1-1-2012

Motion Primitive-Based Graph Planning for Mobile Manipulation With Closed-Chain Systems

Steven R. Gray
University of Pennsylvania, stgray@seas.upenn.edu

Christopher Clingerman
University of Pennsylvania, chcl@seas.upenn.edu

Vijay Kumar
University of Pennsylvania, kumar@seas.upenn.edu

Maxim Likhachev
University of Pennsylvania, maxim@cs.cmu.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-12-06.

Motion Primitive-Based Graph Planning for Mobile Manipulation With Closed-Chain Systems

Abstract

Motion primitive-based (lattice-based) graphs have been used extensively in navigation, but application to high-dimensional state-spaces has remained limited by computational complexity. In this work, we show how these graphs can be applied to mobile manipulation. The formation of closed chains in tasks that involve contacts with the environment may reduce the number of available degrees of freedom but add complexity in terms of constraints in the high-dimensional state space. We propose a novel method to reduce dimensionality by abstracting away the constraints associated with closed-chain systems. Proofs are introduced for the application to graph-search and its theoretical guarantees of optimality. The dimensionality-reduction is done in a manner that enables finding optimal solutions to low-dimensional problems which map to correspondingly optimal full-dimensional solutions. We demonstrate the usefulness of our method with simulation results; we apply our approach to moving an object in 2D using a mobile manipulation platform with a planar arm.

Disciplines

Engineering

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-12-06.

Motion Primitive-Based Graph Planning for Mobile Manipulation with Closed-Chain Systems

GRASP Laboratory Technical Report MS-CIS-12-06

Steven Gray, Christopher Clingerman, Vijay Kumar, and Maxim Likhachev

Abstract

Motion primitive-based (lattice-based) graphs have been used extensively in navigation, but application to high-dimensional state-spaces has remained limited by computational complexity. In this work, we show how these graphs can be applied to mobile manipulation. The formation of closed chains in tasks that involve contacts with the environment may reduce the number of available degrees of freedom but add complexity in terms of constraints in the high-dimensional state space. We propose a novel method to reduce dimensionality by abstracting away the constraints associated with closed-chain systems. Proofs are introduced for the application to graph-search and its theoretical guarantees of optimality. The dimensionality-reduction is done in a manner that enables finding optimal solutions to low-dimensional problems which map to correspondingly optimal full-dimensional solutions. We demonstrate the usefulness of our method with simulation results; we apply our approach to moving an object in 2D using a mobile manipulation platform with a planar arm.

I. Introduction

Motion primitive-based graph planning in high dimensional systems is time consuming as planning times increase exponentially with dimensionality. This is particularly a problem for mobile manipulation where the number of degrees of freedom is quite large. In addition, contacts of end-effectors, arms or bases with the environment result in the formation of a *closed-chain linkage*. A closed-chain linkage is one or more pairs of contacts (including the contact between the base and the ground) are formed thus forming a loop. Such loops introduce complex kinematic constraints in an already high-dimensional system. We propose abstracting away the complexity of closed-chain systems to reduce the dimensionality of the planning problem, and give the conditions for completeness and optimality. For example, in the case of motion constrained to a plane, we may replace the manipulator with a two-degree-of-freedom linkage with two prismatic links but with a finite workspace and ignore the specifics of the manipulator in the abstraction. More generally, complex constraints associated with closed chains are replaced by abstractions that model the key aspects of the contact with the environment, removing unnecessary degrees-of-freedom and enabling switching between open and closed chain topologies for mobile manipulation.

Several theoretical results provide the justification for the method and guarantees on optimality. The benefits of this planning methodology are verified through results and statistics from simulations involving a mobile platform with a planar arm moving an object along a plane.

II. Related Work

Sampling-based planners such as RRTs and PRMs [9, 8] have been used with great success in mobile manipulation. These planners excel at tackling problems with large degrees-of-freedom, but are much less successful when dealing with constrained systems. In particular, there are challenges associated with incorporating kinematic constraints that reduce the size of the state space during a single task. As noted in [1], the representation must incorporate models of kinematic chains being interacted with into the representation of the arm and its changes during the task; *e.g.*, a 7-DOF arm opening a 1-DOF door becomes an 8-DOF arm with additional constraints. Multiple plans are required, one to get the end-effector to the door handle, and another to open the door.

Steven Gray, Christopher Clingerman and Vijay Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, 19104, USA. Email: {stgray, chcl, kumar}@seas.upenn.edu

Maxim Likhachev is with the Robotics Insitute, Carnegie Mellon University, Pittsburgh, 15213, USA. Email: maxim@cs.cmu.edu

Search-based planning algorithms that use motion primitives have been developed extensively in recent years [11]. These methods have been applied to navigation problems [10] as well as various indoor tasks [3],[4]. Recent work has also been conducted on search-based planning with adaptive dimensionality [6]. Two-layer planning schemes have been proposed to reduce complexity, in which a high-dimensional local planner is combined with a low-dimensional global planner. The typical benefit of a multi-layer scheme is a significant reduction in planning time. These local planners have been implemented using various techniques, including reactive obstacle avoidance planners [15] and dynamic windows [12], [2]. While these types of planners can result in difficulties with suboptimality and mismatches between the local and global levels, our approach avoids these problems altogether by generating optimal plans in a low-dimensional space that map to much higher-dimensional optimal solutions.

III. Abstractions for Closed Chain Systems

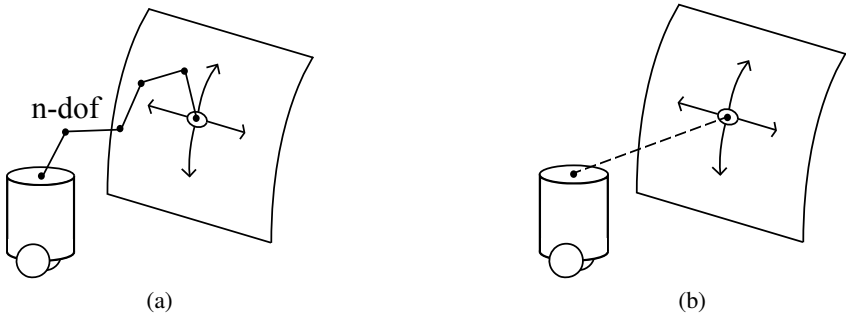

Fig. 1. Example of an abstraction. Planning for an end-effector motion along a constrained manifold may be simplified by planning only for the base and end-effector, then reconstructing the higher-dimensional path afterwards. The n-DOF arm

Consider, for example, a mobile manipulator consisting of an n-degree-of-freedom manipulator atop a differential-drive base as shown in Figure 1. Let $\mathbf{X} \subset SE(2)$ represent the set of configurations of the mobile base, $\mathbf{Y} \subset S^1 \times S^1 \times \ldots \times S^1$ the set of manipulator arm configurations, and $\mathbf{Z} \subset SE(3)$ the set of manipulated object configurations. $\mathbf{W} = \{0, 1\}$ may be used to indicate whether or not the manipulator is in contact with and constrained by the object or the environment. The standard planning paradigm is to plan in $SE(2) \times \mathbb{R}^n$ (where we have replaced S^1 with $\mathbb{R}$), with appropriate constraints on end-effector motion. However, in many settings, mobile manipulation tasks may be encoded solely (but not uniquely) by the motion of the object being manipulated and the motion of the base. Indeed, for a redundant manipulator, there may be infinite motions for the arm satisfying the end-effector motion. But it is clear that a sufficing plan can be found by restricting the search to $\mathbf{X} \times \mathbf{Z}$ provided that, for every sufficing plan, feasible motions in $\mathbf{Y}$ may be calculated by, for example, using inverse kinematics methods. We note that this is feasible in general as long as the reachable arm configurations for a given pair (X, Z) define a path-connected set, ensuring the existence of transitions between consecutive inverse kinematics solutions (see Assumption 1 later in this paper). For manipulators that do not satisfy this condition, and thus whose feasible configurations are in disconnected sets, we must limit ourselves to one such set. In the case of an $n \leq 6$ DOF manipulator interacting with objects in $SE(3)$, replacing the manipulator in $\mathbb{R}^n$ with the object motion in $SE(3)$ does not reduce the dimensionality of the state space. However, the proposed abstractions help in systems with $n \gg 6$ or when $n = 6$ but the object motion is only in $SE(2)$.

Our method analyzes the DOFs associated with the mobile base, manipulator, and end-effector trajectory. Typical planning is done in $SE(2) \times \mathbb{R}^n$ and must obey constraints on the arm required by the desired end-effector motion (the loop-closure constraints). In cases where the end-effector motion is constrained, we show that we may plan in a lower-dimensional space by planning for the base and end effector. The full-dimensional state-space is reconstructed afterward. Thus, we plan for simultaneous base and manipulator motions, but are able to decouple the planning stages so the dimensionality of each planning query is reduced. We demonstrate the method and list the assumptions required to show completeness

and optimality. We illustrate this method using graph-search approaches for planning with applications to the task of opening a door and entering through the doorway.

A. Problem Definition

We represent the full-dimensional planning problem as a graph, $G^f = [S^f, T^f]$, where S^f is the vertex set and T^f is the edge set. Let us define the full-dimensional (of dimensionality h) discretized finite state-space S^f as the 3-tuple (X, Y, Z), where $X \in \mathbf{X}, Y \in \mathbf{Y}, Z \in \mathbf{Z}$. As in Figure 1, $\mathbf{X} \subset SE(2)$ is the set of configurations of the mobile base, $\mathbf{Y} \subset \mathbb{R}^n$ the set of manipulator arm configurations, and $\mathbf{Z} \subset SE(3)$ the set of manipulated object configurations. We emphasize that $\mathbf{Y}$ is finite, containing all valid manipulator configurations associated with positions chosen from $\mathbf{X}$ and $\mathbf{Z}$.

We define a set of transitions $T^f = \{a^f_{i,j} | s^f_i, s^f_j \in S^f\}$. Each transition is associated with a cost $c(a^f_{i,j})$, bounded from below by a positive constant δ. We have an edge-weighted graph G^f with vertex set S^f and edge set T^f. The objective of the planner is to find the least-cost path in G^f from start state s^f_S to goal state s^f_G. Let $\pi(s^f_i, s^f_j)$ denote a path from state i to state j, and let $\pi^*(s^f_i, s^f_j)$ denote the least cost path. The path cost is the sum of the transitions along the path, $\sum_{i,j \in \pi} c(a^f_{i,j})$, denoted as $c(\pi(s^f_i, s^f_j))$.

We note that the 3-tuple S^f is over-defined when an object is attached to the manipulator; $\{(X, Y) | X \in \mathbf{X}, Y \in \mathbf{Y}\}$ maps to a unique $Z \in \mathbf{Z}$ using the forward kinematics mapping f, where

$$f(X, Y) = Z.$$

We define a set of motion primitives as a set of precomputed kinodynamically feasible atomic actions. See Figure 2 for an example. We define a transition from the set T^f to be the result of a motion primitive applied to a state s^f. Let $\mathcal{A}^X$ be the set of motion primitives for $\mathbf{X} \subset SE(2)$ and $\mathcal{A}^Z$ the set of motion primitives for $\mathbf{Z} \subset SE(3)$. Let $\mathcal{A}^Y$ be the set of corresponding motion primitives of $\mathbf{Y} \subset \mathbb{R}^n$ that make the transition in $\mathbf{X}, \mathbf{Z}$ feasible when the manipulator is grasping an object. $\mathcal{A}^Y$ are defined as relative displacements and are dependent on the starting arm configuration Y. When no object is being manipulated, we shall assume the manipulator moves freely. In the full-dimensional state space, we define a motion primitive a^f as a 3-tuple member of the set $\mathcal{A}^f(Y) = \{(a^X, a^Y, a^Z) | a^X \in \mathcal{A}^X, a^Z \in \mathcal{A}^Z, a^Y$ applied to Y enables $(a^X, a^Z)\}$.

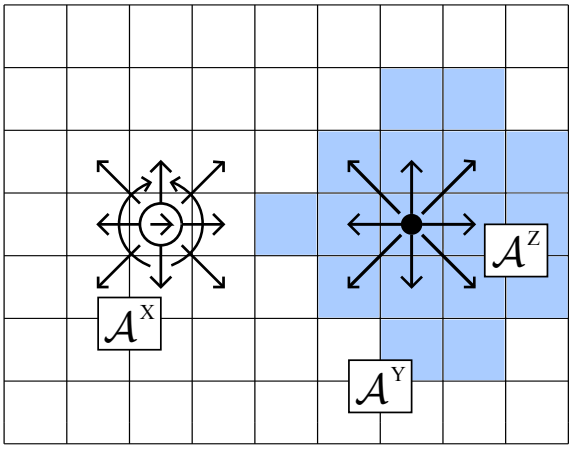

Fig. 2. Example motion primitives for a mobile manipulator attached to an object. $\mathcal{A}^X$ is the set primitives belonging to a mobile base with $X = (x, y, \theta)$ constrained to move along an 8-connected grid or turn in place. $\mathcal{A}^Z$ is the set belonging to an object with $Z = (x, y)$, constrained to move on an 8-connected grid. In this case, $\mathcal{A}^Y$ is the set of arm motions which keep the end-effector on the object during all transitions chosen from the feasible portion of $\mathcal{A}^X \cup \mathcal{A}^Z$. The shaded region represents reachable workspace of the manipulator.

B. Reduced Dimensional Graph

Let us also define a reduced-dimensional (of dimensionality l) discretized finite state-space S^l as the 2-tuple (X, Z). The crux of this paper is that we may also represent the same mobile manipulation motion planning problem as a graph on the reduced-dimensional state-space $G^l = [S^l, T^l]$, where S^l is the vertex set and T^l is the edge set. S^l is a projection of S^f onto the lower-dimensional manifold. We define a many-to-one mapping $\gamma : S^f \to S^l$, in which $\gamma((X, Y, Z)) = (X, Z)$, dropping the manipulator configuration Y. We also define the inverse mapping $\gamma^{-1} : S^l \to S^f$, a one-to-many mapping. When an object is grasped by the manipulator,

$$\gamma^{-1}((X, Z)) = \{(X, Y, Z)|Y \in \mathbf{Y}, f(X, Y) = Z\}.$$

Otherwise, when no object is grasped (i.e., the manipulator forms an open chain),

$$\gamma^{-1}((X, Z)) = \{(X, Y, Z)|Y \in \mathbf{Y}\}.$$

There is also a many-to-one mapping $\varphi : a^f \to a^l$, where a^l is a 2-tuple of the set

$$\begin{aligned}\mathcal{A}^l(Y) = \big\{(a^X, a^Z)|a^X \in \mathcal{A}^X, a^Z \in \mathcal{A}^Z, \\ \exists\, a^Y, Y\ s.t.\ (a^X, a^Y, a^Z) \in \mathcal{A}^f(Y)\big\}\end{aligned} \tag{1}$$

We require that edge costs be such that for every pair of states

$$c(\pi^*(s_i^f, s_j^f)) \geq c(\pi^*(\gamma(s_i^f), \gamma(s_j^f))) \tag{2}$$

The least cost path between any two states in the high-dimensional state-space is at least the cost of the least-cost path between their images in the low-dimensional state-space. The transition cost in the high-dimensional graph is

$$c(\pi(s_i^f, s_j^f)) = c_1(\pi(s_i^f, s_j^f)) + c_2(\pi(s_i^f, s_j^f)) \tag{3}$$

the sum of two terms that are not interrelated. The first term in Equation 3 is also the transition cost function of the lower-dimensional state-space; it is a function of only X, Z, a^X, and a^Z. The additive second term is a positive cost as a function of Y and a^Y.

C. Algorithm

The overall algorithm is to construct and search the reduced-dimensional graph for a least-cost path from start to goal, then use that path to reconstruct one of the corresponding least-cost full-dimensional paths. This decouples the planning for the manipulator from the planning for the mobile platform. Reconstruction is done by traversing the path in the lower-dimensional space. Beginning with the first state, a Y is generated such that $s_1^f = (X, Y, Z)$ is a valid configuration. Then the set of $\mathcal{A}^f$ is examined to find those transitions that produce the desired (X, Z) of the next state in the path. A Y is generated such that $s_2^f = (X, Y, Z)$ is a valid configuration, and then any a^Y that produces s_2^f and satisfies Equation 1 is selected (either by interpolation of inverse kinematics solutions or using a motion planner for the arm).

When constructing the reduced dimensional graph, we must verify the existence of some a^Y such that $(a^X, a^Y, a^Z) \in \mathcal{A}^f(Y)$. This check can either be done using an inverse kinematics query at plan time, or done in advance as a precomputation of the reachable workspace in Z for a given X (the shaded region in Figure 2). The precomputation is also used eliminate transitions with self-collisions. When constructing the graph during planning, we also must check transitions for collisions with the world. It is worth noting that both the high- and low-dimensional graphs include explicit transitions between grasping and not grasping an object.

D. Theoretical Properties

We proceed to show that a graph search on G^l is sound, complete, and optimal. First, for convenience, let us define $\sigma : \pi(s_i^l, s_j^l) \rightarrow \pi(s_i^f, s_j^f)$, which maps a path in the lower-dimensional state-space to a set of corresponding paths in the higher-dimensional state-space.

Assumption 1. *We assume that when the manipulator is connected to the object, the set* $\mathbf{Y}$ *of feasible manipulator arm configurations for a given lower-dimensional state* s_i^l *occupies a path-connected set. For manipulators that do not satisfy this condition, and thus the feasible configurations are in disconnected sets, we limit ourselves to one such set.*

That is, for any given $s_i^l = (X_i, Z_i)$, any corresponding feasible Y_i can be reached from any other feasible Y_j. $\mathbf{Y}$ forms a fully-connected set, though the connections are not required to be enumerated as part of $a^Y \in \mathcal{A}^Y$. When no object is grasped in the manipulator, $\mathbf{Y}$ contains all possible manipulator configurations.

Theorem 1. *Soundness. Any path* $\pi(s_i^l, s_j^l)$ *in* G^l *can be executed in the full dimensional space. That is, every* $\pi(s_i^l, s_j^l)$ *corresponds to at least one path* $\pi(s_i^f, s_j^f)$ *given by* $\sigma(\pi(s_i^l, s_j^l))$.

Proof: As our base case, we know the starting configuration may be mapped to the full-dimensional space. Assume the mapping σ exists and has produced $\pi(s_i^f, s_n^f)$, with $j > n > i$, terminating in (X_n, Y_n, Z_n). From the lower dimensional path, we have $a_{n,n+1}^l = (a_n^X, a_n^Z)$ and $s_{n+1}^l = (X_{n+1}, Z_{n+1})$. By Equation 1, there exists an a^Y such that for some starting configuration (X_n, Y_j, Z_n), there is $(a_n^X, a^Y, a_n^Z) \in \mathcal{A}^f(Y_j)$. However, Y_j may not be equal to Y_n. Assumption 1 maintains that Y_j and Y_n are path connected, so we may transition from Y_n to Y_j. Then, by definition, applying (a_n^X, a^Y, a_n^Z) produces a valid state $s_{n+1}^f = (X_{n+1}, Y_{j+1}, Z_{n+1})$. Thus, by induction, the entire corresponding path is given by $\sigma(\pi(s_i^l, s_j^l))$. ■

Theorem 2. *Completeness. If there exists a path* $\pi(s_i^f, s_j^f)$ *in the* G^f*, then there exists a corresponding path* $\pi(s_i^l, s_j^l)$ *in* G^l.

Proof: As our base case, we know the starting configuration may be mapped to the lower-dimensional space by dropping the Y component. Assume the mapping σ^{-1} exists and has produced $\pi(s_i^l, s_n^l)$, with $j > n > i$, terminating in (X_n, Z_n). From the full-dimensional path, we have $a_{n,n+1}^f = (a_n^X, a_n^Y, a_n^Z)$ and $s_{n+1}^f = (X_{n+1}, Y_{n+1}, Z_{n+1})$. The existence of $a_{n,n+1}^l = (a_n^X, a_n^Z)$ is indicated by Equation 1, because a_n^Y exists. Applying $a_{n,n+1}^l$ to s_n^l results in the retrieval of the next state $s_{n+1}^l = (X_{n+1}, Z_{n+1})$. Thus, by induction, the entire corresponding path is given by $\sigma^{-1}(\pi(s_i^f, s_j^f))$. ■

Theorem 3. *The cost of a least-cost path from start to goal in* G^l *is a lower bound on the cost of a least-cost path in* G^f.

$$c(\pi^*(s_S^l, s_G^l)) \leq c(\pi^*(s_S^f, s_G^f))$$

Proof: Theorem 2 established that the path $\pi_f^*(s_S^f, s_G^f)$ can be mapped onto the lower dimensional state-space S^l. Given the restrictions on edge costs in Equation 2, the costs of any transition in G^l are bounded from above by the cost of any transition it maps to in G^f. Thus, with all transitions comprising the path bounded from above, the cost of a least-cost path from start to goal in G^l is a lower bound on the cost of a least-cost path in G^f. ■

Theorem 4. *Optimality. If* $c(\pi(s_i^f, s_j^f))$ *does not depend on* Y *or* a^Y *(the second term in Equation 3 is zero), the mapping of the least-cost lower-dimensional path into the higher dimensional state-space* $\sigma(\pi^*(s_S^l, s_S^l))$*, is also (one of) the optimal cost path(s)* $c(\pi^*(s_i^f, s_j^f))$ *in* G^f.

Proof: Theorem 1 established the mapping $\pi(s_i^f, s_j^f) = \sigma(\pi(s_i^l, s_j^l))$. Because transition costs are independent of Y and a^Y, and because the full-dimensional states and transitions are mapped to the reduced-dimensional system by dropping only the Y and a^Y terms, the costs remain unchanged. Thus,

the lowest cost path in G^l is one of multiple lowest cost paths in G^f due to the multiplicity of the mapping. ■

By similar arguments, the paths in G^l that are ϵ-suboptimal (are of at most ϵ-times the cost of the least-cost path) are guaranteed to map to ϵ-suboptimal paths in G^f. This result is important when using ϵ-suboptimal searches like Anytime Repairing A*, used in the experiments in this paper.

E. Graph Search

In this paper, we use a lattice-based graph representation [13], [10] to define the transitions between states, allowing motion planning problems to be formulated as graph searches. Lattices are well-suited to planning for constrained robotic systems because, unlike other graph-based representations such as n-connected grids, the feasibility requirement ensures that any solutions found using a lattice will also be feasible.

Given a graph, we need an efficient way to search it for a solution path. We note that our algorithm is independent of the graph search chosen. A* search is a popular method [7], improving upon Dijkstra's algorithm [5] by utilizing a heuristic to focus the search toward promising areas of the search space. However, A* aims to find an optimal path, which may not be possible if deliberation time is limited. Instead, we use an anytime variant of A*, Anytime Repairing A* (ARA*) [11]. The algorithm is guaranteed complete and provides theoretical bounds on suboptimality of solutions. The bound, $\epsilon \geq 1.0$, is an input to the algorithm, specifying the first solution returned is guaranteed to cost no more than ϵ times the optimal. Given additional time, the graph search is able to decrease the bound to 1.0 and provide the optimal solution.

IV. Application to Moving an Object in 2D

To demonstrate the benefits of our method, we test extensively in simulation on a system like that shown in Figures 1 and 2. We use a mobile base ($\mathbf{X} \subset SE(2)$) with an n-DOF planar arm ($\mathbf{Y} \subset \mathbb{R}^n$) to move an object around the ground plane (the object has no notion of directionality, so $\mathbf{Z} \subset \mathbb{R}^2$). The C-space has been inflated so the robot and object can be represented as points. The manipulator may attach and detach from the object, switching between open and closed chains, so $\mathbf{W} = \{0, 1\}$. The object can be thought of as a cylinder with omnidirectional wheels; the planar arms make contact with the cylinder at a height greater than the height of any world obstacles. Thus, obstacles can collide with the mobile base and the object being moved, but not with the arms. When the arm makes contact with the cylinder, we assume it is rigidly attached.

Any state in the full-dimensional state-space is given by

$$s^f = (\underbrace{x_r, y_r, \theta_r}_{X}, \underbrace{\theta_1, \ldots, \theta_n}_{Y}, \underbrace{x_o, y_o}_{Z}, \underbrace{m}_{W})$$

where $(x_r, y_r, \theta_r) \in \mathbf{X}$ is the mobile base pose, $(\theta_1, \ldots, \theta_n)$ is the joint angles of the arm, $(x_o, y_o) \in \mathbf{Z}$ is the cylinder location, and m is a binary value indicating if the object is attached to the manipulator. A state in the reduced-dimensional state-space is given by

$$s^l = (\underbrace{x_r, y_r, \theta_r}_{X}, \underbrace{x_o, y_o}_{Z}, \underbrace{m}_{W})$$

A. Implementation

The goal of the planner is to get the object to a desired location on the 2D grid. The robot must navigate to the object, attach it to the manipulator, then move it along the plane to the goal while avoiding obstacles. There is no fixed goal for the location of the robot base. The heuristic function used to guide the search is the distance between the robot and object plus the distance between the object location and the goal.

When the robot is connected to the object, only the latter is used. Both distances are calculated for the entire map during precomputation using 2D Dijkstra searches.

The cost function is:

$$c(a^l_{i,j}) = (c_{costmap}(a^X, X) + 1)(c_{movement}(a^X) + c_{obj}(a^Z, Z))$$

where $c_{costmap}(a^X, X)$ is the maximum cost cell traversed during the transition, $c_{movement}(a^X)$ the cost associated with moving the robot, and $c_{obj}(a^Z, Z)$ the cost associated with moving the object. The cost is independent of manipulator motion, satisfying Theorem 4. The units for the cost functions are seconds; the movement cost for forward or backwards motion is the distance divided by nominal velocity of the robot and the cost for turning in place is the angular distance divided by the nominal angular velocity. The cost for moving the object is similarly a distance divided by nominal velocity for the object (one could think of it as the speed at which the object may be moved without tipping). The costmap is unitless with a value of 0 for unoccupied space and 254 for the obstacles themselves.

A valid configuration of the arm, Y, when not connected to the object is any configuration not in self-collision. A valid Y when connected to the object must satisfy the (X, Z) pair. Such pairs are constrained such that the object is at least one arm link length distant from the base, but no more than the total length of the arm. The arm is assumed to be above the height of the obstacles and so cannot collide with them. This, coupled with the distance constraint, satisfies the path connectivity requirement of $\mathbf{Y}$).

The reduced-dimensional lattice is constructed using 12 motion primitives, of which 8 are for moving the object in a 8-connected grid, 2 for turning base in place, and 2 for forward and backwards movement. The robot arm is allowed to connect to the object, but not to disconnect from it. ARA* is first run on the reduced-dimensional state-space representation, initially with suboptimality bound $\epsilon = 5.0$, then carried until $\epsilon = 1.0$. Full-dimensional paths are generated by populating the arm joint angles using inverse kinematics (using an iterative method, seeded with the previous state's joint angles). If the interpolation fails to connect two solutions without self-collision, a random arm configuration is generated, checked for collision, then used as the seed for the inverse kinematics call. This method succeeded in generating a full-dimensional path in all trials.

B. Results

We tested the graph planner with closed-chain abstractions on 100 randomly-generated maps of size 100 by 100 cells, 10 cm on a side. 50 of these were pseudo-outdoor environments (random circular obstacles) and 50 were pseudo-indoor environments (grid obstacles). Robots with planar arms of 3 or 10-links were used; the 3-link arms had link lengths of 10 cm, while the 10-link arms had link lengths of 4 cm. The graph planner results were compared against those found by a sampling-based planner, RRT [9, 8]. The RRT was implemented as two successive searches; the first (S1) was to bring the robot end effector into contact with the object, and the second (S2) to move the robot end effector, now with object attached, to the goal location. The RRT was unidirectional, with the root at the robot start location. At each iteration, the probability of sampling from the goal region was 0.02. Goal sampling was accomplished by sampling within an annulus determined by the distance constraints of the object or object goal, then using inverse kinematics to get feasible arm joint angles. Any sample, after being checked for collision (between the robot base or object and obstacles) and self-collision, was connected to the nearest state in the tree provided the interpolation between the two was also collision-free.

For the graph search, planning times and number of states expanded are shown for the initial plan with $\epsilon = 5.0$ and the final plan with $\epsilon = 1.0$. The reduced-dimensional planning time is given by LD, and the full-dimensional path reconstruction from the reduced-dimensional path is listed under FD. Reconstruction was only done for the $\epsilon = 1.0$ paths. Results for indoor and outdoor environments are shown in Tables I and II, respectively. Sample runs (with the arms not pictured for clarity) are shown in Figures 3 and 4.

It is worth noting the RRT failed to plan in under 30 seconds for the 10-link chain in 4 outdoor trial environments, and 3 indoor trial environments. These environments featured narrow passageways that a

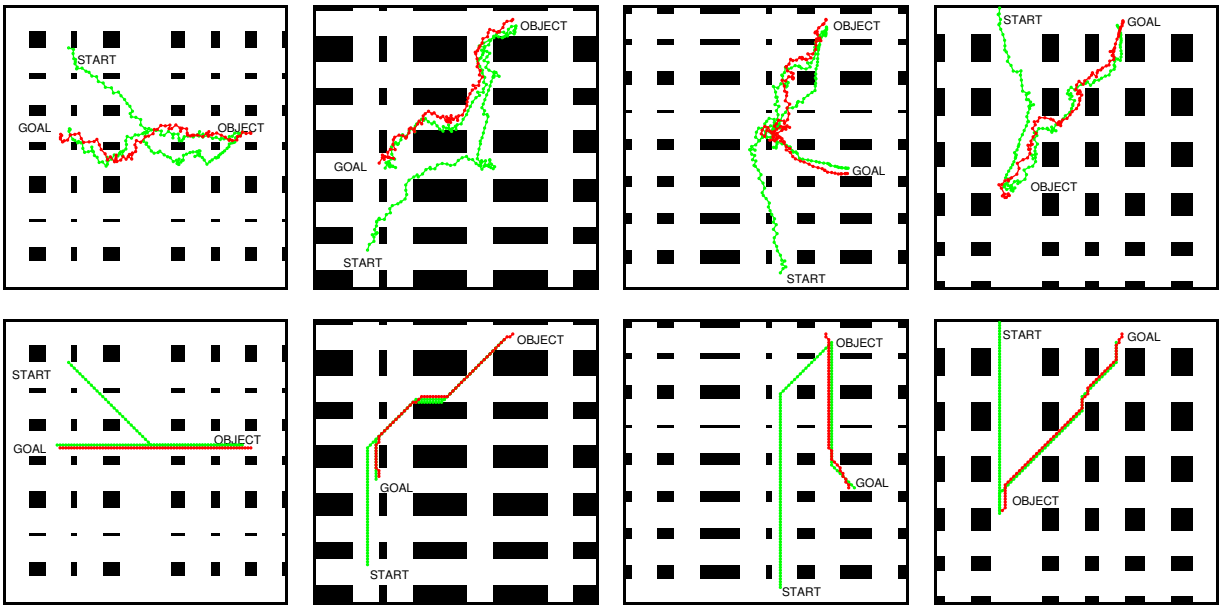

Fig. 3. Comparison of indoor plans between RRT (top row) and graph planner (bottom row). Environments are 100 by 100 cells with 10 cm cell sides. The robot has a 10-link arm, with each link being 4 cm in length. The robot must approach an object, pick it up with the manipulator, and bring it to the goal. The green indicates the robot trajectory and the red indicates the object trajectory.

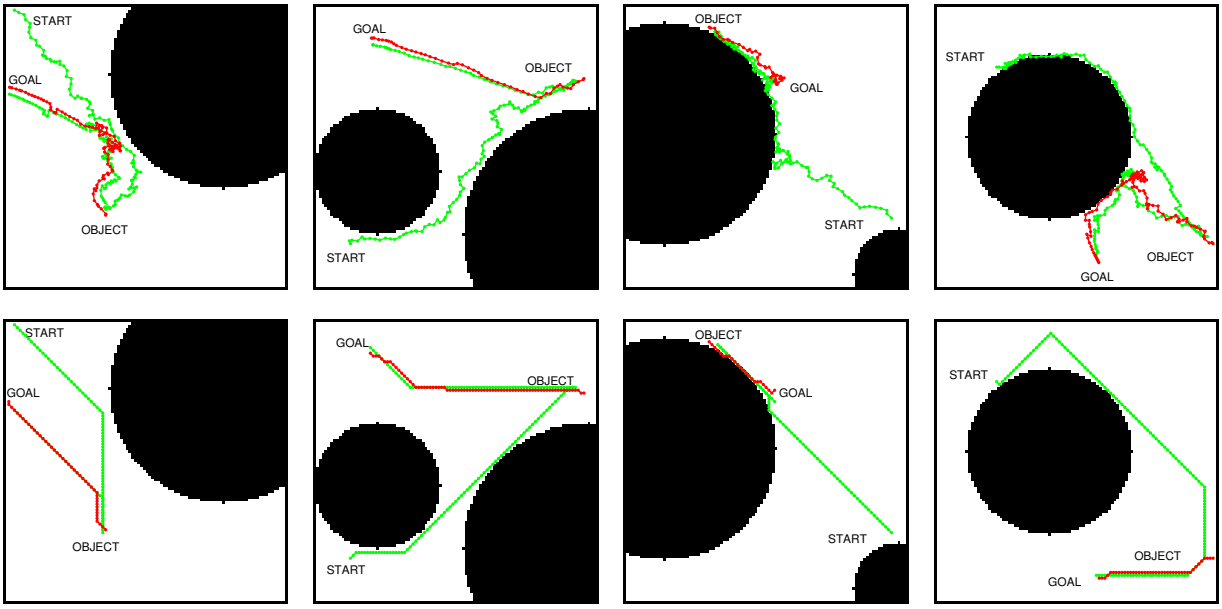

Fig. 4. Comparison of outdoor plans between RRT (top row) and graph planner (bottom row). Environments are 100 by 100 cells with 10 cm cell sides. The robot has a 10-link arm, with each link being 4 cm in length. The robot must approach an object, pick it up with the manipulator, and bring it to the goal. The green indicates the robot trajectory and the red indicates the object trajectory.

vanilla RRT is ill-equipped to handle (though variants such as [14] help handle such regions). The outdoor environments are shown in Figure 5. The graph search was successful on all environments in the same time limit.

Path length comparisons were done on a specific indoor environment, with the start and goal perturbed slightly each time. Results from these runs are given in Table III, and highlight the benefits of the graph search, determinism and repeatability. Similar problems will have consistent, similar, optimal solutions. The graph search path lengths for the base and the object are less than half of those for the RRT, and the

	Graph Search					RRT		
Arm DOFs	Subopt. Bound ϵ	Planning Times (s) LD	Planning Times (s) FD	Expansions	Success	Planning Times (s) S1	Planning Times (s) S2	Success
3	5 1	$< 0.01 \pm < 0.01$ 2.30 ± 2.45	N/A $< 0.01 \pm < 0.01$	287 ± 139 $3.2 \times 10^5 \pm 3.0 \times 10^5$	50/50	0.13±0.12	0.11±0.09	50/50
10	5 1	$< 0.01 \pm < 0.01$ 3.48 ± 3.19	N/A 0.023 ± 0.012	288 ± 164 $4.78 \times 10^5 \pm 4.17 \times 10^5$	50/50	2.81±3.28	4.24±6.08	47/50

TABLE I
GRAPH PLANNER AND RRT PLANNER COMPARISON FOR SIMULATED INDOOR ENVIRONMENTS.

	Graph Search					RRT		
Arm DOFs	Subopt. Bound ϵ	Planning Times (s) LD	Planning Times (s) FD	Expansions	Success	Planning Times (s) S1	Planning Times (s) S2	Success
3	5 1	$< 0.01 \pm < 0.01$ 1.39 ± 1.63	N/A $< 0.01 \pm < 0.01$	280 ± 154 $2.36 \times 10^5 \pm 2.71 \times 10^5$	50/50	0.20±0.44	0.14±0.27	50/50
10	5 1	$< 0.01 \pm < 0.01$ 2.57 ± 2.50	N/A 0.024 ± 0.013	253 ± 139 $3.39 \times 10^5 \pm 3.19 \times 10^5$	50/50	1.87±1.95	1.29±1.03	46/50

TABLE II
GRAPH PLANNER AND RRT PLANNER COMPARISON FOR SIMULATED OUTDOOR ENVIRONMENTS.

corresponding standard deviations are less than one-fifth that of their RRT counterparts.

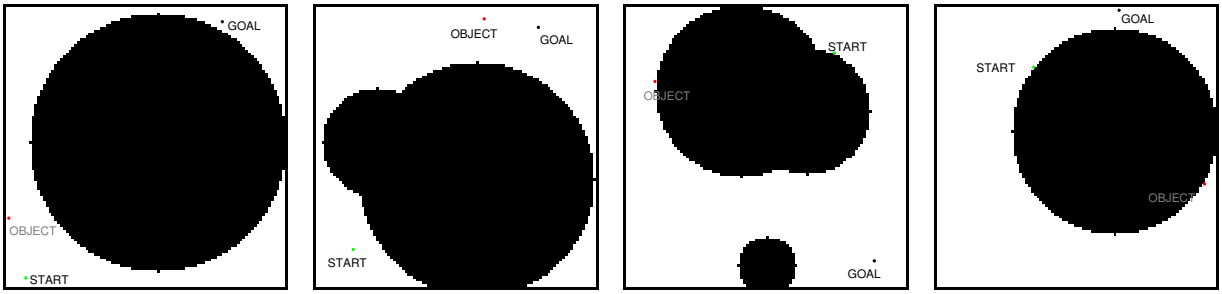

Fig. 5. Environments in which the RRT fails to find a solution in under 30 seconds, but the graph planner succeeds. These environments are characterized by narrow passageways the robot must traverse.

V. CONCLUSION

We propose a methodology for reducing the dimensionality of search- based planning problems for mobile manipulators by creating lower dimensional abstractions. The central idea is to only retain the configuration space of the mobile base and the objects in the environment without explicitly modeling the configuration of the arm or the constraints associated with closed-chain systems that are formed when the arm contacts objects in the environment. The mathematical formulation for our approach allows us to prove optimality and completeness under very reasonable assumptions. While we illustrate the results in a simple scenario, the basic framework and approach are applicable to a wide variety of mobile manipulation tasks in home environments.

REFERENCES

[1] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *International Journal of Robotics Research (IJRR)*, March 2011.

[2] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, pages 341–346, 1999.

	Graph Search		RRT	
Arm DOFs	Base Path Length	Object Path Length	Base Path Length	Object Path Length
3	8.60 ± 0.53	4.86 ± 0.41	19.88± 4.07	13.61± 3.53
10	8.46 ± 0.53	4.81 ± 0.40	18.53± 2.89	12.33± 2.50

TABLE III
GRAPH PLANNER AND RRT PLANNER COMPARISON OF PATH LENGTHS FOR THE ROBOT BASE IN PERTURBATIONS OF A SIMULATED INDOOR ENVIRONMENT.

[3] Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *IEEE International Conference on Robotics and Automation*, 2010.

[4] Benjamin Cohen, Sachin Chitta, and Maxim Likhachev. Search-based planning for manipulation with motion primitives. In *IEEE International Conference on Robotics and Automation*, 2010.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, number 1, pages 269–271, 1959.

[6] Kalin Gochev, Benjamin Cohen, Jonathan Butzke, Alla Safanova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Proc. of the Fourth International Symposium on Combinatorial Search (SoCS)*, 2011.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[8] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[9] Steven M Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Department, Iowa State University, October 1998.

[10] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research (IJRR)*, 2009.

[11] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 2003.

[12] R. Philippsen and R. Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *IEEE International Conference on Robotics and Automation*, pages 446–451, 2003.

[13] M. Pivtoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3231–3237, 2005.

[14] X. Tang JM. Lien N. Amato S. Rodriguez. An obstacle-based rapidly-exploring random tree. In *IEEE International Conference on Robotics and Automation*, 2006.

[15] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, 1998.