have relied either on a hand engineered plan outline [3], or have incorporated task planners at the top level (with varying degrees of specialization) [4]. We discuss these

The authors are with the Computer Science Department in the Viterbi School of Engineering at the University of Southern California in Los Angeles. {pflueger, gaurav}@usc.edu. An earlier version of this work was presented at a RSS 2013 workshop [1]. This work was supported in part by NSF grant IIS-1017134 and by DARPA grant W91CRBG-10-C-0136. MP acknowledges partial scholarship support from the ARCS Foundation. www.arcsfoundation.org

approaches in some more detail in our Related Work section. Our design relies on a simple discrete state planner at the top level.

The multi-step planning process involves combining a discrete planner with a continuous planner, where the continuous planner provides information on the feasibility of discrete steps. The discrete steps correspond to qualitative changes in the nature of the environment, in our case that means points where the robot either grasps or releases an object. Depending on what the robot is holding on to, a continuous planner will be selected that is capable of planning in that situation.

The main contribution of this paper is to introduce multi-step planning as a framework for solving combined task and motion planning problems, and demonstrate its application on a non-trivial planning task. We have implemented a demonstration of this approach for the problem of manipulating an articulated object, in this case a folding chair. This problem is in some ways easier that other multi-step problems (only one degree of freedom outside the robot), but also demonstrates the power of this approach by showing how it can integrate a specialized planning algorithm in some steps of the plan where it is necessary (two arm manipulation of the articulated object), while also using more simple planning algorithms when the robot is differently constrained (standard sampling based planners for free arms).

One exciting aspect of our approach is that is relies on a well studied and well understood class of discrete search algorithms. Currently we rely on a simple breadth first search, which allows us to achieve good results on our problem. Development of a good admissable heuristic could significantly improve performance through heuristic search algorithms such as A*, and could open possibilities to much more complex planning problems. This is an ongoing area of research for us.

II. Related Work

A. Continuous Space Planners

Motion planning problems for modern, high degree-of-freedom robots, such as arms and mobile manipulators, are most frequently solved using one of a group of sampling based motion planning algorithms. This includes techniques such as probabilistic road maps [5] and RRTs [6].

RRTs in particular are well suited to the well behaved non-holonomic problems often encountered in robot navigation (discussed earlier), and many variants have been proposed, including RRT* [7], RRT-connect [8] and others. However, RRTs rely on a notion of locality that works well in these well behaved spaces (specifically, a meaningful distance metric for finding nearest nodes), but tends to break down in manipulation problems such as the type we address with multi-step planning and other combined task and motion planners.

B. Object Perception and Manipulation

Outside of simulated or highly structured environments, any manipulation planning system relies on some form of sensing and perception to detect the presence and position of the objects with which to interact. Our system relies on fiducial markers and knowledge of the structure of the object. At a practical level it can be difficult to supply structured information about articulated objects when a CAD model is not handy. Sturm et. al. studied ways to model and learn the configuration of articulated objects [9]. Katz et. al. have done similar work using a robot to interact with the scene and detect articulated structure [10].

C. Combined Task and Motion Planning

Multi-step planning has been used before in some other specialized applications, Bretl et. al. [11] used it to plan a path for a wall climbing robot. In their experiment a robot climbs a wall using rock climbing holds similar to what would be seen in a rock climbing gym. In this context the discrete states represented the set of holds the robot is currently using to attach to the wall, and state transitions have to account for whether the robot can reach a new hold without falling off the wall. Hauser et. al. [12] demonstrated an application of machine learning to improve the efficiency of the multi-step planner for the same robot.

Some recent work has started to look at ways of solving the combined task and motion planning problem using other techniques. Nedunuri et. al. [3] developed an approach that relied on plan outlines supplied by an expert. In work that is probably most directly comparable to our own, Srivastava et. al. [4] developed a planner based on a new interface layer between known motion planning algorithms and known PDDL based task planning algorithms. In contrast, our work does not use specialized task planners, but instead divides the world state into qualitative classes based on the nature of the robot's interaction with the environment, and asks the programmer to provide motion planners that are valid in each class of world states. In a practical sense, this requirement would exist with any attempt at combined task and motion planning; our formulation allows us to provide planners for specialized situations such as two hand manipulations of articulated objects, a task space that has not, to our knowledge, been demonstrated elsewhere in a combined task and motion planning setting.

III. Problem Formulation

A. General Problem

The full planning space of the problem combines the full pose of the robot along with the poses of all objects in the scene. Planning in that space in a raw form would normally be very difficult due to the high dimensionality of the space, and the need to model state transitions that take into account the physical interaction of objects. We simplify this space by assuming that the scene (non-actuated degrees of freedom) is fixed, and only moved under special circumstances. In this case, objects in the scene only move when the robot is holding on to them.

This forms the notion of qualitative classes of the world state we mentioned earlier, we call the configuration of the robot's end effectors on the world the *stance* of the robot.

Fig. 2. Initial conditions for a 1D toy world, with a robot on the left, and cart on the right.

Fig. 3. Final conditions for the 1D toy world.

In a mathematical sense, a stance can be thought of as a set of manifolds through the configuration space, where on any individual manifold we have a functional continuous planner.

The multi-step planning problem then becomes one of finding at the discrete level a sequence of stance transitions in the state space, and at the continuous level a valid plan across each manifold connecting the transition points.

A limitation of this approach is that it does not directly model the physics of objects in the world. For example, if the robot deposits a small tabletop object into a location in free space, the object tends to fall to the ground (Robonaut [13] will be allowed to disagree). This can be fixed with a constraint that objects may only be released in stable positions, but it requires the programmer to codify these stable positions.

B. A Toy Problem

The need for this solution approach is easy to demonstrate by a simple, 2D toy problem. Consider a robot and object in a 1 dimensional world, such as Fig. 2. Suppose we wish to achieve a final state as in Fig. 3. To visualize a plan we can see the whole state space represented in 2D in Fig. 4.

We observe that since the cart is not actuated, only certain forms of motion in the state space are possible. A plan must either move along the diagonal line, as when the robot is attached to the cart, or it must move vertically under the diagonal line, as when the robot is unattached. We note that most easy notions of locality or distance metrics break down in this problem as two states may be very close, if the cart is near its goal position, but if the robot is not nearby, the actual manipulation distance may be much larger.

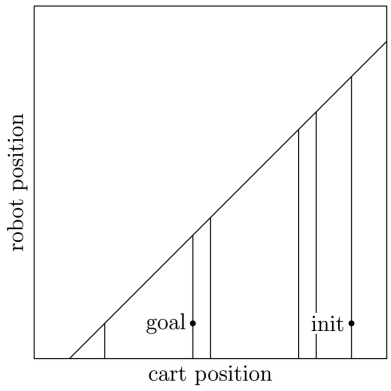

Fig. 4. The full 2D state space for the 1D toy world. Note that motion is only possible along the diagonal line (robot holding the cart) or vertically under the diagonal line (robot unattached to the cart).

For this toy problem it is easy to see how to find a solution: travel up to the diagonal line (grab the cart), traverse the diagonal line until vertically aligned with the goal state (move the cart to its goal position), then move vertically down to the goal state (release the cart and move the robot to its own goal position). Of course, this is an ad hoc solution that would only work for this particular problem, below we describe the multi-step planning algorithm for finding solutions in these sorts of spaces.

IV. Algorithm

Multi-step planning is, fundamentally, a layering of a discrete search algorithm, in this case breadth-first search, on top of specialized continuous planners. Algorithm 1 lists the basic algorithm for multi-step planning as the `getPlan` function.

We consider the full state of our world to include all degrees of freedom of our robot, as well as all the degrees of freedom of any objects we will interact with. Some dimensions of our world can be directly controlled (i.e. robot joints), where as others cannot (e.g. the position of an object).

We further provide the idea of a *stance* in the world as some qualitative and discrete interaction between the robot and objects in the world. For our planner this becomes the question of where and if the robot grippers are attached to another object in the world. We define a *configuration* to be the combination of a stance and the current state of the object(s) in the world. Finally we say that a *state* or *world state* is the combination of a configuration with the current state of the robot (or *robot state*). By restricting the robot to states reachable from a configuration, we create a manifold through our full state space, over which we are able to plan.

The main multi-step planner is run as sketched out in Algorithm 1. It implements a breadth first search process, exploring from the start configuration to adjacent configurations and so on. With the development of a distance metric and a good heuristic we could use a more sophisticated discrete search process, though it would not fundamentally change the structure of the algorithm.

A. Adjacency

An important abstraction here is the idea of an adjacency. Two stances are considered to be adjacent if there exists a world state that is valid in both stances. The implementation of this algorithm requires a way to sample world states that are valid for both the current configuration and an adjacent stance.

B. Reachable

The reachable function is designed to tell the discrete search algorithm if a valid path exists from one world state to another. To do this it has to verify that any changes in the state are valid (i.e. objects cannot move unless the robot is holding them), and use a continuous planner to verify that a feasible path exists for the robot.

Algorithm 1 getPlan

Input: world states $init$ and $goal$
Output: trajectory, if one exists
$state\_queue$.push_back($init$)
while $state\_queue$ is not empty **do**
  $current \leftarrow state\_queue$.pop()
  for all $s \in$ adjacent($current$) and not yet visited **do**
    if s is reachable from $current$ **then**
      mark s as visited
      $state\_queue$.push_back(s)
      if s is the goal stance **then**
        trace steps backwards to construct the trajectory and return
      end if
    end if
  end for
end while

V. Implementation

The implementation of this algorithm involves resolving a number of specific technical challenges. Some of these are specific to the nature of the object we are working with, others are not. The sections below will cover the significant technical challenges and assumptions made by our implementation.

A. The Chair

We examine the problem of a robot that folds (or unfolds) a typical folding chair. The initial and goal states will be specified as poses for the chair, leaving the planner to decide where to hold the chair and how to move. We allow ourselves full prior knowledge of the shape and kinematics of the chair, as well as a list of available grasp points on the chair.

We perform this manipulation with the PR2 robot from Willow Garage, as a practical matter the robot is not strong enough to manipulate the chair arbitrarily, so we add an extra constraint to the state of the chair that requires two legs of the chair to be on the ground at all times, and we assume they will not slip. In most practical configurations this will ensure the robot does not have to support any significant forces with its arms. In this initial implementation we have also assumed that the pose of the chair base link (the back of the chair) will not change.

B. Problem Specifications

We allow the algorithm to start with a complete kinematic model of a real chair, supplied in the standard URDF format. We also provide a list of valid grasp points on the chair. Each grasp point is tied to a particular link on the chair, so it will continue to be valid through manipulation actions. Fig. 5 shows a visualization of available grasp points being evaluated during the planning process, note that some of the grasp points are shown for hypothetical configurations of the chair. Grasps are shown green when a valid kinematic solution could be found to perform a grasp, and red when one could not be found, or because that grasp is considered in collision.

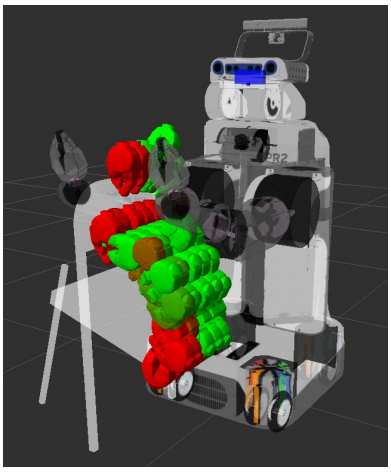

Fig. 5. Visualization of hypothetical grasp points during planning for different chair configurations. Grasps are shown as PR2 grippers, green for valid, red for invalid. Note that some grasps are shown for hypothetical chair positions.

Currently the list of grasp points passed to the algorithm is hand specified, however it would be sensible to do this process automatically. The running time of the algorithm, as well as the existence and quality of a solution, can depend heavily on how many and where grasp points are provided, so having a principled process here would be quite useful.

The initial state of the object (chair) is observed at the beginning of the planning process, the final state is specified as the state of the object and robot. Currently we specify the pose of the root link of the object to be unchanged, and only allow the robot to move the articulated joint of the object.

C. Chair State Estimation

Perceiving the exact state of the chair is also a challenging problem, however a general solution to that problem is outside the scope of this work. We have attached AR Tag style markers to the chair that can be used with a calibrated 2D camera to perceive the pose of the parts of the chair in the frame of the robot. We used the `ar_pose` ROS package to perceive these tags. In ROS we treat the chair as another robot with a URDF file, so it is necessary to extract from multiple observed tracking tags the joint angles of the chair. We calculate the observed rotation between two links of the chair, then project that rotation in quaternion space onto the constraint created by the rotational joint in the URDF specification of the chair. Note that an axis of rotation constraint becomes a hyperplane passing through the origin in 4D quaternion space, so we can use standard linear algebra techniques for this.

D. Adjacent Stance Expansion

For this robotic manipulation task stances can be defined by the state of the robot grippers with respect to what (if anything) they are holding and where they are holding it. This makes it relatively easy to sample stance intersections since they will always be characterized by a change of state for a gripper (grabbing or releasing something).

We maintain an index of sampled world states for every stance intersection, and by choosing an upper limit on how

many states may by sampled for a given stance intersection we avoid the issue of oversampling an intersection if it is revisited.

The sampled world states are maintained in a data structure that associates each world state with the stances it is in, and allows efficient queries of stance intersections to find all world states in that intersection.

E. Reachability Checking

Reachability checking involves determining when a state transition is feasible in 1 step. We have to verify three things, kinematics, collisions, and path validity.

To check kinematics we look at the initial and final states wherever the robot is holding an object. If the robot is holding an object, we use inverse kinematics to verify that a valid kinematic solution exists for the configuration of the object in that state.

Currently we check for collision only by specifically excluding grasp-configuration combinations that are known to cause collisions, however it makes sense to use a more generalized collision checker here.

Checking path validity can be done by a standard robot arm path planner for whatever change in robot joint state is necessary from initial to final conditions. We are currently using joint space interpolation for unconstrained arms, but with further work we plan to incorporate a standard collision avoidance path planner here. Here we add some additional constraints that the configuration of the object may not change unless the robot is holding it with both arms on different links (this is a constraint specific to the chair, but it is conceptually simple to generalize to other objects, a joint cannot be moved unless the robot is holding both the links). When the robot does change the configuration of the object, this creates some special constraints as a closed kinematic chain, discussed below.

F. Closed Kinematic Chains

Solving kinematics for closed chains in general can be difficult. We deal with this problem by prioritizing the pose of the object. When a path must be found where the robot changes the configuration of the object, we perform a linear interpolation in the joint space of the object, and then solve inverse kinematics for the robot to hold the object at points along that path. We keep the joint states of the robot smooth by seeding each IK solver with the solution from the previous state.

G. Discrete Search

We currently use breadth first search for the planning process, thus searching for a plan with a minimum of steps. This may be a significant factor slowing down the search, and we believe heuristic search will be able to offer significant speed up here.

H. Path Execution

We assume that our environment is very predictable and so after finding a solution trajectory we simply execute it without sensor feedback.

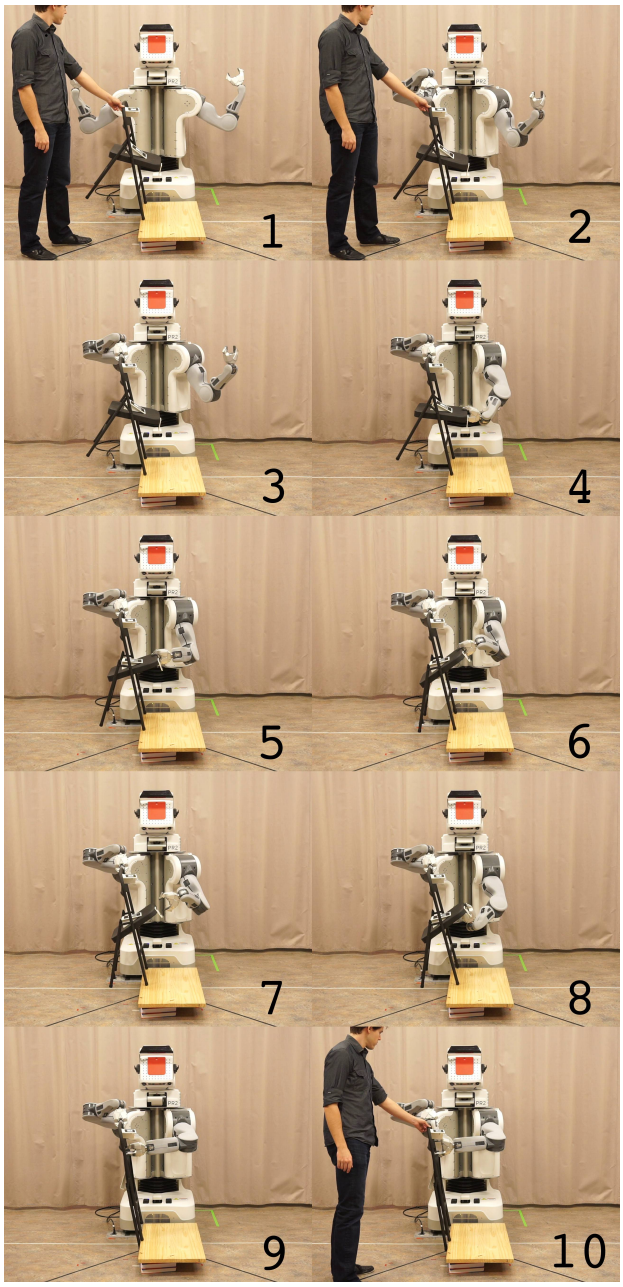

Fig. 6. A sequence of robot states from a plan executed in the lab. The initial state of the chair is shown in tile 1, the goal state is reached in tile 10. Observe how the robot moves the seat part way, then transitions to a different grip (tiles 7-8) from which it is possible to finish the manipulation.

VI. Experiments

We have been able to demonstrate this planner running both in simulation and on the real robot, with successful planning and execution results.

Fig. 6 shows a sequence of states while executing a resulting plan on the real robot. The chair starts in an unfolded state with a goal configuration to have it folded up. The plan is executed blind and is susceptible to perception errors in the initial state of the chair, or small miscalibrations in the sensors, so as a result some undesired motion of the chair is observed. Despite this the plan executes successfully and moves the chair into a folded state. Of particular interest

should be the state where the planner has naturally discovered the need to reposition the gripper part way through the motion. This is a result of the fact that in order of fully close the chair the robot must grasp the seat from below to avoid a collision. However, those grasp poses are not kinematically feasible in the unfolded state, so it is necessary to start with different grasp, fold the chair part way, then change grasp point.

We are currently working to better characterize the performance characteristics of the algorithm, but current trials have shown the computation time to find a 7 step solution at about 10-15 seconds on a quad-core desktop computer.

The full video of the test can be seen here: [`http://youtu.be/cmL4UpjyMng`].

VII. Discussion and Future Work

We are currently working to extend our implementation to to other manipulation problems. It is worth noting some of the complexity bottlenecks that exist in this technique, and some of the ways we might address them for more complex problems in the future.

The complexity of this algorithm (particularly when using breadth first search) is dominated by the branching factor of the discrete search tree. This is determined by how many states are sampled at each stance intersection. In terms of the chair folding problem, there are two factors at play here: how many grasp poses we consider, and how finely we discretize the motion of the chair joint. Both are relatively coarse in our implementation. One can get a flavor for this by looking at the sampled grasps in Fig. 5.

Without enough samples at a stance intersection, the probability of getting within an acceptable distance of the goal position may become very small. As such, a coarse discretization, particularly for more subtle manipulation tasks, could be difficult to use. Also, as we increase the degrees of freedom of the object, this will multiply the number of samples necessary to achieve the same coverage of a stance intersection, and thus multiply the branching factor of the search tree (in nodes where the object can be moved).

We believe the solution to this problem will be switching to a heuristic search approach. Heuristic search approaches will not require expanding every node at a given level of the tree before going deeper, and therefore they should significantly reduce the sensitivity of the algorithm to the branching factor of the discrete graph.

A reasonable extension of this work could also include an algorithm for generating grasp points on the object. The same branching factor issues would apply as above, but a finer grained selection of grasp points may be necessary for problems with tighter kinematic constraints.

A relatively easy extension would be to incorporate the motion of the robot base. This significantly extends the kinematic workspace of the robot, and since it doesn't change the branching factor mentioned above, the only cost would be that associated with a larger continuous planning space.

A limitation of this algorithm is that once a plan is generated it must be executed blind, without any updates based on sensor data. This is usually how traditional motion planners are operated, but when multi-step motion involves interacting with objects, the system becomes less predictable. If a robot gripper is slightly ill-positioned then a grasp may fail, or it may succeed but move the object into an unexpected pose. We observed this behavior in our own experiments. Extensions to this algorithm might provide the opportunity for replanning or adjusting plans based on new sensor data.

VIII. Conclusion

Combined task and motion planning problems represent a new frontier in robotic planning tasks. They allow the programmer to no longer be concerned with the details of where a robot holds an object, or the order of operations necessary for larger problems. In this paper we presented a new approach to combined task and motion planning problems using multi-step planning. We demonstrated the efficacy of the approach by implementing it for a complex manipulation problem involving an articulated object, in this case a folding chair. Our results show that this approach is promising and worthy of extension and further investigation.

References

[1] M. Pflueger and G. S. Sukhatme, "Multi-Step Planning for Robotic Manipulation of Articulated Objects," in *Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications, Robotics: Science and Systems*, 2013.

[2] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[3] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, "Smt-based synthesis of integrated task and motion plans from plan outlines," in *IEEE Intl. Conference on Robotics and Automation*, 2014.

[4] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE Intl. Conference on Robotics and Automation*, 2014.

[5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE Intl. Conf. on Robotics and Automation*, vol. 2. IEEE, 2000, pp. 995–1001.

[9] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal on Artificial Intelligence Research (JAIR)*, vol. 41, pp. 477–626, August 2011.

[10] D. Katz, A. Orthey, and O. Brock, "Interactive perception of articulated objects," in *12th International Symposium on Experimental Robotics*, Delhi, India, Dec 2010.

[11] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *Algorithmic Foundations of Robotics VI*. Springer, 2005, pp. 59–74.

[12] K. Hauser, T. Bretl, and J.-C. Latombe, "Learning-assisted multi-step planning," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 4575–4580.

[13] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter *et al.*, "Robonaut 2-the first humanoid robot in space," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 2178–2183.