Scaling Multi-Modal Planning: Using Experience and Informing Discrete Search

Zachary Kingston and Lydia E. Kavraki, *Fellow, IEEE*

***Abstract*—Robotic manipulation is inherently continuous, but typically has an underlying discrete structure, such as if an object is grasped. Many problems like these are *multi-modal*, such as pick-and-place tasks where every object grasp and placement is a *mode*. Multi-modal problems require finding a sequence of *transitions* between modes—for example, a particular sequence of object picks and placements. However, many multi-modal planners fail to scale when motion planning is difficult (e.g., in clutter) or the task has a long horizon (e.g., rearrangement). This work presents solutions for multi-modal scalability in both these areas. For motion planning, we present an experience-based planning framework ALEF which reuses experience from similar modes both online and from training data. For task satisfaction, we present a layered planning approach that uses a discrete *lead* to bias search towards useful mode transitions, informed by weights over mode transitions. Together, these contributions enable multi-modal planners to tackle complex manipulation tasks that were previously infeasible or inefficient, and provide significant improvements in scenes with high-dimensional robots.**

I. Introduction

Robotic systems are increasingly challenged with complex manipulation tasks [1]. For example, systems with many degrees-of-freedom (DoF) such as mobile manipulators and humanoids will soon be deployed in cluttered kitchens, warehouses, and onboard space stations, such as with NASA's Robonaut 2 [2], [3]. Many manipulation problems such as these can be modeled as *multi-modal* problems [4], a general representation that encapsulates many domains including pick-and-place manipulation.

Multi-modal problems are defined by a finite number of high-level *mode families* each with a continuous set of *modes* that are *parameterized* by continuous values. In pick-and-place domains, an example mode family is the set of object placements, and a mode would be a placement of an object at a specific x, y coordinate on a table. Each mode imposes a set of geometric *constraints* (specifically *manifold constraints*) on the system that must be respected in order for motion to be feasible (e.g., the cup must always be in the hand of the robot when grasped). The goal or *task* of a multi-modal planner is to reach a desired state in a set of modes of the system, which might take many *mode transitions* to reach.

Many multi-modal motion planners have been proposed to address high-dimensional problems [4]–[6]. However, these methods are limited to specific domains, requiring domain-specific implementation, and suffer from two key challenges. First, for high-DoF systems in clutter, motion planning that respects mode constraints and avoids obstacles is difficult, time-consuming, and often infeasible. Second, for long-horizon tasks, it is difficult to find a feasible sequence of mode transitions among the exponential possibilities.

To improve motion planning, experience-based methods [7]–[9] use information from prior problems to expedite search in similar problems. However, these methods are designed for changing scene geometry, not modes. During multi-modal planning, many modes are explored, each imposing different constraints; solutions in one mode might not apply to others in the same family. To improve task search, some planners use a *layered* approach [10]–[12]; a high-level planner informs a low-level planner what transitions to attempt. However, it is challenging to design useful task-relevant abstractions for effective search over mode transitions in general domains.

This work addresses these challenges with two contributions: a novel experience-based framework named ALEF (Augmented Leafs with Experience on Foliations) and a general layered planner that proposes a lead path of mode transitions to guide search. These contributions enable a multi-modal planner to learn online during planning and using prior experience, improving motion planning and task satisfaction performance. In addition, our approach does not required specialized samplers or planners as we use a general formulation of modes as manifold constraints, leveraging a manifold-constrained planning framework [13].

Our experience-based framework ALEF builds a sparse roadmap within a novel manifold-constrained configuration space augmented with the mode family parameterization. Paths from this sparse roadmap are used to bias a sampling-based planner with valid samples directed by a query start and goal configuration. We show that ALEF improves planning performance dramatically only given a few examples and can be run within a multi-modal planner.

Second, our layered planning approach, inspired by SyCLoP [14], provides the multi-modal planner a candidate *leads*—heuristic sequences of mode families and parameters. Informed by a novel weighting scheme over the parameters of the mode family, the lead is generated by finding the lowest-weighted sequence of transitions to reach a desired mode. These weights are updated online by motion planning success or failure. We demonstrate on a variety of complex scenes in 2D and 3D that significant improvements are tied to our lead and weighting scheme.

This article extends initial work [15], [16] with expanded definitions (Sec. III-A1), algorithmic details (Sec. VI-A, Sec. VI-D), and contributions required for experiments in 3D

ZK and LEK are with the Department of Computer Science, Rice University, Houston, TX, USA. {zak, kavraki} at rice.edu. ZK is supported by NASA Space Technology Research Fellowship 80NSSC17K0163. This work is supported by NSF IIS 2008720 and Rice University Funds.

workspaces. To address the complexities of 3D manipulation tasks, we replace our prior explicit encoding of mode transition graphs with an improved, implicitly-defined transition graph (Sec. V-C1), as also factor our redundant configuration space variables by using explicit constraints (Sec. V-B2). These improvements enable our new experiments on 3D robots—without these improvements, the baseline approach fails to solve any of the new problems. These experiments include a complex pick-and-place domain with an 8-DoF Fetch robot (Sec. VII-B4) and a handrail climbing domain with NASA's 21-DoF Robonaut 2 (Sec. VII-B2).

II. Related Work

Manipulation planning is a core problem in robotics [1], [17] and has been studied for decades [18]–[21]. We focus on multi-modal manipulation planners that use sampling-based motion planning. Sampling-based motion planning methods are probabilistically-complete methods able to scale to high-dimensional problems [22]–[24]. In Sec. II-A, we discuss techniques for solving motion planning problems and related methods for reusing experience. We discuss the multi-modal planning problem and related methods in Sec. II-B.

A. Experience and Motion Planning

Planning in a single mode requires planning under *constraints*; we focus on geometric planning problems with *manifold constraints*. There are many approaches for planning under manifold constraints, e.g., trajectory optimization [25], [26] or sampling-based [27]—a survey of sampling-based techniques is given in [28]. For this paper, we use the constrained planning framework presented in [13].

To improve efficiency, many sampling-based methods adapt search *online* with information gathered during the same query. For example, the authors of [29] weight different samplers based on their performance. Other methods use collision checking to adapt sampling [30], improve solution quality [31], and adapt the local planner [32]. While our proposed framework ALEF is itself not an online method, within a multi-modal planner the framework can be trained online to improve future motion planning queries.

Some methods store experience for later retrieval. Experience can be retrieved based on start and goal similarity [7], [33] or workspace similarity [8], [9]. While these methods can be used within a multi-modal planner, they are designed for unconstrained problems, and they cannot transfer experience across mode constraints. The method most similar ALEF is THUNDER [7], which also stores experience in a sparse roadmap. Besides not considering constraints, THUNDER is designed around a retrieve-and-repair paradigm which assumes that recovered experience will be "close" to a solution. This is not the case for ALEF, which uses experience as samples to guide search under constraints, as retrieved paths are unlikely to be able to be used wholesale.

In the context of manipulation planning, learning has been used to infer which action parameterizations are likely to be valid, both offline [34], [35] and online (such as our guiding leads). Other techniques have used demonstrations to infer which constraints are needed in order to perform a task [36]. However, these methods are limited to inferring "good" modes and do not improve motion planning. Most similar to ALEF, the authors of [37] bias heuristic search from demonstrations to solve parameterized constrained problems. Our framework learns from prior queries within a mode family, either in a multi-modal planner or standalone, to improve the performance of motion planning.

B. Multi-Modal and Manipulation Planning

There are a wide variety of approaches to solving multi-modal planning problems—beyond those covered in this related work, [38] contains a survey of related techniques.

Frequently, planners use *layered* planning: a combination of planners that inform each others' searches. Layered planning is a heuristic to speed-up search by solving the problem at different levels of abstraction [14], [39]. One common abstraction is to introduce a *symbolic* or *discrete* representation, e.g., PDDL [40] or workspace discretizations [14].

A common abstraction used is a *mode graph* [18]—a discrete structure encoding possible transitions between modes. For problems with finite modes, mode graphs have been successfully applied to high-dimensional problems [41]–[43] and dynamic environments [44]. However, manipulation problems have continuous modes, e.g., an object can be placed anywhere on the plane of the table. For problems with continuous modes, *transition graphs* [4] capture what transitions between mode families, i.e., which "classes" of modes are possible. We use a similar abstraction to capture allowable transitions (see Sec. V-C).

Many planners discretize continuous modes, but if the discretization is not fine enough a plan will not be found (e.g, [10], [40], [45]). With an offline discretization, [10] used a "fuzzy" PRM over the mode graph to determine what transitions to take, with edge weights proportional to time spent solving single-mode planning problems, biasing search towards "easier" planning problems. Manipulation RRT [42], [43] also weights transitions between modes online, but is limited to a finite set of modes. Similarly, online adaptation has been used to bias search, e.g., approximating collisions [30], [46]. Our lead generation approach uses similar ideas for biasing search but for continuous modes (see Sec. VI-B).

Approaches similar to ours use online discretization via sampling. Manipulation PRM [11], [47], [48] and ASyMov [12], [49], [50] build a PRM over the transition manifold. Other planners [4], [51] build a tree of transitions in the continuous mode space, similar to how sampling-based planners discretize a continuous space. In the case of navigation among movable obstacles, a probabilistically-complete tree-based planner has been proposed [51]. Task and Motion Planning approaches (TAMP) also commonly use online sampling, e.g., [52], [53]. These works require specialized samplers or only work for finite sets of modes.

Our work draws inspiration from [4], a probabilistically-complete tree-based planner for general multi-modal problems. "Utility tables" are used by [4] to capture valuable transitions in multi-modal search, but these are computed offline and require specialized domain knowledge. Our lead generation estimates the value of transitions online (see Sec. VI-C). DARRTH [5], [54] is a multi-modal planner that takes a layered planning approach, but uses a simplified continuous planning domain rather than a discrete layer. [6], [55] propose asymptotically optimal multi-modal planners. These approaches use specialized samplers for transitioning between modes. Similar to our approach, [56] uses general transition samplers and factors the configuration based on relevant modes, but does not guide search beyond immediate transitions.

Our lead generation takes inspiration from SyCLoP, a general framework which uses a "lead" path through a discretization of the workspace [14]. In SyCLoP, planning is informed of promising avenues of exploration via discrete search, while discrete search is informed by feedback from local planning. We use leads through possible mode transitions to inform planning; successes and failures in single-mode planning inform the discrete layer. Notably, our framework and other manipulation planners based on SyCLoP avoid the explicit backtracking found in search-based TAMP frameworks [57] by considering expansion from any point on the search tree. Note we do not address tasks with temporal goals, e.g., [58].

III. Single-Mode Motion Planning

In this work, we consider manipulation planning problems that are *multi-modal*. These are composed of a finite set of *mode families*, each containing a continuous infinity of *modes*, parameterized by some co-parameter (as in [4]). Before we can discuss the structure of multi-modal problems, we first discuss the *single-mode planning problem*, which involves searching for a feasible motion plan in the presence of a mode constraint. Single-mode planning problems are sub-problems in the multi-modal problem—solving them effectively is essential to multi-modal planning performance. Modes are modeled by *manifold constraints*, which are discussed in Sec. III-A. Mode families are modeled as *foliated manifolds*, which are discussed in Sec. III-B. For further information on these concepts from differential geometry, see [59], [60].

A. Manifold Constraints as Modes

In the unconstrained geometric motion planning problem, we are interested in finding a collision-free *path* σ from a configuration $q_{\text{start}} \in \mathcal{Q}$ in a configuration space $\mathcal{Q}$ to some region of interest $Q_{\text{goal}} \subset \mathcal{Q}$, where $\sigma : [0,1] \rightarrow \mathcal{Q}_{\text{free}}$ such that $\sigma(0) = q_{\text{start}}$, $\sigma(1) \in Q_{\text{goal}}$ and $\sigma(t)$ is collision-free for all t. However, we consider solving motion planning problems within a *mode*, which imposes *constraints* on a robot's motion. We consider modes defined by *manifold constraints*, here shortened to "constraints."

Consider a robot system in a mode ξ. A mode ξ is defined by a *constraint function* $F^{\xi} : \mathbb{R}^n \rightarrow \mathbb{R}^{k^{\xi}}$ $(1 \leq k^{\xi} < n)$ which is C^2-smooth and is adhered to when $F^{\xi}(q) = \mathbf{0}$. Planning in a mode requires finding a path in the *mode manifold* $\mathcal{M}^{\xi}$, an $(n - k^{\xi})$-dimensional smooth submanifold of $\mathcal{Q}$ (Fig. 1a):

$$\mathcal{M}^{\xi} = \left\{ q \in \mathcal{Q} \mid F^{\xi}(q) = \mathbf{0} \right\}.$$

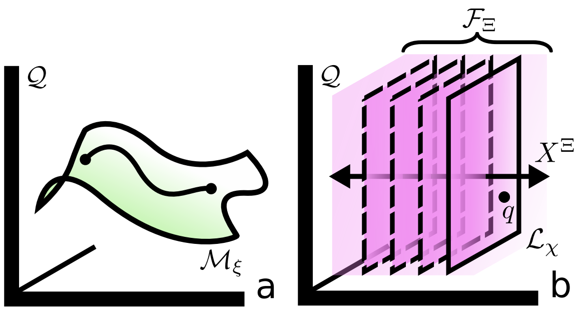


Fig. 1. In **a**, the lower-dimensional manifold $\mathcal{M}^{\xi}$ in a configuration space $\mathcal{Q}$ is shown. A valid path in mode ξ lies on the surface of this manifold. **b** shows a foliation $\mathcal{F}_{\Xi}$ for a mode family Ξ, with a transverse manifold X_{Ξ}. Leaves are shown along the transverse, including $\mathcal{L}_{\chi}$, which is determined by the configuration q. Although shown here as planes, leaf manifolds are not necessarily Euclidean.

Algorithm 1 P, a Projection Operator

```
    Input q, an initial configuration to project from
    Output P(q) ∈ M, the projected state configuration.
        On failure, "false" is returned.
1: procedure $P(q)$
2:     $x \leftarrow F(q)$
3:     while $\|x\|_2 > \epsilon$ and iterations remain do
4:         $q \leftarrow q - J(q)^{+}x$
5:         $x \leftarrow F(q)$
6:     return $q$ if $\|x\|_2 \leq \epsilon$ else false
```

Thus, the single-mode motion planning problem is finding a path from a point $q_{\text{start}} \in \mathcal{M}^{\xi}$ to some region of interest $Q_{\text{goal}} \subset \mathcal{M}^{\xi}$ such that the path is collision-free and satisfies mode constraints.

1) Solving for Constraints: Essential for planning under manifold constraints is finding constraint satisfying configurations. As F^{ξ} is a C^2-smooth function, the first derivative can be taken at a configuration $q \in \mathcal{Q}$, $J^{\xi} : \mathcal{Q} \rightarrow \mathbb{R}^{k^{\xi} \times n}$ (the *Jacobian*). $J^{\xi}(q)$ is of full rank when $F^{\xi}(q) = \mathbf{0}$, i.e., the gradients of the constraint functions are not linearly dependent at q. To find constraint adhering configurations, we use a *projection operator*. For manifold constraints, the basic gradient descent method presented in Alg. 1 can be used. Note that this is a basic algorithm for gradient descent, see [61] for more on solving non-linear equations such as these. This projection operator is used in both projection-based sampling-based planning for single modes as well as for sampling constraint satisfying configurations for transitions between modes (see Sec. V-A). In general—and especially in higher dimensions or scenarios with many constraints (e.g., Sec. V-B1)—the projection operator may fail to converge for many initial configurations. While this does not affect completeness, as sampling eventually finds a valid initial configuration, this does affect the performance of methods that use projection-based sampling. See [13], [28] for more details on planning under manifold constraints and projection-based methods.

B. Mode Families as Foliated Manifolds

We consider multi-modal problems which can be decomposed into a set of parameterized *mode families*—intuitively, these are the "classes" of modes. For example, consider the "monkey" robot to the right, which is grasping the green handrail. Here, modes correspond to all grasps of the handrail, the parameter in this case being the one-dimensional coordinate along the bar. The set of all these grasps comprises the mode family—there is a mode family for grasping the handrail with each limb. For a system considering pick-and-place manipulation (e.g., the Fetch robot in Fig. 15), modes could correspond to all placements of an object on a table: the object could be placed anywhere on the table, the parameter in this case being the two-dimensional coordinate of the objects on the table, with the set of all placements composing the mode family.

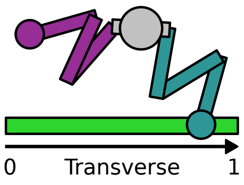


Fig. 2. "Monkey" robot.

Consider Ξ, the set of all modes, which can be partitioned into m disjoint *mode families* $\Xi_1, \ldots, \Xi_m$, each of which defines a *foliation*.

DEFINITION III.1: *Foliation [59]*
An n-dimensional manifold $\mathcal{M}$ is a *foliation* if there is a smooth fiber bundle $\mathcal{F}_{\mathcal{M}} = (X, \mathcal{L}, \pi)$. $\mathcal{F}_{\mathcal{M}}$ contains a *transverse manifold* X of dimension k, a set of disjoint, connected $(n-k)$-dimensional *leaf manifolds* $\mathcal{L}_\chi$ for all $\chi \in X$, and a smooth surjective *bundle projection* $\pi : \mathcal{M} \to X$. The union of all leaves $\bigcup_{\chi \in X} \mathcal{L}_\chi = \mathcal{M}$.

An example of a foliation is shown in Fig. 1b. A mode family is a foliation following their definition by a constraint function F^{Ξ_i} and *co-parameters* $\chi \in X^{\Xi_i}$, where X^{Ξ_i} is the transverse manifold of dimension k_{Ξ_i}. Furthermore, we assume the co-parameters are explicitly parameterized, e.g., with real numbers $X^{\Xi_i} \subset \mathbb{R}^{k_{\Xi_i}}$, such as the position along the bar grasped in Fig. 2. These co-parameters are essential to the efficiency of our methods—they enable us to succinctly encode relevant geometric state of the problem. This fact is leveraged by our methods, discussed in Sec. IV-A for the augmented space and in Sec. VI-B for weighting transitions.

A *mode* $\xi = \langle \Xi_i, \chi \rangle$ in a mode family is defined by its constraint function and a specific co-parameter χ, $F^{\Xi_i}(q) = \chi$, e.g., a specific grasp of the handrail at a position χ. This defines the leaf manifold $\mathcal{L}_\chi$:

$$\mathcal{M}^\xi = \mathcal{L}_\chi = \left\{ q \in \mathcal{Q} \mid F^{\Xi_i}(q) = \chi \right\}.$$

The *bundle projection* of the foliation π allows us to map from a configuration q that lies in the foliation manifold (that is, satisfies some mode in the mode family Ξ_i) to the co-parameter χ of the mode it satisfies. To ground this in the example, the bundle projection gives the coordinate along the handrail grapsed by the robot, for any configuration that grasps the handrail. With the guarantees of continuity afforded by a foliation, we have guarantees that the mode constraints between two modes "close by" (in parameter space) will be "similar." Modeling mode families as foliations puts manifold constraints and the definition of modes under one umbrella, enabling use of the general constrained planning framework presented in [13]. The relationship between the concepts of modes and mode families and their modeling as manifolds is summarized in Table I.

TABLE I
CONCEPTS AND NOTATION

Concept	Function Representation	Implicit Manifold
Mode $\xi = \langle \Xi_i, \chi \rangle$	Constraint Function $F^{\Xi_i}(q) = \chi, F^\xi = \mathbf{0}$	Leaf/Mode Manifold $\mathcal{L}_\chi = \mathcal{M}^\xi$
Mode Family Ξ_i	Foliation Constraint F^{Ξ_i}	Foliation $\mathcal{F}_{\Xi_i} = (X^{\Xi_i}, \mathcal{L}, \pi_{\Xi_i})$

IV. Accelerating Single-Mode Planning

We present the ALEF framework (Augmented Leafs with Experience on Foliations) for experience-based multi-modal planning under manifold constraints. ALEF learns from valid paths from solving single-mode planning problems and applies this experience to problems constrained by modes within the same mode family, improving the performance of single-mode planning. An overview of ALEF is given in Fig. 3.

From valid paths (Fig. 4a), a sparse roadmap is constructed in an *augmented foliated space* (AFS) (Fig. 4b). An AFS is a manifold-constrained configuration space with respect to the foliation constraint. Configurations in the AFS are augmented to include their co-parameters, i.e., what mode they are in. The AFS and the sparse roadmap are explained in Sec. IV-A and Sec. IV-C.

Upon a new planning query, ALEF uses experience to bias a sampling-based algorithm. Given a start and goal configuration, nearby vertices within the sparse roadmap are retrieved. If a valid path exists in the roadmap between the retrieved start and goal vertices (Fig. 4c), the path is projected onto the current query's leaf manifold using a projection operator (shown in Fig. 4d, see Sec. III-A1). Configurations from this path that are valid are used as samples. Experience retrieval is discussed in Sec. IV-D.

A. Augmented Foliated Space (AFS)

Recall from Sec. III-B that we model mode families as foliated manifolds with constraint functions F^Ξ. Foliations are parameterized by a transverse manifold X, where each $\chi \in X$ corresponds to a different mode. To relate configurations across different modes, we introduce a composite space $\mathcal{Q} \times X$, where each configuration is indexed with the parameters of the mode it satisfies. This can be seen in Fig. 3c, which visualizes configurations only in $\mathcal{Q}$, and Fig. 3d, which visualizes the same configurations in $\mathcal{Q} \times X$.

However, not all configurations satisfy the foliation constraint. The augmented foliated space (AFS) is a submanifold within the composite space $\mathcal{Q} \times X$. The AFS manifold is formally defined as:

$$\mathcal{M}^\Xi = \left\{ (q, \chi) \in \mathcal{Q} \times X \mid F^\Xi(q) = \chi \right\}.$$

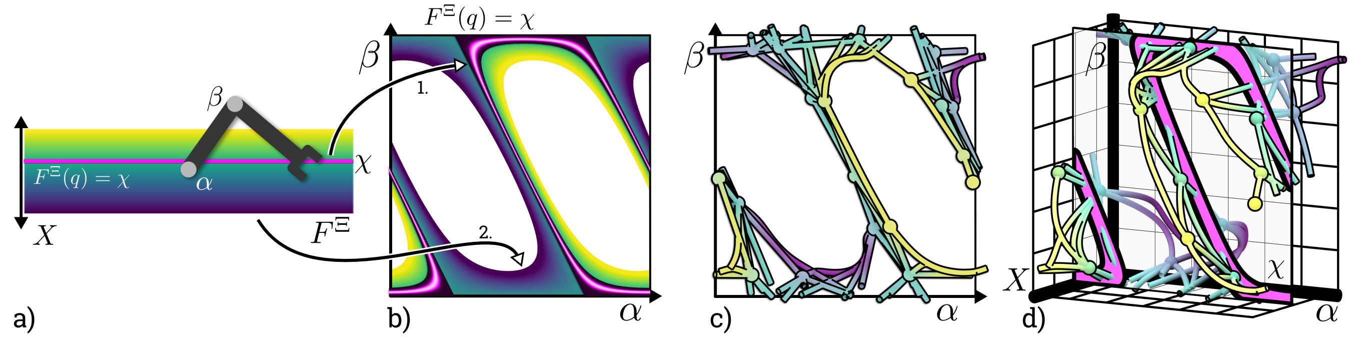


Fig. 3. Our experience-based constrained planning framework, ALEF, applied to a simple foliation with a two-link planar manipulator. **a)** The manipulator has two continuous joints α, β, ranging from $(-\pi, \pi]$. The foliation specifies a set of line constraints on the position end-effector, with the choice of line parameterized by values from X, which determines the height of the line (a mode family). An example line at $\chi \in X$ is shown in pink. The set of these lines forms a plane—the parameter of each line is visualized with a color gradient. Following the line imposes a *manifold constraint* on the system (a mode). **b)** Visualization of the set of manifold constraints defined by this foliation in the robot's configuration space. The manifold constraint corresponding to χ (white line) is highlighted in pink, indicated by arrow 1. The color gradient is shared with *a)*—the continuum of colors in the configuration space correspond to the different constraints (indicated by arrow 2). **c)** The sparse roadmap generated by ALEF. The color gradient is shared with *b)*, indicating each configuration's parameters χ. Although vertices in the roadmap satisfy different constraints (modes), our method makes connections between them. **d)** The manifold constraint corresponding to χ intersecting our sparse roadmap. The parameters X are another dimension (see Sec. IV-A). Edges between vertices traverse through this manifold, showing related experience our method can apply.

We leverage the general manifold-constrained sampling-based planning framework of [13] in order to model the AFS. Through the transverse dimension, connections can be made between configurations that satisfy different leaf constraints, since the AFS manifold is a superset of all leaf manifolds (Fig. 4b). The connections are such that all intermediate states satisfy the aforementioned foliation constraint.

B. Metrics in the AFS

As sampling-based methods require a distance metric to explore and find nearby configurations, we define a weighted metric for the AFS. This metric uses a weighted sum of the metrics of the configuration space and the transverse space. By default, we use the Euclidean metric for the transverse space (although any metric space is admittable). This weight relates to the relative importance of the co-parameters versus the configuration. For example, consider the "monkey" robot (Fig. 2): there are many configurations that are close in configuration space but possibly distant in terms of co-parameters. Simply shifting the pose of the base along the axis of the bar will change parameter values but not cause large configuration space distances. This can be visualized in Fig. 3d as "stretching" the X dimension. In our experiments, we heuristically weight the co-parameter three times more than the configuration space, which gave good performance.

C. Sparse Roadmap in the AFS

In this work, we represent experiences as valid paths gathered from leaf-constrained motion planning problems. Information from these paths is stored within a *sparse roadmap* that resides in the augmented foliated space. In particular, we employ SPARS2 [62], a method that does not require the maintenance of a dense roadmap. SPARS2 has guarantees of asymptotic near-optimality—as the roadmap "fills out" the probability of inserting a new configuration goes to zero, and paths within the roadmap are within some bound of optimal. A sparse roadmap has the benefit of finite memory requirements; the small size of the sparse roadmap, as compared to a dense roadmap, enables fast search times on new retrieval queries (Sec. IV-D). Note that we use SPARS2 within the manifold-constrained AFS. We conjecture that the theoretical properties of SPARS2 hold in this case given theoretical results from [13].

As our sparse roadmap is built in the AFS, edges can be added between nodes from different leaves if the edge satisfies the foliation constraint and is collision-free (Fig. 4c). In the example shown in Fig. 3, nodes of the roadmap that lie on different lines are connected by edges that correspond to the motion of the manipulator that moves between these lines. Valid connections made in the AFS are indicative of potentially valid motion on all leaves in-between, due to continuity between the leaves of the foliated manifold. An example roadmap is shown in Fig. 3c. Vertices and edges are colored according to their co-parameter. An example of this continuity can be seen in Fig. 3d, where a leaf manifold is shown intersecting the roadmap in the AFS.

The idea of using SPARS2 as a database to store and retrieve experiences was first introduced in the THUNDER algorithm [7]. However, inserting, retrieving, and reusing paths for SPARS2 in the AFS demands different methods as compared to the standard unconstrained roadmap used in [7]. In particular, the nature of the foliation constraint leads to far less exact reuse of configurations from the roadmap—configurations retrieved must be adjusted to the current constraint, and retrieved paths cannot be used as a whole. This motivates the following more efficient (but less complete) insertion heuristic into the roadmap as well as the sampling-based retrieval detailed in Sec. IV-D.

As noted by [7], a naïve insertion of the waypoints in sequential order will most likely result in the vertices of the path being disconnected within the roadmap. This stems from the way the SPARS2 chooses which vertices to connect in order to maintain sparseness. Connections between disconnected components of the roadmap are only attempted when a new vertex acts as a *connectivity* node, meaning it is "visible" from two vertices that do not belong to the same connected component. A node is visible by another if it is within a certain radius and there is a valid connection between them. Additionally, nodes are added as *guard nodes* when there are

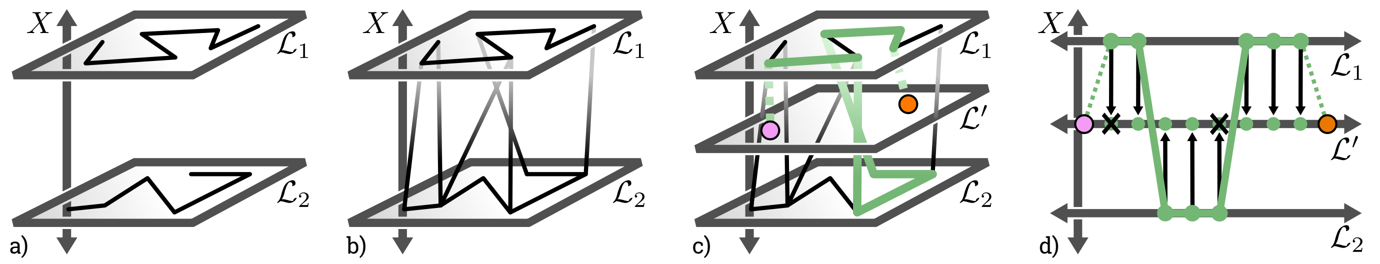


Fig. 4. Sketch of ALEF, the proposed method. **a)** Valid paths from different action parameterizations are gathered as experience. These paths exist in the leaves $\mathcal{L}_1$ and $\mathcal{L}_2$ of the foliated manifold of the action. Note that these are paths in the configuration space, but are such that each configuration satisfies their leaf's constraint. **b)** ALEF builds a sparse roadmap in the augmented foliated space (AFS, Sec. IV-A), finding feasible connections across leaves. Connections are such that the foliation constraint is always satisfied. **c)** Upon retrieval for a query in new leaf $\mathcal{L}'$, a path is found in the roadmap (highlighted) given a start and goal (pink and orange circles). **d)** This path is projected to $\mathcal{L}'$. Valid configurations are used as samples in the new query.

no other nodes that are visible. However, with this insertion policy, many paths will end up disconnected. For example, when sequentially inserting a straight-line path, only nodes that act as *guards* will be added, and no connections between them will be attempted. Thus, we use the ordering heuristic proposed by [7]—first, the inserted path is interpolated at high resolution, then, nodes that are likely *guards* are inserted, and then select nodes between these guards are added as *connectivity* nodes.

To improve connectivity, [7] inserts all remaining configurations in the path randomly, resulting in possibly long roadmap construction times. For efficiency, we do not do this step. Instead, to improve roadmap connectivity, we employ a heuristic that attempts to connect newly inserted nodes to other connected components. This heuristic effectively increases the visibility radius of SPARS2, possibly yielding longer paths in the roadmap, but quickly improving connectivity. Our heuristic enables ALEF to be trained in real-time within a multi-modal planner.

D. Retrieving Experience from the AFS Roadmap

During motion planning, a planner generates many candidate goal configurations, e.g., from a goal sampler. Given a start and goal configuration pair, we find their nearest neighbor in the AFS roadmap using the weighted metric (Sec. IV-B). A collision-free path within the roadmap is then found using A* search (Fig. 4c), again with respect to the weighted metric. As there might be changes within the environment (e.g., changing obstacles between queries), we lazily evaluate edge validity in roadmap search, which invalidates edges as they are discovered. Paths retrieved from the roadmap hopefully contain configurations important for the current planning problem.

After a path is retrieved, the waypoints of the path are projected to satisfy the query's leaf constraint and are checked for collision (Fig. 4d). It is unlikely that the entire retrieved path is successfully projected onto the new leaf due to projection failing to converge or collisions with changed obstacle geometry, thus, we cannot use the retrieved path directly. The ratio of states that are successfully retrieved is given as the valid state ratio in our experiments (Sec. VII-A). Instead, we use the retrieved waypoints as samples to bias the search of a sampling-based planner. ALEF uses *all* valid waypoints from retrieved paths given *all* start and goal pairs. Heuristically, these samples are used ordered by their place along the retrieved path. Samples are used with some probability $\lambda \in (0, 1)$, otherwise, default constrained state space sampling is used (which is probabilistically complete [13]). In our experiments, we use $\lambda = 0.5$, but we did not see much sensitivity to this value.

V. Multi-Modal Motion Planning

We now discuss *multi-modal* planning, which involves not only solving single-mode problems, but solving for a sequence of single-mode problems to reach a desired mode. We also highlight here extensions necessary for complex manipulation planning, demonstrated by our experiments with 3D manipulation tasks. Critical to finding valid sequence is finding valid transitions between modes, which is discussed in Sec. V-A. Additionally, important to scalability is the ability to factor the problem into relevant components, discussed in Sec. V-B. Possible transitions between modes are encoded in a discrete abstraction discussed in Sec. V-C. Finally, the multi-modal planning problem is presented in Sec. V-D.

A. Transitions between Modes and Mode Families

Consider two modes ξ and ξ' of a robot, as shown in Fig. 5a. To transition between ξ and ξ', a path in $\mathcal{M}^{\xi}$ must be found that ends at a *transition configuration* q' that simultaneously satisfies the manifold constraints of ξ and ξ'. That is, $q' \in \mathcal{M}^{\xi} \cap \mathcal{M}^{\xi'}$, where $\mathcal{M}^{\xi} \cap \mathcal{M}^{\xi'}$ is the *transition manifold* between ξ and ξ' and is of zero volume relative to $\mathcal{M}^{\xi}$ and $\mathcal{M}^{\xi'}$.

$$\mathcal{M}_{\xi \cap \xi'} = \left\{ q \in \mathcal{Q} \mid F^{\xi}(q) = \mathbf{0} \wedge F^{\xi'}(q) = \mathbf{0} \right\}.$$

$\mathcal{M}_{\xi \cap \xi'}$ can be empty if no configuration satisfies both mode constraints (i.e., transitioning is impossible between ξ and ξ'). The transition manifold is evaluated by combining F^{ξ} and $F^{\xi'}$ as described in Sec. V-B1.

1) Foliations as Transition Goals: To transition from a mode to another mode family, our method considers not only a single transition configuration or mode, but a *range* of modes in the destination mode family. This desired range is modeled as a *sub-foliation* constructed from a contiguous interval of co-parameters in the desired foliation. For example, consider a foliation with co-parameters $[0, 1]$—the total foliated manifold is $\bigcup_{\chi \in [0,1]} \mathcal{L}_{\chi} = \mathcal{M}$. We take a "slice" of this foliation to create a sub-foliation $\mathcal{M}'$, e.g., a sub-foliation for the interval

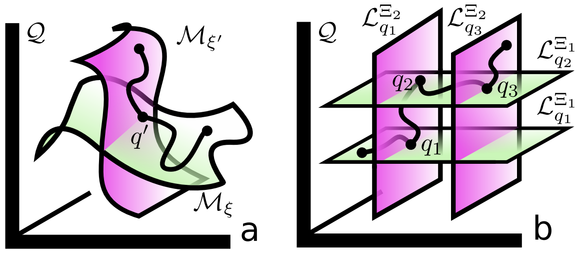

$\mathcal{Q}$
$\mathcal{M}_{\xi'}$
q'
$\mathcal{M}_{\xi}$
a
$\mathcal{Q}$
$\mathcal{L}^{\Xi_2}_{q_1}$
$\mathcal{L}^{\Xi_2}_{q_3}$
$\mathcal{L}^{\Xi_1}_{q_2}$
$\mathcal{L}^{\Xi_1}_{q_1}$
q_2
q_3
q_1
b

Fig. 5. In **a**, two mode manifolds $\mathcal{M}^{\xi}$ and $\mathcal{M}^{\xi'}$ are shown, along with a path that transitions between these modes at a transition configuration q'. **b** shows a multi-modal path over two mode families with foliations $\mathcal{F}_{\Xi_1}$ and $\mathcal{F}_{\Xi_2}$. Here, transitions occur at the configurations q_1, q_2, and q_3. Each configuration determines the leaf of the foliation.

$[0.1, 0.2]$ is $\bigcup_{\chi\in[0.1,0.2]} \mathcal{L}_{\chi} = \mathcal{M}'$. We denote the sub-foliation for a mode family Ξ as $\Xi_{[0.1,0.2]}$. As above, the intersection the current mode and sub-foliation's constraint is used to sample transition configurations.

B. Factoring and Composing Modes

In manipulation-focused multi-modal problems, it is not only the controllable robot(s) $R_1, \dots, R_{n_R}$ that are of interest, but also the collection of manipulable objects and kinematic structures $O_1, \dots, O_{n_O}$ in the scene. The configuration space of the system is the n-dimensional Cartesian product of the configuration spaces of the robots and objects:

$$\mathcal{Q} = \mathcal{Q}_{R_1} \times \cdots \times \mathcal{Q}_{R_{n_R}} \times \mathcal{Q}_{O_1} \times \cdots \times \mathcal{Q}_{O_{n_O}}.$$

Typically objects are rigid bodies in $SE(3)$, but could also be articulated systems such as drawers and doors. Usually this composite system is under-actuated, as only the robot(s) are controllable—objects and kinematic structures are kept stationary by imposed mode constraints (e.g., a placed object remains in the same position), and are moved when an appropriate mode is entered (e.g., an object moves when grasped).

1) Combining Multiple Modes: In complex manipulation problems, a mode is more easily specified as a combination of modes. For example, consider a pick-and-place domain with multiple objects. The system as a whole is in a single mode composed of multiple objects that are placed, stacked, or held. This mode is more easily specified as a composition of multiple modes, i.e., $\xi = \xi_1 \cap \cdots \cap \xi_k$. The constraints $F^{\xi}_1, \dots, F^{\xi}_k$ are composed together to define a composite constraint function which defines the composite mode:

$$F^{\xi}(q) = \begin{bmatrix} F^{\xi}_1(q) & \dots & F^{\xi}_k(q) \end{bmatrix}^T.$$

In our approach, we use projection-based sampling of transition manifolds to generate candidate transitions, which has a non-zero probability of sampling the entire transition manifold [13] (see Sec. III-A1). Note that in general this can lead to poorly-conditioned functions with many local minima. However, mode constraint functions usually apply to *disjoint* subsets of the composite configuration space or at least different end-effectors, and thus gradient descent works reasonably well in practice. Some attention might be needed for cases where there are overlaps or interactions between constraint functions, e.g., by using a damped pseudo-inverse in the gradient descent [63].

2) Implicit and Explicit Constraints: Manifold constraints discussed so far are *implicit constraints*, as they do not have an explicit representation that specifies valid configurations—valid configurations are defined *implicitly* by adhering configurations of the constraint function. However, sometimes an explicit parameterization of a constraint exists, particularly when a constraint fully determines a subset of configuration variables in terms of another disjoint subset of configuration variables. For example, when a robot has grasped an object, the object's configuration is fully determined by the robot's configuration. These constraints are said to be *explicit*:

DEFINITION V.1: *Explicit Constraint [64]*
Let $I = \{1, \dots, n\}$ be the set of configuration indices, where $\mathcal{Q}_{I'}, I' \subset I$ is a subspace of the configuration space given those dimension indices. An *explicit constraint* $E = (I_{\text{in}}, I_{\text{out}}, F_{\text{exp}})$ is a mapping from $\mathcal{Q}$ to $\mathcal{Q}$ defined as:

- Input indices $I_{\text{in}} \subset I$,
- Output indices $I_{\text{out}} \subset I$ where $I_{\text{in}} \cap I_{\text{out}} = \emptyset$,
- A C^2 smooth mapping $F_{\text{exp}} : \mathcal{Q}_{I_{\text{in}}} \rightarrow \mathcal{Q}_{I_{\text{out}}}$ such that $q_{I_{\text{out}}} = F_{\text{exp}}(q_{I_{\text{in}}})$

Note that explicit constraints can be used as implicit constraints—this is useful for sampling transitions as described above. If a mode imposed is explicit, all configuration variables that are determined by that mode constraint can be *factored* from the problem, as these configuration variables are redundant for planning. For example, when an object is grasped by the robot, the object's configuration variables are fully determined by the robot's configuration and grasp mode, and thus the object's configuration does not need to be selected when planning.

C. Mode Transition Graph

Recall from Sec. III-B the set of mode families $\Xi_1, \dots, \Xi_m$. In many domains, we have knowledge of what feasible transitions between mode families exist. For example, consider a pick-and-place domain with two objects. There are mode families for picking and placing either object, and we know the robot cannot grasp another object if an object is already grasped. We encapsulate this knowledge with a *mode transition graph*, similar to [4], which encapsulates domain knowledge of which mode family foliations intersect with each other, i.e., what valid transitions exist.

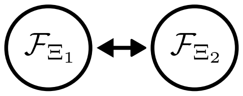
A mode transition graph $\mathcal{G}$ is composed of mode family vertices $V = \Xi_1, \dots, \Xi_m$ with edges $\langle \Xi_i, \Xi_j \rangle \in E \subseteq V \times V$ denoting valid transitions between mode families. Note, transitions are only possible between *different* mode families, and not within the mode family itself. For example, a grasp cannot be instantaneously changed; a regrasp is necessary, which corresponds to at least two mode transitions.

The transition graph only describes possible transitions in the domain and does not include information about mode family co-parameters. Our multi-modal planner leverages the transition graph (along with corresponding mode co-parameters) in order to bias multi-modal search along promising directions (see Sec. VI-B).

```
(define (domain simple)
  (:types bar)
  (:functions (current-bar) - bar)
  (:action transition
    :parameters (?dst - bar)
    :precondition (not (= (current-bar) ?dst))
      :effect (= (current-bar) ?dst))

  ;; Encodes when mode families are imposed on
      state
  (:family bar-foliation
    :parameters (?x - bar)
    :condition (= (current-bar) ?x))
  )

(define (problem example)
  (:domain simple)
  (:objects bar1 bar2 - bar)
  ;; …
  )
```



$$\begin{tikzcd} \mathcal{F}_{\Xi_1} \arrow[r, leftrightarrow] & \mathcal{F}_{\Xi_2} \end{tikzcd}$$

Fig. 6. A simple PDDL domain that encodes an implicit transition system between grasping two bars and the resulting explicit transition graph for a simple problem. In the above PDDL, a new definition :family encodes the imposition operator $\mathfrak{X}$, the condition in which a mode family is imposed on a symbolic state. If conditions hold, then the mode family is imposed given the corresponding grounding. Here, $\mathcal{F}_{\Xi_1}$ corresponds to grasping bar 1 and $\mathcal{F}_{\Xi_2}$ to bar 2. This domain is a simplified version of the domain used for handrail climbing in Fig. 13, minus preconditions to require alternating legs.

1) Implicitly-Defined Transition Graphs: For complex manipulation domains, explicitly encoding the transition graph is difficult as there are possibly exponentially many mode families and transitions to encode, e.g., pick-and-place domains with respect to objects. Common in many manipulation planners is the use of a *task language* that uses symbolic states and actions that implicitly describe an underlying transition graph:

DEFINITION V.2: *Task Language [40]*
The task language is a set of strings of actions, defined by $\mathfrak{L} = (\mathfrak{P}, \mathfrak{A}, \mathfrak{T}, \mathfrak{p}^{t_0}, \mathfrak{G})$, where:

- $\mathfrak{P}$ is the finite *state space* ranging over predicate variables $\mathfrak{p}_0, \ldots, \mathfrak{p}_n$. That is, a state $\mathfrak{s} \in \mathfrak{P}$ contains the value of each variable $\mathfrak{p}_0, \ldots, \mathfrak{p}_n$.
- $\mathfrak{A}$ is the finite set of task operators, *i.e.*, *terminal symbols* and the *actions* in the domain.
- $\mathfrak{T} \subseteq \mathbb{P}(\mathfrak{P}) \times \mathfrak{A} \times \mathbb{P}(\mathfrak{P})$ is the finite set of symbolic transitions, where $\mathbb{P}(\mathfrak{P})$ is the power set of $\mathfrak{P}$. Each $\mathfrak{t}_i \in \mathfrak{T}$ denotes transitions $\text{pre}(\mathfrak{t}_i) \xrightarrow{\mathfrak{a}_i} \text{eff}(\mathfrak{t}_i)$, where $\mathfrak{a}_i \in \mathfrak{A}$ is the operator, $\text{pre}(\mathfrak{t}_i) \subseteq \mathfrak{P}$ is the precondition set, and $\text{eff}(\mathfrak{t}_i) \subseteq \mathfrak{P}$ is the effect set. A transition at step t with action $\mathfrak{a}_i$ is such that $\mathfrak{s}^t \in \text{pre}(\mathfrak{t}_i), \mathfrak{s}^{t+1} \in \text{eff}(\mathfrak{t}_i)$, and for all $\mathfrak{p}_k^t \notin \text{eff}(\mathfrak{t}_i), \mathfrak{p}_k^t \in \mathfrak{s}^t$, $\mathfrak{p}_k^t = \mathfrak{p}_k^{t+1}$.
- $\mathfrak{G} \subseteq \mathfrak{P}$, the finite set of accepting states, *i.e.*, the goal.

This is encoded with notations such as STRIPS [65], ADL [66], and PDDL [67]–[69]. We use PDDL to model complex domains. See Fig. 6 for a simplified domain similar to the handrail climbing example in Sec. VII-B2, and Fig. 16 for the domain used for block stacking with the Fetch robot.

We additionally define for each mode family an *imposition operator* $\mathfrak{X} \subseteq \mathbb{P}(\mathfrak{P}) \times \Xi$ which encodes, similar to the set of symbolic transitions, which mode families imposed their constraints on the system given a symbolic state $\mathfrak{s}$. Each

Algorithm 2 Multi-Modal Planner Skeleton

```
    Input $\mathcal{P}$, a multi-modal planning problem
    Output A multi-modal path
 1: procedure $MultiModalPlanner(\mathcal{P})$
 2:     $T \leftarrow (q_{\text{start}}, \mathfrak{s}_{\text{start}})$
 3:     while iterations remain do
 4:         Sample a system state $s_{\text{sample}} = (q_{\text{sample}}, \mathfrak{s}_{\text{sample}})$,
            with a small chance $< \lambda_{\text{goal}}$ of being from the
            goal $\mathcal{Q}_{\text{goal}}$
            // Use an RRT-like expansion strategy
 5:         Select $s_0 = (q_0, \mathfrak{s}_0)$ in $T$ closest to $q_{\text{sample}}$
 6:         $s_1, \ldots s_k \leftarrow$ EXTENDTREE$(s_0, s_{\text{sample}}, T)$
 7:         if $k > 0$ then
 8:             Add $s_1, \ldots, s_k$ as descendants of $s_0$ in $T$
 9:             if $q_k \in \mathcal{Q}_{\text{goal}}$ then
10:                 return path from $q_{\text{start}}$ to $q_k$
```

$\mathfrak{x}_i \in \mathfrak{X}$ denotes the condition (similar to action preconditions) $\text{pre}(\mathfrak{x}_i) \subseteq \mathfrak{P}$ where if $\mathfrak{s} \in \text{pre}(\mathfrak{x}_i)$, mode family Ξ_i is imposed (see Fig. 6). Note that multiple mode families can be simultaneously imposed—these constraints are combined as described in Sec. V-B1. Additionally, no specific implementation is required for specifying transition manifolds; transition manifolds are constructed as stated in Sec. V-A1.

D. Multi-Modal Planning

A multi-modal problem is defined by:

- The robot and object configuration spaces (Sec. V-B).
- Mode families Ξ
- The implicit transition graph defined by task language $\mathfrak{L}$ and imposition mapping $\mathfrak{X}$ (Sec. V-C).
- Initial configuration of the system $q_{\text{start}} \in \mathcal{Q}$ and initial symbolic state $\mathfrak{s}_{\text{start}}$.
- Goal region $\mathcal{Q}_{\text{goal}} \times \mathfrak{G}$ of configurations and states.

The multi-modal planning problem is thus to find a continuous sequence of collision-free paths, each of which is a valid path in some mode ending in a valid transition to the subsequent mode. A multi-modal path is shown in Fig. 5b.

VI. Guiding Multi-Modal Planning with Leads

With guiding leads, multi-modal planners are able to handle problems that interleave complex discrete structures with difficult continuous single-mode planning. This discrete structure (e.g., information about which objects are grasped by which grippers) is captured by the *mode transition graph* which we augment with information from single-mode planning (Sec. VI-B). Based on this graph, we compute *leads*—promising sequences of modes to reach the goal (Sec. VI-D). Leads are used to guide the search of a lower-level motion planner, which then informs weights in the transition graph (Sec. VI-C). We combine these layers of planning synergistically to better inform the overall search.

A. Baseline Method

Our lead generation builds on concepts from [4], which plans given a similar model of a multi-modal problem. The

Algorithm 3 Random ExtendTree

```
1: procedure ExtendTree($s_0, s_{\text{goal}}, T$)
2:     $\mathfrak{a}_1, \ldots, \mathfrak{a}_k \leftarrow$ all valid actions from $\mathfrak{s}_0$ based on $\mathfrak{L}$
3:     Sample a mode family $\mathfrak{a}_{\text{goal}}$ from the set
       // Plan to the goal family given the mode constraints
4:     $\Xi_0 \leftarrow \mathfrak{X}(\mathfrak{s}_0)$    // Imposition op. to get mode family
5:     $\chi_0 \leftarrow \pi_{\Xi_0}(q_0)$    // Bundle projection to get mode
6:     $\mathfrak{s}_1 \leftarrow \mathfrak{a}_{\text{goal}}(\mathfrak{s}_0)$
7:     return SMP($q_0, \xi_{\chi_0}, \mathfrak{X}(\mathfrak{s}_1)$), $\mathfrak{s}_1$
```

Algorithm 4 Single-Mode Planning SingleModePlan

```
1: procedure SMP($q_{\text{start}}, \xi, \Xi'_{\text{goal}}$)
2:     Sample transitions $Q_{\text{goal}} = q_1, \ldots, q_k$ from $\xi \cap \Xi'_{\text{goal}}$
       using projection sampling
3:     If no samples, return NO SAMPLES
4:     Plan under mode $\xi$ from $q_{\text{start}}$ to $Q_{\text{goal}}$
5:     if planning fails, return PLANNING FAILED,
6:     else, return $q_{\text{goal}}$
```

algorithm presented in [4] transitions between modes by choosing the next mode using an RRT-like extension scheme, extending at random, reproduced in Alg. 2. Our approach modifies the extension step by informing search with a discrete lead, described in Sec. VI-D. Additionally, [4] uses a "utility-centered expansion strategy," where promising transitions are taken based on a precomputed "utility." This is reproduced in the random extension method shown in Alg. 3, albeit with a weighted sampling method based on this precomputed utility. Our approach also approximates how likely a transition is to succeed, but does so online (Sec. VI-C). This is similar to the weighting done in [10], but for continuously infinite modes.

Moreover, we use a very general formulation of mode families: a constraint function F^{Ξ} (recall Sec. III-A). For example, in Fig. 7c, the constraint that defines the mode family for grasping the handrail is given as a function of the robot's kinematics, requiring the end-effector to be positioned along the rail. We leverage the general constrained sampling-based planning framework described in [13] to enable single-mode planning given general mode constraints (using a manifold-constrained PRM, which is reused if the mode is revisited). Moreover, if ALEF (Sec. IV) is used, the appropriate sparse roadmap is retrieved based on the mode family of the current mode constraint. Mode transitions are sampled using projection-based sampling over the intersection of the leaf manifold and destination manifold.

B. Weighted Mode Transition Graphs

Recall the mode transition graph from Sec. V-C. We augment the mode transition graph with statistics on what transitions are likely to succeed. Attached to every directed edge $\langle \Xi_{\text{src}}, \Xi_{\text{dst}} \rangle$ are *transition weights* $\mathcal{D}^{\Xi_{\text{src}},\Xi_{\text{dst}}}(\chi_{\text{src}}, \chi_{\text{dst}})$. The transition weights $\mathcal{D}^{\Xi_{\text{src}},\Xi_{\text{dst}}}(\chi_{\text{src}}, \chi_{\text{dst}})$ are a distribution with support over the transverse manifolds for the source and destination mode families,

$$\mathcal{D}^{\Xi_{\text{src}},\Xi_{\text{dst}}}(\chi_{\text{src}}, \chi_{\text{dst}}) : X^{\Xi_{\text{src}}} \times X^{\Xi_{\text{dst}}} \to \mathbb{R}.$$

Informally, $\mathcal{D}^{\Xi_{\text{src}},\Xi_{\text{dst}}}(\chi_{\text{src}}, \chi_{\text{dst}})$ captures the difficulty of the single-mode motion planning problem of transitioning to the mode $\xi_{\text{dst}} \in \Xi_{\text{dst}}$ (where ξ_{dst} is the mode determined by the co-parameter $\chi_{\text{dst}} \in X^{\Xi_{\text{dst}}}$) from the mode $\xi_{\text{src}} \in \Xi_{\text{src}}$ (where ξ_{src} is the mode determined by the co-parameter $\chi_{\text{src}} \in X^{\Xi_{\text{src}}}$). Specifically, the weights are inversely proportional to the probability of single-mode motion planning succeeding within a fixed time budget on the leaf manifold $\mathcal{L}_{\chi_{\text{src}}}$, $\chi_{\text{src}} \in X^{\Xi_{\text{src}}}$ to any transition configuration (that may or may not exist) on the leaf manifold $\mathcal{L}_{\chi_{\text{dst}}}$, $\chi_{\text{dst}} \in X^{\Xi_{\text{dst}}}$.

Recall from Def. III.1 that the transverse manifolds for a foliation are parameterized. The transverse weights are a distribution maintained over the Cartesian product of the underlying spaces $X^{\Xi_{\text{src}}} \times X^{\Xi_{\text{dst}}}$ for a $k_{\Xi_{\text{src}}} + k_{\Xi_{\text{dst}}}$ dimensional space.. An example of these weights is visualized in Fig. 7b. However, these distributions are not known *a priori*. Estimating these distributions online is key to effective multi-modal planning. We cover estimating these distributions in Sec. VI-C.

Consider the example shown in Fig. 7. A robot with two end-effectors that can climb by grasping bars must position its base in a goal region above bar 2 ("Base_Goal"). The graph contains all possible transitions; there is no transition from "{Left, Right}_Bar1" to "Base_Goal" because it is out of reach, and none between the same end-effector on a bar as the robot must always be grasping a bar. The obstacle on one side of the bar makes movement more difficult, corresponding to a higher weight (seen in the distribution in Fig. 7b and c). Intuitively, this shows that if the robot is trying to transition from "Right_Bar1" at the grasped location to "Left_Bar2", then it is important to consider where it grasps Bar2. Note that the weights for Fig. 7b were generated by many offline runs, while our experiments approximate this distribution online without precomputation.

C. Informing the Transition Graph

As mentioned, the transition weights are meant to capture the likelihood that single-mode planning will be successful in moving from one ⟨mode family, co-parameter⟩ pair to the next. We update the weights using a simple weighting scheme, based on information gathered during the search. There are three events that we can use to update the information, given a single-mode planning instance of $\langle \Xi_{\text{src}}, \chi_{\text{src}} \rangle$ to $\langle \Xi_{\text{dst}}, \chi_{\text{dst}} \rangle$.

- *Planning is Successful*: We add a small penalty δ_{success} to the weight to encourage exploration of alternate routes.
- *Planning is Unsuccessful*: We add a larger penalty δ_{failure} to the weight to discourage attempting this transition again.
- *No Transition Found*: That is, no transition configuration was sampled from $\langle \Xi_{\text{src}}, \chi_{\text{src}} \rangle$ to $\langle \Xi_{\text{dst}}, \chi_{\text{dst}} \rangle$. We add the largest penalty to the weight, δ_{sample}.

Moreover, we add the penalty to all "nearby" co-parameters, as defined by the composite distance metric in the joint $X^{\Xi_{\text{src}}} \times X^{\Xi_{\text{dst}}}$ space. The weight δ is distributed by a bump function, given by $\exp\left(1 - 1/(1 - d^2)\right)$, where d is the distance of the co-parameter to $(\chi_{\text{src}}, \chi_{\text{dst}})$, normalized to a maximum distance (a quarter of the space's extent for experiments).

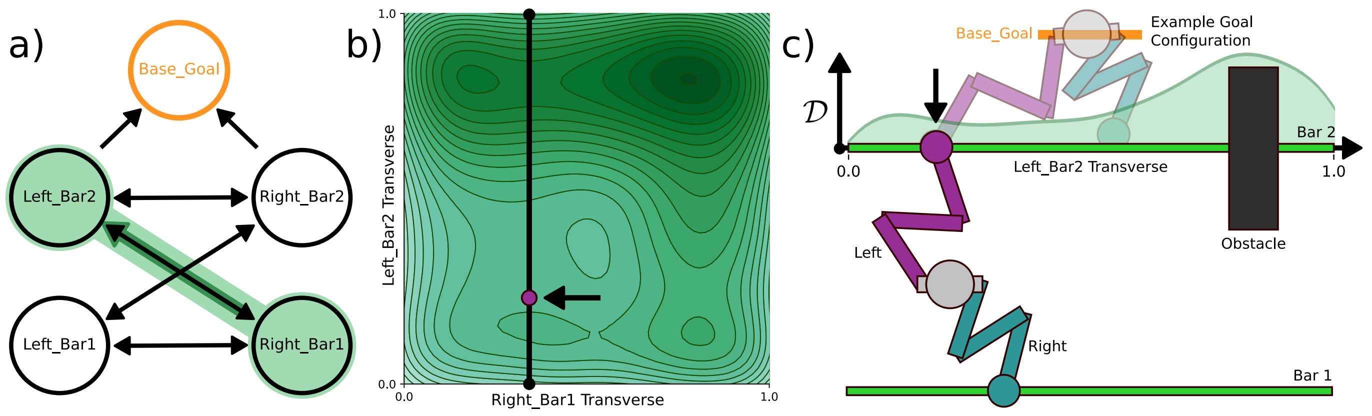


Fig. 7. Visualization of transition graph weights. **a)** A transition graph for the "simple monkey" domain. The transition from grasping bar 1 with the right limb ("Right_Bar1") to grasping bar 2 with the left limb ("Left_Bar2") is highlighted. **b)** A contour map of the weights for the highlighted transition (gathered offline). The co-parameter for grasping bar 1 with the right limb is plotted on the X-axis, while the co-parameter of grasping bar 2 with the left limb is plotted on the Y-axis. Darker shading represent higher-weight (transitions we expect to be difficult). A slice for a specific X-axis co-parameter (i.e., initial placement of the right gripper) is marked in black, and shown in **c)**. **c)** The "simple monkey" domain, with the weights of the slice from **b)** visualized over bar 2. The left arm of the robot is shown grasping bar 2 (highlighted by the arrow), and thus is a transition configuration. The transition is marked at its co-parameters by a purple point in **b)**. To reach "Base_Goal," the robot must grasp bar 2, visualized as an orange region for the base and an example satisfying configuration.

Thus, search is biased not only against attempting this transition again, but also nearby "similar" transitions, as nearby co-parameter pairs will have similar planning conditions due to the assumptions of continuity from transverse of the foliated manifold. We conjecture this weighting scheme maintains probabilistic completeness as we never rule out a possible transition.

Our experiments show that this simple weighting scheme can dramatically improve runtime on a variety of scenes. We used fixed weights for all experiments of 3, 5, and 10 respectively for the three weights. Other weighting approaches are possible, but this simple scheme gave us excellent results.

D. Building Leads

Given a mode family transition graph with suitable weights, we compute leads to bias search. Essentially, the idea of a lead is to suggest the most likely sequence of mode transitions to reach a destination. Here, a destination is a sample at random or from the goal, and the source is an existing configuration in the search tree. The lead generation is used within the multi-modal planning algorithm's EXTENDTREE method. Two variants are used in our experiments: "Dijkstra" and "Augmented", shown in Alg. 5 and Alg. 6 respectively.

In the "Dijkstra" method, we compute a sequence of mode families that is effective for reaching the desired destination, shown in Alg. 5. In practice, this means finding a shortest path in the mode transition graph from the mode of the current state to the mode of the desired destination mode—which results in a sequence of mode families to transition between. For this approach, we simply use Dijkstra's shortest path algorithm over the transition graph (either implicit or explicit). Note that this approach does not use any weighting information over co-parameters, but instead over the transitions between mode families themselves—rather than a distribution over the parameters, a single number is used.

In the "Augmented" method, we compute not only a sequence of likely mode families that will be useful for search,

Algorithm 5 Dijkstra ExtendTree

```
1: procedure EXTENDTREE($\mathfrak{s}_0, s_{\text{goal}}, T$)
2:     Compute shortest sequence of states $\mathfrak{s}_1, \ldots, \mathfrak{s}_k$
       from $\mathfrak{s}_0$ to $\mathfrak{s}_{\text{goal}}$    // We use Dijkstra's algorithm
3:     $\Xi' \leftarrow \mathfrak{X}(\mathfrak{s}_0), \quad \chi' \leftarrow \pi_{\Xi'}(q_0)$
4:     $S \leftarrow \{\}$
5:     for $\mathfrak{s}' \in \mathfrak{s}_1, \ldots, \mathfrak{s}_k$ do
6:         $\Xi^t \leftarrow \mathfrak{X}(\mathfrak{s}')$
7:         if $q' \leftarrow$ SMP$(q_0, \xi_{\chi'}, \Xi^t)$ then
8:             $\chi^t \leftarrow \pi_{\Xi'}(q')$
9:             Append $(q', \mathfrak{s}')$ to $S$
10:            Update $\mathcal{D}^{\Xi',\Xi^t}$ with $\delta_{\text{success}}$
11:        else if NO SAMPLES then
12:            Update $\mathcal{D}^{\Xi',\Xi^t}$ with $\delta_{\text{sample}}$
13:            break
14:        else if PLANNING FAILED then
15:            Update $\mathcal{D}^{\Xi',\Xi^t}$ with $\delta_{\text{failure}}$
16:        $\Xi' \leftarrow \Xi^t, \quad \chi' \leftarrow \chi^t$
17:    return $S$
```

but also co-parameters to those mode families that will likely result in a sequence of successful transitions. Formally, the "Augmented" method solves the following problem:

DEFINITION VI.1: *Augmented Shortest Path Problem* Given a transition graph $\mathcal{G}$, a starting mode specified by a mode family and co-parameter $\langle \Xi_{\text{src}}, \chi_{\text{src}} \rangle$, and destination mode $\langle \Xi_{\text{dst}}, \chi_{\text{dst}} \rangle$, find the lowest cumulative weight path, $\langle \Xi_1, \chi_1 \rangle \ldots \langle \Xi_k, \chi_k \rangle$, that minimizes the total cost:

$$\sum_{i=1}^{k-1} \mathcal{D}^{\Xi_i, \Xi_{i+1}}(\chi_i, \chi_{i+1})$$

Thus, a lead is a sequence of $\langle$mode family, co-parameter$\rangle$ pairs, realized in sequence by single-mode planning. The EXTENDTREE variant that uses the augmented shortest path

Algorithm 6 Augmented ExtendTree

```
1: procedure ExtendTree($s_0, s_{\text{goal}}, T$)
2:     $\Xi' \leftarrow \mathfrak{X}(\mathfrak{s}_0), \quad \chi' \leftarrow \pi_{\Xi'}(q_0)$
       // We use Dijkstra's algorithm over the
       augmented graph, which returns a sequence of
       mode families and co-parameter intervals
3:     Compute shortest sequence of states
       $(\mathfrak{s}_1, [\chi_1^l, \chi_1^h]) \ldots, (\mathfrak{s}_k, [\chi_1^l, \chi_1^h])$ from $(\mathfrak{s}_0, \chi')$ to $\mathfrak{s}_{\text{goal}}$
4:     $S \leftarrow \{\}$
5:     for $\mathfrak{s}', (\chi^l, \chi^h) \in (\mathfrak{s}_1, [\chi_1^l, \chi_1^h]) \ldots, (\mathfrak{s}_k, [\chi_1^l, \chi_1^h])$ do
6:         $\Xi^t \leftarrow \mathfrak{X}(\mathfrak{s}')$
           // Plan to sub-foliation defined by interval
7:         if $q' \leftarrow \text{SMP}(q_0, \xi_{\chi'}, \Xi^t_{[\chi^l, \chi^h]})$ then
8:             $\chi^t \leftarrow \pi_{\Xi'}(q')$
9:             Append $(q', \mathfrak{s}')$ to $S$
10:            Update $\mathcal{D}^{\Xi', \Xi^t}(\chi', \chi^t)$ with $\delta_{\text{success}}$
11:        else if NO SAMPLES then
12:            Update $\mathcal{D}^{\Xi', \Xi^t}(\chi', \chi^h + \chi^l/2)$ with $\delta_{\text{sample}}$
13:            break
14:        else if PLANNING FAILED then
15:            Update $\mathcal{D}^{\Xi', \Xi^t}(\chi', \chi^h + \chi^l/2)$ with $\delta_{\text{failure}}$
16:        $\Xi' \leftarrow \Xi^t, \quad \chi' \leftarrow \chi^t$
17:    return $S$
```

is shown in Alg. 5.

In practice, we do not produce enough samples to make exact computations of this shortest path worthwhile. Instead, we approximate the continuous distribution by discretizing the co-parameter for each mode family into intervals, tracking weights over transitions from interval to interval. For our implementation, we simply use Dijkstra's shortest path algorithm over the discretized mode family transition graph to solve Def. VI.1, generating a lead that consists of a sequence of mode families and intervals of co-parameters.

For both the "Augmented" and "Dijkstra" approach, searching the mode transition graph is time-bounded for efficiency's sake, and the current "best" lead is returned after time runs out. For our experiments, we use a time bound of 5 seconds for both methods.

E. *Putting It All Together*

The complete multi-modal planning algorithm is presented in Alg. 2. In an iteration of planning, a random configuration and mode or goal configuration is sampled with some bias. Using an RRT-like scheme, a node from the existing tree is selected to expand from. A lead is generated from this node to the sample, guiding single-mode search to extend the search tree—we presented three variants of the EXTENDTREE function: one that randomly expands as a baseline (Alg. 3), one that is informed at the level of mode family transitions (Alg. 5), and our method which exploits transition information over the co-parameters of the mode families (Alg. 6). The planner uses a single-mode constrained planner (Alg. 4) to attempt to resolve the desired transitions as proposed by the lead. All successful transitions are added to the planning tree.

VII. Experiments

Both of our contributions are implemented with the Open Motion Planning Library (OMPL)'s [70] general constrained planning framework [13] and use Robowflex [71] for 3D workspace experiments. All experiments were performed on a desktop computer with an Intel® Core™ i7-6700K processor at 4GHz with 32GB of DDR4 RAM at 2400 MHz. Unless otherwise specified, all trials succeeded in planning.

A. ALEF *Experiments*

We evaluate ALEF on an example "monkey" robot tasked with climbing across a set of bars and a "handoff" problem with two manipulators. We use PRM for all single-mode planning problems under manifold-constraints [13].

1) Monkey Example: First, we illustrate ALEF's ability to improve single-mode planning in a foliation in isolation. The robot is shown in Fig. 8a and has two end-effectors that can grasp the bars. There are two foliations we consider in this problem: one with all grasps of the right limb on bar 1 (the source foliation), and one with all grasps of the left limb on bar 2 (the destination foliation). The co-parameter is the location along the bar the robot has grasped (labeled in Fig. 8a). Problems are generated by randomly sampling grasps on bar 1, which determine which leaf the problem lies on. The goal is to reach the destination foliation. All tested problems within both the test and training sets are solvable.

Fig. 8b shows timing results for planning on 500 randomly sampled problems with a timeout of 30 seconds. Our framework achieves notable speed-up even with a few examples and continues to improve performance as the training set size increases. Additionally, even with few examples, the variance in solution time decreases, showing that our framework learns to solve "hard" problems faster. This is also visible in Fig. 8c, which shows the cumulative distribution of solving the planning problem versus planning time.

Fig. 8d shows the path retrieval and valid state ratio distributions for our framework over the 500 tested queries. The path retrieval ratio is the ratio of how many paths were successfully retrieved for all start/goal query pairs. A ratio of 1 means that all queries had relevant experience retrieved from the roadmap, while 0 means that no relevant experience was found. The valid state ratio is the ratio of the states from retrieved paths that were successfully projected onto to the new leaf. A high valid state ratio means that the experience retrieved was "useful" to the new problem. Even with only 10 plans inserted into the roadmap, a high ratio of experience is retrieved. However, the average of the ratio of valid states was low (the peak at 0 in Fig. 8d). As the amount of experience in the roadmap increases so does the ratio of the valid states, improving the performance of ALEF.

We now show how ALEF improves the performance of a multi-modal planner. We test the following variations:

- "None": This is the baseline multi-modal planner (essentially an emulation of [4]).
- "Adaptive": Here, ALEF is learns online for each foliation while the baseline planner is running. The results of every planning query the multi-modal planner makes are

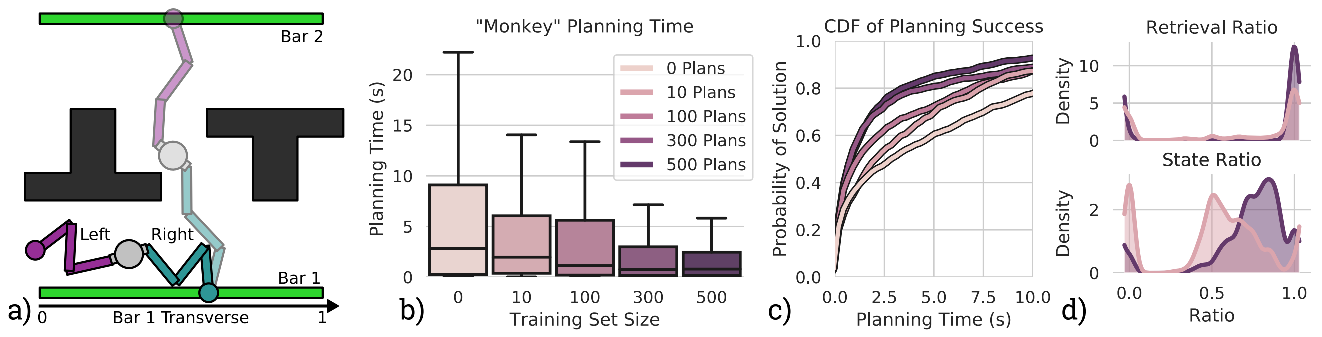


Fig. 8. **a)** A two-limbed "monkey" robot must grasp the other bar. Start configurations are sampled such that the robot grasps the first bar. The location of the robot's grasp on the bar corresponds to the co-parameter of the foliation, indicated below the bar. **b)** Timing results for 500 trials are presented. On the $x-$axis, we show results for ALEF trained on increasing numbers of training paths. We achieve significant speed-up given only a few examples. **c)** Cumulative probability of finding a solution versus time. Our method solves more problems faster as experience is gathered. **d)** Retrieval ratio and valid state ratio for ALEF trained on 10 and 500 examples. More paths are retrieved and the number of relevant experiences increases given more training data.

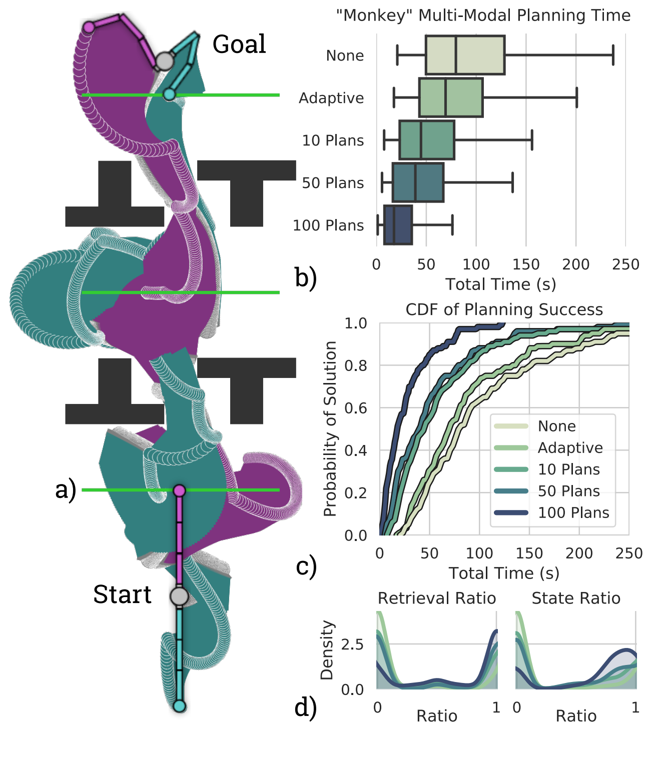


Fig. 9. **a)** The swept volume of a multi-modal plan. The start configuration and an example goal configuration are highlighted. **b)** Total multi-modal planning time for 100 trials for each method, including training time for ALEF. Obstacles vary in position and rotation between each query. Shown are a baseline method with no experience ("None"), ALEF which learns from an empty roadmap ("Adaptive"), and ALEF trained on n multi-modal queries ("n plans"). Starting with an empty roadmap, our method provides a small benefit over the baseline. Training from prior queries gives substantial benefit for multi-modal planning. **c)** Timing results presented as a cumulative distribution curve. textbfd) Retrieval and valid state ratios for all motion planning queries. ALEF retrieves more relevant experience as its training set increases.

inserted into ALEF. The planner can only learn during its current query.

- "n Plans": Here, ALEF was trained incrementally using all motion plans generated during n prior multi-modal planning queries.

Fig. 9a shows an extension of the environment from Fig. 8a, where there are now three bars. The robot is tasked with climbing to reach a goal past the far bar. In this environment, the monkey starts always from the same configuration, but the obstacles are varied between different multi-modal queries. Specifically, each obstacle can rotate 20 degrees around its center and vary in position as much as their thinnest width. Depending on the location of the obstacles, the robot might have to discover an alternate route to the goal, as the middle path might be closed.

Timing results for total multi-modal planning time are presented in Fig. 9b and Fig. 9c. Starting from nothing, the "Adaptive" planner provides a small benefit over baseline performance, showing that our framework helps solve queries faster even with limited experience. Note that these reported times include training time for ALEF, which is negligible. Offline training from other queries gives substantial benefit and accelerates multi-modal planning.

Fig. 9d shows path retrieval and valid state ratio distributions for all single-mode plans made by the multi-modal planner. Here, "Adaptive" does not retrieve much experience, and the experience it retrieves is typically unhelpful, given the low valid state ratio. As our method is trained on more plans, the retrieval ratio and valid state ratio both increase significantly, which is corroborated by their performance.

2) Handoff Example: Fig. 10a shows a "handoff" environment, where an object must be transferred from one side to the other (stills 1 and 3 in Fig. 10a). However, due to the length of the manipulators and the obstacle in the middle, the object must be handed off (still 2 in Fig. 10a). Additionally, the end-effector of the manipulator is constrained to always remain upright. Here, there is a mode family for grasping the object anywhere along its length, and another mode family for placing the object anywhere on the flat surface. Similar to before, the problem starts from the same configuration, but the gray obstacle varies between queries. Here, the gray obstacle can vary up to 20 degrees about its center, and in position by the height of its peak in both the x- and y-axes.

Results for total multi-modal planning time are presented in Fig. 9b. As before, the "Adaptive" planner provides a small benefit over baseline, while training ALEF with multiple queries gives dramatic performance improvements. This example reveals the generality of our framework.

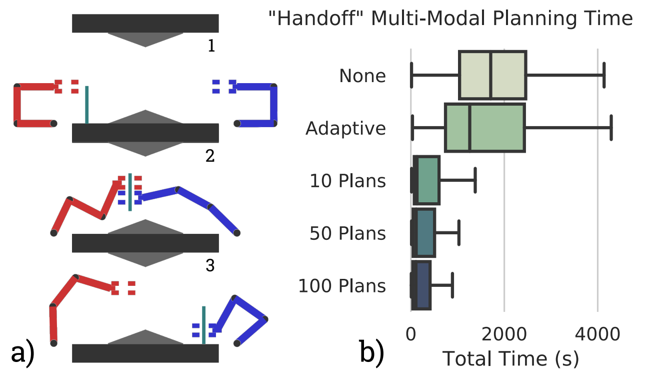


Fig. 10. **a)** Three stills from a multi-modal plan in the “handoff” environment. The thin object must be handed off to the far robot to reach the other side of the environment. Many motion plans are attempted in this environment, as the choice of parameterization determines if a handoff is possible. **b)** Total multi-modal planning time for 100 trials for each method, including training time for ALEF. Obstacles vary in position and rotation between each query. ALEF shows significant improvement over the baseline method.

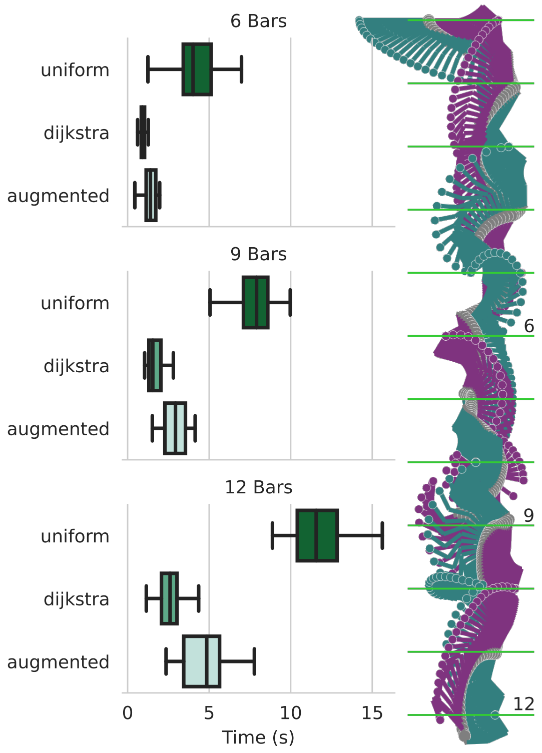


Fig. 11. Timing results for three “long monkey” domains. On the right the “12 Bars” version of the environment is shown, along with the swept volume of by an example multi-modal plan. On the left, timing results for “6 Bars,” “9 Bars,” and “12 Bars” are shown for *Uniform*, *Dijkstra*, and *Augmented*—each box plot represents 30 trials. Note that as the problems become more challenging (i.e., there are more bars and thus a longer sequence of transitions needed), the benefits of using the discrete leads become more pronounced. In this scenario the specificity of *Augmented* does not provide any benefits over *Dijkstra*, as the location of the grasps matters less than the sequence.

B. *Guiding Lead Experiments*

The following experiments are chosen to measure the importance of the discrete component (building leads) and continuous component (updating transition weights) of our algorithm in various scenarios. Note that in all of the following experiments, ALEF is not used. We compare three algorithms:

1) *Uniform*, which chooses mode transitions uniformly at random from the neighboring modes (essentially, an emulation of [4] by our planner). This uses the EXTENDTREE presented in Alg. 3.
2) *Dijkstra*, which searches for a sequence of mode family transitions, but does not select co-parameters. This uses the EXTENDTREE in Alg. 5.
3) *Augmented*, our proposed method which searches for a sequence of mode family transitions as well as co-parameters. This uses the EXTENDTREE in Alg. 6.

Uniform is the baseline. Many manipulation problems have challenging discrete structure, in that many transitions are necessary to achieve the task. Intuitively, *Dijkstra* should perform well when this discrete structure is important, because good choices of which mode to transition to make the search much more efficient (e.g., Fig. 11, Fig. 15a). *Augmented* should perform well relative to *Dijkstra* when the continuous co-parameters are relatively important (e.g., Fig. 12, Fig. 17). Using the augmented transition graph also allows us solve difficult manipulation problems (see Fig. 14) more efficiently.

1) Handrail Climbing: Fig. 11 shows results for the “long monkey” domains. Here, a “monkey” robot with two end-effectors grasps handrails to move through the environment (similar to Fig. 7). The “monkey” has nine degrees of freedom (three for each arm and three for the pose of the base). There are two mode families for each bar, corresponding to grasping the bar with either end-effector—the co-parameter corresponds to the location grasped on the bar. Fig. 11 shows that *Dijkstra* works well when the discrete aspect of the problem is the primary challenge. Intuitively this makes sense as the robot must traverse many bars—choosing a good order makes the problem significantly easier. *Dijkstra* helps the robot choose *which* rung to grasp.

Fig. 12 shows results in the “lateral monkey” domains, each domain increasing in clutter. There are eight bars the monkey can grasp in each domain, with the same monkey robot as before. Fig. 12 shows that *Augmented* improves significantly over *Dijkstra* in problems where choosing the right co-parameters becomes a significant aspect of the problem. While similar to the above problem, the bars are much longer; the greater latitude means *where* the robot grasps the bar (the co-parameter) is much more important in this problem.

2) Robonaut 2: Fig. 13 shows results for NASA’s Robonaut 2 in a similar handrail climbing domain as presented before for the “monkey” robot. Here, Robonaut 2 has two seven DOF limbs, a single DOF waist joint, and is free-flying, for a total of 21 DOF. The mode families are identical to the “monkey” domains. Moreover, Robonaut 2 must keep its waist upright and facing forward within a small angular tolerance, adding an additional manifold constraint to each mode family. Fig. 13b shows results for both *Dijkstra* and our proposed *Augmented* method—*Uniform* was unable to solve this problem within the allotted time of 1200 seconds, due to the long-horizon of the task and complexity of single-mode planning. *Augmented* was able to outperform *Dijkstra*

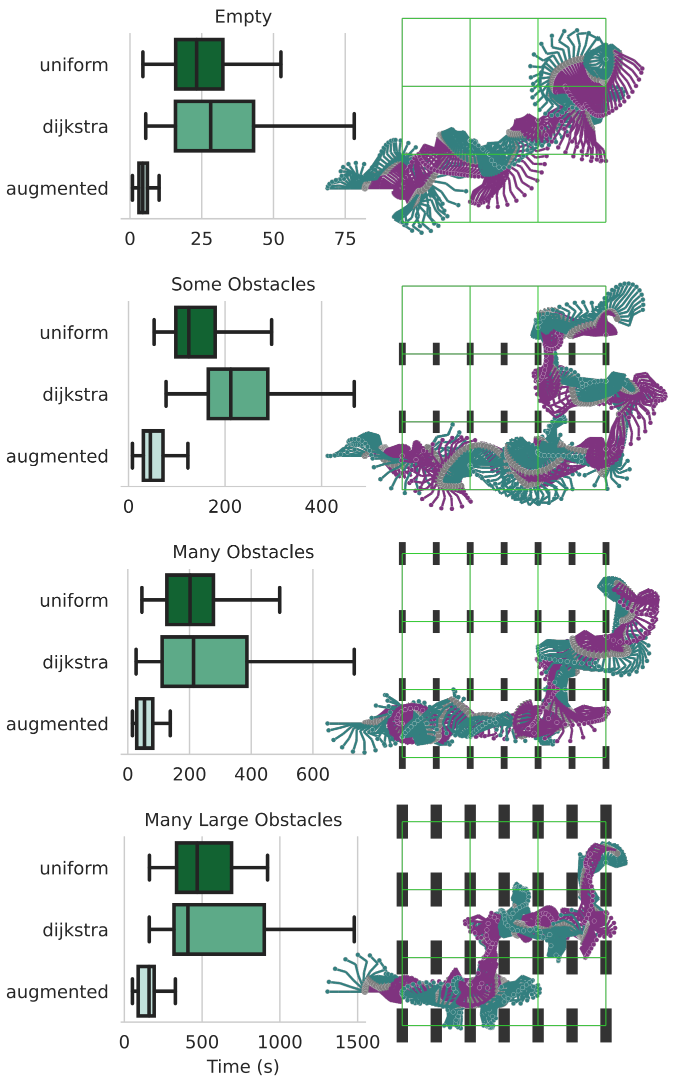


Fig. 12. Timing results for "lateral monkey" domains. On top, the swept volume of an example multi-modal plan is shown for each of the domains. Obstacles are shown as dark rectangles, and the graspable handrails are shown in green. On bottom, timing results for each environment are shown: each dot corresponds one of the 30 trials done for each scenario and planner. As the problems become more challenging (i.e., the clutter in the scene increases), the benefits of *Augmented* become more pronounced. This happens because as the weighting scheme helps bias away from known failure regions and toward unexplored transitions. Note the Y-axis changes between each plot.

in this environment due to the presence of the obstacles which prevented certain transitions from occurring. *Augmented* is able to more effectively focus search by learning the co-parameter distributions.

3) Handoff Example: Fig. 14 shows results for the "tall handoff" domain. In this case, there are two translating end-effectors that can grasp a long object. Here, there are seven mode families: five corresponding to the placement of the object along the flat surfaces, and a mode family corresponding to each end-effector grasping the object—the co-parameter corresponds to where the end-effector grasps the object. Fig. 14 shows that we can solve complex TAMP-like problems. This scenario is difficult due to obstacles preventing the object from being removed from the narrow passage while grasped, necessitating a sequence of handoffs to reach the goal. The robot needs to repeatedly place the object in order to regrasp the object, so it may be handed off. This combines a difficult motion planning problem with the discrete structure inherent to the problem. Our approach can solve problems intractable in typical TAMP frameworks (e.g., [40]), as search looks for short task plans. Such plans are not feasible, so these methods waste time by assuming the "best" plan has fewer actions, while our approach inherently biases towards solving the motion planning problem using discrete structure as a guide.

4) Fetch Block World: Fig. 15 shows results for two experiments using an 8-DoF Fetch manipulator [72]. Note that in these experiments, *Uniform* was unable to solve any problem within the allotted time of 60 seconds. In these problems, the implicit mode graph of the problem is important for scalability, as there are an exponential number of mode families and transitions as objects are added to the problem—a truncated version of the domain specified in PDDL is presented in Fig. 16. Moreover, explicit constraints are leveraged heavily in this domain to avoid the exponential blow-up of considering all block configuration variables—object variables are factored out when placed or grasped, and thus, only the robot configuration needs to be considered.

Fig. 15a shows a complex, long-horizon task planning problem. The Fetch must reorganize the blocks on two tables into an ordered tower. Here, *Dijkstra* outperforms the *Augmented* method, as there are no complex geometric elements to consider: motion planning is straight-forward (as in Fig. 11). However, when the geometry of the problem is important, as in Fig. 15b, *Augmented* outperforms *Dijkstra*. In this problem, a tower needs to be constructed from the three long blocks on two tables. If the tower is started too close or too far from the Fetch, it will be unable to complete the tower. Moreover, to complete the tower, the last block cannot be grasped from the top. *Augmented* method here quickly learns which of these transitions are infeasible.

Finally, Fig. 17 shows results for a complex "puzzle" domain. The Fetch must maneuver the dumbbell-like object first to the middle of the puzzle box (which can only be done from certain grasps), so that it can be regrasped near the top to be extracted from the box. As above in Fig. 14, *Augmented* outperforms *Dijkstra* given the importance of geometry in the problem.

VIII. Conclusion

We have presented two contributions to improving the efficiency and scalability of multi-modal planning: ALEF, an experience-based learning framework for multi-modal problems, and a guiding lead informed by weights over co-parameters.

ALEF builds a sparse roadmap in an augmented, manifold-constrained space that considers the parameters of the mode family and configuration space, and uses the roadmap to retrieve experience as a sampler on future queries. Our framework achieves significant speedup given only a few examples and improves the performance of multi-modal planning. ALEF increases in usefulness as planning on the constraint manifold

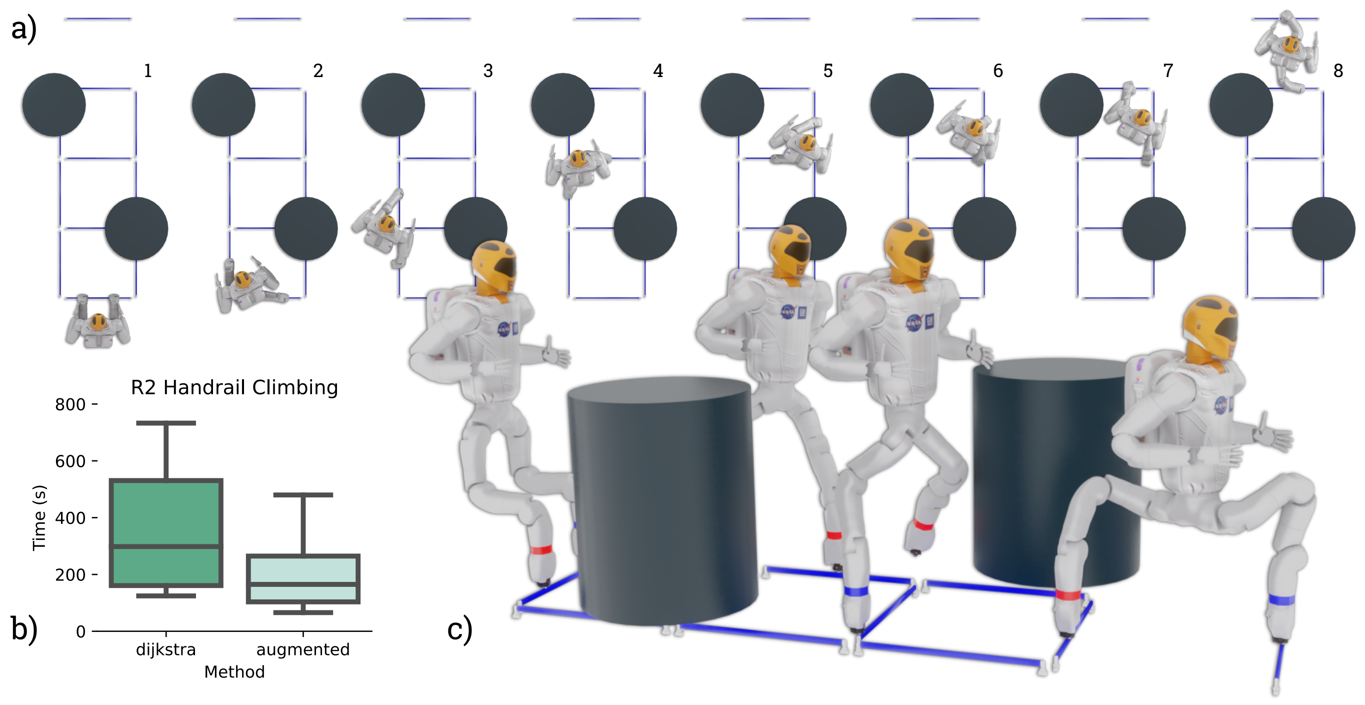


Fig. 13. Timing results for NASA's Robonaut 2 climbing in a handrail domain with obstacles. **a)** shows the sequence of transition states found in one trial of the multi-modal planner. **b)** shows timing results for 30 trials of both *Dijkstra* and *Augmented*. *Uniform* was unable to solve this problem with 1200 seconds per trial for all 30 trials. *Augmented* achieves more consistent performance due to learning to avoid obstacles and challenging transitions. **c)** shows another view of the same trial shown in **a)**.

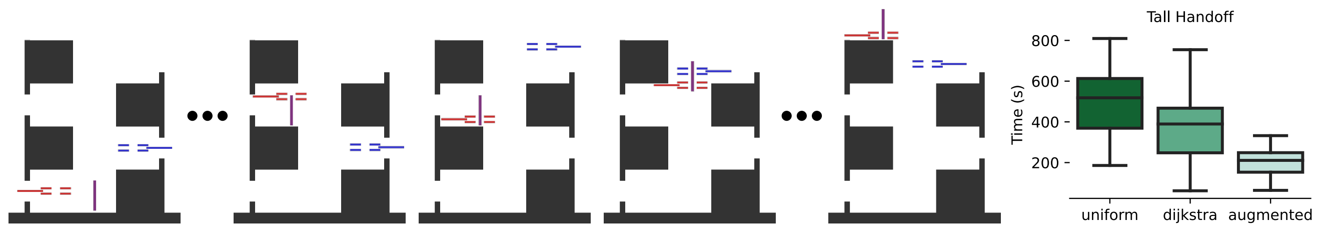


Fig. 14. Timing results for the "tall handoff" domain. On the left, five stills are shown from a multi-modal plan in this domain. The start and goal are shown on the far left and right respectively. The end-effectors are red and blue, the graspable object is purple. An example of a necessary regrasp to achieve a handoff is shown in the middle. Timing results are shown on the right, 30 trials for each planner. Even in difficult TAMP-like scenarios, *Augmented* learns important information about feasibility to make search more efficient. *Uniform* had four failures given 1200 seconds, which are not included in the box plot.

defined by the foliation increases in difficulty, e.g., due to non-linearities, dimensionality, or clutter. Moreover, we believe that using an experience-based planning framework is essential for efficient manipulation planning, given the amount of redundant effort in single-mode planning. Currently, ALEF can adapt to changing environment obstacles, but not changes in the foliation itself. For example, consider a foliation for grasping a bar—if the bar moves in the workspace, the underlying foliation constraint function will change as well, invalidating experience. While small changes in the foliation will most likely result in usable experience, it is not clear how to associate this experience between larger problem changes, e.g., between two bars nearby but of different lengths. In the future, we would like to improve our method by considering workspace features such as in [8] to generalize to retrieving experience from multiple, similar foliations to apply to novel problem instances. We would also like to investigate performance when the co-parameter is less informative. For example, in the bar grasping example, if the bar is very long relative to the size of the robot, experience is applicable in a much smaller neighborhood.

Our guiding leads enable us to inform search across a wide variety of problems with continuous modes. Both the lead and its choice of continuous parameters yield more efficient planning, which enables scaling to complex problems, such as the 21-DoF handrail climbing example with NASA's Robonaut 2. Guiding leads, like many guiding heuristics, are most useful for long-horizon tasks that have complex structure. As demonstrated in our experiments, information about continuous parameters only provides benefit when the parameterization is important for the task, e.g., due to obstacles or goal preconditions. This is shown in experiments where the *Dijkstra* variant outperforms *Augmented*, and vice versa. Adaptively choosing the amount of information that should be considered at each transition would improve efficiency and efficacy of lead generation, and is left for future work. In the future, we would like to improve the representation of

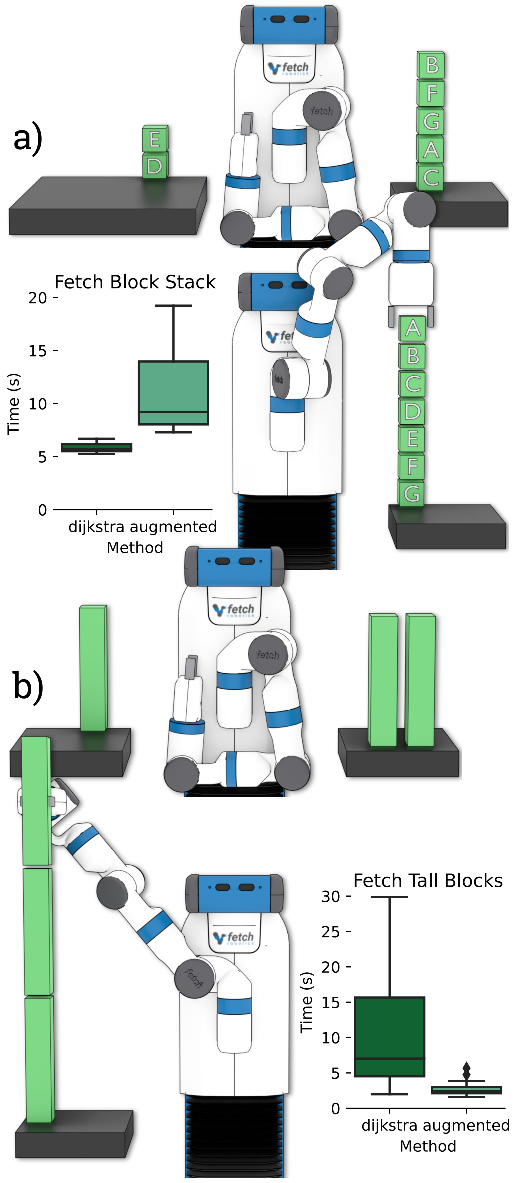


Fig. 15. In **a)**, an ordered tower needs to be built from an initial random configuration of the initial block towers. In this case, the specificity of the *Augmented* works causes it to perform worse than *Dijkstra*, as the placement of the blocks is not as important as the task itself. **b)** poses a block stacking problem with "long" blocks that the Fetch can grasp anywhere along the block's length. Placement of the tower and the grasp of the block are important, as the tower cannot be completed if started too far or close to the Fetch, and the last block cannot be grasped from the top. Here, the *Augmented* method outperforms *Dijkstra*, demonstrating that when geometry is important, our co-parameter learning is useful. *Uniform* was unable to solve either problem in the allotted time of 60 seconds. Timing results represent 30 trials.

the weights on co-parameters to a continuous representation, and investigate transferring learned weights to novel problems similar to experience-based methods. Moreover, our method provides feasible multi-modal motion, but not "optimal" motion with respect some cost (e.g., minimizing the number of transitions, total path length). Although our single-mode plans are heuristically optimized [73], how to effectively optimize multi-modal paths is an open question, and a fruitful line of future work, e.g., similar to the heuristics used in [12] or joint optimization [74]–[76].

```
(define (domain tabletop-blocks)
  (:types block table)
  (:predicates (on ?x - block ?y - block)
               (ontable ?x - block ?y - table)
               (clear ?x - block)
               (handempty)
               (holding ?x))

  ;; … Standard pick-up, place, stack, unstack
      actions …

  ;; When a block is placed on a table
  (:family placed
           :parameters (?x - block ?y - table)
           :condition (ontable ?x ?y))

  ;; When a block is grasped by the robot
  (:family held
           :parameters (?x - block)
           :condition (holding ?x))

  ;; When a block is stacked on another block
  (:family stacked
           :parameters (?x - block ?y - block)
           :condition (on ?x ?y))
  )
```

Fig. 16. The PDDL domain used in block stacking experiments (including Fig. 17). This is a simple block world domain—the standard "pick up," "place," "stack," and "unstack" actions are not shown. There are three classes of mode families here: when a block is placed, held, or stacked on another block. Each of these mode families has a specific grounding according to the relevant grounded expression in `:condition`. All mode families with satisfied conditions are composed together into a composite mode family.

We believe both ALEF and weight-guided leads to be broadly applicable to other manipulation planners besides our baseline approach, and that in general adaptive and *a priori* biased methods will be essential to scalability of these planners. Currently, our methods require extensive modeling of the environment—every foliation requires definition of a function, Jacobian, and bundle projection. For explicit constraints, information about which configuration space variables are controlled is also required. A PDDL description of the domain and when foliations are applied is also required, as well as a problem description that contains all geometry. Lowering the burden of specification either through learning [77] or other means is of interest for realistic usage of these methods, and to deploying complex model-based manipulation planners in general. Additionally, while our abstract constraint representation provides great benefits for combining constraints, these implicit constraints become very difficult to solve for in complex scenarios, e.g., the combination of handrail grasp constraints and waist upright constraints for Robonaut 2.

In the future, we want to investigate the performance of combinations of approaches to improve single-mode planning and task guidance, including our own proposed methods. Moreover, we want to push the scalability of these methods by increasing the dimension of the co-parameters and the branching factor of the discrete task. In this work, we only investigated foliations with a transverse dimension of 1 and 2. Scaling to high-dimensional co-parameters (e.g., $SE(2)$, $SE(3)$) will introduce challenges with requiring more data in order to be informative over the larger volume of space—both

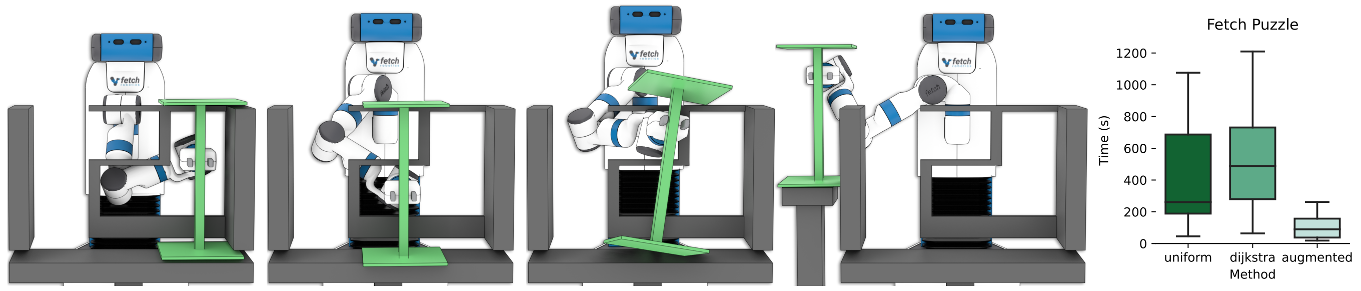


Fig. 17. A complicated "puzzle" domain for the Fetch block stacking problem. Here, a dumbbell-shaped object can be grasped anywhere along its spine. The Fetch must move the dumbbell from the initial configuration to the middle of the box, so it can be regrasped from the top to be extracted from the box. Similar to Fig. 14, given the importance of grasp and placement location, *Augmented* outperforms the other methods. Timing results represent 30 trials.

ALEF and the guiding leads are affected by this dimensionality (recall as well that guiding leads weight in the *product space* of the source and destination transverse). Using low-dimensional projections (e.g., as in KPIECE [78]) might be effective for guiding search.

References

[1] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robot., and Autom. Syst.*, vol. 1, no. 1, pp. 1–28, 2018.

[2] M. A. Diftler, J. S. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. A. Permenter, B. K. Hargrave, R. Platt, R. T. Savely, and R. O. Ambrose, "Robonaut 2—the first humanoid robot in space," in *IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2178–2183.

[3] W. Baker, Z. Kingston, M. Moll, J. Badger, and L. E. Kavraki, "Robonaut 2 and you: Specifying and executing complex operations," in *IEEE Wksp. on Advanced Robot. and its Social Impacts*, 2017, pp. 1–8.

[4] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *Int. J. of Robotics Research*, vol. 30, no. 6, pp. 678–698, 2011.

[5] J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *IEEE Int. Conf. Robot. Autom.*, 2013, pp. 1799–1806.

[6] W. Vega-Brown and N. Roy, "Asymptotically optimal planning under piecewise-analytic constraints," in *Int. Wksp. on the Algorithmic Foundations of Robotics*. Springer, 2016, pp. 528–543.

[7] D. Coleman, I. A. Sucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *IEEE Int. Conf. Robot. Autom.*, 2015, pp. 900–905.

[8] C. Chamzas, A. Shrivastava, and L. E. Kavraki, "Using local experiences for global motion planning," in *IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 8606–8612.

[9] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *IEEE J. Robot. Autom.*, vol. 34, no. 2, pp. 111–127, 2013.

[10] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy PRM for manipulation planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2000, pp. 1716–1721.

[11] T. Siméon, J.-C. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. of Robotics Research*, vol. 32, no. 7-8, pp. 729–746, 2004.

[12] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *Int. J. of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[13] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *Int. J. of Robotics Research*, vol. 38, no. 10–11, pp. 1151–1178, 2019.

[14] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 469–482, 2010.

[15] Z. Kingston, A. M. Wells, M. Moll, and L. E. Kavraki, "Informing multi-modal planning with synergistic discrete leads," in *IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3199–3205.

[16] Z. Kingston, C. Chamzas, and L. E. Kavraki, "Using experience to improve constrained planning on foliations for multi-modal problems," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2021.

[17] J. J. Kuffner and J. Xiao, *Springer Handbook of Robotics*, 2nd ed. Springer, 2016, ch. Motion for Manipulation Tasks, pp. 867–896.

[18] R. Alami, J.-P. Laumond, and T. Siméon, "Two manipulation planning algorithms," in *Int. Wksp. on the Algorithmic Foundations of Robotics*, 1994, pp. 109–125.

[19] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *IEEE Int. Conf. Robot. Autom.*, 1994, pp. 945–952.

[20] J. Barraquand and P. Ferbach, "A penalty function method for constrained motion planning," in *IEEE Int. Conf. Robot. Autom.*, 1994.

[21] P. Ferbach and J. Barraquand, "A method of progressive constraints for manipulation planning," *IEEE Trans. Robot. Autom.*, no. 4, pp. 473–485, 1997.

[22] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

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

[24] L. E. Kavraki and S. M. LaValle, *Springer Handbook of Robotics*, 2nd ed. Springer, 2016, ch. Motion Planning, pp. 139–162.

[25] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[26] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[27] D. Berenson, S. S. Srinivasa, and J. J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. J. of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, Oct. 2011.

[28] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robot., and Autom. Syst.*, vol. 1, no. 1, pp. 159–185, 2018.

[29] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *IEEE Int. Conf. Robot. Autom.*, 2005, pp. 3874–3880.

[30] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics: Science and Syst.*, 2005, pp. 105–112.

[31] S. S. Joshi and T. Panagiotis, "Non-parametric informed exploration for sampling-based motion planning," in *IEEE Int. Conf. Robot. Autom.*, 2019, pp. 5915–5921.

[32] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian local sampling-based planning," vol. 5, no. 2, pp. 1954–1961, April 2020.

[33] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3671–3678.

[34] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Active model learning and diverse action sampling for task and motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2018, pp. 4107–4114.

[35] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *Int. J. of Robotics Research*, vol. 38, no. 7, pp. 793–812, 2019.

[36] G. Ye and R. Alterovitz, "Guided motion planning," in *Springer Tracts in Advanced Robot.* Springer, 2017, pp. 291–307.

[37] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev, "Learning to plan for constrained manipulation from demonstrations," *Autonomous Robots*, vol. 40, no. 1, pp. 109–124, 2016.

[38] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robot., and Autom. Syst.*, vol. 4, no. 1, pp. 265–293, 2021.

[39] E. Vidal Garcia, M. Moll, N. Palomeras, J. D. Hernández, M. Carreras, and L. E. Kavraki, "Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles," in *IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 8936–8942.

[40] N. T. Dantam, Z. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *Int. J. of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[41] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *Int. J. of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[42] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2016, pp. 383–389.

[43] J. Mirabel and F. Lamiraux, "Manipulation planning: addressing the crossed foliation issue," in *IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4032–4037.

[44] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. von Wichert, and W. Burgard, "Modeling and planning manipulation in dynamic environments," in *IEEE Int. Conf. Robot. Autom.*, 2019, pp. 176–182.

[45] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *Int. J. of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[46] J. Pan, S. Chitta, and D. Manocha, "Faster sample-based motion planning using instance-based learning," in *Int. Wksp. on the Algorithmic Foundations of Robotics*, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 381–396.

[47] T. Siméon, J. Cortés, A. Sahbani, and J.-P. Laumond, "A manipulation planner for pick and place operations under continuous grasps and placements," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, May 2002, pp. 2022–2027.

[48] A. Sahbani, J. Cortés, and T. Siméon, "A probabilistic algorithm for manipulation planning under continuous grasps and placements," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, vol. 2, Sep. 2002, pp. 1560–1565.

[49] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *European Conference on Artifical Intelligence*. IOS Press, 2004, pp. 895–899.

[50] F. Gravot, S. Cambon, and R. Alami, "aSyMov: a planner that deals with intricate symbolic and geometric problems," in *Int. Symp. on Robotics Research*. Springer, 2005, pp. 100–110.

[51] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Int. Wksp. on the Algorithmic Foundations of Robotics*. Springer, 2009, pp. 599–614.

[52] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *Int. J. of Robotics Research*, vol. 37, no. 13–14, pp. 1796–1825, 2018.

[53] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[54] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with multiple action types," in *Int. Symp. on Experimental Robotics*. Springer, 2013, pp. 531–545.

[55] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard, "Optimal, sampling-based manipulation planning," in *IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3426–3432.

[56] W. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *Int. Symp. on Robotics Research*, 2019.

[57] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *IEEE Int. Conf. Robot. Autom.*, 2015, pp. 346–352.

[58] A. Bhatia, M. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robot. Autom. Magazine*, vol. 18, no. 3, pp. 55–64, 2011.

[59] M. Spivak, *A Comprehensive Introduction to Differential Geometry*. Publish or Perish, 1999.

[60] J. M. Lee, *Introduction to Smooth Manifolds*, 1st ed., ser. Graduate Texts in Mathematics. Springer-Verlag New York, 2003, vol. 218.

[61] J. Nocedal and S. Wright, *Numerical Optimization*. Springer-Verlag New York, 2006.

[62] A. Dobson and K. E. Bekris, "Improving sparse roadmap spanners," in *IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4106–4111.

[63] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *J. of Graphics, GPU, and Game Tools*, vol. 10, no. 3, pp. 37–49, 2005.

[64] J. Mirabel and F. Lamiraux, "Handling implicit and explicit constraints in manipulation planning," in *Robotics: Science and Syst.*, 2018.

[65] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[66] E. P. Pednault, "ADL: Exploring the middle ground between STRIPS and the situation calculus," in *Int. Conf. on Principles of Knowledge Representation and Reasoning*, vol. 1, 1989, pp. 324–332.

[67] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL — the planning domain definition language," Yale Center for Computational Vision and Control, Tech. Rep., 1998.

[68] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *J. of Artificial Intell. Research*, vol. 20, pp. 61–124, 2003.

[69] S. Thiébaux, J. Hoffmann, and B. Nebel, "In defense of PDDL axioms," *Aritifical Intell.*, vol. 168, pp. 38–69, 2005.

[70] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robot. Autom. Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[71] Z. Kingston and L. E. Kavraki, "Robowflex: Robot motion planning with MoveIt made easy," *IEEE Robot. Autom. Letters*, 2021, Under Review.

[72] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and Freight: Standard platforms for service robot applications," in *Wksp. on Autom. Mobile Service Robots*, 2016.

[73] R. J. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *Int. J. of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[74] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Int. Joint Conf. on Artificial Intell.*, 2015, pp. 1930–1936.

[75] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Syst.*, 2018.

[76] C. Phiquepal and M. Toussaint, "Combined task and motion planning under partial observability: An optimization-based approach," in *IEEE Int. Conf. Robot. Autom.*, 2019, pp. 9000–9006.

[77] I. M. R. Fernández, G. Sutanto, P. Englert, R. K. Ramachandran, and G. S. Sukhatme, "Learning manifolds for sequential motion planning," *CoRR*, vol. abs/2006.07746, 2020.

[78] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Int. Wksp. on the Algorithmic Foundations of Robotics*. Springer, 2008.

Zachary Kingston received his Ph.D. degree in computer science from Rice University, Houston, TX, in 2021, and is a postdoctoral research associate under the direction of Dr. Lydia E. Kavraki. His research interests lie in robotic manipulation, task and motion planning, and motion planning under constraints.

Lydia E. Kavraki received her Ph.D. degree in computer science from Stanford University, Stanford, CA, in 1995, and is the Noah Harding Professor of computer science and bioengineering at Rice University, Houston, TX. She is an author of over 200 papers and co-authored a robotics textbook: Principles of Robot Motion (MIT Press, 2005). Her work pioneered the area of sampling-based motion planning, and her group developed and maintains the Open Motion Planning Library (OMPL). Prof. Kavraki's research interests include motion planning for continuous and hybrid systems, task and motion planning, mobile manipulation, and applications of robotics methods in biomedicine. She is a Fellow of the IEEE, AAAI, AIMBE, ACM, AAAS, and a member of the National Academy of Medicine.