Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task

Kris Hauser Victor Ng-Thow-Hing

February 3, 2011

Abstract

Robots that perform complex manipulation tasks must be able to generate strategies that make and break contact with the object. This requires reasoning in a motion space with a particular *multi-modal* structure, in which the state contains both a discrete mode (the contact state) and a continuous configuration (the robot and object poses). In this paper we address multi-modal motion planning in the common setting where state is high-dimensional, and there are a continuous infinity of modes. We present a highly general algorithm, Random-MMP, that repeatedly attempts mode switches sampled at random. A major theoretical result is that Random-MMP is formally reliable and scalable, and its running time depends on certain properties of the multi-modal structure of the problem that are not explicitly dependent on dimensionality. We apply the planner to a manipulation task on the Honda humanoid robot, where the robot is asked to push an object to a desired location on a cluttered table, and the robot is restricted to switch between walking, reaching, and pushing modes. Experiments in simulation and on the real robot demonstrate that Random-MMP solves problem instances that require several carefully chosen pushes in minutes on a PC.

1 Introduction

A wide variety of challenging problems in robotics involve the motion of certain type of *multi-modal* hybrid system in which the continuous motion switches between discrete modes, and each mode limits motion to a submanifold of configuration space. Multi-modal systems occur in manipulation using multiple grasp and regrasp operations [1, 13, 23, 37], dexterous manipulation [10, 42, 44], legged robot locomotion [2, 4, 11, 16, 24, 35], navigation among movable obstacles [31, 38, 39], assembly planning [41], compliant contact motion [22], and reconfigurable robots [8, 9]. In general they occur in systems that make and break contact, because each discrete change of contact introduces or removes a closed-chain constraint on the robot's motion. In order to unify these disparate application domains, there is a need for a common mathematical language as well as versatile, domain-independent algorithms for multi-modal planning.

Motivated by a humanoid robot manipulation task, we consider the setting where the set of possible modes (the *mode space*) is a continuous infinity (e.g., corresponding to all possible contact placements of the robot against the object). In earlier work we addressed multi-modal planning in feasible spaces that consist of a discrete, finite number of overlapping submanifolds, and we presented a probabilistically complete algorithm that interleaves motion planning queries across submanifolds [17]. But to plan in a continuous infinity of modes, a multi-modal planner must select a finite subset of modes in which to perform single-mode planning queries. The discretization is critical because each query is relatively expensive: too coarse, and no feasible multi-modal path will be found; too fine, and planning will be exceedingly slow. Furthermore, the mode space is high-dimensional itself, so random sampling is necessary to avoid the curse of dimensionality. The primary theoretical contribution of this paper is to study the conditions under which a sparse sampling of modes can capture the connectivity of mode space with high probability.

We present a motion planner, Random-MMP (originally presented in [19]), that adaptively discretizes the mode space by building a tree, where each iteration extends the tree with a mode switch sampled at random. At each mode switch, it queries a sample-based planner to find a continuous single-mode path. Our theoretical analysis shows that given enough time, Random-MMP will find a solution if one exists, as long as the hybrid configuration/mode space satisfies certain reachability characteristics. Reachability requires that any mode can reach a significant subset of mode space within a finite number of mode switches, and that single-mode planning among those switches has a nonzero probability of success. If these properties are satisfied and a solution exists, then the probability that Random-MMP finds a solution approaches 1 quickly as the number of iterations grows large. Furthermore, running time is not explicitly dependent on dimensionality.

To ground our work in a concrete problem, we apply Random-MMP to the task of enabling the Honda humanoid robot to push an object on a table to a desired location using little visual feedback. This task is challenging because it requires planning in an 18-dimensional space under complex kinematic and dynamic constraints. For example, an object must remain within a certain distance of the edge of the table or else it cannot be retrieved (Figure 1), and furthermore this reach

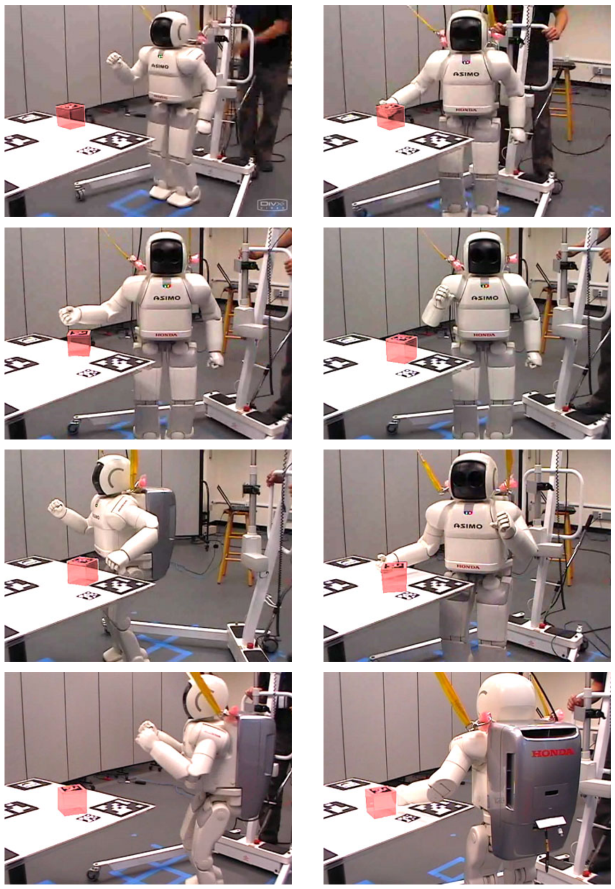


Figure 1: Pushing a block with the Honda humanoid robot, switching between walking, reaching, and pushing modes. The block is shaded for contrast.

constraint is a complex function of the geometry of the robot and surrounding obstacles. The robot can use the surfaces of either hand to push, but to ensure the results of pushing are predictable, we restrict the robot to use a class of stable pushing actions [27]. Integrating with the robot platform imposes the further constraint that the robot must stand while pushing, because its hand cannot be positioned accurately while it walks. These constraints introduce a multi-modal structure to the problem, where the system moves between *walking*, *reaching*, and *stable pushing* modes.

To help improve planning time as well as execution time, we encode problem-specific domain knowledge into the sampling distributions used in Random-MMP. This domain knowledge takes the form of precomputed tables that measure the expected distance the object can be pushed, as a function of its position relative to the robot. These tables are used in a sampling strategy that selects configurations of the robot in locations that are likely to execute long-distance pushes. Experiments show that this strategy improves the running time of Random-MMP such that it can solve problems on an empty table in less than a minute, and difficult problems with cluttered obstacles in minutes on a PC. Results are demonstrated in simulation and in experiments on the real Honda humanoid.

The rest of the paper is organized as follows. Section 2 describes background and related work. Section 3 provides a general formulation of multi-modal problems and describes the formulation of three example problems as well as the humanoid pushing task. Section 4 describes the general Random-MMP algorithm, and Section 5 analyzes its running time. Section 6 describes the implementation of Random-MMP subroutines on the pushing task, and Section 7 describes an improved expansion strategy. Section 8 describes the integration of Random-MMP on the Honda humanoid, and Section 9 concludes with directions for future work.

2 Related Work

Many hybrid dynamical systems, such as systems with contact, have multi-modal motion spaces. Hybrid system modeling, verification, controllability, and optimality have been studied heavily [3, 5, 6, 15, 29]. The distinguishing characteristic of multi-modal problems is that they possess discrete dynamics that induce submanifold structures (e.g., discontinuities in the dynamics of rigid body contact). In many systems of interest, these submanifolds are high-dimensional, nonlinear, and also impose complex operational constraints, so the problem of finding a feasible path can be quite challenging.

Traditionally, planning algorithms have been problem-specific and application-driven, but recent work is moving toward more general modeling frameworks such as STRIPS-like languages for task-and-motion planning [7]. This section will compare prior multi-modal planning algorithms in a problem-independent framework.

2.1 Problem-Specific Planners

Some low-dimensional or geometrically tractable problems have been solved by searching a graph whose vertices enumerate the connected components of each mode [1, 2, 41], but these approaches do not scale well to geometrically complex, high-dimensional problems. Another approach is to use a sample-based planner that randomly samples control inputs and mode switching actions (e.g., pp. 270–274 of [25], [42]). However, this may fail to make mode switches, because in many problems transition regions (the set of configurations in which mode switches are applicable) have zero or miniscule volume and are unlikely to be sampled at random. Random-MMP and other planners address this issue by explicitly sampling configurations in transition regions as a bridge between independent single-mode planning queries.

2.2 Mode Graph Search for Finite-Mode Problems

A general *mode graph* approach was introduced in a manipulation planning application [1], where the vertices of the mode graph identify modes, and arcs identify adjacencies between modes. The planner searches along the mode graph while explicitly constructing single-mode motions between neighboring modes. This approach scales well to high-dimensional problems when sample-based techniques are used for single-mode planning queries [4, 16, 30].

To improve planning speed, a common heuristic decouples the choice of modes and the planning of single-mode paths [8, 9]. In manipulation this takes the form of planning an object trajectory first, and then planning motions to make or break contact with the object [10, 23, 44]. Although this approach may improve efficiency, it is also unreliable, because it cannot be guaranteed in general that the modes produced by the first stage contains a solution path. In recent work we presented a variant that achieves probabilistic completeness by interleaving planning across modes in the mode graph [17]. In this variant, heuristics are used only to bias planning toward promising modes but do not exclude the possibility of finding a path elsewhere.

2.3 Discretizations for Continuous-Mode Problems

For problems with a continuous infinity of modes, the mode graph approach cannot be directly applied. Rather, a discretization step must be performed either before [16, 30, 31] or during [1, 39, 37, 19] planning. The choice of discretization is critical: if too few modes are chosen, the planner cannot find a path; too many, and planning will be extremely slow. A primary contribution of this work is to study the conditions necessary for the planner to find a path with a small number of modes sampled during planning.

Most similar to our work, van den Berg et. al. proved probabilistic completeness of a planner for navigation among movable obstacles under a certain path clearance assumption [39]. This planner discretizes a continuous mode space by growing a tree, and on each iteration samples a mode switch uniformly from existing modes in the tree. They also

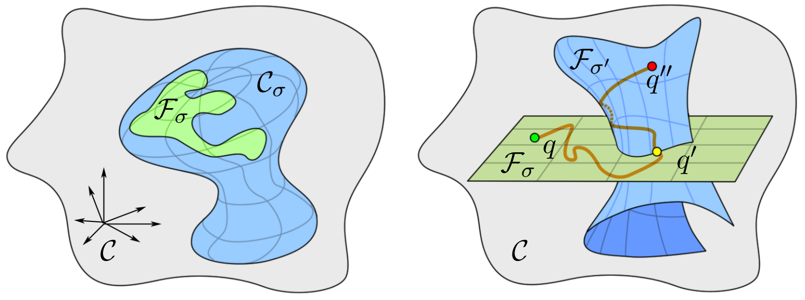


Figure 2: (a) At a mode σ, motion is constrained to a subset $\mathcal{F}_\sigma$ of a submanifold $\mathcal{C}_\sigma$ with lower dimension than $\mathcal{C}$. (b) To move from configuration q at mode σ to q'' at an adjacent mode σ', the motion must pass through a transition configuration q'.

assume that the connected components of a single-mode feasible space can be computed exactly, which restricts the applicability of their work to low-dimensional problems. Our work also grows a tree of modes, but 1) does not require the single-mode planner to be complete, 2) uses a reachability assumption that generalizes to more problems, and 3) achieves a better exponential convergence bound using a nonuniform sampling strategy.

3 Multi-Modal Problem Definition

This section presents a general formulation of multi-modal planning problems, and illustrates its instantiation on two example problems and on the humanoid pushing task. The terminology introduced here will be used throughout the paper.

3.1 Definitions

The system's *hybrid state* (q, σ) consists of a configuration q in a continuous configuration space $\mathcal{C}$, and a *mode* σ in a mode space Σ. Each mode $\sigma \in \Sigma$ defines a *feasible space* $\mathcal{F}_\sigma \subseteq \mathcal{C}$, the set of configurations that satisfy certain mode-specific operational constraints. A motion planning query asks for a feasible path connecting the start state $x_s = (q_s, \sigma_s)$ to the goal state $x_g = (q_g, \sigma_g)$. The problem also defines an *adjacency* condition such that each subsequent mode along the path must be adjacent to the prior mode.

Feasible spaces are often lower dimensional submanifolds of $\mathcal{C}$ (Figure 2a). In general, a mode may also impose constraints on the derivative of the trajectory, but for simplicity we do not consider them here. So, the dynamics of the system allow either 1) single-mode motion along a submanifold:

$$\begin{aligned} &\dot{q} = u \text{ for some } u \in T_\sigma(q) \\ &\sigma \to \sigma \end{aligned} \tag{1}$$

where $T_\sigma(q)$ is the tangent space of the submanifold at q (see Section 3.2), or 2) a mode switch

$$\begin{aligned} &\dot{q} = 0 \\ &\sigma \to \sigma' \text{ for some adjacent mode } \sigma'. \end{aligned} \tag{2}$$

Note that adjacency only means that a mode switch is not forbidden, not necessarily that a feasible motion connects σ and σ'. The difference is that adjacency can be tested easily, while feasibility cannot. To feasibly switch from σ to σ', the system must reach a *transition* configuration $q \in \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ that satisfies the constraints of both modes (Figure 2b).

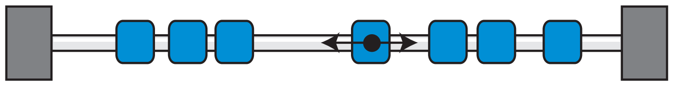
3.1.1 Example 1: Multi-Robot Coordination

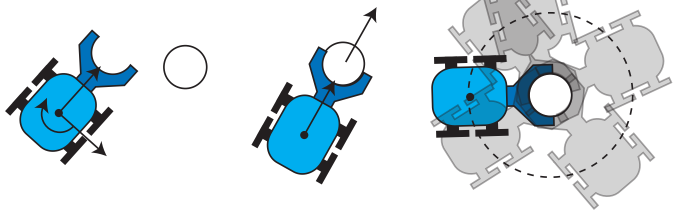
Consider a system of N objects that can translate one-by-one on a line segment $[a, b]$ without overlapping, like a single row of an abacus (Figure 3a). The N dimensional configuration space is $\mathcal{C} = [a, b]^N$. In a given mode, one object translates while the others remain fixed, so each feasible space is a 1-dimensional subset of $\mathcal{C}$.

An important semantic point throughout this work is that a single mode is defined such that σ specifies the values of all stationary degrees of freedom. Thus, in this example there are a *continuous infinity* of modes, which can be partitioned into N discrete "families" corresponding to which object is moving. This definition is chosen because although it would be somewhat natural to say that the system is in "mode k" if object k is moving, this statement is ambiguous as to which submanifold of $\mathcal{C}$ contains the object's motion. Rather, we say that the system is in "a mode in family k". This will be discussed further in Section 3.3.

3.1.2 Example 2: Mobile Robot Pushing a Barrel

Consider a mobile robot that can translate and rotate on the plane among static obstacles, and can push a cylindrical barrel (Figure 3b). The robot's configuration $q_R = (x_R, y_R, \theta_R)$ has 3 parameters, the barrel's configuration $q_B = (x_B, y_B)$ has 2 (here, orientation does not matter), and thus the overall configuration space is 5-dimensional. When the robot is not contacting the barrel (a *transit* mode), it moves in


(a) Coordinating N robots on a line segment, where only one robot moves at once.


(b) A mobile robot pushing a barrel. During transit, the system moves in the 3D subspace of the robot configuration. During transfer, only forward motion is allowed, so the system moves in a 1D subspace. The set of configurations that transition from a transit to transfer modes is a nonlinear 1D submanifold.

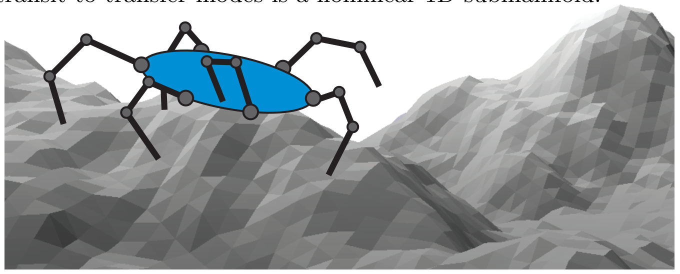

(c) A legged robot on rough terrain. For any set of fixed foot placements, the range of allowed motions defines a submanifold in configuration space.

Figure 3: Three examples of multi-modal problems.

a 3D feasible space. Suppose that for safety reasons the robot must cradle the barrel fully with its gripper before pushing, and also that it can only drive in a straight line while pushing. Then, when the robot is contacting the barrel (a *transfer* mode), it moves in a 1-D feasible space. To switch from a transit to a transfer mode, a point (x_C, y_C) in the local frame of the robot must meet the center of the barrel.

3.1.3 Example 3: Legged Locomotion

An N-legged robot with j joints can be represented in a configuration space of dimensionality $j + 6$, where 6 parameters describe the transition and orientation of the body and j represent the joint angles (Figure 3c). Assuming each foot is a point contact, any placement of any subset of feet against the terrain defines a mode. A mode σ with n contacts imposes closed-loop constraints, which restricts motion to a submanifold $\mathcal{C}_\sigma$ with dimensionality $j + 6 - 3n$, and also imposes balance and collision constraints that reduce the volume of $\mathcal{F}_\sigma$.

3.2 Varying Dimensionality of Feasible Spaces and Transitions

Although $\mathcal{C}$ may be very high dimensional, the operational constraints of a mode σ usually contain functional equalities, denoted $C_\sigma(q) = 0$, that restrict $\mathcal{F}_\sigma$ to lie on a lower-dimensional submanifold $\mathcal{C}_\sigma$. Additional constraints may further reduce the volume of $\mathcal{F}_\sigma$ (Figure 2a). It is not uncommon for $\mathcal{F}_\sigma$ to be empty; in some problems more than 95% of modes have empty feasible spaces [4].

The equalities $C_\sigma(q) = 0$ typically arise if a subset of degrees-of-freedom are allowed to move at once (e.g., in the barrel-pushing example), or if contact constraints impose closed kinematic loops (e.g., in the legged locomotion example). To find configurations that satisfy closed-chain constraints, the submanifolds $\mathcal{C}_\sigma$ may be parameterized explicitly using analytical inverse kinematics (IK) solutions [12], or configurations may be moved onto $\mathcal{C}_\sigma$ using numerical techniques [43]. The latter approach will be discussed further in Section 6.1.

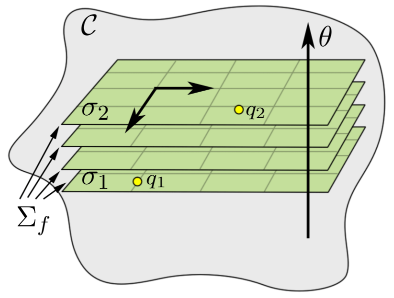
To make a feasible switch from σ to σ', a motion must reach a transition configuration in $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$, so $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ must be nonempty (Figure 2b). But this region may be of lower dimension than $\mathcal{F}_\sigma$, $\mathcal{F}_{\sigma'}$, or both. If so, special techniques (e.g., IK) must be used to explicitly find a transition configuration.

3.3 Continuous Mode Families

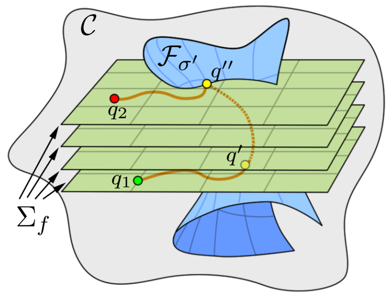
For systems where Σ is infinite and uncountable, we partition Σ into F disjoint *families*, $\Sigma_1 \ldots \Sigma_F$. Every mode $\sigma \in \Sigma_f$ is defined uniquely by a *coparameter* θ in a manifold Θ_f (Figure 4a). We say $\dim(\Theta_f)$ is the *codimension* of σ.

Families are chosen such that no two modes of a family overlap, so to switch between two modes of the same family, the system must first switch to (at least one) other family (Figure 4b). Also, any configuration q in $\sigma \in \Sigma_f$ identifies a single coparameter θ (Figure 4a). So, a mode $\sigma = (f, q)$ can be represented by an integer f to describe the family, and a *representative* configuration q from which the coparameter is uniquely derived. This model is sufficiently general for most hybrid systems.

Families usually arise when the dimension-reducing equality $C_\sigma(q) = 0$ can be written in the form $C_f(q) = \theta$. The submanifolds $\mathcal{C}_\sigma$ are then



(a)



(b)

Figure 4: (a) A continuous family of modes Σ_f. Each value of the coparameter θ defines a different mode $\sigma \in \Sigma_f$ (four are shown). Each mode is depicted as a linear subspace, but modes may be nonlinear in general. A mode in Σ_f can be identified by a representative configuration, e.g. q_1 identifies σ_1 and q_2 identifies σ_2. (b) Moving within Σ_f, from q_1 to q_2, requires transitioning to a different mode σ' at q', and back again at q''.

the level sets of $C_f(q)$. In a rough mathematical sense, a family is a foliation, and its modes are the leaves.

3.3.1 Example 1: Multi-Robot Coordination

Again consider the system of N objects moving one-by-one on a line segment. Let the coordinates of the fth object be x_f. Then a mode σ in the fth family allows object f to be moved while keeping all other objects fixed, and hence x_f varies. We describe σ using coparameters $x_1, \ldots, x_{f-1}, x_{f+1}, \ldots, x_N$, the positions of all objects except for f. Thus, each mode has codimension $N-1$.

A mode σ in Σ_f and a mode σ' in Σ_g, $f \neq g$, are adjacent if all object positions in σ except for object g match all object positions in σ' except for object f. So, there is a 1-D space of possible transitions from σ to Σ_g, and each transition corresponds to a position of object f when the transition is taken.

3.3.2 Example 2: A Mobile Robot Pushing a Barrel

In the mobile robot example, the coparameters of the transit mode family are specified by the position of the barrel q_B, so its codimension is 2. The transfer mode family also has codimension 2, with coparameters given by the orientation of the robot and the position of the object in the direction *perpendicular* to the robot's heading, which together identify the line on which the barrel is allowed to travel. A transit mode can transition to a 1-D set of transit modes, parameterized by the angle at which the robot makes contact with the barrel. A transfer mode can transition to a 1-D set of transfer modes, corresponding to how far the barrel is pushed.

3.3.3 Example 3: Legged Locomotion

In the legged locomotion example, suppose the terrain surface is parameterized by two parameters (u, v), and all footfalls are point contacts. A robot with N feet has 2^N mode families, each one describing a subset of feet in contact with the terrain. A mode σ, with $n \leq N$ feet in contact with the terrain, assigns foot i_1 to touch the terrain at (u_{i_1}, v_{i_1}), foot i_2 to touch the terrain at (u_{i_2}, v_{i_2}), ..., and foot i_n to touch the terrain at (u_{i_n}, v_{i_n}). The mode belongs to a family Σ_f, corresponding to the foot indices $S_f = \{i_1, \ldots, i_n\}$. Σ_f has $2n$ coparameters describing the positions of the feet, $\theta = (u_{i_1}, v_{i_1}, \ldots, u_{i_n}, v_{i_n})$.

Assuming one contact can be made or broken at once, the robot can switch from σ to one of n modes by removing a contact. If $n < N$, then it can also switch to a continuous infinity of modes by placing a free foot against the terrain. These adjacencies form a 2D subspace in each of the $N - n$ adjacent mode families.

3.4 Specification of the Humanoid Pushing Task

Pushing is an especially useful manipulation capability for a humanoid robot, because it permits the manipulation of objects that are too large or heavy to be grasped. Any part of the hands – palm, back, fingertips, knuckles – may be used for pushing; this greatly increases the workspace in which manipulation can be applied.

Integrating with the Honda humanoid platform imposes constraints that divides motion into walking, reaching, and stable pushing modes. Because each arm has a small workspace, it is difficult to keep the hand from oscillating while the robot is walking, and thus the robot must stand still while reaching for and pushing the object. Furthermore, visual feedback is not possible during pushing because the robot's cameras cannot be tilted downward to view objects near its waist, so it is constrained to use a class of stable pushes [27].

3.4.1 Modeling Assumptions

We plan for the robot to push an object across a horizontal table. We move one arm at a time for convenience. We assume the object moves quasi-statically (slides without toppling and comes to rest immediately), and can be pushed without affecting the robot's balance. The planner is given a perfect geometric model of the robot, the object, and all obstacles. Other physical parameters are specified, e.g.

Type	Parameters (dims)	Constraints (dims)	Dims	Coparameters	Codims
Walk	$R_b(3)$		3	$O(3)$	3
Reach	$R_a(5)$, $R_h(1)$		6	$R_b(3)$, $O(3)$	6
Push	$O(3)$, $R_a(5)$	contact (5)	3	$R_b(3)$, $C_h(2)$, $C_o(2)$	7

Table 1: Dimensionality of mode families in the humanoid pushing task. We report the non-fixed configuration parameters (Parameters), dimension-reducing constraints (Constraints), feasible space dimensionality (Dims), coparameters, and codimension (Codims). Parameter subsets are abbreviated are as follows: O = object, R_b = robot base, R_a = robot arm, R_h = robot hand, C_h = hand contact point, C_o = object contact point.

the object's mass and the hand-object friction coefficient. Given a desired translation and/or rotation for the object, it computes a path for the robot to follow, and an expected path for the object.

The robot must always avoid collision with itself and the table, and may only contact the surface of the object in a push mode. It must obey kinematic limits. The object may not collide with obstacles or fall off the table. We also require that the object be in front of the robot while pushing to avoid some unnatural motions, such as behind-the-back pushes.

3.4.2 Hybrid Configuration Space

A configuration q combines a robot configuration q_{robot} and an object configuration q_{obj}. The Honda humanoid's walking subsystem allows fully controllable motion in the plane, so leg joint angles can be ignored. Thus, q_{robot} consists of a planar transformation $(x_{robot}, y_{robot}, \theta_{robot})$, five joint angles for each arm, and a degree of freedom for each hand ranging from open to closed. Since the object slides on the table, q_{obj} is a planar transformation $(x_{obj}, y_{obj}, \theta_{obj})$. In all, the configuration space $\mathcal{C}$ is 18 dimensional.

The pushing task contains the following types of mode:

- *Walking.* Only the base of the robot $(x_{robot}, y_{robot}, \theta_{robot})$ moves. The arms must be raised to a "home configuration" that avoids colliding with the table while walking. The coparameters of walk modes consist of the object configuration q_{obj}.
- *Reach Left/Right.* Only a single arm and its hand may move. The arm must avoid collision with the table or object. The co-

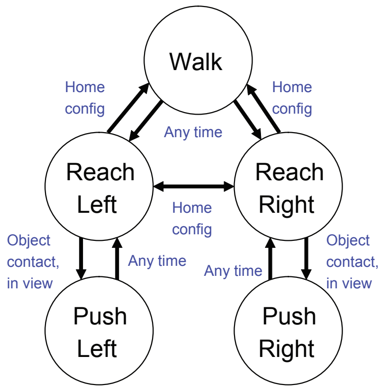


Figure 5: Diagram of permitted mode transitions.

Type	Parameters (dims)	Constraints (dims)	Dims	Coparameters	Codims
Walk→Reach	$R_b(3)$		3	O (3)	3
Reach→Walk		R_a = home	0	$R_b(3)$, $O(3)$	6
Reach→Reach		R_a = home	0	$R_b(3)$, $O(3)$	6
Reach→Push	$R_a(5)$, $C_h(2)$, C_o (2)	contact (5)	4	$R_b(3)$, $O(3)$	6
Push→Reach	$O(3)$, $R_a(5)$	contact (5)	3	$R_b(3)$, $C_h(2)$, $C_o(2)$	7

Table 2: Dimensionality of transitions. Parameter subsets are abbreviated are as follows: O = object, R_b = robot base, R_a = robot arm, R_h = robot hand, C_h = hand contact point, C_o = object contact point.

parameters consist of both q_{obj} and the robot's body location $(x_{robot}, y_{robot}, \theta_{robot})$.

- *Push Left/Right.* The hand is in contact with the object. The object moves in response to the arm motion according to push dynamics, and reciprocally, the dynamics impose constraints on arm motions (see Section 6.4.1). Additionally, the object must lie in the robot's field of view. The coparameters include the robot body location and the contact points on both the hand and the object, which must remain fixed during the push.

The parameters, dimensionalities, coparameters, and codimensionalities are summarized in Table 1.

The following mode transitions are permitted (Figure 5). Walk-to-reach and push-to-reach are allowed from any feasible configuration.

Reach-to-walk and reach-to-reach are allowed if the arms are returned to the home configuration. Reach-to-push is allowed when the hand of the moving arm contacts the object. The dimensionalities of transition regions are summarized in Table 2. In particular, note that reach-to-push transitions have lower dimensionality than the reach space, so performing a mode switch requires finding an inverse kinematics solution that places the robot hand on the object surface (see Section 6.1).

4 Multi-Modal Planning with Random-MMP

Random-MMP performs a forward tree search in the continuous mode space by iteratively switching modes sampled at random. Its operation is analogous to kinodynamic planners that grow trees through random sampling of control inputs and then checking the forward-simulated paths for feasibility [26, 21]. The difference is that here the tree is grown by sampling mode switches at random, and each mode switch requires planning a feasible single-mode path.

This section will present the basic algorithm along with generic sampling strategies that attempt to explore the mode space quickly. In Section 5 we will study the theoretical conditions under which random exploration is successful. Later in Section 6 we will describe the implementation of Random-MMP subroutines for the humanoid pushing task and problem-specific sampling strategies that speed up planning.

4.1 Description

Random-MMP maintains a search tree T of hybrid states (q, σ) and iteratively extends it by constructing local feasible paths to new states (Figure 6). The tree is grown by attempting random extensions selected by a subroutine `ExtendTree`, which attempts to plan a path from some node x_0 in T to a destination state x_k using one or more mode switches at intermediate states $x_0, x_1, \ldots, x_k$. The algorithm terminates when it finds a path from the start state x_s to an *endgame region* E, which is assumed to be a neighborhood of the goal state x_g. This neighborhood may represent a tolerance with which the goal must be reached. Or, if a control strategy is available to drive states to the goal, E may represent set of states that satisfy the preconditions. Pseudocode for Random-MMP is listed in Algorithm 1.

4.2 Extension Sampling

Good implementations of `ExtendTree` will avoid placing samples too densely in any region of state space. We describe two possible general-purpose implementations of `ExtendTree`that attempt to explore the

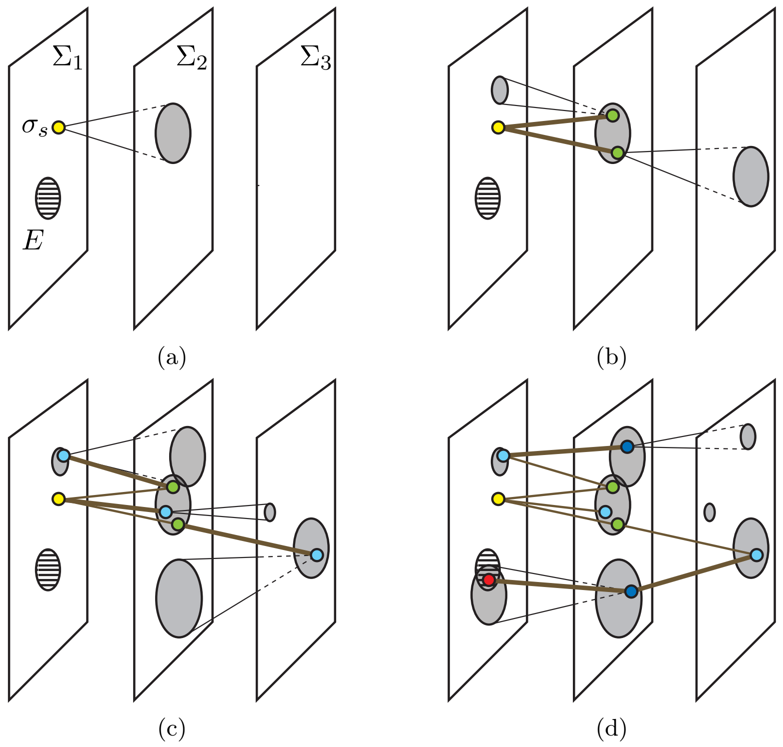


Figure 6: Illustrating the operation of Random-MMP in a mode space with three mode families of codimension 2. Shaded ovals indicate the 1-mode reachable set of the search tree. (a) The start state x_s (with mode σ_s) and the endgame region E in Σ_1. (b) Two successful mode switches. The resulting states are added to the search tree as children of x_s, and the reachable set grows. (c) Three states are added to the tree. (d) After another three iterations, Random-MMP terminates once it finds a five-mode path that reaches E.

space uniformly. The first is an "ideal" implementation that is easy to analyze, and operates much like the EST planner [21]. The second is a distance metric-based sampler similar to the RRT planner [26], which is slightly easier to implement. We use the latter in our humanoid planning system.

As described in [21], an "ideal" implementation of `ExtendTree` samples an extension of the tree *uniformly* from the set of *locally reachable* states, so that each iteration steadily grows the states covered by T. We define the locally reachable states as those that can be reached from the current search tree within k mode switches. We call such an

```
Input : $x_s$, $E$
$T \leftarrow \{x_s\}$ ;
for $n = 1, 2, \ldots$ do
    $(x_0, \ldots, x_k) \leftarrow$ ExtendTree($T$) ;
    if $x_0 \neq nil$ then
        Add the path $x_0 \rightarrow \cdots \rightarrow x_k$ as descendants of $x_0$ in $T$ ;
        if $x_i \in E$ for some $i \in 1, \ldots, k$ then return the path leading
        from $x_s$ to $x_i$;
    end
end
```

Algorithm 1: Random-MMP

extension a *k-mode extension.* To facilitate the analysis in Section 5, a value of k will be chosen such that the locally reachable states comprise a nonzero volume of $\mathcal{C}$. A practical approximation of the ideal `ExtendTree` is employed in the EST algorithm using a density estimation scheme to approximately distribute states uniformly over the reachable set, and simply using $k = 1$ [21]. A similar technique could potentially be used in the multi-modal setting.

Our humanoid robot planner uses an implementation that draws its inspiration from the RRT planner by biasing samples toward large Voronoi regions of the search tree. It operates as follows. First, a random state x is drawn at random from $\mathcal{C} \times \Sigma$. To encourage goal-seeking behavior, some fraction of the time x is chosen from the endgame E. Then, the state x_0 in T is picked that minimizes the distance to x. Several metrics might be defined to encourage exploration of useful modes, but we simply use the $\mathcal{C}$ distance metric. From x_0, we plan a 1-mode switch to reach a state x_1, and return (x_0, x_1) if successful. Although we have not been able to adapt the theoretical arguments that establish the probabilistic completeness of RRT [26] into the multi-modal setting, this implementation has been quite successful in our experiments.

4.3 Sampling a Mode Switch

To generate k-mode extensions, `ExtendTree` calls a subroutine `PlanModeSwitch` that samples a mode switch from a given state $x = (q, \sigma)$ to a state (q', σ') in an a different mode. This requires sampling an adjacent mode as well as planning a single-mode path in σ. As discussed in Section 3.2, `PlanModeSwitch` must pay special attention to the transition region, because it is a bottleneck for planning. We implement this subroutine using the pseudocode in Algorithm 2.

```
$\sigma' =$ SampleAdjacentMode($\sigma$);
$q' =$ SampleTransition($\mathcal{C}_\sigma, \mathcal{C}_{\sigma'}$);
**if** $q' \notin \mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ **then return** *nil*;
**if** PlanSingleModePath($\mathcal{F}_\sigma, q, q'$) fails, **then return** *nil*;
Otherwise, return $(q', \sigma')$;
```

Algorithm 2: PlanModeSwitch

`PlanModeSwitch` requires problem-specific implementations of the operations `SampleAdjacentMode`, `SampleTransition`, and `PlanSingleModePath`. We will assume that these operations are defined so that if a feasible mode switch exists, `SampleAdjacentMode` samples one with nonzero probability; if a feasible transition configuration exists, `SampleTransition` samples one with nonzero probability; and `PlanSingleModePath` is a probabilistically complete sample-based planner.

5 Theoretical Analysis of Random-MMP

In this section we will prove that, with the ideal sampler and under certain conditions that are expected to hold in practice, the probability that Random-MMP finds a multi-modal path approaches 1 if one exists. Furthermore the convergence rate is exponential, which is a stronger condition that implies that expected running time and variance in running time are finite [17]. There are three key properties that affect the running time and completeness of Random-MMP.

The first is a mode-to-mode reachability property (Section 5.1.1). If Σ contains structures that require several modes to be chosen carefully in succession, then a huge number of random samples will be needed to pass through those structures. This phenomenon is analogous to the "narrow passage" problem that causes difficulties for sample-based planners in standard configuration spaces. We will define reachability properties of Σ (analogous to visibility properties in standard configuration spaces) such that if reachability is favorable, a small number of modes drawn from Σ will contain a feasible path with high probability.

The second property concerns the difficulty of planning a path to achieve a mode switch (Section 5.1.2). Say we are given some time limit T_{max} for each single-mode planning query, after which the mode switch is declared a failure. If the goal is reachable with only "easy" mode switches where queries succeed with high probability, then Random-MMP will find a path quickly. If, on the other hand, some "hard" mode switches are required, then Random-MMP may fail to find a path. If so, either more iterations of Random-MMP are needed, or T_{max} must be increased to have a better chance of solving those hard queries.

The third property is the number of single-mode feasible spaces that are disconnected (Section 5.2). Multi-modal planners must account for the possibility of disconnections if they are to solve problems reliably [17]. We show that Random-MMP tolerates disconnections, but running time may be affected if solution paths travel through many subsequent disconnected spaces.

5.1 Random Trees With Connected Feasible Spaces

In this section we perform a preliminary analysis that assumes all feasible spaces $\mathcal{F}_\sigma$ are singly-connected. `PlanModeSwitch` will succeed starting from any $q \in \mathcal{F}_\sigma$ if and only if the transition $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ is nonempty. So, the feasibility of a mode switch *is a function of the modes only.* The start and goal configuration are connected by a feasible path if and only if there exists a sequence of modes, with feasible transitions in between, that connect the start mode to the goal mode. This means that configuration is essentially unimportant to the analysis and allows us to focus on the properties of Σ.

We analyze Random-MMP as a function of the reachability properties of Σ. We need a few preliminary definitions to specify these properties. Our definitions largely follow those for sample-based planners in configuration spaces [21], except we treat Σ as the space in question. First, let μ be a measure on Σ that assigns nonzero measure to each mode family in Σ (no matter the codimensionality). We require that $\mu(\Sigma)$ is finite. Define, for every subset $X \subseteq \Sigma$, the reachable set $\mathcal{V}(X)$ as the set of modes $y \notin X$ such that y is reachable from some $x \in X$. We will interpret the `ExtendTree` subroutine as 1) drawing an endpoint y uniformly using the sampling measure μ, and then 2) calling a *local planner* $L(x, y)$ to construct a path from a node x in the tree to y. If $L(x, y)$ succeeds we say y is *L-reachable* from x, and similarly define the L-reachable set $\mathcal{V}_L(X)$.

The critical property required of Σ is known as *expansiveness.* It states, roughly, that given any set of modes, there is a significant probability of expanding the L-reachable set by sampling a mode in the L-reachable set. More precisely, Σ is (α, β, L)-expansive if there exist constants α and β such that for *any* subset A of Σ, at least an α fraction of A can "see" at least a β fraction of $\mathcal{V}(A)$ using the local planner L. To satisfy this condition, every mode must be able to be connected to at least an α fraction of its reachable set using the local planner.

Theorem 1. *If:*

- *The volume $\gamma = \mu(E \cap \mathcal{V}(x_s))$ of the reachable portion of the endgame region satisfies $\gamma > 0$,*
- *The endpoints of extensions sampled by* `ExtendTree` *are uniformly drawn from $\mathcal{V}_L(T)$, and the local planner L is determin-*

istic,

- Σ *is* (α, β, L)*-expansive,*

then after n *iterations, the probability that Random-MMP fails to find a path is no more than* $c \exp(-dn)$*, for some positive constants* c *and* d *that depend on* α*,* β*, and* γ*.*

Now we relax the assumption that the local planner is complete. For example, if a feasible space has high dimensionality (for practical purposes, greater than 3 or 4), then `PlanModeSwitch` must be implemented using sample-based planners that are cutoff after a finite time limit T_{max}, and so the local planner may fail with some probability. For "hard" mode switches, `PlanModeSwitch` may have a only a small chance of success, and for even harder ones, this chance may be zero. The next theorem states that if the mode space is still expansive when hard mode switches are excluded, then Random-MMP is probabilistically complete. Because each extension succeeds only with some probability, $L(x, y)$ can be considered a stochastic variable.

Theorem 2. *Suppose* $L(x, y)$ *is stochastic and succeeds with probability* $p_L(x, y)$*. Define, for a constant* $p > 0$*, a hypothetical deterministic local planner* $L_p(x, y)$ *that returns 1 when* $p_L(x, y) \geq p$ *and 0 otherwise. If:*

- $\gamma = \mu(E \cap \mathcal{V}(x_s)) > 0$,
- *The endpoints of extensions sampled by* `ExtendTree` *are uniformly drawn from* $\mathcal{V}_L(T)$*,*
- Σ *is* (α', β', L_p)*-expansive,*

then the probability that Random-MMP fails to find a path by n *iterations is no more than* $(1 + c) \exp(-n \min(dp/2, p^2/2))$*, where* c *and* d *are the same functions of* α'*,* β'*, and* γ *as in Theorem 1.*

The proofs are given in the Appendix.

In general there exists a family of bounds that trade-off between increasing p and reducing α' and β', so the running time of Random-MMP will be bounded by the best of those exponentially convergent bounds. In problems with "easy" mode switches, Random-MMP converges faster because p can be raised, and expansiveness will still hold for some α' and β'. In problems with harder mode switches, p cannot be raised without reducing α' and β', so Random-MMP will be slower.

It is important to note that the running time depends jointly on properties of the multi-modal space, the mode-space sampling measure, and the local planner. This suggests that the two best ways of improving the performance of Random-MMP are improving the success rate of local planners, and placing modes more densely where they are needed.

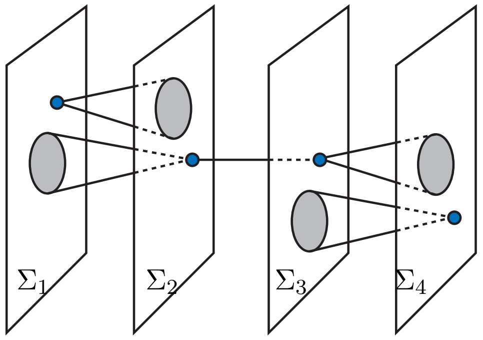


Figure 7: A mode space that is k-mode good, but is nonexpansive. The reachable set of every mode is the entire space, and every mode has a 1-mode reachable set with nonzero volume, but mode families 2 and 3 are connected only at single mode.

5.1.1 Expansiveness and k-Mode Goodness

The intuition behind the expansiveness condition is that it requires `ExtendTree` to have a significant probability of expanding the locally reachable set by a significant volume. To be expansive, a mode space must also possess a *k-mode goodness* property, which states that a significantly large volume of Σ can be reached within k mode switches starting at (almost) all modes. Aside from the connectivity technicalities that are captured by the expansiveness condition, k-mode goodness is the most essential property for a random sampling to discretize Σ well.

Definition. Consider the set of modes $R_k(\sigma)$ that can be reached using no more than k mode switches starting from σ. A mode space possesses *k-mode goodness* if $\mu(R_k(\sigma)) > 0$ for all σ such that $R_k(\sigma)$ is nonempty, except for possibly a set of measure zero.

In most problems we have studied, k-mode expansiveness holds for some $k > 1$. The $k = 1$ case rarely holds because the 1-mode reachable sets of most modes are of lower dimension and thus have volume zero. For example, recall the toy examples of Section 3. The example with n movable objects is $n - 1$-mode expansive because a significant volume of modes can be reached by moving each of the $n-1$ other objects. The mobile robot pushing example is 4-mode expansive because the barrel must be pushed twice to reach a significant subset of its configuration space, which requires two transit modes and two transfer modes. The legged locomotion example is $2n$-mode expansive because each foot must be lifted and placed. The humanoid pushing task is 4-mode expansive, because any mode can reach a significant volume of modes using the sequence reach, push, reach, walk.

Expansiveness is actually a stronger condition than k-mode goodness, because it is possible to construct problems that are k-mode good but have infinitesimally narrow passages between certain regions in mode space, which make them non-expansive (Figure 7). Our intuition is that such pathological structures are extremely rare in practice.

5.1.2 Examining Local Planner Failure

Consider when `PlanModeSwitch` fails (where a feasible path actually exists) with probability approaching 1. Given the assumptions in Section 4.3, this only occurs when not enough time was given to the sample-based planner `PlanSingleModePath`. Although `PlanSingleModePath` is assumed to be probabilistically complete, if the time cutoff is too low, and the planning query is sufficiently "hard", then the probability of failure may be 1.

In our experience in household manipulation problems and legged locomotion in rough terrain, single-mode planning queries are relatively easy, and finding a suitable time cutoff is typically a matter of parameter tuning on a few benchmark queries. In nonadversarial problems, such as those in household environments, hard single-mode queries rarely appear or can usually be circumvented by a better choice of mode.

But this is not an entirely satisfactory solution; new queries may be harder than expected, leading to planner failures, or easier than expected, leading the planner to waste time on infeasible queries. One improvement might use the adaptive strategy of [30], which interleaves multiple planning queries without setting a priori time cutoffs. In prior work we adapted this technique to multi-modal planning in a discrete mode space [17], and we are currently studying how Random-MMP might balance the discretization of continuous mode spaces with adaptive single-mode planning time in a principled fashion.

5.2 Random Trees with Disconnected Feasible Spaces

Now consider returning to the original problem where each feasible space may be disconnected, which requires our analysis to return the hybrid configuration/mode space setting. We will show that randomness in Random-MMP allows it to maintain probabilistically completeness, except when problems contain pathological patterns of disconnections that are very unlikely to occur naturally. Disconnections will, however, increase running time.

Disconnections introduce three new conditions in which a given k-mode extension may fail. The first two do not strongly affect prior completeness arguments, but the third requires a more nuanced analysis:

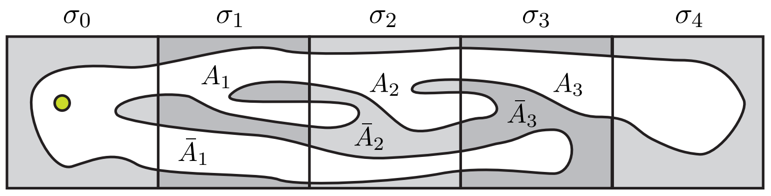


Figure 8: In a series of subsequent disconnected feasible spaces, Random-MMP must be lucky several times over to find transition configurations in the correct components A_1, A_2, and A_3.

- `PlanModeSwitch` fails if the feasible space is disconnected and the destination mode cannot be reached from the start configuration. If so, fewer mode switches are allowed than if all modes were connected, which affects expansiveness and running time.
- A mode switch is possible, but `PlanModeSwitch` samples a transition in the wrong connected component, causing `PlanSingleModePath` to fail. So, the value of $p_L(x, x')$ used in the derivation of Theorem 2 will be lower than if the feasible space were connected, but will still be nonzero as long as every transition connected component can be sampled with at least a nonzero probability.
- To reach a certain set of reachable modes, Random-MMP must sample several subsequent extensions precisely in the right connected components.

The third condition is interesting because it is possible for the planner to have arrived in the wrong connected component but not know it, because connectivity cannot be easily tested. This rather subtle point affects the sampling distribution of subsequent extensions in a nontrivial way, such that it is less likely to reach the correct component, and even less likely to reach the correct next component, and so on. We analyze this case here.

Consider the possibility of encountering a sequence of C disconnected modes, where each consecutive extension must be in the correct connected component (we will call this a *disconnection of order* C). Figure 8 depicts an example where C=3. We will show that rate at which Random-MMP generates extensions to the final mode is proportional to r^C, where r measures the probability of generating an extension to the correct connected component. Intuitively, it is unlikely for disconnections of high order to exist in realistic problems, although in pathological problems C can grow arbitrarily large.

Let a disconnection of order C denote a problem structure as follows. Let some mode set M_g be reachable from a configuration x_0 in the search tree, via a sequence of C subsequent k-mode extensions between mode sets $M_1, \ldots, M_C, M_g$. Let the feasible spaces of each mode set M_k, $k = 1, \ldots, C$ be divided into two sets of hybrid states, A_k and $\bar{A}_k$, such that

1. A_0 and $\bar{A}_0$ are k-mode reachable from x_0,
2. Every state in A_K is k-mode reachable from A_{K-1} but not $\bar{A}_{K-1}$,
3. M_g is reachable from A_C but not $\bar{A}_C$.

That is, to reach M_g, Random-MMP must make a sequence of extensions among A_k, but the sets $\bar{A}_k$ lead to "dead ends". The key difficulty is that Random-MMP can detect that it has reached the mode set M_k, but not which subset A_k or $\bar{A}_k$. So, once it has reached M_k, it may attempt an extension to M_{k+1}, but may fail because the extensions will be sourced from states in $\bar{A}_k$. The probabilities that this occur can be computed by studying a Markov chain, and are summarized below.

Let p denote the probability of expanding a state from any given M_k. Let r, s, and t denote, respectively, the probability of generating a sample from A_k to A_{k+1}, from A_k to $\bar{A}_{k+1}$, and from $\bar{A}_k$ to $\bar{A}_{k+1}$. As the number of iterations grows large, the expected fraction of states in M_k that are located in A_k is $r^k/(s+r^{k-1}(r+t-s))$. Using an analysis of the convergence rates of irreversible, absorbing Markov chains [32], it can be shown that the asymptotic rate of generating a feasible extension to M_C is bounded by the rate of sampling a feasible extension from A_C. This rate is $pr^{C+1}/(s + r^C(r + t - s))$, and so the probability of success by iteration n is bounded by a function proportional to $exp(-npr^{C+1}/(r + t))$. This bound shrinks up to $1/r^C$ times slower than the bound obtained when all feasible spaces are connected. So, as long as there is a finite bound C_{max} where all disconnections of order C that occur in the problem satisfy $C \leq C_{max}$, Random-MMP will explore all disconnections of order C with probability (exponentially) approaching 1.

6 Implementation of Planner Subroutines for the Humanoid Manipulation Task

The Random-MMP subroutine `PlanModeSwitch` requires that the operations `SampleAdjacentMode`, `SampleTransition`, and `PlanSingleModePath` be implemented for each mode. We use standard techniques. On average, mode switch takes a small but not negligible amount of time to plan (typically between 10 and 100 ms). Running time is dominated by collision checking during single-mode planning queries.

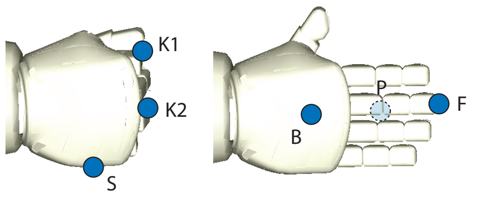


Figure 9: The planner chooses from three candidate closed-hand contact points side(S), and knuckles (K1 and K2); and three candidate open-hand contact points: fingers (F), palm (P), back (B).

6.1 Sampling Mode Transitions

Given a configuration q at mode σ, Random-MMP must be able to sample a transition configuration q' at an adjacent mode σ'. We first choose an adjacent mode family, then sample q' to achieve that type of transition. We sample q' as follows:

- *Walk-to-reach.* Sample a body position between a minimum and maximum distance from the object and a body orientation such that the object lies within the robot's field of view.
- *Reach-to-walk.* Move the arm to the home configuration.
- *Reach-to-reach.* Move the arm to the home configuration.
- *Reach-to-push.* We predefine a number of contact points C_{cand} on the surface of the hand that are candidates for stable pushes (Figure 9). Sample a point p_{hand} from C_{cand} and a point p_{obj} on the surface of the object, with normals n_{hand} and n_{obj}. Starting from a random arm configuration, use numerical IK to simultaneously place p_{hand} at p_{obj} and orient n_{hand} to $-n_{obj}$.
- *Push-to-reach.* A push-to-reach transition can be taken at any time. So we choose a transition implicitly by planning a single-mode path y, and set q' to the endpoint of y.

If q' is feasible, Random-MMP plans a single-mode path y to connect q and q'. These single-mode planning subroutines are described in the following sections.

6.2 Planning in Walk Modes

While walking, the robot can be treated as a mobile robot in the 3D space $(x_{robot}, y_{robot}, \theta_{robot})$. A variety of path planning methods may be used, such as A* footstep search [11].

6.3 Planning in Reach Modes

The configuration space of reach modes is 6D, requiring the use of sample-based planners. We use a variant called SBL [36] that randomly grows two trees of feasible milestones bidirectionally, rooted at the start and the goal configurations. It achieves major speed gains by delaying feasibility tests for straight-line paths, until it believes it has found a sequence of milestones connecting the start to the goal. In our implementation we impose a time cutoff of 0.5 seconds, which is adequate to solve point-to-point reaching tasks in moderately cluttered environments.

Like other sample-based planners, the raw output of SBL is a piecewise-linear path where the arm must stop at each vertex. We speed up execution using a postprocessing step that smooths the path into a piecewise-quadratic trajectory. We use a variant of a common shortcutting procedure that repeatedly samples two points along the trajectory, and replaces the intermediate subpath with a straight line segment if the segment is collision free [14]. Our variant, presented in [18], constructs smooth curves rather than straight lines as shortcuts. Experiments show this technique reduces execution time by approximately 45%.

6.4 Planning in Push Modes

We use a forward-simulation planner as follows. Let p_{hand} be a contact on the hand touching a point p_{obj} on the object, with normals n_{hand} and n_{obj}. First, sample a stable center of rotation (COR) c (Section 6.4.1). Rotating the object by distance D about this COR defines an object trajectory, which is checked for collisions. Then, we ensure that the object trajectory can be executed with an arm trajectory that positions p_{hand} at p_{obj} and orients n_{hand} to $-n_{obj}$ along the entire trajectory. This requires solving an inverse kinematics (IK) problem (Section 6.4.2). Finally, collisions are checked along the object and arm trajectory. We discretize the object trajectory in tiny increments (5 mm in our implementation), discretize the arm trajectory with the same increments, and then run the collision checker at each configuration. If the path is feasible, the process is repeated to push the object further.

6.4.1 Stable Push Dynamics

A sliding object pushed with a point contact does not move deterministically, because the friction forces at the support surface are indeterminate [28, 34]. TheHonda humanoid robot hardware currently does not have the visual or tactile feedback needed for stable point-pushing. Thus, we restrict the robot to use multi-contact pushes that rotate the object predictably under open-loop control. These *stable pushes* must be applied with at least two simultaneous collinear contacts, such as flat areas on the robot's hand. Lynch and Mason [27] studied the conditions necessary for stable pushing; these will be summarized here.

Assume the object moves quasistatically, i.e. the object's velocity is low enough such that dynamic effects are negligible, and it stops sliding immediately after contact forces are removed. Given a known center of friction, surface friction, and contact points, one can use the the process described in [27] to calculate the set of centers of rotation c such that a rigid transformations of the hand about c will yield a stable push. In other words, if we choose an appropriate motion of the hand about c, the object will move as though it were rigidly fixed to the hand, along a helical configuration space trajectory about axis c.

6.4.2 Inverse Kinematics

Given specified contact points and the object configuration q_{obj}, the inverse kinematics subroutine solves for the five arm joint angles q_{arm} to maintain contact between the hand and the object. If there is no solution, the subroutine returns failure.

Denote the contact point and normal in the hand-local frame as p^L_{hand} and n^L_{hand}. Let R_{hand} and t_{hand} denote, respectively, the hand frame's rotation and translation as determined by forward kinematics, such that the world-space coordinates of the hand contact and normal are $p_{hand} = R_{hand}(q_{arm})p^L_{hand} + t_{hand}(q_{arm})$ and $n_{hand} = R_{hand}(q_{arm})n^L_{hand}$. Denote the contact point and normal on the object as p_{obj} and n_{obj}, respectively. Then, the inverse kinematic subroutine must solve for q_{arm} that simultaneously satisfy the six nonlinear equations:

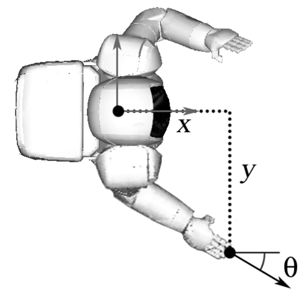
$$R_{hand}(q_{arm})p^L_{hand} + t_{hand}(q_{arm}) = p_{obj} \tag{3}$$

$$R_{hand}(q_{arm})n^L_{hand} = -n_{obj}. \tag{4}$$

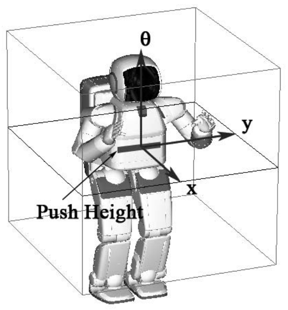
Or more compactly, $C(q_{arm}) = 0$. At first glance this seems like an overdetermined system (5 parameters, 6 constraints). However, the normal-matching constraint is degenerate, and the Jacobian of C is therefore at most rank 5. We solve this system of equations using a Newton-Raphson numerical solver [33], which iteratively moves a start configuration q_0 onto the solution set using the Jacobian pseudoinverse.

7 An Informed Expansion Strategy using Prior Knowledge

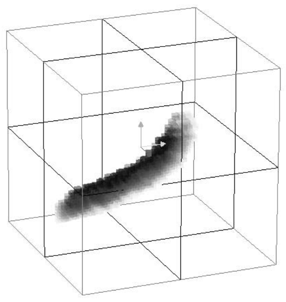
As described Random-MMP is probabilistically complete, but its performance can be improved using an *informed* expansion strategy that



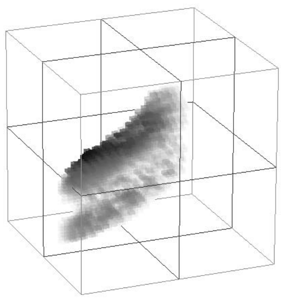
(a) Workspace coordinates of the right fingers.



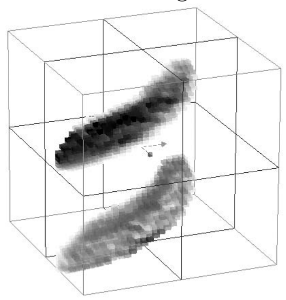
(b) Reference frame F.



(c) Utility table for the right fingers.



(d) Utility table for the right knuckles.



(e) Utility table for the right palm.

Figure 10: Utility tables (c–e) are drawn in frame F (b). Reachable cells are drawn with darkness increasing with utility.

is biased to pick high-utility expansions that help it explore the hybrid configuration/mode space quickly. We provide prior knowledge to the algorithm in the form of precomputed *utility tables*, which assess the expected distance the object can be pushed from a given location. These tables can then be used in `ExtendTree` in a form of importance sampling that favors high-utility robot configurations and mode switches. Experiments in Section 7.3 show that this improves the speed at which `ExtendTree` explores the space an order of magnitude. Furthermore, we introduce an additional parameter q_{des} that is used in the RRT-like expansion heuristic (Section 4) to guide the distribution of configurations in T, for example, toward the goal or toward unexplored regions in configuration space. Our planner achieves large speedups with these heuristics.

7.1 Precomputed Utility Tables

In reach-to-push sampling, only a small portion of S_{obj} is reachable from a given point on the hand. For each p in C_{cand}, we precompute one table that allows us to quickly compute the reachable region R on S_{obj} during planning, and another table that we use to estimate the utility of points in R.

When pushing, the normal n at p must be horizontal in world space. We fix a height of pushing h, constraining the vertical coordinate of p. This define a 3D workspace $\mathcal{W}$ of points (x, y, θ), where (x, y) are the horizontal coordinates of p and θ is the orientation of n, relative to the robot's frame (Figure 10a). We precompute two grids, `Reachable` and `Utility`, over $\mathcal{W}$ as follows.

`Reachable` stores 1 if the contact is reachable and 0 otherwise (Figure 10). We initialize `Reachable` to 0, and then sample the 5D space of the arm joints in a grid in $10°$ increments along each axis. Starting at each sampled configuration, we run IK to bring the height of p to h and reorient n to be horizontal. If successful, we check if the arm avoids collision with the body and the point p is in the robot's field of view. If so, we mark `Reachable`$[(x, y, \theta)]$ with 1, where (x, y, θ) are the workspace coordinates of p and $[\cdot]$ denotes grid indexing. This preprocessing step takes approximately 20-30 minutes.

`Utility` stores the expected distance the contact can be pushed in the absence of obstacles, calculated by Monte Carlo integration through `Reachable` (Figure 10). In $\mathcal{W}$, a push traces out a helix that rotates around some COR. We assume a prior probability distribution P over stable CORs for a reasonable range of physical parameters of the object. Starting from $w_0 = (x, y, \theta)$ we generate a path $w_0, w1, \ldots, w_K$. For all k, w_{k+1} is computed by rotating w_k a short distance along some COR sampled from P. The sequence terminates when `Reachable`$[w_{K+1}]$ becomes 0. After generating N paths, we record the average length in `Utility`$[(x, y, \theta)]$.

Given robot and object positions, contacts along S_{obj} at height h form a set of curves B in $\mathcal{W}$. By intersecting B with the marked cells of `Reachable`, we can quickly compute the set of reachable object edges R. This set is useful because we can reject infeasible reach-to-push transitions by testing whether R is empty, and furthermore we can filter out many unreachable contact points by drawing candidate contact points from R rather than the entirety of B. Some contact points in R may still be unreachable due to obstacles in the environment, and further some may not lead to fruitful pushes.

Our first attempt to use this prior knowledge was in a *utility-based importance sampling* strategy. When choosing a reach-to-push transition, this strategy picks contacts from R with probability proportional to `Utility`. Other transitions were kept the same as in the blind

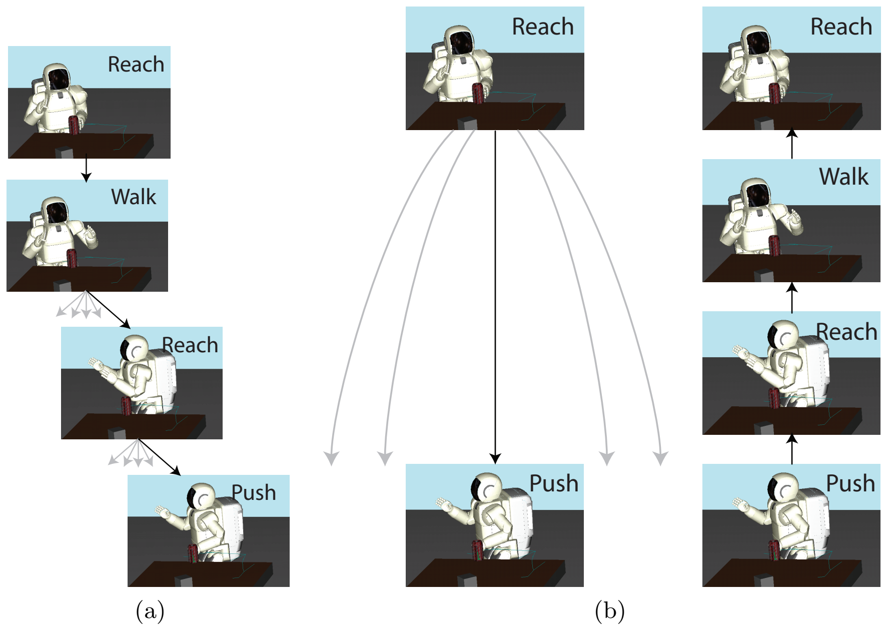


Figure 11: (a) To achieve a push mode, a forward search must branch on walk-to-reach and reach-to-push transitions. (b) The utility-centered expansion strategy first attempts to sample a reach-to-push transition (middle), then connects the transition backwards to the initial mode (right). The backward connections require no branching.

strategy. With this simple change, we observed a dramatic improvement in the fraction of feasible pushes chosen by the planner. But, we found even greater benefits using the strategy described below, which uses the utility tables to improve the success rates of the walk-to-push transition as well.

7.2 Utility-Centered Expansion Strategy

Utility-centered expansion explicitly stages the body position in a position that is ready to execute a high-utility push (Figure 11). Given a start configuration q at configuration σ, and a target object configuration q_{des}, this strategy 1) chooses a robot's body and arm configuration and a high-utility push for a reach-to-push transition q_{push}, 2) plans a sequence of three modes backwards from q_{push} to q (which requires no branching), and 3) plans a push path forward from q_{push}.

We elaborate on step 1. We first choose a point p_{obj} on S_{obj} (at height h and normal n_{obj}) and a stable push such that the object will

Strategy	Walk-to-reach	Reach-to-push	Pushes > 1cm	Overall
Blind	0.63	0.20	0.29	0.037
Utility importance	0.59	0.33	0.79	0.16
Utility-centered	1	0.47	0.66	0.31

Table 3: Fraction of feasible walk-to-reach transitions, reach-to-push transitions, acceptable pushes, and walk-reach-push cycles.

Strategy	Modes / push	Time / push (s)	Average push (cm)	Push rate (m/s)	Tgt. seek rate (m/s)
Blind	10.0	0.956	6.7	0.070	0.0302
Utility importance	5.99	**0.325**	8.2	0.254	0.111
Utility-centered	**5.08**	0.404	**13.3**	**0.329**	**0.257**

Table 4: Expansion strategy timing experiments. Bold indicates best in column.

be pushed toward q_{des}. Next, we sample a contact p_{hand} from C_{cand} and a workspace coordinate $(x_{hand}, y_{hand}, \theta_{hand})$ with probability proportional to `Utility`. Then, we compute the body coordinates that transform (x_{hand}, y_{hand}) to p_{obj} and rotate θ_{hand} to θ_{obj}, where θ_{obj} is the orientation of $-n_{obj}$. Repeat until the body position is collision free. Fixing the body, we sample the arm configuration of q_{push}, using IK to position p_{hand} to p_{obj} and orient n_{hand} to $-n_{obj}$.

7.3 Comparing Expansion Strategies

In a series of experiments we compared the three expansion strategies, blind, utility-based importance sampling, and utility-centered expansion. The utility-centered expansion has a better success rate, pushes the object farther, and has the useful ability to bias the direction of pushing toward a desired configuration.

First, we measure the fraction of feasible mode transitions attempted by each strategy. We ran the three strategies starting from a configuration similar to the first frame of Figure 1. Each strategy is initialized with a walk mode, and terminates after a single push has been executed (requiring three transitions, from walk to reach to push). For the utility-centered strategy, we picked random target positions q_{des}. We reject any "insignificant" pushes, defined as moving the object less than

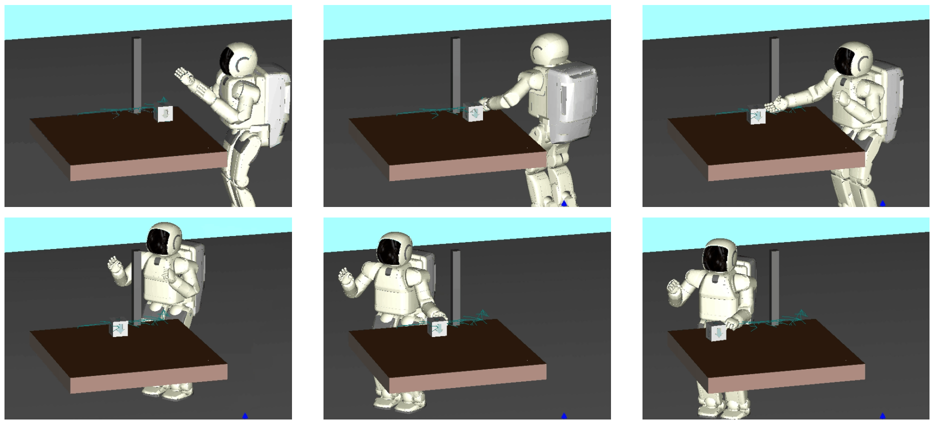


Figure 12: Pushing a block while avoiding a vertical obstacle.

1cm. The results, averaged over 1,000 runs, are shown in Table 7.3. The blind strategy produces very few significant pushes. Utility-based importance sampling does much better, but reach-to-push transitions are still a bottleneck. Utility-centered expansion improves both the fraction of feasible walk-to-reach and reach-to-push transitions.

Next, Table 4 measures the expansion time and rate, and highlights the distance the object is pushed per unit of planning time. The first column measures the number of modes explored before terminating, and the second measures the time elapsed. The third and fourth columns measure the distance of the terminal push, and the distance divided by computation time. The final column measures the distance (if positive) the object was pushed toward q_{des}. So, besides improving the transition success rate, utility-centered expansion can be focus pushing more effectively by almost a factor of two.

We also tested Random-MMP in several simulated pushing tasks amongst cluttered tables. In all examples, the user specified a task to move the block within 5cm of a target position, with arbitrary orientation. Figure 12 shows the solution to a moderately difficult problem, where the robot must carefully choose a series of pushes to avoid colliding with the vertical obstacle. The relatively long-distance push chosen in frame 3 enables the robot to continue pushing the object in frame 5. Using the utility-centered expansion strategy, this trajectory was found in approximately four minutes on a 2GHz PC. The blind strategy did not find a path after 30 minutes.

8 Experiments on the Honda Humanoid Robot

This section describes the integration of Random-MMP into a system that enables the real Honda humanoid robot to push an object on a table, given knowledge of the geometry (but not pose) of the object and table. The system uses the robot's native sensing and actuation hardware, but image processing and planning are performed off-board. The robot's vision system is designed in such a way that prevents it from observing the motion of the object as it pushes it, so it must operate in a "quasi-sensorless" fashion that executes pushes in open-loop, and can only resense the object and table by stepping back from the table by approximately 1.5 m.

8.1 Visual Sensing and Pose Uncertainty

The object and table are assumed to have known geometry but unknown pose. We estimate the object and table pose using an open-source augmented reality system (ARToolKitPlus [40]) that makes use of fiducial markers of known size. Markers are affixed the top of the object, and the center and corners of the table. The redundant markers on the table both improve accuracy and help estimate pose even when some markers are out of view or occluded. Experiments show that at a distance of 1.5m, errors in the marker pose estimation are bounded by approximately 2cm in translation and $4°$ in orientation. To help avoid colliding with the object and table under this uncertainty, we slightly grow the geometry of the object and table used for collision detection before planning.

8.2 Execution of Walking Trajectories

In practice, the footsteps produced by the walk planner are not achieved exactly by the robot's low-level footstep controller, because the controller compensates for unmodeled perturbations (e.g., variations in the ground surface, joint friction, residual motion in the arms, etc.). We employ a high-level walk controller to monitor the execution of the path, and make local corrections to ensure the robot does not stray off course. The robot's odometry measures the leg joint angles at foot touch-down to estimate the actual footstep taken. The walk controller uses this estimate to determine a new footstep command to steer the robot back toward the planned path.

In practice, the odometry's estimate of each footstep has errors on the order of a few millimeters, causing the estimated robot pose to drift. Though drift did not significantly performance up to a few

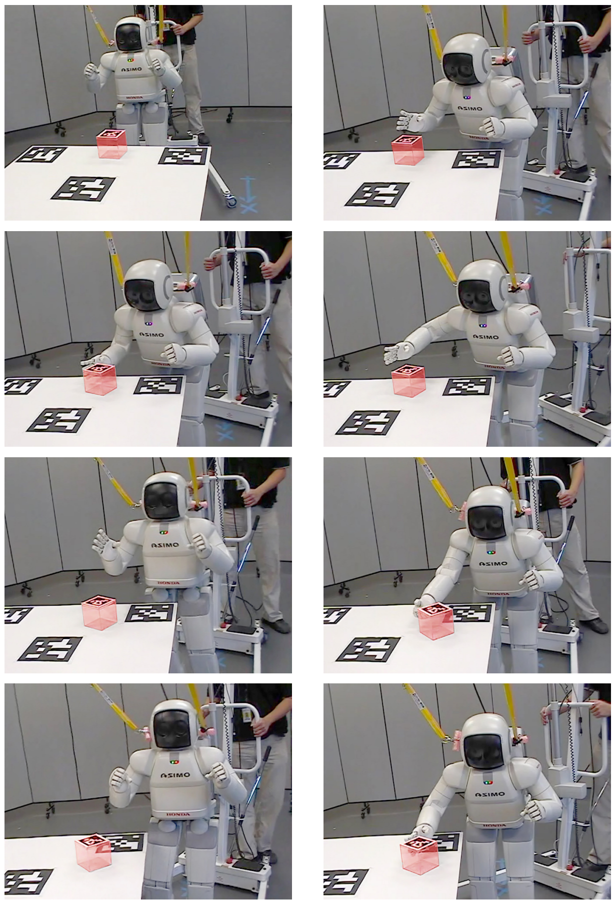


Figure 13: An execution of a series of pushes with an additional mode where the robot bends at the waist.

pushes (approximately 4–6), periodic visual localization is necessary for longer-term executions.

8.3 Experiments

Figure 1 shows a sequence of pushes executed on the Honda humanoid, which was planned in about one minute on an off-board PC. Environment sensing is performed once at the beginning of execution. No additional visual feedback is performed while the planned path is executed. Despite errors in initial sensing and in the robot's odometry, it executes several pushes successfully. In our experiments, 4–6 pushes typically can be executed before the object deviates too much from the planned path (approximately 2–3cm), and resensing is needed.

9 Conclusion and Future Work

This paper presented the Random-MMP algorithm for general-purpose multi-modal motion planning in high-dimensional spaces with continuous sets of modes. It uses random sampling both for selecting a discrete set of modes and for planning single-mode motions, and we present a detailed theoretical analysis of the conditions under which it reliably finds a path. We applied Random-MMP to the problem of enabling the Honda humanoid robot to push an object to a precise location on a table. In experiments both in simulation on the real robot, we used Random-MMP to plan complex sequences of pushes amongst cluttered obstacles in a few minutes on a 2Ghz PC.

As a sampling-based approach, the performance of Random-MMP is determined largely by the choice of sampling strategy. We present a sampling strategy that used prior information to speed up the planning of short local paths comprised of a few mode switches, coupled with longer-term Voronoi exploration biases inspired by the RRT planner. It is still an open question how to best incorporate prior knowledge into the sampling strategy in order to speed up long-term planning.

Additional manipulation modes like grasping, multi-handed pushing, and pushing while walking, can be integrated into the humanoid pushing system with modest effort. In about a week, we were able to integrate an additional mode, where the robot pushes while bending at the waist (Figure 13). In future work, general task-and-motion description languages will facilitate the specification of multi-modal problems, and the work of [7] is a significant step in this direction. A final avenue for future work is integrating multi-modal planning with hybrid control to construct robust plans. We are optimistic that progress in these areas may enable breakthroughs in challenging applications like full-body object manipulation, dexterous manipulation with tools, and automation of surgical tasks.

9.1 Acknowledgment

The authors thank Jingru Luo and the anonymous reviewers for their insightful feedback on this manuscript. Hector Gonzalez-Baños was instrumental in shaping the early stages of this project at Honda Research Institute.

A Appendix

This Appendix proves Theorems 1 and 2 as stated in Section 5.1. We require the preliminary definition of expansiveness.

Definition 1. *For any set $X \in \Sigma$ and constant $\beta > 0$, let (β, L)-Lookout(X) be the subset of modes*

$$(\beta, L)\text{-}Lookout(X) = \{x \in X | \mu(\mathcal{V}_L(x) \setminus X) \geq \beta\mu(\mathcal{V}(X))\} \tag{5}$$

Roughly speaking, a lookout is the subset of a set X that can reach a significant fraction of the reachable set of X.

Definition 2. *Σ is (α, β, L)-expansive if for any $S \subseteq \Sigma$, there exist positive constants α, β such that $\mu((\beta, L)\text{-}Lookout(S)) \geq \alpha\mu(S)$.*

The two theorems will also make use of the elementary fact that if events Y_1 and Y_2 imply X, then

$$Pr(\neg X) \leq Pr(\neg Y_1) + Pr(\neg Y_2|Y_1)Pr(Y_1) \leq Pr(\neg Y_1) + Pr(\neg Y_2|Y_1). \tag{6}$$

So, if $Pr(\neg Y_1) \leq c_1 e^{-d_1 n}$ and $Pr(\neg Y_2|Y_1) \leq c_2 e^{-d_2 n}$, then

$$Pr(\neg X) \leq (c_1 + c_2)e^{-n \min(d_1, d_2)}. \tag{7}$$

Claim 1. *Theorem 1 holds.*

Proof. The proof is a straightforward adaptation of the argument in [21]. Consider the L-reachable set of the search tree at step n, $\mathcal{V}_L(\mathcal{T}_k)$. We will examine how the volume of the lookout set $\mu_n \equiv \mu(\mathcal{V}_L(\mathcal{T}_k))$ converges toward the volume of the reachable set, $\mu(\mathcal{V}(x_s))$, as a function of n. For simplicity, assume the measure is normalized such that $\mu(\mathcal{V}(x_s)) = 1$

By assumption, the destination y of a random extension is drawn randomly from $\mathcal{V}_L(\mathcal{T}_n)$. By the expansiveness condition, there is at least a α probability of y lying in the lookout set (β, L)-Lookout$(\mathcal{V}_L(\mathcal{T}_n))$.

If y lies in the lookout set, then the new tree $\mathcal{T}_{n+1}$ produced by adding the extension has a significantly larger L-reachable set. Specifically,

$$\mathcal{V}_L(\mathcal{T}_{n+1}) = \mathcal{V}_L(\mathcal{T}_n) \cup \mathcal{V}_L(y), \tag{8}$$

so,

$$\mu_{n+1} = \mu_n + \mu(\mathcal{V}_L(x') \setminus \mathcal{V}_L(\mathcal{T}_n)) \geq \mu_n + \beta\mu(\mathcal{V}(\mathcal{V}_L(\mathcal{T}_n))) \tag{9}$$

Since $\mathcal{V}(\mathcal{V}_L(\mathcal{T}_n)) = \mathcal{V}(x_s) \setminus \mathcal{V}_L(\mathcal{T}_n)$, we have

$$\mu_{n+1} \geq \mu_n + \beta(1 - \mu_n). \tag{10}$$

If, on the other hand, y does not lie in the lookout set, we trivially have $\mu_{n+1} \geq \mu_n$.

If after n iterations of Random-MMP, k expansions have lied in the lookout set, then

$$\mu_n \geq \sum_{i=0}^{k-1} (1-\beta)^i \beta = 1 - (1-\beta)^k \tag{11}$$

Now consider the event that k is high enough such that the L-reachable set of $\mathcal{T}_n$ overlaps the endgame region by volume $\gamma/2$ (or some other significant fraction). Using some algebraic manipulation of (11), we see that this is guaranteed to occur when $k \geq \ln(\gamma/2)/\ln(1-\beta)$. Let $k^\star = \lceil \ln(\gamma/2)/\ln(1-\beta) \rceil$ be the number of expansions needed to satisfy the condition. Once $k^\star$ of these expansions occur, then each subsequent iteration of Random-MMP samples an expansion into the endgame region with probability at least $\gamma/2$.

Finally, consider the event that Random-MMP succeeds after n iterations. Success is implied if (A) the first $n/2$ iterations of Random-MMP draws at least $k^\star$ lookout expansions, and then (B) the last $n/2$ iterations successfully samples an expansion into the endgame region.

Because each lookout expansion occurs with probability at least α, Hoeffding's inequality [20] shows that the probability that A is false is no more than

$$Pr(\neg A) \leq \exp(-2\frac{(\alpha(n/2) - k^\star)^2}{n/2}) = \exp(-\frac{(\alpha n - 2k^\star)^2}{n}). \tag{12}$$

Once A is true, then the probability of reaching the endgame region is a coin flip with probability at least $\gamma/2$. So, we have

$$Pr(\neg B | A) \leq (1-\gamma/2)^{n/2} \leq e^{-n\gamma/4} \tag{13}$$

where the latter inequality uses the fact that $(1-x)^y \leq e^{-xy}$ for $x \in [0,1]$ and $y > 0$. Substituting these two equations in (7) and performing a few substitutions, we have the probability that Random-MMP fails in n iteration is no more than

$$Pr(\neg A) + Pr(\neg B|A) \leq (\exp(4\alpha k^\star - k^{\star 2}) + 1)\exp(-n \min(\alpha^2, \gamma/4)). \tag{14}$$

This equation gives exponentially decreasing bound $c \exp(-nd)$ with constants $c = \exp(4\alpha k^\star - k^{\star 2}) + 1$ and $d = \min(\alpha^2, \gamma/4)$. □

Claim 2. *Theorem 2 holds.*

Proof. Let Random-MMP′ denote the algorithm Random-MMP using L_p as the local planner. From the assumptions and Theorem 1, it follows that the probability that Random-MMP′ fails to find a path by

k iterations is bounded by ce^{-dn} for some constants c and d derived from α', β', and γ.

Construct an algorithm A, that performs n flips of a coin with probability p of landing heads, counts the number of heads k, and runs k iterations of Random-MMP$'$. A is guaranteed to succeed if (event X) $k \geq np/2$, and then (event Y) Random-MMP$'$ succeeds after $np/2$ iterations. Using Hoeffding's inequality [20], the probability that event X does not hold satisfies $P(\neg X) \leq \exp(-p^2 n/2)$. If $np/2$ iterations of Random-MMP$'$ are run, then by assumption, $Pr(\neg Y|X) \leq ce^{-dnp/2}$. Substituting these values into (7) gives:

$$P(A \text{ fails}) \leq (1+c)\exp(-n\min(dp/2, p^2/2)). \tag{15}$$

Since n iterations of Random-MMP using L as the local planner is at least as likely to succeed as algorithm A, the desired result is implied. □

References

[1] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Alg. Found. Rob.*, pages 109–125. A K Peters, Wellesley, MA, 1995.

[2] J.-D. Boissonnat, O. Devillers, L. Donati, and F. Preparata. Motion planning of legged robots: The spider robot problem. *Int. J. of Computational Geometry and Applications*, 5(1-2):3–20, 1995.

[3] M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[4] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock. Multi-step motion planning for free-climbing robots. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Zeist, Netherlands, 2004.

[5] F. Bullo and M. Žefran. On modeling and locomotion of hybrid mechanical systems with impacts. In *IEEE Conf. on Decision and Control*, pages 2633–2638, Tampa, FL, 1998.

[6] F. Bullo and M. Žefran. Modeling and controllability for a class of hybrid mechanical systems. *IEEE Trans. Robot. and Autom.*, 18(4):563–573, 2002.

[7] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *Intl. J. of Robotics Research*, 28(104), 2009.

[8] A. Casal. *Reconfiguration Planning for Modular Self-Reconfigurable Robots.* PhD thesis, Stanford University, Stanford, CA, 2001.

[9] A. Casal and M. Yim. Self-reconfiguration planning for a class of modular robots. In *SPIE Int. Symp. on Intel. Sys. and Adv. Manufacturing*, pages 246–257, 1999.

[10] M. Cherif and K. Gupta. Planning for in-hand dextrous manipulation. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*, pages 103–117. AK Peters, Ltd., 1998.

[11] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *IEEE Int. Conf. Hum. Rob.*, Munich, Germany, 2003.

[12] J. Cortés and T. Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Zeist, Netherlands, 2004.

[13] P. Ferbach and J. Barraquand. A method of progressive constraints for manipulation planning. *IEEE Trans. Robot. and Autom.*, 13(4):473–485, 1997.

[14] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *Intl. J. of Rob. Res.*, 26(8):845–863, 2007.

[15] B. Goodwine and J. Burdick. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Hanover, NH, 2000.

[16] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox. Motion planning for legged robots in varied terrain. *Int. J. of Rob. Res.*, 27(11–12):1325–1349, 2008.

[17] K. Hauser and J.-C. Latombe. Multi-modal planning in non-expansive spaces. *Int. J. of Robotics Research*, 29(7):897–915, 2010.

[18] K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *IEEE Intl. Conf. on Robotics and Automation*, Anchorage, AK, May 2010.

[19] K. Hauser, V. Ng-Thow-Hing, and H. G.-B. nos. Multi-modal planning for a humanoid manipulation task. In *Intl. Symposium on Robotics Research*, Hiroshima, Japan, 2007.

[20] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. of the American Statistical Association*, 58(301):13–30, March 1996.

[21] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. *Int. J. Rob. Res.*, 21(3):233–255, Mar 2002.

[22] X. Ji and J. Xiao. On random sampling in contact configuration space. In B. Donald, K. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 265–277. AK Peters, Ltd., 2001.

[23] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. Rob. Aut.*, pages 945–952, San Diego, CA, 1994.

[24] J. J. Kuffner, Jr., S. Kagami, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. In *First Int. Conf. on Humanoid Robotics*, Boston, MA, 2000.

[25] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[26] S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *Int. J. Rob. Res.*, 20(5):379–400, 2001.

[27] K. Lynch and M. Mason. Stable pushing: Mechanics, controllability, and planning. *Int. J. Rob. Res.*, 15(6):533–556, Dec 1996.

[28] M. Mason. Mechanics and planning of manipulator pushing operations. *Int. J. Rob. Res.*, 5(3), 1986.

[29] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control, LNCS 1790*, Mar 2000.

[30] C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. In *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, pages 1716–1721, Takamatsu, Japan, 2000.

[31] D. Nieuwenhuisen, A. F. van der Stappen, and M. H. Overmars. An effective framework for path planning amidst movable obstacles. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, New York, NY, 2006.

[32] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics (No. 2). University of Cambridge, 1998.

[33] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. SIAM, 2000.

[34] M. Peshkin and A. Sanderson. The motion of a pushed, sliding workpiece. *Int. J. Robotics and Automation*, 4(6):569–598, Dec 1988.

[35] J. Pettré, J.-P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Eurographics/SIGGRAPH Symp. Comp. Anim.*, 2003.

[36] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *Int. J. of Rob. Res.*, 21(1):5–26, 2002.

[37] T. Simeon, J.-P. Laumond, J. Cortes, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *Int. J. of Robotics Research*, 23(7-8):729–746, 2004.

[38] M. Stilman. *Navigation Among Movable Obstacles*. PhD thesis, Carnegie Mellon University, 2007.

[39] J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *Workshop on the Algorithmic Foundations of Robotics*, 2008.

[40] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Computer Vision Winter Workshop 2007*, Lambrecht, Austria, February 2007.

[41] R. Wilson and J. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71(2):371–396, 1995.

[42] J. Xu, T. J. Koo, and Z. Li. Sampling-based finger gaits planning for multifingered robotic hand. *Autonomous Robots*, 28(4):385–402, May 2010.

[43] J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Trans. Robot. and Autom.*, 17(6):951–958, 2001.

[44] M. Yashima and H. Yamaguchi. Dynamic motion planning whole arm grasp systems based on switching contact modes. In *IEEE Int. Conf. Rob. Aut.*, Washington, D.C., 2002.