Using Experience to Improve Constrained Planning on Foliations for Multi-Modal Problems

Zachary Kingston, Constantinos Chamzas, and Lydia E. Kavraki

Abstract—Many robotic manipulation problems are multi-modal—they consist of a discrete set of mode families (e.g., whether an object is grasped or placed) each with a continuum of parameters (e.g., where exactly an object is grasped). Core to these problems is solving single-mode motion plans, i.e., given a mode from a mode family (e.g., a specific grasp), find a feasible motion to transition to the next desired mode. Many planners for such problems have been proposed, but complex manipulation plans may require prohibitively long computation times due to the difficulty of solving these underlying single-mode problems. It has been shown that using experience from similar planning queries can significantly improve the efficiency of motion planning. However, even though modes from the same family are similar, they impose different constraints on the planning problem, and thus experience gained in one mode cannot be directly applied to another. We present a new experience-based framework, ALEF, for such multi-modal planning problems. ALEF learns using paths from single-mode problems from a mode family, and applies this experience to novel modes from the same family. We evaluate ALEF on a variety of challenging problems and show a significant improvement in the efficiency of sampling-based planners both in isolation and within a multi-modal manipulation planner.

I. INTRODUCTION

Solving manipulation planning problems can be complex and time-consuming, as both a sequence of actions and their corresponding valid motions must be found. During search, a manipulation planner will evaluate many different variations of an action (e.g., different grasps and placements of an object), as a variation may not have a corresponding feasible motion (e.g., due to obstacles). To find valid motions, manipulation planners such as task and motion planners [1] or multi-modal planners [2], [3] use motion planning algorithms as a subroutine. Thus, improving the efficiency of motion planning improves the efficiency of manipulation planning.

Many manipulation problems are multi-modal, and contain a discrete set of actions (mode families) that are parameterized by continuous values, e.g., the placement of an object on a table is given by x, y-coordinates. Each parameterization of an action defines a mode which imposes different constraints on the problem, namely a manifold constraint. Moreover, as the parameters are continuous, problems with parameters “nearby” other parameters will be similar, albeit under different constraints. As each mode within a mode family defines similar problem constraints, results from prior plans could be used as experience to improve future queries.

Experience-based planning methods [4]–[6] learn from prior motion planning problems to expedite search in new, similar problems. For these methods, similarity is typically defined in terms of the obstacles in a changing environment, and cannot cope with changing problem constraints. In multi-modal planning, as solutions from different modes lie on disjoint manifolds, experience in one mode cannot be directly applied to another, and thus these methods cannot be used.

This work proposes a novel experience-based framework, ALEF, that can effectively reuse experiences in multi-modal problems to improve the efficiency of manipulation planning. ALEF builds a sparse roadmap within an augmented, manifold-constrained state space which unifies and relates experience gathered from different single-mode problems within a mode family. Our method learns by using paths from these problems to construct the sparse roadmap. Upon a new query, paths from the sparse roadmap are retrieved and used to bias sampling in a sampling-based planner. Learning is quick and can be run online within a manipulation planner. We demonstrate the effectiveness of our approach on challenging manipulation problems with varying environments.

II. PRELIMINARIES

In this work, we consider manipulation planning problems that are parameterized by a continuum of values. We use terminology from multi-modal planning, a type of manipulation planning, to describe these types of tasks (as in [2], [3]). Parameterized actions are mode families, where each parameterization of the action is a mode. We use a multi-modal planner to demonstrate our framework within the context of a manipulation planner (Sec. V-B).

Consider a robot with a configuration space Q. Typically, in motion planning we are interested in finding a collision-free path σ from a point qstart ∈ Q to some region of interest Qgoal ⊂ Q, where σ : [0, 1] → Qfree such that σ(0) = qstart, σ(1) ∈ Qgoal and σ(t) is collision-free for all t.

A. Single-Mode Planning

A specific parameterization of an action defines a mode ξ which imposes a set of manifold constraints on a robot's motion. Manifold constraints are determined by a constraint function Fξ : Rn → Rκξ (1 ≤ κξ < n). Constraints are satisfied when Fξ(q) = 0; the set of all configurations which satisfy the constraint define the mode manifold Mξ:

\[ M^ξ = \{ q ∈ Q | F^ξ(q) = 0 \} \]
Thus, single-mode planning requires finding a valid path in \( \mathcal{M}^k \), an \((n - k^2)\)-dimensional submanifold of \( \mathcal{Q} \). See [7] for more on planning under manifold constraints.

**B. Motion Planning within a Mode Family**

We consider manipulation problems where actions have continuous parameterizations that parameterize how an action is done. Parameterized actions are defined as a *mode family* \( \Xi \), as each parameterization defines a mode. For example, a robot can grasp a bar anywhere along its length: each grasp location is given by a parameter that defines a mode in the family. An example of a mode family is given in Fig. 1a.

The modes from a mode family are similar—crucially, there is continuity between the manifold constraints of a mode family’s modes. This is codified by defining mode families as *foliated manifolds* [8]. A foliated manifold is a manifold with additional structure: there exists a *transverse manifold* \( X^\Xi \) of parameters \( \chi \in X^\Xi \) of dimension \( k \), which parameterizes a set of \((n - k)\)-dimensional *leaf manifolds* (the mode manifolds) \( \mathcal{L}_\chi \) for all \( \chi \in X^\Xi \). A foliated manifold can also be represented as a constraint function, \( F^\Xi \).

A mode \( \xi_\chi \in \Xi \) is defined by a parameter \( \chi \), \( F^\Xi(q) = \chi \). The mode’s manifold is the leaf manifold \( \mathcal{L}_\chi \):

\[
\mathcal{M}^\xi_\chi = \mathcal{L}_\chi = \{ q \in \mathcal{Q} \mid F^\Xi(q) = \chi \}.
\]

An example foliated manifold and a leaf manifold is given in Fig. 1b. The definition of mode families as foliated manifolds enables us to use the general manifold-constrained motion planning framework presented in [9] used by our experience-based framework.

**III. Related Work**

There are many kinds of manipulation planning algorithms, such as Task and Motion Planning algorithms [1], [10]–[12] and multi-modal planning algorithms [2], [3], [13]–[15]. Each algorithm handles the continuity of action parameterization differently, but all use motion planning to determine if a parameterization is valid. See [16] for a survey of techniques.

Sampling-based planners are probabilistically-complete methods able to scale to high-dimensional problems [17]–[19]. To find motions within a mode, there are many approaches for planning under manifold constraints (e.g., [9], [20]). A survey of sampling-based techniques is given in [7].

To improve efficiency, many methods adapt search *online* with information gathered during the same query. For example, the authors of [21] weight different samplers based on their performance. Other methods use prior collision checking to adapt sampling [22], improve solution quality [23], or adapt the local planner [24]. Our method too can be trained online during a single query of a manipulation planner, improving performance during search as well as on future queries.

Other experience-based methods store gathered experiences for later retrieval. Experiences can be retrieved based on start and goal similarity [4], [25] or workspace similarity [5], [6]. These methods can only retrieve experiences that satisfy the constraints they were trained on, and cannot transfer experiences between constraints. Our method uses experience from one mode and transfers that experience to other modes within the same family. The method most similar to ours is THUNDER [4], which also uses a sparse roadmap to store previous paths. However, THUNDER does not consider constraints, retrieves a path for repair rather than sampling, and requires significant processing time to store experiences.

In the context of manipulation planning, learning has been used to infer which action parameterizations are likely to be valid, both offline [26], [27] and online [3]. Other techniques use demonstrations to infer which constraints are needed to perform a task [28] or approximate constraints observed from data [29]. However, these methods learn information about the constraints themselves and do not improve motion planning. More similar to our method, the authors of [30] bias search from demonstrations to solve parameterized constrained problems. Our method learns from prior planning queries within a mode family, either in a manipulation planner or standalone, to improve performance on motion planning problems within the same mode family.
IV. THE ALEF FRAMEWORK

We present the ALEF framework (Augmented Leafs with Experience on Foliations) for experience-based manipulation planning under manifold constraints. ALEF learns using valid paths from leaf-constrained problems and applies this experience to problems constrained by different leaves within the same foliation. That is, experience from one mode can be transferred to another within the same mode family.

From valid paths (Fig. 2a), a sparse roadmap is constructed in an augmented foliated space (AFS) (Fig. 2b). The AFS is a manifold-constrained configuration space with respect to the foliation constraint. Configurations in the AFS are augmented to include their transverse parameters, i.e., what mode/leaf they are in. The AFS and sparse roadmap are explained in Sec. IV-A and Sec. IV-B.

Upon a new 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. 2c), the path is projected onto the current query’s leaf manifold using a projection operator (Fig. 2d). Configurations from this path that are valid are used as samples. Experience retrieval is discussed in Sec. IV-C.

A. Augmented Foliated Space (AFS)

Recall from Sec. II-B that we model mode families as foliated manifolds with constraint functions \( F^\Xi \). 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 \( Q \times X \), where each configuration is indexed with the parameters of the mode it satisfies. This can be seen in Fig. 1c, which visualizes configurations only in \( Q \), and Fig. 1d, which visualizes the same configurations in \( Q \times X \).

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

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

We leverage the general manifold-constrained sampling-based planning framework of [9] 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. 2b). The connections are such that all intermediate states satisfy the aforementioned foliation constraint.

As sampling-based methods require a 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. This weight relates to the relative importance of the transverse parameters versus the configuration. This can be visualized in Fig. 1d as “stretching” the \( X \) dimension. In our experiments, we weight the transverse parameter three times more than the configuration space.

B. 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 [31], 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-C). 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 [9].

The idea of using SPARS2 as a database to store and retrieve experiences was first introduced in the THUNDER algorithm [4]. However, inserting, retrieving, and reusing paths for SPARS2 in the AFS demand different methods as compared to the standard unconstrained roadmap used in [4].

As noted by [4], 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 (visibility radius) and there is a valid connection between them. Additionally, nodes are added as guard nodes when there are 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 [4]—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. Additionally, to improve connectivity, we check if these newly inserted nodes can be connected to the other connected components of the roadmap. Effectively, this increases the visibility radius of SPARS2 (and thus affects SPARS2’s asymptotic near-optimality property), possibly resulting in longer paths but quickly improving roadmap connectivity. This enables ALEF to be trained online within a manipulation planner.

As the sparse roadmap is built in the AFS, edges are added between nodes from different leaves if the edge satisfies the foliation constraint and is collision-free (Fig. 2c). In the example shown in Fig. 1, nodes in the roadmap from problems that were constrained to different lines are connected by edges that correspond to a motion of the manipulator that moves between these lines. Valid connections made in the AFS are...
indicative of possibly valid motions on all leaves in-between two nodes, due to the continuity between the leaves of the foliated manifold. A roadmap generated by ALEF is shown in Fig. 1c. Vertices and edges are colored according to their transverse parameter. An example of this continuity can be seen in Fig. 1d, where a leaf manifold is shown intersecting the roadmap in the AFS.

C. Retrieving Experience from the AFS Roadmap

In manipulation planning, many candidate goal configurations are considered during motion planning in order to find a valid transition from one mode to the next (the intersection of manifolds). ALEF considers all goal configurations that are sampled. Given each start and goal configuration pair, their nearest neighbors in the AFS roadmap are found using the weighted AFS metric. Then, a collision-free path between these configurations within the roadmap is found using A* search (Fig. 2c). As there might be changes within the environment (e.g., changing obstacles between queries), edge validity is lazily evaluated in roadmap search, which invalidates edges as they are discovered. Paths retrieved from the roadmap hopefully contain configurations important for the current planning problem.

Given a retrieved path, the waypoints of the path are projected to satisfy the query’s leaf constraint and are checked for collision (Fig. 2d). For projection, we use gradient descent with respect to the leaf’s constraint function, but any projection operator would suffice (see [9] for further details). It is unlikely that the entire retrieved path is successfully projected onto the new leaf, thus, we cannot use the retrieved path directly. Instead, the waypoints of the retrieved path are used 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. Samples are used with some probability $1 > \lambda > 0$. For all experiments we use $\lambda = 0.5$.

V. EXPERIMENTS

We evaluate the performance of ALEF on a “monkey” robot tasked with climbing across a set of bars and a “handoff” problem with two manipulators. Although these problems have a 2D workspace, they contain complex robots with 9 and 8 degrees-of-freedom (DOF) respectively, and are subject to non-trivial end-effector constraints. ALEF is implemented with the Open Motion Planning Library (OMPL) [9], [32].

We use PRM for all single-mode planning problems under manifold constraints. We demonstrate ALEF’s ability to learn given only a few examples and improve planning over a foliation in Sec. V-A. Then, we show how ALEF improves the performance of manipulation planning by using it within a multi-modal planner (Sec. V-B).

A. Planning in a Mode Family

The robot has 9 DOF and has two end-effectors, shown in Fig. 3a. There are two foliations we consider in this problem: all grasps of the right limb on bar 1 (the source foliation), and all grasps of the left limb on bar 2 (the destination foliation). The transverse parameter is the location on the bar the robot has grasped, shown in Fig. 3a. Problems were generated by randomly sampling grasps on bar 1, which determine the leaf of the problem. The goal is to reach the destination foliation. All problems in both the test and training sets are solvable.

Fig. 3b 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. 3c, which shows the cumulative distribution of solving the planning problem versus planning time.

Fig. 3d 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 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. 3d). As the amount of experience in the roadmap increases so does the ratio of the valid states, improving the performance of ALEF.

B. Multi-Modal Planning

Our framework also improves the performance of a manipulation planner, where there are multiple mode families...
to traverse in a single query. ALEF was implemented in the multi-modal planner of [3]. We test the following variations:

- “None”: This is the baseline motion planner in [3] and does not utilize our framework at all.
- “Adaptive”: Here, ALEF learns online while the multi-modal planner is running. The results of every planning query the multi-modal planner makes are inserted into the roadmap. ALEF can only learn during the current multi-modal planning query. Learning time is included.
- “n Plans”: Here, ALEF was trained using all motion plans generated from n multi-modal planning queries.

1) Monkey Example: Fig. 4a shows an extension of the environment from Fig. 3a with three bars (6 foliations). The robot is tasked with climbing to reach a goal past the far bar. That is, the planner must find a sequence of feasible mode transitions to reach the goal, requiring at least three mode transitions (e.g., from grasping the initial bar to grasping the middle, then middle to far, and then far to the goal). The monkey starts always from the same configuration, but obstacles are varied between queries. Specifically, each obstacle can rotate 20 degrees about its center and vary in position by as much as its thinnest width. Depending on the pose of the obstacles, an alternate route to the goal might need to be taken, as the middle corridor might be closed.

Timing results for total multi-modal planning time are presented in Fig. 4b and Fig. 4c. Starting from nothing, the “Adaptive” planner provides a small benefit over baseline performance, showing that our framework helps solve queries 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.

2) Handoff Example: Fig. 5a shows a “handoff” environment (8 DOF), where an object must be transferred from one side to the other (stills 1 and 3 in Fig. 5a). However, due to the length of the manipulators and the obstacle in the middle, the object must be handed off (still 2 in Fig. 5a). 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. 4b. 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.

VI. CONCLUSION

We have presented our framework, ALEF, for experience-based planning in the context of manipulation planning. ALEF transfers experience between a continuum of manifold-constrained problems, specifically problems that are drawn from a “mode family”, or foliation. ALEF builds a sparse roadmap in an augmented space that makes connections between problems with different constraints and uses this roadmap to retrieve experience as a sampler on future queries. ALEF provides significant speedup in isolation and within a manipulation planner given only a few examples. In the future, we plan to demonstrate ALEF on problems with 3D workspaces using realistic robots, as well as foliations with higher dimensional traverse manifolds. Moreover, we plan to investigate other experience storage methods and apply our method within other manipulation planning algorithms.

REFERENCES


