Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

Motion Planning in Foliated Manifolds using Repetition Roadmap

Jiaming Hu*§, Shrutheesh R Iyer*†§, Jiawei Wang* and Henrik I Christensen*
*Contextual Robotics Institute, University of California, San Diego
La Jolla, CA 92093
{jih189, siyer, jwjoel, hichristensen}@ucsd.edu
†Aurora Innovation, USA

***Abstract*—Numerous classes of robotics motion planning problems involve searching in constrained configuration spaces where the constraints change during different stages of the motion, and these kinds of motion planning problems are named multi-modal problems. The most common method to solve these problems is to represent them as a set of manifolds and search for a trajectory across them. Often, instead of using manifolds alone, foliated manifolds, which are a union of disjoint manifolds, are a better way to model the manipulation problem. However, the complexity of planning in foliated manifolds is significant due to the increased number of manifolds, hard task constraints, and complex environments. To tackle these challenges, we propose an efficient planning framework that leverages a dynamic roadmap structure to learn from accumulated experience acquired during previous planning attempts in similar foliated manifolds. When planning in a new foliated manifold, this experience, captured in configuration distributions and an atlas, which are tangential charts approximating the new manifold with constraints, is effectively utilized to guide motion planning. We demonstrate the framework's performance for manipulation problems with different foliated manifold structures in simulation and real-world scenarios. https://jih189.github.io/RSS2024_planning_in_foliation**

I. Introduction

Non-trivial robot manipulation tasks are typically broken down into sub-tasks, each with unique constraints. Consider the example of a robot executing a pick-and-place task with a cup of water. This task requires distinct motion planning for different phases. When delivering the cup, the robot arm must adhere to the critical constraint of keeping the cup upright to prevent spillage. In contrast, such a constraint is unnecessary during other phases of the task. These varied requirements categorize the task as a multi-modal problem in motion planning. Previous studies [8, 14] have approached these problems by segmenting them into sequential sets of manifolds, each representing a different action phase. As Kingston underscores in their study [25], representing each action phase with a single manifold often lacks precision. A more accurate approach involves the use of foliation, which entails representing a single action phase as a collection of manifolds, each manifold parameterized by a specific variable, such as grasp or placement. In a scenario involving the pick-and-place of a cup of water, a key constraint is maintaining the cup horizontally throughout the entire process. However, this constraint on the robot arm can vary differently based on the selected grasp pose over the object. Consequently, planning within foliated manifolds has garnered increased attention , through methodologies like the Mode Transition Graph [25] and Markov Decision Process [15].

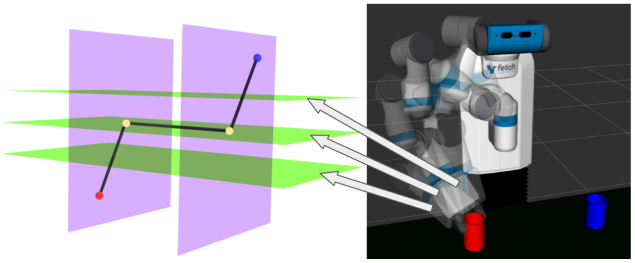


Fig. 1: Delivering a cup horizontally (right) can be considered a foliated manifolds problem (eft). The initial and the target object placements (red and blue) define the un-grasping foliated manifolds—visualized abstractly as purple planes. Each potential grasp introduces another foliated manifold, depicted abstractly as green planes. The problem structure requires finding a path, the black line, across these foliated manifolds from the starting point (red dot) to the endpoint (blue dot), where yellow dots represent intersections between manifolds. It must be noted that the manifolds that are shown as planes in this figure (and subsequent figures) represent an abstract representation of the manifolds (abstract manifold). In practice, they are non-euclidian spaces, with complex structures.

Nonetheless, as the number of manifolds within each foliation increases, the complexity of the motion planning problem escalates accordingly. Recognizing this challenge, [26] emphasized a crucial property of foliations, wherein two foliated manifolds exhibit similarity when their co-parameters are in proximity. Leveraging this property, the Augmented Leafs with Experience on Foliations framework (ALEF) [26] uses solutions derived from foliated manifolds as experiential knowledge to aid in planning within new, similar manifolds. With more examples of solutions for tasks, the performance of ALEF improves. It is worth noting that, however, this approach relies on the availability of existing trajectory solutions for the task to achieve good performance, which imposes limitations on its applicability. In addition, their focus is solely on leveraging successful planning experiences, while experiences of

§Equal contribution

failed planning attempts are disregarded. Consequently, ALEF doesn't guide the motion planner to avoid areas where many planning attempts fail across different foliated manifolds.

Thus, to fully leverage planning experiences from different foliated manifolds, we propose a foliated planning framework, the FoliatedRepMap, that captures the configuration space as a roadmap to encode successful and unsuccessful motion planning experiences. As proposed in ALEF [26], we allow each manifold to contain a roadmap, but we formalize our roadmap to cover the complete self-collision-free configuration space. To represent this space, instead of using a probabilistic roadmap (PRM), which can only encode successful motion plans, we attempt to leverage and extend the notion of the repetition roadmap introduced by [29] that utilizes a Gaussian Mixture Model (GMM).

The GMM decomposes the overall space and represents both valid and invalid regions in the configuration spaces of each foliated manifold. Therefore, after motion planning in different foliated manifolds, the FoliatedRepMap can be updated using gathered sampling data rather than relying on pre-existing solution trajectories. Consequently, in a new planning phase, the FoliatedRepMap offers a lead (i.e. a sequence of foliated manifolds to traverse [25]), where each manifold is accompanied by an approximated indication, learned from the previous experience. During planning in a manifold, those approximated indications provide valuable guidance to navigate, ultimately leading to enhanced performance.

While the planner demonstrates superior performance, we identify two key areas for improvement to (i) tackle hard problems with exceedingly narrow constraint manifold volumes and (ii) improve the efficiency of the planner by focusing on regions of interest. Thus, we extend the FoliatedRepMap by incorporating tangential manifold information as an Atlas [16] to enable more precise modeling of the local manifold region. In addition, we develop a dynamic variant of the planner to cover only relevant parts of the planning space and dynamically adjust its size as exploration continues to improve efficiency.

The rest of the manuscript is organized with related work in Section II. Background information is presented in Section III. The overall problem is detailed in Section IV and the planning solution is presented in Section V. Evaluation results are presented in Section VI.

II. RELATED WORK

A. Task and Motion Planning

Task and Motion planning (TAMP) merges high-level abstract reasoning with low-level motion feasibility checks to accomplish a complex task. Early TAMP methods [19, 31, 21] focus on its hierarchical structure, with some [27] zooming in on constraints. Nowadays, TAMP research, such as PDDLstream [10], aims to find complete feasible solutions within set boundaries, using methods like sampling. The transformer-based method [33] exhibits scalability and potential for handling complex tasks by leveraging large datasets to predict the feasibility of individual sub-tasks. Nonetheless, TAMP attempts to solve problems with many abstract states.

B. Multi-Modal Motion Planning

Planning with multiple sub-tasks with different constraints can be likened to the problem of identifying a solution trajectory within a collection of constraint manifolds. In these classes of problems known as Multi-modal motion planning (MMMP), the robot is in different modes during different phases of action. For a pick and place task, when the robot is holding the object, it is in transfer mode, while it is in transit mode when not grasping [11]. MMMP is a subclass of TAMP, which involves solving long-horizon problems by planning in large hybrid spaces of abstract and geometric states. In MMMP, the motion planner is required to search for a path traversing different modes, with modes parameterizing collision-free spaces of the robot configuration [13]. [13] introduced a method to build local maps for each mode as a manifold, which are then connected to create a roadmap for the entire task. This allows for smooth transitions between different task stages. Later, [14] developed a more advanced technique called Random-MMP. This method systematically searches for paths in various manifolds, using a tree-like structure for efficient planning by moving toward the goal. In PSM* [8], the authors explore the problem of solving a specific subclass of multi-modal planning problems, in which the subtasks performed in a sequence are represented as sequential manifolds. The method can explore trajectories crossing such manifolds even more efficiently.

C. Sampling-Based Motion Planning with Constraints

Motion planning in a single mode has been studied extensively for decades. Sampling-based motion planning, such as the rapidly exploring random tree (RRT) [28] and the probabilistic roadmap (PRM) [22] have become the standard approaches towards motion planning. Many robot manipulation tasks require the robot to generate actions satisfying specific constraints, necessitating constrained motion planning. Within this domain, the prevalent approach is to use constrained sampling-based motion planning. Notable examples include CBiRRT2 [2], the tangent space-based method [30], the Tangent bundle RRT [23] and the Atlas sampling [17]. These methodologies concentrate on generating sampled configurations that fulfill constraints and expanding the tree towards these configurations within the constraint manifold. However, they focus on planning within the single manifold.

D. Experience-based motion planning

To improve the efficiency of planning repeatedly in similar environments, a class of approaches, known as experience-based planners aim to store and leverage past planning attempts to help guide future planning. [3] introduced the Lightning framework, which creates a path library to retrieve and repair similar paths for new queries. [7] later expanded upon this to develop the THUNDER framework that stores experiences in a sparse roadmap for efficient retrieval. Several learning-based approaches have also been developed that decompose the workspace into local regions that can be reused [4], or use deep-learning based approaches to retrieve

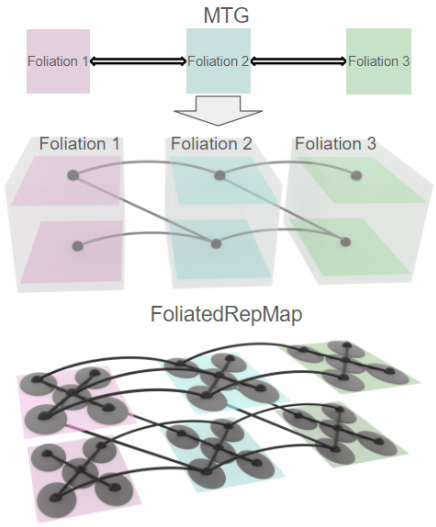


Fig. 2: MTG vs FoliatedRepMap: FoliatedRepMap uses GMM (gray ellipses in the planes) to partition the configuration space, achieving a higher granular level than MTG due to its more detailed representation of foliated manifolds.

relevant experiences conditioned on the environment and the query [5]. Our work is inspired by the Repition Roadmap (RepMap) [29], which learns sampling distributions that are problem invariant in the form of Gaussian Mixture Models. The RepMap constructs an abstract roadmap in the form of a probability distribution to consolidate all prior experiences of the motion planner and searches for a sequence of distributions, given a new query, from which configurations can be sampled to guide motion planning. [18] expand on this framework by encoding the robot configuration space as a set of dictionaries using a Transformer network.

E. Planning in Foliated Manifolds

Foliated manifolds are a class of manifold structures where a group of parallel manifolds share certain constraints. When problems become complex, the number of manifolds can grow significantly, making planning difficult. To address this, [25] proposed using a mode transition graph (MTG) to simplify the problem. Each MTG node represents a foliation, while an edge represents a connection between two foliations. The difficulty in transitioning between manifolds is captured as edge weights. When the MTG planner fails to find a motion plan for a transition between M_a and M_b, the weights of all edges connecting manifolds similar to M_a and M_b increase. This helps replan by providing a new manifold sequence different from previous failed ones. [15] instead, attempts to use a Markov Decision Process (MDP) to traverse the foliated manifolds by dynamically updating the edge weights based on transition probability computed during the previous planning iteration. However, these methods does not retain motion planning information. To improve this, [26] introduced ALEF, which uses solution trajectories from previous motion planning to help with the next query, making the process more efficient. However, it only keeps successfully planned motion paths and discards configurations that do not contribute towards the plan. This design limits its applicability since it requires prior successful multi-modal queries to initialize the system to achieve good performance.

As illustrated in Fig. 2, the FoliatedRepMap advances the MTG by integrating a GMM-based roadmap. This enhancement refines the roadmap's granularity, enabling it to encompass a broader spectrum of information. Consequently, this enriched detail significantly improves the guidance provided for motion planning within foliated manifolds.

III. PRELIMINARIES

This section gives a brief overview of relevant concepts and notations used in this domain. The notations are derived from the previous works, most notably [25] and [26].

A. Motion Planning on Constraint Manifolds

A constraint manifold is an implicit configuration space within the ambient configuration space, which is defined by a constraint function $F(q)$. Mathematically, given a configuration space Q, a manifold M can be represented as following

$$M = \{q \in Q | F(q) = 0\}$$

Given a constraint manifold $M_c = \{q \in Q | F_c(q) = 0\}$, the constrained motion planning requires a projection operator $Project(q, M_c)$ that takes a configuration q_i and project it to a nearby configuration $q_{proj} \in M_c$, as shown in part (a) of Fig. 3, where an iterative optimization method [1] is used,

$$q_{n+1} = q_n - J_{M_c}(q_n)^+ F_c(q_n)$$

where $J^+_{M_c}$ is the Jacobian pseudo-inverse of the constraint function F_c. It is important to note that this projection function may not always yield a valid configuration that satisfies the constraint due to the pseudo-inverse operation. Consequently, a critical factor influencing the cost of constrained motion planning is the success rate at which the initial sampling point can be successfully projected onto the manifold.

To improve this rate, we incorporate the concept of an atlas, drawing inspiration from the field of differential geometry where a manifold is represented as a collection of charts, each amenable to approximation via a tangent space. As demonstrated in [16], using tangent spaces for manifold approximation enables the planner to produce sampled configurations within the constraint manifold locally. Consequently, configurations sampled this way increase the likelihood of successful projection onto the constraint manifold, enhancing constrained motion planning efficiency.

As shown in part (b) of Fig. 3, given a valid configuration q residing on the manifold M_c, its associated tangent space, denoted as T_q, represents a real vector space encompassing all conceivable tangent movements at a point q within the manifold M_c. The tangent space has an orthonormal basis, denoted as Φ_q. Consequently, during motion planning, instead of sampling from the free space as shown in part (a), the sampling process involves selecting a point t from within the

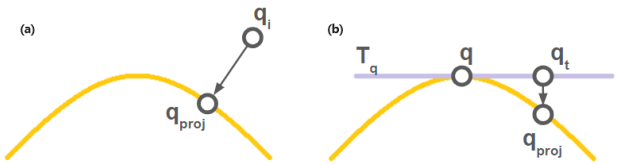


Fig. 3: Orange curves are the constraint manifold M_c, while the blue line is the tangent space T_q. (a) Project q_i to p_{proj} on M_c. (b) Project q_t, mapped from t in T_q, to q_{proj} orthogonally.

tangent space T_q and mapping it to produce a new configuration $q_t = q + \Phi_q t$. Subsequently, the projection function can generate a new valid sampled point, denoted as q_{proj}, derived from the transformed configuration q_t. Therefore, the sampling process is calling a function on t given by,

$$q_{proj} = \psi_{q,M_c}(t) = Project(q + \Phi_q t, M_c)$$

which is a bijective map that locally parametrizes the region around the configuration q on the manifold M_c. Then, a chart c_{q,M_c} is a set of points t such that

$$||q_{proj} - (q + \Phi_q t)|| \leq \epsilon$$
$$||\Phi_q^T \Phi_{q_{proj}}|| \geq cos(a)$$
$$||t|| \leq \rho$$

where ϵ, a, and ρ control the size of the charts, and $\Phi_{q_{proj}}$ is the tangent space basis at q_{proj}. Then all configurations q_{proj} satisfying the above equation define the validity area of the chart c_{q,M_c}. For more details, refer to [24].

B. Foliation

A foliation, denoted as Ξ, constitutes a collection of manifolds, each characterized by a constraint function F^{Ξ} and a corresponding co-parameter set Θ^{Ξ}. More precisely, within a foliation, each manifold is uniquely defined by the combination of the constraint function F^{Ξ} and a co-parameter θ in Θ^{Ξ}, as outlined below:

$$M^{<\Xi,\theta>} = \{q \in Q | F^{\Xi}(q) = \theta, \theta \in \Theta^{\Xi}\}$$

As highlighted in [25], two crucial properties characterize foliations. First, within the same foliation, two manifolds are inherently non-intersecting. Second, manifolds belonging to the same foliation exhibit notable similarity, particularly when their co-parameters closely align.

In this work, all foliated manifolds are subsets of the self-collision-free space. Similar to MTG [25] and ALEF [26], we represent them as a layered structure by using co-parameter as a new dimension as shown in Fig. 4. However, for each co-parameter θ, we merely know the manifold function, but do not know the range of the function, i.e. we do not know the space occupied by the manifold before exploration. We use the abstract foliated manifold to represent where the actual manifold lies. All foliated manifolds are a subset of the self-collision-free space, so we consider all abstract foliated manifolds to be the self-collision-free space here. In other words, all abstract manifolds are the same self-collision-free space and each of them is associated with the co-parameter, that identifies the mode.

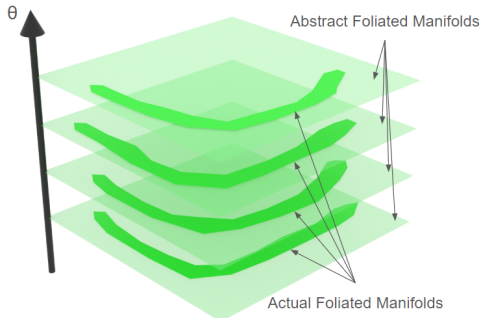


Fig. 4: Abstract foliated manifolds and actual foliated manifolds. The dark green similar regions represents the actual foliated manifolds, while the light green planes represent the abstract foliated manifolds. The arrow with θ is the co-parameter dimension.

IV. PROBLEM DESCRIPTION

Given a set of foliated manifolds $\boldsymbol{\Xi} = \{\Xi^1, \Xi^2, \ldots\}$ and a corresponding list of intersections $\boldsymbol{I} = \{I_1, I_2, \ldots\}$, a foliated manifold planning problem involves the exploration of trajectories, to navigate from a start configuration, q_s, within a foliated manifold $M^{<\Xi_s,\theta_s>}$ to a target configuration, q_g, situated in a different foliated manifold, $M^{<\Xi_g,\theta_g>}$, while adhering to constraints and avoiding collisions. Depending on the application, the intersection can be a configuration, an action, or a trajectory, allowing the state to switch between manifolds. For example, to go from an un-grasping manifold to a grasping manifold, the close-finger action represents the intersection. Initially, the actual manifolds are unknown, so we need to explore them in the abstract manifolds and find the solution trajectory. The generation of these abstract manifolds and intersections will be discussed in the next section. In this work, since the collision-free space can be different for each manifold, $C_{free,M^{<\Xi,\theta>}}$ represents the collision-free space of the manifold $M^{<\Xi,\theta>}$.

Using the example of Fig. 1, for delivering a cup horizontally from start to goal location, the problem is constructed as follows. This task comprises three steps: 1) grasping the cup, 2) delivering it, and 3) resetting the robot. The first and third stages belong to one foliation, with one manifold each for the start and goal locations of the cup, with the cup's location serving as the co-parameter. For the second stage, each possible grasp pose provides a different constraint. Then, this stage belongs to another foliation, with each grasp pose as its co-parameter. As an example, Fig. 1, using an off-the-shelf grasp estimation algorithm, we sample a fixed number of grasp poses that define the foliated manifolds represented by the horizontal green planes. The start and the goal cup locations serve as the start and goal manifolds for the un-grasping foliation, shown as two vertical purple planes. To generate the intersections $\boldsymbol{I}$, for each grasp-placement pair,

using inverse kinematics, actions like grasping (i.e. closing the fingers) and releasing (opening the fingers) are generated. The similarity between manifolds is computed as the inverse of the position difference between the co-parameters of the manifolds. For this example, for the un-grasping manifolds, the similarity function is the placement position differences while the position difference between grasp poses serves as the similarity metric for the slide foliation. Then, each of those actions represents an intersection. It is important to note that this problem-generation step of constructing the manifolds and intersections is done before we start planning.

The objective is to search for a trajectory $\mathcal{T}$ consisting of a list of trajectory segments $(\tau_1, \tau_2 ..., \tau_f)$ where each of them is a trajectory under one foliated manifold as follows:

$$\mathcal{T} = (\tau_1, \tau_2 ..., \tau_f)$$

s.t.

$$\tau_1(0) = q_s, \tau_f(1) = q_g \quad (1)$$

$$\tau_1(t) \in C_{free, M^{<\Xi_s, \theta_s>}} \quad \forall_{t\in[0,1]} (2)$$

$$\tau_f(t) \in C_{free, M^{<\Xi_g, \theta_g>}} \quad \forall_{t\in[0,1]} (3)$$

$$\tau_i(t) \in C_{free, M^{<\Xi, \theta>}} \quad \exists_{\theta\in\Theta^{\Xi}}, \exists_{\Xi\in\boldsymbol{\Xi}}, \forall_{t\in[0,1]}, \forall_{i=2,...,f-1} (4)$$

$$\tau_i(1) = I_j(0), I_j(1) = \tau_{i+1}(0) \quad \exists_{I_j\in\boldsymbol{I}}, \forall_{i=1,...,f-1} (5)$$

Constraint (1) dictates the trajectory should start with q_s and end with q_g. within the start manifold $M^{<\Xi_s, \theta_s>}$. While constraints (2-3) require the initial and final segments of the trajectory to be in the collision-free space of the start manifold $C_{free, M^{<\Xi_s, \theta_s>}}$ and the collision-free space of the goal manifold, $C_{free, M^{<\Xi_g, \theta_g>}}$ respectively. Constraint (4) further imposes the condition that each segment of the trajectory must reside within a certain foliated manifold, ensuring collision-free movement. Lastly, the last constraint (5) underscores the necessity for interconnecting segments from different manifolds with designated intersections from $\boldsymbol{I}$.

A. Generation of Abstract Foliated Manifolds & Intersections

For this work, we need to understand the search space of the problem by pre-generating all the abstract manifolds and intersections for each problem, using a sampling method. For the example of delivering the cup, a set of grasp poses are sampled over the object. Each of these grasps defines an abstract manifold of the delivering foliation, with the grasp pose serving the co-parameter. Similarly, for each type of foliation, a fixed number of co-parameters are sampled and are used to define the corresponding abstract manifold. It must be noted that this pre-generation of abstract manifolds is a necessary step in most prior works [26, 25], to discretize the problem search space. For intersections (black dots in Fig. 5), we iteratively select two randomly connected abstract manifolds from distinct foliations and sample an intersection until a predetermined number of intersections is achieved. For example, if the intersections in the problem are configurations, we can sample them by ensuring they satisfy the constraints

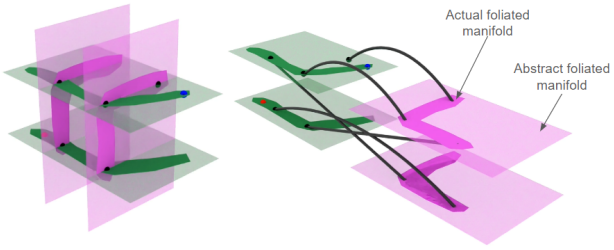


Fig. 5: Different perspectives of a foliation with intersections. The foliated manifolds with the same color are one foliation, while the black dots or lines are the intersections. The red and blue dots are the start and goal.

from the two abstract manifolds they connect. In the delivering cup example, we sample intersections between an abstract manifold defined by a grasp g and another defined by placement p using an IK solver to find a feasible configuration to grasp the object with g in placement p. This discretization for each problem is detailed in the experiments (VI-A) section.

V. PLANNING IN FOLIATED MANIFOLDS

Our study introduces the FoliatedRepMap, a framework designed to incorporate motion planning experience collected from prior foliated manifolds and continuously update itself after each motion planning iteration. Initially, the FoliatedRepMap generates a lead [25], a promising sequence of manifolds to traverse from start to goal, each accompanied by navigation guidance. Then, the motion planner explores each manifold of the lead in order. After each motion planning attempt, the sampled data during this planning is collected and used to update the FoliatedRepMap. If the motion planner fails to plan in any manifold from the lead sequence, the FoliatedRepMap produces a new lead in the next planning iteration. This iterative process continues until a viable solution trajectory is successfully identified. The following sections detail each key component of our framework: (i) the FoliatedRepMap, (ii) the Atlas, and (iii) the Dynamic Roadmap.

A. Foliated Repetition Roadmap

The FoliatedRepMap is an abstract graph spanning multiple abstract foliated manifolds. Converting the foliated manifolds planning problem into a roadmap format simplifies management. This is because representing the problem as a graph allows for the application of efficient graph algorithms in pathfinding. Fig. 5 shows foliated manifolds with provided intersections in different perspectives, while Figures 6 and 7 illustrates the roadmap construction process from the given abstract foliated manifolds $\boldsymbol{\Xi}$ and intersections $\boldsymbol{I}$. The next section will delve into the roadmap's construction, its updates, and how it is utilized to plan in foliated manifolds.

1) Constructing Foliated Repetition Roadmap: Construction of the FoliatedRepRoadmap is divided into two steps : (a) a one-time GMM self-collision-free roadmap step for a robot arm, followed by (b) conversion of GMM-based roadmap to FoliatedRepMap.

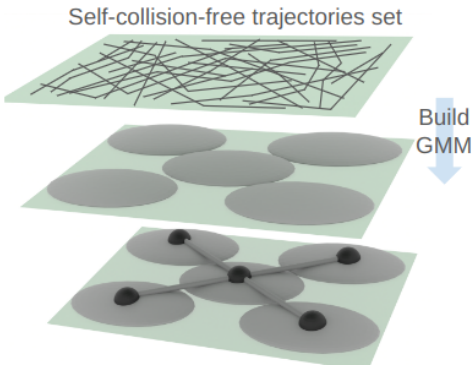


Fig. 6: GMM Self-collision-free space roadmap construction. Starting with self-collision-free trajectories, we partition the waypoints to create a GMM. From this GMM, we create a roadmap that incorporates these self-collision-free paths. This step is independent of any task or environment.

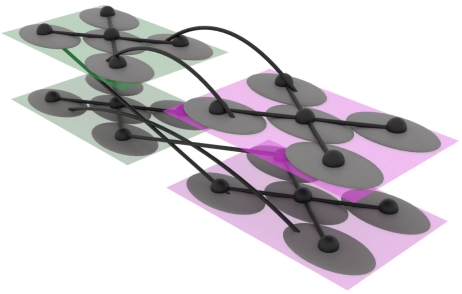


Fig. 7: FoliatedRepMap construction. Based on abstract manifolds and intersections of the task(Fig. 5), the GMM roadmap is duplicated for each abstract manifold, and connections between them are maintained through the intersection list $\boldsymbol{I}$.

a) Creation of GMM-based self-collision-free-space roadmap: As demonstrated in Fig. 6, we start with a set of self-collision-free trajectories by repeatedly running the RRT-star [20] to connect two random configurations in the self-collision-free space. The waypoints of these trajectories are then processed using a clustering algorithm, such as the Dirichlet process algorithm [9], to produce a GMM, which serves as a versatile model for partitioning the robot's self-collsion-free configuration space. Similar to [29], we establish a connection between two distributions if a self-collision-free trajectory across them directly exists. The end result is a roadmap. It is important to note that this is a pre-processing step, is independent of the scene and the task, and depends purely on the self-collision-free space of the robot. Thus, this is a one-time step that remains the same for different tasks and scenes, and thus, does not need to be time-critical.

b) Conversion to FoliatedRepMap: Now, we have individual roadmaps for each abstract manifold. To complete the FoliatedRepMap, we utilize the intersections from $\boldsymbol{I}$ to interconnect those individual roadmaps, as depicted in Fig. 7. Finally, in this FoliatedRepMap, each node is defined by three variables and represented as $n^{<\Xi,\theta,d>}$ where Ξ is foliation, θ is a co-parameter from Θ^{Ξ}, and d is a distribution from the generated GMM. Both foliation Ξ and co-parameter θ define a foliated manifold $M^{<\Xi,\theta>}$. This is shown in Fig. 7. It must be noted that the roadmap spans the abstract foliated manifold (which is merely the self-collision-free space defined by a co-parameter) and does not span the actual manifold, since we do not know the actual manifold space in advance.

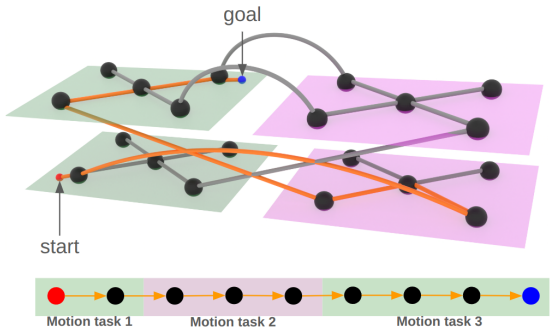


Fig. 8: Given start and goal configurations from different manifolds, an optimal node sequence (orange) is returned and divided into a sequence of motion tasks segmented by manifold intersections.

This GMM-based self-collision-free space roadmap offers a distinct benefit over the online PRM roadmap constructed in the ALEF [26] planner. Each abstract manifold contains a copy of a roadmap constructed from the same GMM, allowing it to individually adjust its nodes' weights. Using the same GMM ensures that similar areas in different abstract manifolds share similar properties, which facilitates easier weight updates of the roadmap in one abstract manifold in response to modifications in the roadmap values of other abstract manifolds.

2) Planning with Foliated Repetition Roadmap: Given a foliated manifold $M^{<\Xi_s,\theta_s>}$ with a start configuration q_s and a foliated manifold $M^{<\Xi_g,\theta_g>}$ with a goal configuration q_g, the FoliatedRepMap generates a lead to guide the motion planner in foliated manifolds. First, the start distribution d_s to which q_s belongs and, the goal distribution d_g for q_g are identified. Subsequently, Dijkstra's algorithm is run to produce the optimal node sequence between the start $n^{<\Xi_s,\theta_s,d_s>}$ and the goal nodes $n^{<\Xi_g,\theta_g,d_g>}$, as illustrated in figure Fig. 8 (top). The weights associated with each node (discussed in the next section) provide the distance function for the shortest path algorithm. This node sequence is then segmented at the intersections crossing the abstract manifolds into motion tasks, as shown on the lower side of Fig. 8. In this arrangement, consecutive nodes within the same abstract manifold of M_i constitute a single motion task $Task_i$. This task inherits its constraints from the constraint of M_i. Furthermore, the intersections in which these nodes enter and leave the abstract manifold of M_i define the start and goal configurations of the motion task $Task_i$, respectively.

Therefore, given a motion task planning in $M^{<\Xi,\theta>}$, we have its consecutive nodes $\{n^{<\Xi,\theta,d_1>}, n^{<\Xi,\theta,d_2>}, ...\}$.

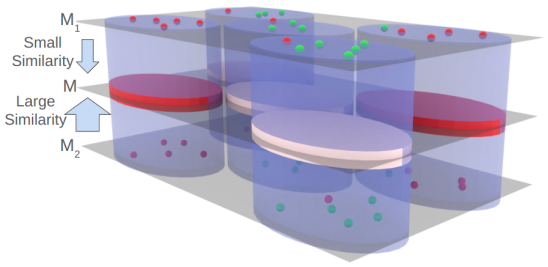


Fig. 9: Updating process in one foliation. For abstract manifold of M, its roadmap node weights are increased based on the number of both valid (green dots) and invalid (red dots) sampled configurations from all abstract manifolds from the same foliation, such as M_1 and M_2. In this figure, due to the higher similarity between M and M_2, the weights in M are impacted more by the sampled configurations from M_2. In this figure, the darker color represents higher weight.

Corresponding to those nodes, a sequence of distributions $\{d_1, d_2, ...\}$ is used to approximate the actual manifold, so the motion planner can navigate from the local start configuration to the local goal configuration in $M^{<\Xi,\theta>}$ by sampling within these distributions with higher priority.

3) Updating Foliated Repetition Roadmap: After motion planning in a foliated manifold, FoliatedRepMap must be updated by the sampled data in this manifold as experience to guide the planning step in the next iteration. As shown in Fig. 9, after motion planning in a abstract manifold of M_1, a set of sampled configuration sets with their respective status are returned like $[(q_1, status_1), (q_2, status_2), ...]$. Then, the FoliatedRepMap uses the gathered sampling data from M_1 to adjust the weights of roadmap nodes in similar abstract manifolds of M. This update ensures that future searches in similar manifolds avoid areas that have been explored or where sampling has frequently failed. In this work, we exclusively consider the following four possible configuration statuses:

- Valid: Configurations that are collision-free and satisfy constraints.
- RE-Collision Invalid: Configurations where collisions occur between the robot and the environment.
- MO-Collision Invalid: Configurations where collisions involve the manipulated object.
- Const-Invalid: Configurations that violate end-effector constraints.

Therefore, every node $n^{<\Xi,\theta,d>}$ has four configuration counters ($C_{valid}^{<\Xi,\theta,d>}$, $C_{RE}^{<\Xi,\theta,d>}$, $C_{MO}^{<\Xi,\theta,d>}$,$C_{Const}^{<\Xi,\theta,d>}$) where each corresponding to the different status described above, for each distribution d of foliated manifold $M^{<\Xi,\theta>}$.

Following the motion planning within the foliated manifold $M^{<\Xi,\theta>}$, for every pair $(q, status)$ derived from the motion planner, we determine the distribution d to which q belongs. Subsequently, we increment the relative counter of the node $n^{<\Xi,\theta,d>}$ based on the value of $status$. Then, we calculate the weight $w^{<\Xi,\theta,d>}$ of each node $n^{<\Xi,\theta,d>}$ as:

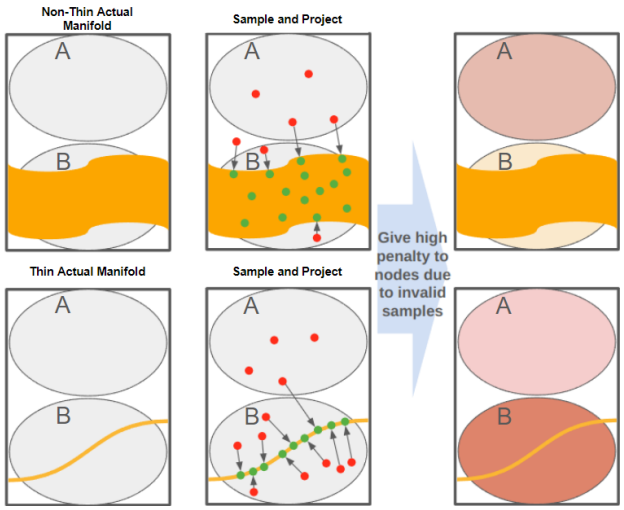


Fig. 10: One limitation of FoliatedRepMap: The box is the abstract manifold, while the orange region is the actual manifold. Constrained motion planning involves sampling a set of configurations and projecting them onto the actual manifold. The FoliatedRepMap assigns a high penalty to nodes with invalid pre-projected configurations in their distributions. In the case of a non-thin actual manifold (where the constraint is relaxed), the motion planner is guided to explore node B where the actual manifold is since it has a higher fraction of valid configurations than A, leading to lower weight. However, in a thin actual manifold, B has a high weight (even darker color) due to a large fraction of pre-projection invalid configurations despite the presence of the actual manifold. As a result, the motion planner will be misguided to explore node A.

$$w^{<\Xi,\theta,d>} = \sum_{\theta'} \{p^+ C_{RE}^{<\Xi,\theta',d>} + s_{\theta,\theta'} \{p^- C_{valid}^{<\Xi,\theta',d>} + p^+ (C_{MO}^{<\Xi,\theta',d>} + C_{Const}^{<\Xi,\theta',d>})\}\} \tag{1}$$

Here, θ' represents all co-parameters in the foliation Ξ. The term $s_{\theta,\theta'}$ is the co-parameter similarity ratio between θ and θ', p^+ denotes a large penalty, while p^- is a small penalty. There are two points of note here.

1) the component $C_{RE}^{<\Xi,\theta',d>}$ is applied irrespective of the similarity ratio because the likelihood of a collision between the robot and the environment remains constant irrespective of manifolds.
2) the number of invalid pre-projection configurations contributes to a node's weight. This is because a region is not promising if sampled configurations from d_i are projected onto other distributions. Therefore, this distribution's weight is increased to disincentivize the motion planner from considering it in the future.

Consequently, in the next planning iteration, regions corresponding to high weights can be effectively avoided.

4) Limitations of Foliated Repetition Roadmap: Using FoliatedRepMap alone still presents two challenges. First, the update method may lead to inappropriate guidance for motion planning in regions where its actual manifold is thin. Second, the large size of the roadmap results in significant computational overhead.

The main goal of the FoliatedRepMap is to guide the motion planner to avoid regions with a lower chance of sampling success based on experience. In the absence of task constraints (where the actual manifold spans a large space in the abstract manifold), it helps the planner choose regions such that sampled configurations from these regions with lower collision with the environment.

However, the presence of a restrictive task constraint typically leads to a thin actual manifold, where the probability of sampling a valid configuration before projection is infinitesimally small. In this case, the update step may mislead the motion planner to search in unpromising regions. As discussed, if a sampled configuration violates constraints, it is projected onto the actual manifold. In thinner manifolds, most of the sampled pre-projected configurations are invalid, even though they can be successfully projected onto the actual manifolds, as shown in Fig. 10. Since the number of pre-projected invalid configurations contributes to a weight increase, even though the distribution contains the actual manifold, its weight will be prohibitively high.

When the actual manifold is typically concentrated around a few distributions for a challenging problem, (as opposed to easier problems where the actual manifold spans multiple distributions), the update step increases the weights of most nodes that contain valid configurations (due to their proximity). This causes the FoliatedRepMap to guide the motion planner into exploring regions devoid of valid configurations, resulting in higher planning times.

Moreover, the complexity of the FoliatedRepMap is compounded by the large number of co-parameters associated with each manifold, resulting in a huge roadmap. At the same time, it is important to note that only a small fraction of this roadmap is utilized in the planning process. Consequently, processing the entire roadmap in each planning iteration is inefficient, leading to unnecessary computational overhead.

B. Foliated Repetition Roadmap with Atlas

To overcome the first limitation, we integrate atlas information into the roadmap during the update and the planning phase. When exploring thin actual manifolds, as shown in Fig. 11, we utilize the atlas from different similar foliated manifolds to approximate the predicted actual manifolds more precisely. Each node now incorporates an atlas, denoted as α, which stores the tangent space information of a local region related to that node. In the previous iteration of planning within a manifold $M_i = M^{<\Xi,\theta_i>}$, the motion planner returns a set of projected valid configurations. For each roadmap node $n^{<\Xi,\theta_i,d_k>}$ in abstract manifold of M_i, we construct an atlas $\alpha^{<\Xi,\theta_i,d_k>}$ using these projected valid configurations belonging to the distribution d_k. In the next iteration of planning, when encountering a new manifold, $M_j = M^{<\Xi,\theta_j>}$ within the same foliation, the FoliatedRepMap extracts the Atlas information from similar manifolds based on the generated node sequence, and combines them to estimate the atlas $\alpha^{<\Xi,\theta_j,d_k>}$ of the new manifold M_j. This new atlas approximates the characteristics of the manifold M_j. This is illustrated in 11 (Bottom).

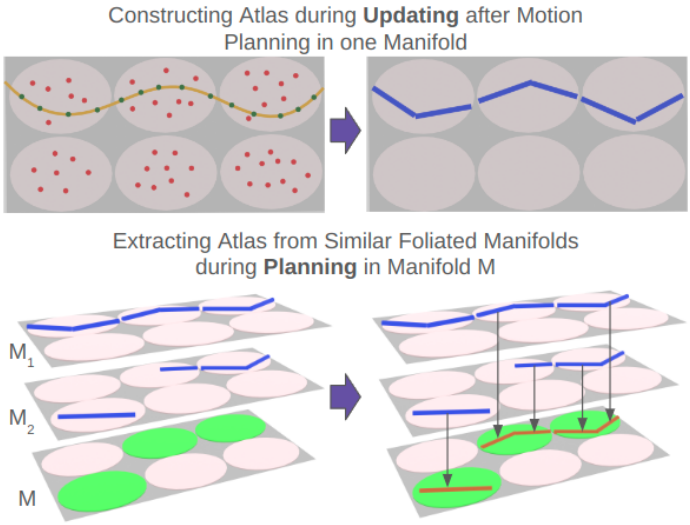


Fig. 11: **Top**: Given a thin actual manifold(yellow curve), after motion planning, a set of projected valid configurations (green dots) and invalid configurations (red) are returned. FoliatedRepMap constructs each node's atlas (dark blue lines) from the projected valid configurations. **Bottom**: While planning in manifold M, assume the FoliatedRepMap produces a list of nodes (green ellipses) as guidance. It extracts the Atlas from related nodes from all manifolds in that foliation, such as M_1 and M_2, and combines them into one atlas (orange line segments) to estimate the actual manifold.

It must be noted that this Atlas construction is expensive and redundant for non-thin regions of the manifold. This is because sampling from the Gaussian distribution d_k that the node belongs to, itself has a fairly high probability of satisfying the manifold constraints. The next section explains the framework to guide the motion planner across various manifolds with different conditions.

1) Planning with Atlas: The goal of this step is to generate a task sequence of nodes and perform motion planning with the generated sequence.

a) Generating task sequence: This step is identical to the previous method. Using Dijkstra's algorithm, the optimal node sequence from the start $n^{<\Xi_s,\theta_s,d_s>}$ and the goal nodes $n^{<\Xi_g,\theta_g,d_g>}$. This node sequence is segmented into a sequence of motion tasks, where each motion task itself is composed of a node sequence, N in a manifold, M_c

b) Constructing Atlas information for a node sequence: To plan with Atlas in a new manifold, M_c, given a motion task (of a node sequence) N, this step computes the Atlas corresponding to the specific region of interest within the

Algorithm 1 BuildGuidance (Node Sequence N, Manifold M_c)

1: $\alpha Buffer$ = []
2: $\beta Buffer$ = []
3: $d_{result} = []$
4: **for** $n^{<\Xi,\theta,d>} \in N$ **do**
5: $d_{result}.add(d)$
6: **for** $\theta' \in$ co-parameters of Ξ **do**
7: **if** HasAtlas($n^{<\Xi,\theta',d>}$) **then**
8: $\alpha^{<\Xi,\theta',d>} =$ get$\alpha(n^{<\Xi,\theta',d>})$
9: $\beta^{<\Xi,\theta',d>} =$ get$\beta(n^{<\Xi,\theta',d>})$
10: $s =$ GetSimilarity(θ, θ')
11: $\alpha Buffer$.add$(< s, \alpha^{<\Xi,\theta',d>} >)$
12: $\beta Buffer$.add$(< s, \beta^{<\Xi,\theta',d>} >)$
13: **end if**
14: **end for**
15: **end for**
16: α_{result} = NewAtlas(M_c)
17: $Sort_{similarity}(\alpha Buffer)$
18: **for** $< s, \alpha > \in \alpha Buffer$ **do**
19: **for** Chart $c \in \alpha$ **do**
20: q=$GetOrigin$(c)
21: $AddToAtlas(\alpha_{result}, q, s, M_c)$
22: **end for**
23: **end for**
24: $SoftMax_{similarity}(\beta Buffer)$
25: $\beta_{result} = 0$
26: **for** $< s, \beta > \in \beta Buffer$ **do**
27: $\beta_{result} += s \cdot \beta$
28: **end for**
29: **return** α_{result}, d_{result}, β_{result}

actual manifold.

Algorithm 1 returns both an Atlas α_{result} and a distribution sequence d_{result} to estimate the actual manifold of M_c. It also returns the sampling ratio β_{result}, which decides the ratio of sampling from either α_{result} or d_{result}.

From lines 1 to 15, the algorithm first collects related distributions from nodes in N. Two buffers $\alpha Buffer$ and $\beta Buffer$ accumulate the relevant Atlas and Atlas-Distribution ratio, from similar manifolds. For each node $n^{<\Xi,\theta,d>}$ in the sequence N, for all co-parameters θ' of Ξ, we obtain its corresponding node $n^{<\Xi,\theta',d>}$, and extract their $\alpha^{<\Xi,\theta',d>}$ and $\beta^{<\Xi,\theta',d>}$ and adds them to the respective buffers, along with the similarity score between θ and θ'. The Atlas-distribution ratio, β is maintained by each node, which will be discussed later in the update step. The next step is to combine all the Atlases from $\alpha Buffer$ to estimate the new manifold's Atlas. From lines 17 to 23, the algorithm calls the $AddToAtlas$(2) on all origin points of the charts from all Atlas in descending order of similarity. The sorting ensures that the charts with higher similarity are given precedence over charts with lower similarity.

The $AddToAtlas$ algorithm is the same as the one introduced in [17] to a new configuration to Atlas, but with one

Algorithm 2 AddToAtlas (Atlas α, config q, **weight** w, Manifold M)

1: **if** not $\alpha.owningChart(q)$ **then**
2: $c = CreateChart(q, M)$
3: **c.SetWeight(w)**
4: **for** $c' \in \alpha.ChartNearby(q)$ **do**
5: $\alpha.GenerateHalfSpace(c, c')$
6: **end for**
7: $\alpha.add(c)$
8: **end if**

modification (in bold). In addition to building the chart, we assign each chart a weight. This weight will later be used as an indicator of similarity between two manifolds containing Atlas information, as indicated by Line 27 in Alg. 1.

The final step is to calculate the sampling ratio β_{result}. The beta buffer values are normalized using the soft-max function. The β_{result} is then calculated as the cumulative sum of the product of all pairs in the buffer, to ensure that β_{result} effectively represents a weighted sum, where the weights are adjusted following the similarity values.

c) Motion planning with Atlas information: Subsequently, the motion planner utilizes β_{result} to determine the source of its sampling of configurations, either from α_{result} or from the distributions within N, and computes motion plans.

2) Updating Roadmap: Similar to the FoliatedRepMap variant without Atlas, the goal of this step is to gather the sampling data from motion plans and update the roadmap nodes' weights. In addition, we construct an Atlas for the actual manifolds from the sampled information.

a) Updating the Atlas: After motion planning in a manifold $M^{<\Xi,\theta>}$, the FoliatedRepMap receives all valid configurations that were projected onto its actual manifold. For each configuration q, we determine its distribution, represented by d, and add this configuration into the Atlas $\alpha^{<\Xi,\theta,d>}$ corresponding to the node $n^{<\Xi,\theta,d>}$ with a default weight 1.0 by calling $AddToAtlas(\alpha^{<\Xi,\theta,d>}, q, 1.0, M^{<\Xi,\theta>})$.

b) Computing Atlas distribution ratio: To update the weight of the roadmap, it must consider both thin and non-thin local regions on the manifolds. Specifically, when the local region on the actual manifold exhibits a non-thin volume, a significant proportion of sampled configurations within this local region are valid without projection. In this case, sampling from the distribution directly becomes more effective for exploration. On the other hand, in thinner regions, it is illogical to increase its weight due to the presence of invalid configurations that violate constraints before projection. Therefore, the approach for updating the node weight responding to each local region should be flexible and responsive to its specific characteristics. To achieve this, we introduce a parameter called the Atlas-Distribution ratio, symbolized by β, which adjusts the sampling methodology according to the nature of the local region.

$$\beta = \frac{\text{count of invalid configurations before projection}}{\text{total count of configurations before projection}}$$

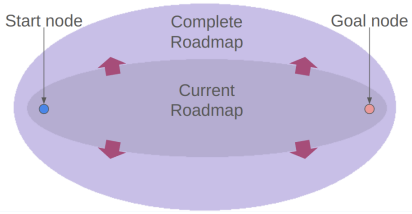


Fig. 12: Dynamic roadmap. During planning, the planner only processes a subsection of the total roadmap, and this current roadmap is selected if the node's distance to both the start and goal node is less than a threshold. After roadmap updating, if the cost from the start to the goal is higher than a certain value, the current roadmap will be expanded with more nodes.

This ratio serves as an indicator of the local region's thickness within the actual manifold, which is determined by the number of valid and invalid configurations sampled in this local region before projection. A higher β is indicative of a thinner region on the actual manifold, suggesting that sampling via Atlas would be more suitable.

Since the projection function is not guaranteed to generate a configuration in the same region, we can have instances where a node lacks an Atlas but has pre-projection configurations and nodes that have an Atlas but no pre-projection configurations. A $beta$ score of 0 and 1 are assigned here respectively.

c) Updating weights of nodes: The update function is similar to the previous method. Contrary to the non-atlas variant, for each node $n^{<\Xi,\theta,d>}$, we allocate two separate weights: the first, $w^{<\Xi,\theta,d>}_{\text{before projection}}$, is a score calculated from sampled configurations before projection, and the second, $w^{<\Xi,\theta,d>}_{\text{after project}}$, is a score derived from sampled configurations assessed after projection. Both $w^{<\Xi,\theta,d>}_{\text{before projection}}$ and $\beta^{<\Xi,\theta,d>}w^{<\Xi,\theta,d>}$ are computed as per Eqn 1. The final weight for each node, $w^{<\Xi,\theta,d>}$, is then a weighted sum given by.

$$w^{<\Xi,\theta,d>} = (1-\beta^{<\Xi,\theta,d>})w^{<\Xi,\theta,d>}_{\text{before projection}} + \beta^{<\Xi,\theta,d>}w^{<\Xi,\theta,d>}_{\text{after project}}$$

Thus, by creating an incremental atlas from the result of the motion planner, and computing β_{result} to capture the characteristic of the actual manifold, we increase the likelihood of the success of the planner in the next step.

C. Dynamic Roadmap

With the current FoliatedRepMap, a significant challenge arises due to the high number of nodes in the roadmap. This leads to prohibitively high task planning and updating time costs. At the same time, for a given problem, most of the nodes are irrelevant and not considered in the planning process since they are too far away from the regions of interest. To solve this, we develop the dynamic roadmap strategy, shown in Fig. 12. This method uses only a part of the full roadmap, focusing on the most important areas for planning the node sequence. We also update only this smaller part of the roadmap, making the whole process of planning and updating more efficient.

We leverage a heuristic function to extract the sub-roadmap from the original roadmap. Each edge in the roadmap is initially assigned a cost, reflecting the Euclidean distance between the means of distributions corresponding to the edge's end nodes, $||q_1 - q_2||^2$ where q_1 and q_2 are the means of the Gaussian distributions of the two nodes of the edge. Subsequently, for each node n, we compute a heuristic value $h(n)$. This value represents the total distance cost starting from $n^{<\Xi_s,\theta_s,d_s>}$ to the node n, and then from n to $n^{<\Xi_g,\theta_g,d_g>}$, as shown in the following:

$$h(n) = Distance(n^{<\Xi_s,\theta_s,d_s>}, n) + Distance(n, n^{<\Xi_g,\theta_g,d_g>})$$

where $Distance(m,n)$ calculates the shortest accumulated distance, achieved by summing the Euclidean distances of all edges along the shortest path between nodes m and n.

The sub-graph is initialized with a small subset of nodes, that have a low heuristic value determined by a threshold r_{th}, which can be thought of as the radius of the sub-graph. This starting threshold is computed as twice the $Distance(n_{start}, n_{goal})$ to include the smallest possible set of nodes from start to goal.

With each planning iteration, the weight cost between the start and goal nodes may exceed a specified threshold, which indicates the difficulty of finding a solution in the current sub-graph. Therefore, the planning framework now expands the sub-graph to include nodes with higher heuristic values in the current roadmap, as illustrated by the arrows in Fig. 12, increasing the radius. These new nodes are assigned a zero weight, enabling the FoliatedRepMap to explore these potential paths as a priority in the following planning iteration.

VI. EVALUATION

This section compares the performance of our approach against established methods under various setups. We consider the following planning methods in our experiment:

1) *MTG*: The baseline motion planner using MTG [25].
2) *ALEF*: A simplified implementation of adaptive ALEF [26].
3) *FoliatedRepMap*: The motion planner with the FoliatedRepMap.
4) *FoliatedRepMap with Atlas*: The motion planner, incorporating the Atlas into FoliatedRepMap.
5) *FoliatedRepMap+Dynamic Roadmap*: The motion planner using FoliatedRepMap with dynamic updating.
6) *FoliatedRepMap with Atlas+Dynamic Roadmap*: The motion planner, incorporating the Atlas into FoliatedRepMap with dynamic updating.

As the baseline, we implement MTG to build a graph where each edge contains the transition difficulty between different manifolds via given intersections. Furthermore, we also implemented the adaptive ALEF (learns online from success queries). The ALEF implementation is slightly different from the original implementation; it uses a distance-based heuristic to connect foliated manifolds, as opposed to the actual feasibility of the motion plan (which is expensive for large dimensionalities). For constrained motion planning, we modified and used the MoveIt! [6]. A graph library [12] is used for storing and traversing the roadmap of FoliatedRepMap.

Task	Planner	Fetch (7 Joints) Success %	Fetch (7 Joints) Planning Time (s)	Fetch (7 Joints) Soln. Length	UR5 (6 Joints) Success %	UR5 (6 Joints) Planning Time (s)	UR5 (6 Joints) Soln. Length
Slide Cup	MTG	**100.000**	1.082	24.696	30.000	190.483	**2.126**
	ALEF	**100.000**	**0.917**	23.232	40.000	204.214	2.258
	FoliatedRepMap	**100.000**	4.689	18.043	**100.000**	12.972	11.400
	FoliatedRepMap+Dynamic Roadmap	**100.000**	4.206	16.424	**100.000**	13.974	19.457
	FoliatedRepMap with Atlas	**100.000**	4.613	13.919	**100.000**	11.981	13.542
	FoliatedRepMap with Atlas+Dynamic Roadmap	**100.000**	2.997	**13.673**	**100.000**	**6.832**	20.552
Rearrange Shelf	MTG	16.000	14.048	18.768	2.000	316.276	**0.664**
	ALEF	38.000	101.350	**2.558**	2.000	285.399	0.819
	FoliatedRepMap	82.000	13.435	24.579	56.000	**36.407**	23.537
	FoliatedRepMap+Dynamic Roadmap	84.000	14.670	21.322	46.000	77.364	18.658
	FoliatedRepMap with Atlas	**100.000**	12.560	21.259	70.000	71.166	18.934
	FoliatedRepMap with Atlas+Dynamic Roadmap	**100.000**	**12.460**	21.182	**76.000**	77.961	12.948
Pour Water	MTG	92.000	8.846	33.006	0.000	N/A	N/A
	ALEF	96.000	11.117	39.447	0.000	N/A	N/A
	FoliatedRepMap	98.000	14.832	26.086	28.000	415.390	**2.570**
	FoliatedRepMap+Dynamic Roadmap	98.000	9.559	27.128	34.000	250.139	4.712
	FoliatedRepMap with Atlas	98.000	7.947	**25.040**	32.000	173.684	6.117
	FoliatedRepMap with Atlas+Dynamic Roadmap	**100.000**	**7.023**	26.074	**54.000**	**155.752**	6.002
Open Bottle	MTG	20.000	11.106	11.944	0.000	N/A	N/A
	ALEF	22.000	38.456	**8.298**	0.000	N/A	N/A
	FoliatedRepMap	60.000	9.580	14.766	**96.000**	24.818	9.804
	FoliatedRepMap+Dynamic Roadmap	58.000	**8.410**	15.093	92.000	**30.437**	10.192
	FoliatedRepMap with Atlas	64.000	11.848	11.402	84.000	56.329	**8.296**
	FoliatedRepMap with Atlas+Dynamic Roadmap	**68.000**	13.103	10.801	94.000	30.734	8.714
Open Door	MTG	20.000	14.991	9.933	0.000	N/A	N/A
	ALEF	28.000	40.528	**8.217**	0.000	N/A	N/A
	FoliatedRepMap	26.000	11.676	14.743	20.000	110.650	**8.549**
	FoliatedRepMap+Dynamic Roadmap	24.000	14.691	12.801	16.000	71.895	12.647
	FoliatedRepMap with Atlas	**42.000**	**10.840**	13.495	22.000	99.367	10.522
	FoliatedRepMap with Atlas+Dynamic Roadmap	38.000	11.847	12.711	**26.000**	**70.618**	10.311
Open Drawer	MTG	70.000	44.096	**10.434**	0.000	N/A	N/A
	ALEF	74.000	51.358	10.502	0.000	N/A	N/A
	FoliatedRepMap	88.000	5.471	23.695	92.000	**15.682**	11.169
	FoliatedRepMap+Dynamic Roadmap	90.000	**5.323**	23.908	96.000	24.681	**9.986**
	FoliatedRepMap with Atlas	**100.000**	9.187	19.530	**98.000**	33.644	12.503
	FoliatedRepMap with Atlas+Dynamic Roadmap	**100.000**	6.400	20.365	**98.000**	22.115	11.110
Navigate Maze	MTG	38.000	4.601	15.935	0.000	N/A	N/A
	ALEF	44.000	21.669	**8.494**	0.000	N/A	N/A
	FoliatedRepMap	76.000	5.524	17.638	14.000	13.823	**9.730**
	FoliatedRepMap+Dynamic Roadmap	72.000	**4.287**	18.904	16.000	**8.470**	18.255
	FoliatedRepMap with Atlas	82.000	4.993	16.185	22.000	10.117	12.984
	FoliatedRepMap with Atlas+Dynamic Roadmap	**84.000**	4.349	18.002	**36.000**	10.779	11.523

TABLE I: Evaluation Across Different Tasks and Robots. (**Bold font**: the best value; underlined font: the second-best value)

For each problem, we pre-generate all abstract manifolds and intersections and use the same pre-generated set for all planners for a fair comparison.

The performance of planners is evaluated based on the following metrics:

- **Success Rate**: Success rate of planning tasks, out of 50 trials with a random setup, which is related to the task.
- **Total Planning Time**: Average total planning time for tasks that were successfully solved, excluding those that were not. The planning time also includes the time to construct the roadmap.
- **Solution Length**: Average solution trajectory length for successfully solved tasks.

Each experiment is executed for 50 iterations with a random start and goal setup based on the task, with 50 max attempts. We consider an iteration to fail if it exceeds the attempt limit without a solution. Furthermore, the GMM is constructed using the Dirichlet Process [9], with 91 distributions to partition the robot's self-collision-free space. We choose 91 distributions since it is equal to the maximum-path length of the self-collision free motion plans generated during construction. This heuristic is borrowed from the Repetition Roadmap work [29]. Different distributions are discussed briefly in the appendix.

A. Tasks with Different Foliated Manifolds Structures

Although foliated manifold structures are prevalent in manipulation problems, the difficulty levels vary significantly. Two factors define the difficulty of a problem. (1) the complexity of the environment and (2) the restrictiveness of the end effector constraint. A complex environment increases the likelihood of collisions of sampled (and projected) configurations, while the restrictions imposed by the end-effector constraint significantly reduce the valid configuration space. For instance, consider the task of sliding a cup across a table: the given constraint requires the arm configuration to maintain the cup in a horizontal position on the table. With no obstacles present on the table surface, most of the projected configurations are valid, making the solution relatively straightforward. Solving a maze however poses a greater challenge, despite sharing a similar constraint with the sliding cup scenario. In navigating a maze, the complexity of the environment elevates the probability of collisions between maze walls and the robot/object. On the other hand, a task like opening a drawer or bottle is hard since the end-effector constraint significantly reduces the range of valid configuration space, which increases the probability of projecting a sampled configuration to lie outside joint limits.

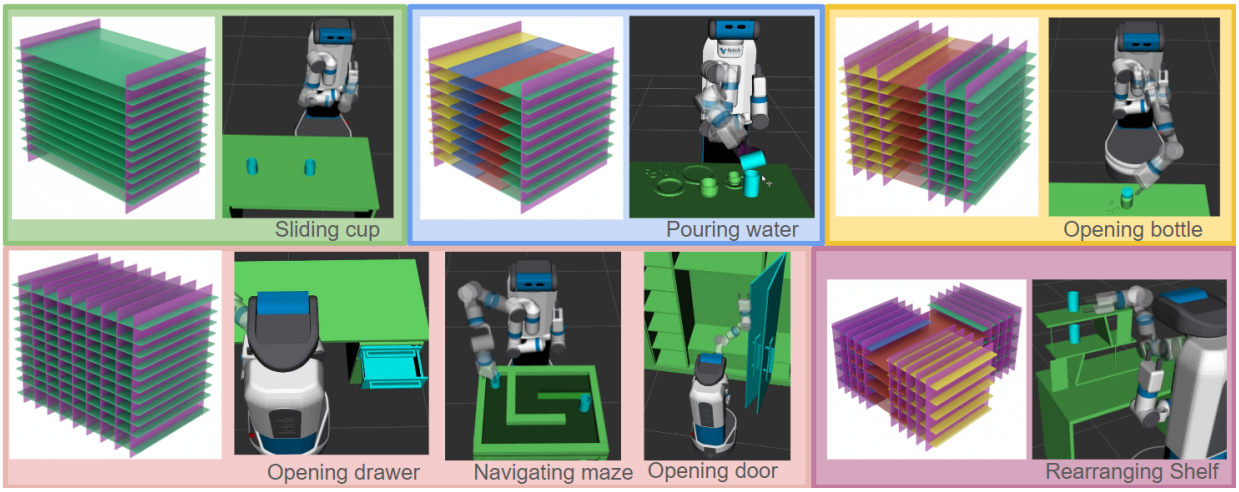


Fig. 13: Each task with its abstract manifold structure. We group the tasks with a similar structure in the same color. Each plane represents an abstract manifold. In all tasks, the vertical planes represent the un-grasping manifolds, defined by placements, while the horizontal planes represent sliding/transporting manifolds, defined by grasp poses. **Sliding cup**: Each green plane is defined by sliding the cup with a certain grasp. **Pouring water**: Horizontal planes with different colors are defined by the different stages of pouring water. For all subsequent tasks, each purple plane represents re-grasping at intermediate placements. **Opening bottle**: Yellow and green planes are defined by rotating the lid with a certain grasp; each red plane is defined by the initial half of the rotation. **Opening drawer**: Each green plane is defined by dragging the drawer with a grasp. **Navigating maze**: Each green plane is defined by sliding the cup with a grasp. **Opening door**: Each green plane is defined by opening the door with a grasp. **Rearranging shelf**: Blue, green, and yellow planes are defined by sliding the cup at different levels; each red plane is defined by lifting or lowering the first column of the cup between different heights.

We design tasks with different abstract foliated manifold structures as shown in Fig. 13. Despite the presence of a mobile base, we restrict our usage to the arm only, to avoid the added complexity that would arise from considering the robot's position. Consequently, we assume that the robot's base is fixed in place.

This section provides an overview of the abstract manifold structure associated with each task. For additional details regarding the methodology for determining the set of foliated manifolds $\boldsymbol{\Xi}$, the intersections $\boldsymbol{I}$ between them, and how random setups are established, please refer to the Appendix B in the supplementary material.

- **Sliding cup:** This task involves sliding a cup on an empty table. It has two foliations: $\Xi_{un-grasp}$ and Ξ_{slide}. $\Xi_{un-grasp}$ deals with picking up the cup and resetting the robot. Ξ_{slide} focuses on sliding the cup, considering different grasps.
- **Pouring water:** Pouring water from one cup to another contains 5 foliations: $\Xi_{transport}$, Ξ_{pour}, $\Xi_{pour\ done}$, Ξ_{place}, $\Xi_{un-grasp}$. The $\Xi_{un-grasp}$ step describes picking up the cup and resetting the robot. $\Xi_{transport}$ is for moving the cup without spilling. Ξ_{pour} is for tilting the cup to pour the water. $\Xi_{pour\ done}$ marks the end of pouring. Ξ_{place} is for placing the cup down. It's key to have the $\Xi_{pour\ done}$ to make sure the water is in fact poured before putting the cup down.
- **Rearranging Shelf:** Rearranging self involves transporting a cup from one level to another level, then sliding it on that level. This task involves 5 foliations: $\Xi_{un-grasp}$, $\Xi_{slide\ table}$, $\Xi_{slide\ shelf1}$, $\Xi_{slide\ shelf2}$, and $\Xi_{transport}$. $\Xi_{un-grasp}$, is different from previous tasks since the robot is allowed re-grasping (zero or more times). $\Xi_{slide\ table}$ involves sliding the cup on the table surface. $\Xi_{slide\ shelf1}$ and $\Xi_{slide\ shelf2}$ refer to sliding the cup on the first and second levels of the shelf. $\Xi_{transport}$ is about the transition of the cup between different heights.

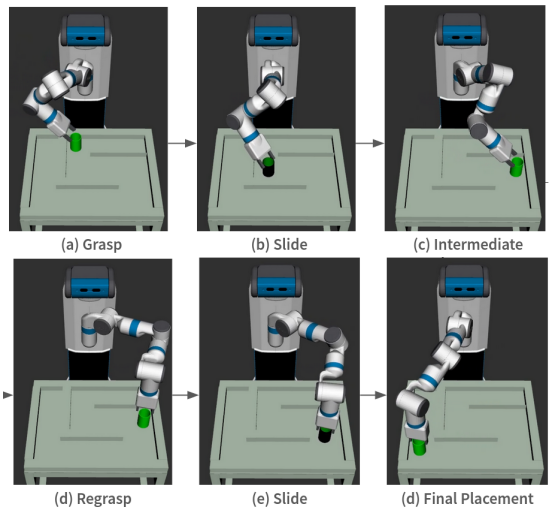


Fig. 14: Example of a successfully completed maze task.

- **Navigating maze:** Moving a cup in a maze from one location to another and allowing re-grasping involves two foliations: $\Xi_{un-grasp}$ and Ξ_{slide}. Similar to the shelf task, $\Xi_{un-grasp}$ here considers re-grasping by having extra manifolds for each intermediate placement. Ξ_{slide} represents the action to slide the cup in the maze.
- **Opening drawer:** Opening drawer with re-grasping has also two foliations: $\Xi_{un-grasp}$ and Ξ_{pull}. Similar to the shelf task, $\Xi_{un-grasp}$ here considers re-grasping by providing extra manifolds. Meanwhile, Ξ_{pull} is about pulling the drawer open using a single grip.
- **Opening door:** Similar to opening a drawer, opening the door considering re-grasping has two foliations: $\Xi_{un-grasp}$ and Ξ_{pull}. $\Xi_{un-grasp}$ considers re-grasping the door by having extra manifolds. Meanwhile, Ξ_{pull} is about pulling the door open using a single grip.
- **Opening bottle:** Opening the bottle requires the robot to rotate its lid 360 degrees. Its manifold structure is slightly complicated and has 4 foliations: $\Xi_{un-grasp}$, $\Xi_{first\ rotate}$, $\Xi_{second\ rotate}$, and $\Xi_{first\ rotate\ done}$. $\Xi_{un-grasp}$ represents the initial grasping and resetting robot, including additional manifolds to account for any re-grasping needed during the rotation. $\Xi_{first\ rotate}$ represents the first 180-degree rotation. $\Xi_{first\ rotate\ done}$ represents the end of first half-rotation. $\Xi_{second\ rotate}$ represents the second half lid rotating.

B. Results and Discussion

Table. I presents all results of the different algorithms across tasks with different abstract foliated structures.

The first three columns of the table I show the results of the Fetch robot, which is the primary robot of interest for the implementation. Compared to MTG and ALEF, the FoliatedRepMap shows a significant improvement in all metrics. The success rate is consistently higher, especially for hard problems, such as rearranging the shelf and solving the maze, thereby highlighting that incorporating both successful and unsuccessful motion planning experiences into the roadmap can help the motion planner by giving better leads. In addition, the FoliatedRepMap with Atlas shows even better improvement, thereby proving that incorporating the Atlas only enhances the planner. The gain in performance is particularly noticeable in harder problems such as opening the door, rearranging the shelf, and opening the lid.

The FoliatedRepMap can find solutions faster than MTG and ALEF planners in hard problems. Past experiences can quickly help the planner to converge on a valid solution path in fewer iterations. Easier problems can be solved in a tiny number of iterations, so the overhead of the FoliatedRepMap to store experiences causes it to be slower than the MTG and ALEF. In addition, the Atlas Planner can use prior experience more efficiently by sampling more successfully on manifolds and consequently can solve the problem faster. The Atlas-based planner is consistently faster for hard problems. It must be noted that the planning time also includes the time to construct the roadmap, which is roughly 0.1s for the MTG and ALEF planners, and 1s for the FoliatedRepMap. (since it has to duplicate the GMM roadmap for each abstract manifold).

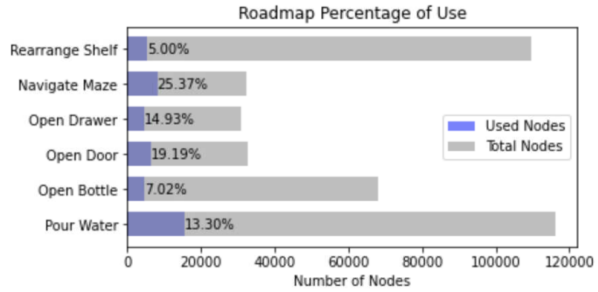


Fig. 15: The dynamic roadmap percentage of use on each task. (Shown for the Atlas variant here)

The FoliatedRepMap can also provide shorter solution paths. By constructing the configuration roadmap in each manifold, FoliatedRepMap achieves a finer granularity thus increasing the density. Thus distances between nodes in the FoliatedRepMap reflect the actual distances between the configurations associated with those nodes, which others don't consider. As a result, the motion tasks generated by the FoliatedRepMap tend to have shorter distances.

The dynamic roadmap design improves the efficiency of the planner. It always achieves a similar success rate as its non-dynamic counterpart, thus confirming that only a small subset of nodes contributes towards planning. As the number of iterations increases, the dynamic planner can eventually reach the size of the total planning graph, ensuring that it can use all the available nodes for planning for harder problems. Fig. 15 shows the final size of the dynamic roadmap at the end of planning compared to the size of the total roadmap size (which is used in the non-dynamic setting). As is evident, the dynamic planner uses only a small fraction of the total roadmap. For a hard problem such as the shelf, where the roadmap size is huge, the dynamic planner uses less than 5% of the total nodes, while maintaining the success rate, which demonstrates the strength of the dynamic strategy. In most cases, it consistently has a lower planning time while still achieving a high success rate. In addition, the higher solution lengths of the dynamic planner can be attributed to the lower density of nodes, which can lead to slightly inefficient paths. Nonetheless, the trade-off in planning times can overshadow the increase in path length.

We perform ablation studies with different numbers of distributions (91, 112, 262, and 1310). The performance of the planners is stable for all distribution numbers with minor differences in their success rates. An increase in the number of distributions increases the planning time as expected. The results are presented in Appendix A.

It must be noted that comparing the planning time across planners must be in the context of success rate. A planner with a higher success rate for a task may also show an increase in average planning time since the additional success rate was likely due to the planner's ability to successfully solve a harder problem, which would naturally take longer to solve.

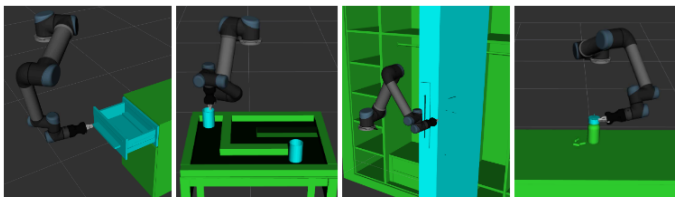


Fig. 16: UR5 experiments on opening drawer, navigating maze, opening door, and opening bottle.

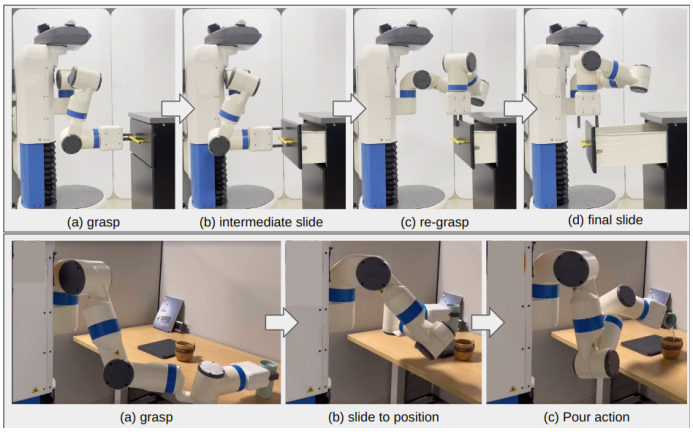


Fig. 17: Real-world experiments: Open drawer and pour water.

To show the adaptability of our system, we also tested it on a UR5 robot arm with a different number of DOF compared to the Fetch robot, as shown in Fig. 16. Although, we did not tune the planners for the UR5, our planning framework consistently achieves a higher level of success, highlighting its ability to generalize effectively. The dynamic planner results for the UR5 robot are however not consistent, which hints at the requirement of additional tuning for the dynamic planner.

C. Real-world Experiment on Fetch Robot

To showcase the feasibility of our planner, we conducted real-world tests using a Fetch Robot [32] to perform two tasks: opening a drawer and pouring water, as shown in Fig. 17. We assume that the locations of objects involved in these tasks are known in advance. The number of manifolds and intersections are generated similar to the simulation experiments. As seen in the figure, the robot is capable of planning and executing these plans in the real world. The motions of the robot during these tasks are captured in the supplementary video.

VII. Conclusion

In this work, we introduce the FoliatedRepMap, a framework for multimodal manipulation problems that can be represented as a foliation problem. By capturing the self-collision-free space in the form of a repetition roadmap using GMMs, which spans across all abstract manifolds, we can encode all prior planning experiences, and use this information to inform the re-planning in the next iteration strategically. The experiment results show that the FoliatedRepMap can efficiently solve different classes of foliated tasks. The atlas and the dynamic enhancements further improve the performance of the planner, by addressing two key bottlenecks. We believe that further refinement of (i) the heuristic function for the dynamic planner, (ii) the similarity function to compare manifolds and (iii) alternative partitions of the configuration space, to encompass a broader range of task-relevant features could lead to even greater improvements in performance.

VIII. Limitations and Future Work

Despite the improved performance of the planner in most tasks, there are a few limitations of that system that could be addressed in future works. Some of them are:

- The manifolds and the intersections need to be pre-generated for each problem set-up. The generated manifolds and intersections are valid only for a specific task, and a specific environment (but are valid for all start and goal positions and all planners). Although the MTG and ALEF can sample intersections while planning, this paper limits the scope to pre-generated intersection. A future work of sampling intersections during planning is in progress.
- Discretization of the space: A fixed number of grasps and intermediate grasps have to be sampled as part of the pre-generation step.
- The heuristic function and the consequent expansion step of the dynamic planner are naive, and the subgraph may expand too quickly for complex problems.
- The method to partition the robot's self-collision-free space using a GMM may be too naive, and a more sophisticated partitioning might improve performance.

It must be noted that most prior works also suffer from the first two limitations.

References

[1] Dmitry Berenson, Siddhartha S. Srinivasa, Dave Ferguson, and James J. Kuffner. Manipulation planning on constraint manifolds. In *2009 IEEE International Conference on Robotics and Automation*, 2009.

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

[3] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012.

[4] Constantinos Chamzas, Zachary Kingston, Carlos Quintero-Peña, Anshumali Shrivastava, and Lydia E Kavraki. Learning sampling distributions using local 3d workspace decompositions for motion planning in high dimensions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[5] Constantinos Chamzas, Aedan Cullen, Anshumali Shrivastava, and Lydia E Kavraki. Learning to retrieve relevant experiences for motion planning. In *2022*

International Conference on Robotics and Automation (ICRA). IEEE, 2022.

[6] Sachin Chitta. Moveit!: an introduction. *Robot Operating System (ROS) The Complete Reference (Volume 1)*, 2016.

[7] David Coleman, Ioan A Şucan, Mark Moll, Kei Okada, and Nikolaus Correll. Experience-based planning with sparse roadmap spanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[8] Peter Englert, Isabel M. Rayas Fernández, Ragesh Kumar Ramachandran, and Gaurav S. Sukhatme. Sampling-based motion planning on sequenced manifolds. In *Proceedings of Robotics: Science and Systems*, 2021.

[9] Thomas S Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, 1973.

[10] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, 2020.

[11] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 2021.

[12] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[13] Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 2010.

[14] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 2011.

[15] Jiaming Hu, Zhao Tang, and Henrik I. Christensen. Multi-modal planning on rearrangement for stable manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[16] Léonard Jaillet and Josep M. Porta. Path planning with loop closure constraints using an Atlas-based RRT. In *International Symposium of Robotics Research*, 2011.

[17] Léonard Jaillet and Josep M. Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 2013.

[18] Jacob J Johnson, Ahmed H Qureshi, and Michael C Yip. Learning sampling dictionaries for efficient and generalizable robot motion planning with transformers. *IEEE Robotics and Automation Letters*, 2023.

[19] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, 2011.

[20] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 2011.

[21] Bernd Kast, Philipp S Schmitt, Sebastian Albrecht, Wendelin Feiten, and Jianwei Zhang. Hierarchical planner with composable action models for asynchronous parallelization of tasks and motions. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2020.

[22] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996.

[23] Beobkyoon Kim, Terry Taewoong Um, Chansu Suh, and Frank C Park. Tangent bundle RRT: A randomized algorithm for constrained motion planning. *Robotica*, 2016.

[24] Zachary Kingston, Mark Moll, and Lydia E. Kavraki. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[25] Zachary Kingston, Andrew M. Wells, Mark Moll, and Lydia E. Kavraki. Informing multi-modal planning with synergistic discrete leads. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[26] Zachary Kingston, Constantinos Chamzas, and Lydia E. Kavraki. Using experience to improve constrained planning on foliations for multi-modal problems. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[27] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 2014.

[28] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[29] Peter Lehner and Alin Albu-Schäffer. The repetition roadmap for repetitive constrained motion planning. *IEEE Robotics and Automation Letters*, 2018.

[30] Mike Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 2010.

[31] Sebastian Stock, Masoumeh Mansouri, Federico Pecora, and Joachim Hertzberg. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[32] Melonee Wise, Michael Ferguson, Daniel King, Eric Diehr, and David Dymesich. Fetch & freight : Standard platforms for service robot applications. Technical report, Fetch Robotics, 2016.

[33] Zhutian Yang, Caelan Reed Garrett, and Dieter Fox. Sequence-based plan feasibility prediction for efficient task and motion planning. In *CoRL 2022 Workshop on Learning, Perception, and Abstraction for Long-Horizon Planning*, 2022.

APPENDIX

A. Ablation Studies on the number of distributions

The heuristic for determining the number of distributions used in this work is borrowed from the Repetition Roadmap [29] work. During the construction of the GMM-roadmap, a large number of motion plans are generated in the self-collision-free space, using a sampling motion planner, such as RRT [28]. The path length (in terms of the number of intermediate configurations sampled) is recorded for each motion plan. The number of distributions is then chosen as the maximum path length across all motion plans generated for the self-collision-free space. For our work, we generate about 10,000 motion plans. The maximum path length yields 262 distributions, the 99.9th percentile gives 112 distributions while the 99th percentile gives 91 distributions. For the results presented in Table I, we choose 91 distributions to remove outliers in the longest path. To test the sensitivity of the distributions, we perform the experiments with 91, 112, 262, and 1310 distributions and record the success rate and planning times. We perform this experiment on the Dynamic FoliatedRepMap with Atlas planner.

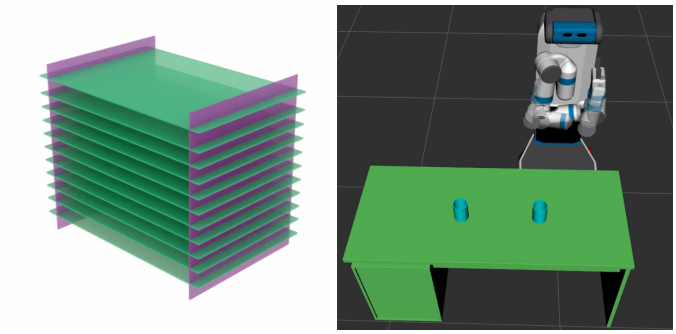
Task	**91** (99th)	**112** (99.9th)	**262** (100%)	**1310** (500%)
	Success Rate %			
Rearrange Shelf	**100.000**	**100.000**	98.000	**100.000**
Open Door	56.000	54.000	52.000	**60.000**
Open Drawer	**100.000**	**100.000**	**100.000**	**100.000**
Navigate Maze	**92.000**	88.000	**92.000**	90.000
	Planning Time (s)			
Rearrange Shelf	6.635	**6.394**	9.141	10.855
Open Door	**68.863**	128.463	76.062	106.010
Open Drawer	**2.363**	2.418	3.588	5.893
Navigate Maze	3.007	**2.866**	4.813	3.942

TABLE II: Success Percentage and Planning Time Across Different Number of Distributions.
(**Bold font**: the best value; underlined font: the second-best value. This table includes results from a new set of experiments, distinct from TABLE I.)

From the table, it is evident that the success rate of the planners is fairly stable across large distribution numbers, and is significantly better than the ALEF and MTG planners. On average, an increase in the number of distributions seems to improve the success rate marginally. However, the planning time increases generally as we increase the number of distributions, since the number of nodes in the roadmap scales quadratically with the distribution number. Thus, we have presented the results with 91 distributions.

B. Foliation Detail of Each Task

This section details the generation of the foliated manifolds and intersections for each task, and describes the task setup.



(a) Foliated structure (b) Setup in RViz

Fig. 18: Visual representation of the sliding cup task.

1) Sliding Cup: The task aims to grasp and slide a cup across an empty table, as shown in Fig 18.

Foliation / Parameter	**Symbol**	**Value**
Un-grasp	$\Xi_{un-grasp}$	
Number of Placements	$\mathcal{P}$	2
Slide	Ξ_{slide}	
Orientation Tolerance	$\sigma_{\text{orient\_slide}}$	$0.01, 0.01, 0.01$ (rad)
Position Tolerance	$\sigma_{\text{pos\_slide}}$	$\infty, \infty, 0.0008$ (m)
Ref Pose Position	$\mathcal{R}_{\text{pos\_slide}}$	$0.68, -0.55, 0.78$
Ref Pose Orientation	$\mathcal{R}_{\text{orient\_slide}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{slide}}$	150

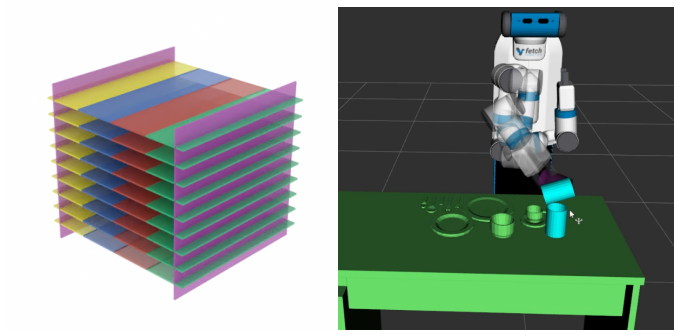
TABLE III: Configurations for Sliding Cup Task

The $\Xi_{un-grasp}$ is defined by the start and goal placements. Ξ_{slide} is defined by a set of $\mathcal{G}_{\text{slide}}$ grasp poses, where each pose corresponds to a manifold and is subject to the constraints $\sigma_{\text{orient\_slide}}$ for orientation and $\sigma_{\text{pos\_slide}}$ for position.

Intersection	**Symbol**
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{slide}}$	$\mathcal{I}_{\text{un-grasp slide}}$

TABLE IV: Intersections for the Sliding Cup Task

Each sampled grasp pose, denoted by $\mathbf{g} \in \Xi_{\text{slide}}$, is paired with a placement pose, denoted by $\mathbf{p} \in \Xi_{\text{un-grasp}}$. Intersections $\mathcal{I}_{\text{slide}}$ comprise a collection of valid motion sequences that transition between the $\Xi_{un-grasp}$ and Ξ_{slide} manifolds. The intersections $\mathcal{I}_{\text{slide}}$ are generated through a random sampling process that selects a combination of $\mathbf{g}$ and $\mathbf{p}$ and checks if the grasp action $\mathbf{g}$ for the selected placement is feasible (valid collision-free configuration). If a valid solution is found, the trajectory to $\mathbf{g}$ is verified and the pair of $\mathbf{p}$ and $\mathbf{g}$ is considered an intersection in $\mathcal{I}_{\text{slide}}$. The process is repeated to accumulate a sufficient number of samples, as specified by N_{samples}. Details of the foliation parameters and intersections detail are provided in Table III and IV.



(a) Foliated structure (b) Setup in RViz

Fig. 19: Visual representation of the pouring water task.

2) Pouring water: The task involves a sequence of actions to pick up a cup of water, transport it, pour the water into another container, and place the cup at a designated location, as shown in Fig 19.

Foliation / Parameter	Symbol	Value
Un-grasp	$\Xi_{un-grasp}$	
Number of Placements	$\mathcal{P}$	9
Transport	$\Xi_{\text{transport}}$	
Orientation Tolerance	$\sigma_{\text{orient\_trans}}$	$0.01, 0.01, 0.01$ (rad)
Position Tolerance	$\sigma_{\text{pos\_trans}}$	$0.0008, \infty, \infty$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_trans}}$	$0.68, -0.55, 0.78$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_trans}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{trans}}$	150
Pour	Ξ_{pour}	
Orientation Tolerance	$\sigma_{\text{orient\_pour}}$	$\pi/3, 0.01, \pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_pour}}$	$0.0008, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_pour}}$	$0.68, 0.05, 0.9$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_pour}}$	$0.95, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{pour}}$	150
Pour Done	$\Xi_{\text{pour-done}}$	
Orientation Tolerance	$\sigma_{\text{orient\_pour-done}}$	$0.01, 0.01, 0.01$ (rad)
Position Tolerance	$\sigma_{\text{pos\_pour-done}}$	$0.0008, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_pour-done}}$	$0.68, 0.05, 0.9$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_pour-done}}$	$0.95, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{pour-done}}$	150
Place	Ξ_{place}	
Orientation Tolerance	$\sigma_{\text{orient\_place}}$	$\pi, \pi/3, \pi/3$ (rad)
Position Tolerance	$\sigma_{\text{pos\_place}}$	$\infty, 0.0008, \infty$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_place}}$	$0.75, 0.05, 0.78$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_place}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{place}}$	150

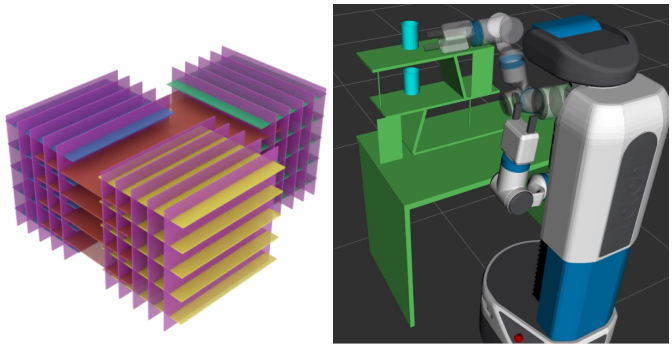
TABLE V: Configurations for Pouring Water Task

The manifolds in $\Xi_{\text{un-grasp}}$ are defined by the start placement and goal placement of the cup. Manifolds in all other foliations are defined by a set of grasps. $\Xi_{\text{transport}}$ is constrained by keeping the cup upright during movement to prevent water spillage. Ξ_{pour} is constrained by tilting the cup to pour the water, while $\Xi_{\text{pour-done}}$ aims to push the cup by confirming the completion of the pouring action. Ξ_{place} is constrained by the setting down of the cup in its final position. All foliations, $\Xi_{\text{transport}}$, Ξ_{pour} and $\Xi_{\text{pour-done}}$ need to be completed using the same grasp, which is represented by the yellow, red and the green parallel manifolds (planes) in the foliated structure. There is only one intersection between the three manifolds(planes) since the robot cannot change its grasp mid-action.

Intersection	Symbol
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{transport}}$	$\mathcal{I}_{\text{un-grasp\_transport}}$
$\Xi_{\text{transport}} \cap \Xi_{\text{pour}}$	$\mathcal{I}_{\text{transport\_pour}}$
$\Xi_{\text{pour}} \cap \Xi_{\text{pour-done}}$	$\mathcal{I}_{\text{pour\_pour-done}}$
$\Xi_{\text{pour-done}} \cap \Xi_{\text{place}}$	$\mathcal{I}_{\text{pour-done\_place}}$
$\Xi_{\text{place}} \cap \Xi_{\text{un-grasp}}$	$\mathcal{I}_{\text{place\_un-grasp}}$

TABLE VI: Intersections for the Pouring Water Task

The first and the last intersections $\mathcal{I}_{\text{un-grasp\_transport}}$ and $\mathcal{I}_{\text{place\_un-grasp}}$ are generated similarly to the sliding cup task. For the other intersections, since each triplet of horizontal manifolds (yellow, green, red) is connected by a single grasp, all configurations that satisfy the three manifolds and have the same grasp are added as intersections.



(a) Foliated structure (b) Setup in RViz

Fig. 20: Visual representation of the rearrange shelf task.

3) Rearrange Shelf: The task entails relocating a cup from one shelf level to another and subsequently sliding it to a specified target position on the new level, as shown in Fig 20. The manifold in $\Xi_{\text{un-grasp}}$ is defined by the placements of the cup. The manifold in Ξ_{trans} is defined by the grasps and constrained by the vertical displacement of the cup between different shelf levels. $\Xi_{\text{slide\_table}}$, $\Xi_{\text{slide\_shelf\_1}}$, and $\Xi_{\text{slide\_shelf\_2}}$, are defined by the grasps and constrained by moving the cup along the surface of different level.

To facilitate the transfer between different heights, as illustrated in Fig 20a, the red panel represents a foliation with small constraints in the y dimensions, ensuring that the cup's movement is restricted to the first column of the shelf.

Each foliation is defined similarly to the previous tasks, with a set of grasp poses and associated orientation tolerances σ_{orient}. The intersections $\mathcal{I}_{\text{un-grasp\_slide\_table}}$, $\mathcal{I}_{\text{un-grasp\_slide\_shelf\_1}}$, $\mathcal{I}_{\text{un-grasp\_slide\_shelf\_2}}$, and $\mathcal{I}_{\text{un-grasp\_trans}}$ are generated through a similar sampling process, ensuring the validity of grasp and placement combinations within the constraints of the task. For comprehensive foliation and intersection settings, please refer to Table VII and VIII.

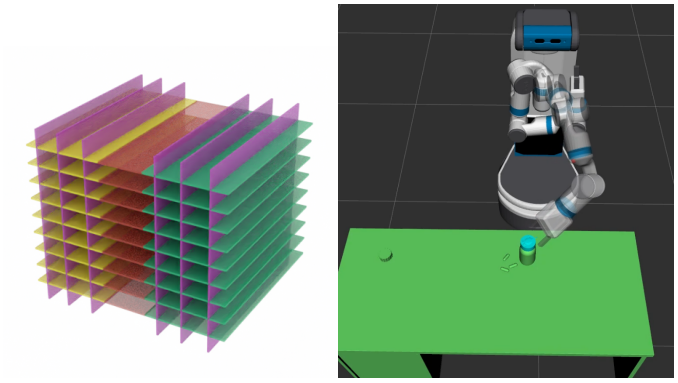
4) Opening Bottle: The task of opening a bottle is defined by the rotational manipulation required to unscrew the bottle lid, as shown in Fig 21. The rotational foliations Ξ_{fr}, Ξ_{frd}, and Ξ_{sr} constrained to distinct angular segments within the lid's 360-degree unscrewing motion. The manifold in $\Xi_{\text{un-grasp}}$ is defined by the set of placements that may be necessary for re-grasping the lid during the rotation.

Foliation / Parameter	Symbol	Value
Un-grasp	$\Xi_{\text{un-grasp}}$	
Number of Placements	$\mathcal{P}$	11
Slide Table	$\Xi_{\text{slide\_table}}$	
Orientation Tolerance	$\sigma_{\text{orient\_slide\_table}}$	$0.01, 0.01, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_slide\_table}}$	$\infty, \infty, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_slide\_table}}$	$0.5, 0.25, 0.78$
Num of Grasps	$\mathcal{G}_{\text{slide\_table}}$	150
Slide Shelf 1	$\Xi_{\text{slide\_shelf\_1}}$	
Orientation Tolerance	$\sigma_{\text{orient\_slide\_shelf\_1}}$	$0.001, 0.001, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_slide\_shelf\_1}}$	$\infty, \infty, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_slide\_shelf\_1}}$	$0.77, 0.25, 1.02$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_slide\_shelf\_1}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{slide\_1}}$	150
Slide Shelf 2	$\Xi_{\text{slide\_shelf\_2}}$	
Orientation Tolerance	$\sigma_{\text{orient\_slide\_shelf\_2}}$	$0.001, 0.001, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_slide\_shelf\_2}}$	$\infty, \infty, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_slide\_shelf\_2}}$	$0.77, 0.25, 0.24$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_slide\_shelf\_2}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{slide\_2}}$	150
Transport	Ξ_{trans}	
Orientation Tolerance	$\sigma_{\text{orient\_trans}}$	$0.001, 0.001, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_trans}}$	$\infty, 0.0008, \infty$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_trans}}$	$0.0, 0.25, 0.0$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_trans}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{trans}}$	150

TABLE VII: Configurations for Transporting Cup on a Shelf Task

Intersection	Symbol
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{slide\_table}}$	$\mathcal{I}_{\text{un-grasp\_slide\_table}}$
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{slide\_shelf\_1}}$	$\mathcal{I}_{\text{un-grasp\_slide\_shelf\_1}}$
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{slide\_shelf\_2}}$	$\mathcal{I}_{\text{un-grasp\_slide\_shelf\_2}}$
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{trans}}$	$\mathcal{I}_{\text{un-grasp\_trans}}$

TABLE VIII: Intersections for the Transporting Cup on a Shelf Task



(a) Foliated structure (b) Setup in RViz

Fig. 21: Visual representation of the open bottle task

To facilitate a complete rotation, $\mathcal{P}$ intermediate angles are sampled within the $\Xi_{\text{un-grasp}}$, capturing the incremental opening stages of the lid. The intersections $\mathcal{I}_{\text{un-grasp\_fr}}$ and $\mathcal{I}_{\text{un-grasp\_sr}}$ are generated through a similar sampling process, which ensures that when a single grasp pose is insufficient for a full rotation, re-grasping can occur at specific intermediate states. $\mathcal{I}_{\text{un-grasp\_frd}}$ aims to ensure that the bottle indeed rotates 360 degrees. As illustrated in Fig 21a, the red panel represents a constrained region within the lid's rotation, specifically a 180-degree segment. This segmentation is purposefully designed

Foliation / Parameter	Symbol	Value
Un-grasp	$\Xi_{\text{un-grasp}}$	
Number of Placements	$\mathcal{P}$	20
First Rotate	Ξ_{fr}	
Orientation Tolerance	$\sigma_{\text{orient\_fr}}$	$0.01, 0.01, \pi/2$ (rad)
Position Tolerance	$\sigma_{\text{pos\_fr}}$	$0.0008, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_fr}}$	$0.842, -0.074, 0.865$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_fr}}$	$0.0, 0.0, \pi/2$
Num of Grasps	$\mathcal{G}_{\text{first\_rotate}}$	80
First Rotate Done	Ξ_{frd}	
Orientation Tolerance	$\sigma_{\text{orient\_frd}}$	$0.01, 0.01, 0.01$ (rad)
Position Tolerance	$\sigma_{\text{pos\_frd}}$	$0.0008, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_frd}}$	$0.842, -0.074, 0.865$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_frd}}$	$0.0, 0.0, \pi$
Num of Grasps	$\mathcal{G}_{\text{first\_rotate\_done}}$	80
Second Rotate	Ξ_{sr}	
Orientation Tolerance	$\sigma_{\text{orient\_sr}}$	$0.01, 0.01, \pi/2$ (rad)
Position Tolerance	$\sigma_{\text{pos\_sr}}$	$0.0008, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pos\_sr}}$	$0.842, -0.074, 0.865$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_sr}}$	$0.0, 0.0, 3\pi/2$
Num of Grasps	$\mathcal{G}_{\text{second\_rotate}}$	80

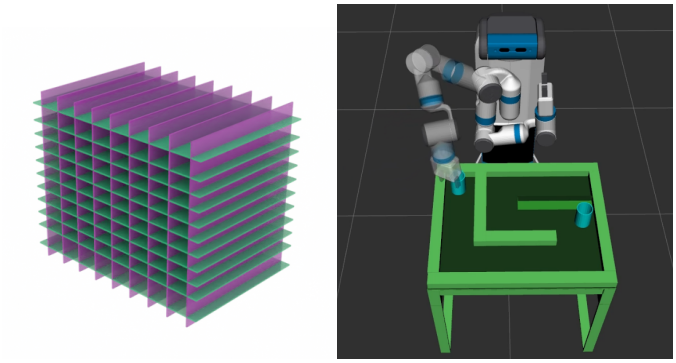
TABLE IX: Configurations for Opening Bottle Task

to ensure that the lid can be rotated through the entire 360-degree range by enforcing pass-through at the intermediate orientations defined by the foliations.

Intersection	Symbol
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{fr}}$	$\mathcal{I}_{\text{un-grasp\_fr}}$
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{frd}}$	$\mathcal{I}_{\text{un-grasp\_frd}}$
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{sr}}$	$\mathcal{I}_{\text{un-grasp\_sr}}$

TABLE X: Intersections for the Opening Bottle Task

Detailed configurations of the foliations are provided in Table IX, and the intersections are specified in Table X.



(a) Foliated structure (b) Setup in RViz

Fig. 22: Visual representation of the maze task.

5) Maze: The maze task requires the manipulation of a cup through a labyrinthine path from an initial to a final location, involving both sliding movements and multiple re-grasping actions. The manifold in $\Xi_{\text{un-grasp}}$ is defined by the set of all potential placements, while the manifold in $\tau_{\text{slide\_table}}$ is defined by the sliding motion of the cup within the constraints.

To navigate the maze, a series of intermediate sliding and grasping actions are sampled. Similar to the open bottle task, the intersections $\mathcal{I}_{\text{un-grasp\_slide\_table}}$ are sampled randomly,

Foliation / Parameter	Symbol	Value
Un-grasp	$\Xi_{\text{un-grasp}}$	
Number of Placements	$\mathcal{P}$	21
Slide	$\tau_{\text{slide\_table}}$	
Orientation Tolerance	$\sigma_{\text{orient\_slide}}$	$0.001, 0.001, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pos\_slide}}$	$\infty, \infty, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pose\_slide}}$	$0.65, 0.00, 0.78$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_slide}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{slide}}$	150

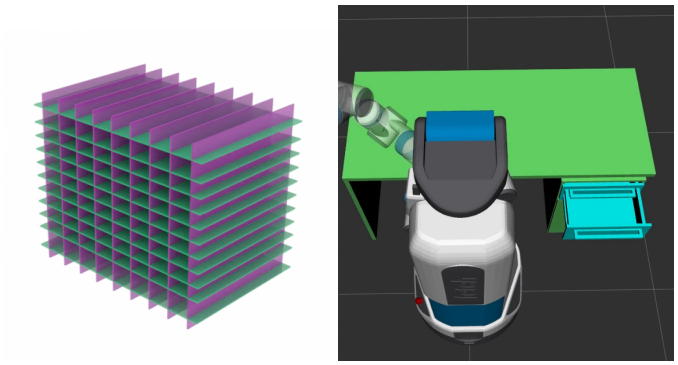
TABLE XI: Configurations for Maze Task

ensuring that the cup can transition between sliding and re-grasping. However, unlike the bottle opening task, there are no predefined intermediate states that must be traversed.

Intersection	Symbol
$\tau_{\text{slide\_table}} \cap \Xi_{\text{un-grasp}}$	$\mathcal{I}_{\text{un-grasp\_slide\_table}}$

TABLE XII: Intersections for the Maze Task

The detailed configurations of the foliations are provided in Table XI, and the intersections are specified in Table XII.



(a) Foliated structure (b) Setup in RViz

Fig. 23: Visual representation of the open drawer task.

6) Opening Drawer: The task of opening a drawer involves a sequence of actions that include grasping the drawer handle, pulling it to open, and re-grasping the handle at intermediate stages if necessary. The manifold in $\Xi_{\text{un-grasp}}$ is defined by a set of $\mathcal{P}$ intermediate placements. The manifold in Ξ_{pull} is given by $\mathcal{G}$ grasp poses on the handle to pull it open.

Foliation / Parameter	Symbol	Value
Un-grasp		
Number of Placements	$\mathcal{P}$	15
Pull	Ξ_{pull}	
Orientation Tolerance	$\sigma_{\text{orient\_pull}}$	$0.001, 0.001, 0.001$ (rad)
Position Tolerance	$\sigma_{\text{pose\_pull}}$	$\infty, 0.0008, 0.0008$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pose\_pull}}$	$0.4, -0.650, 0.51$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_pull}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{pull}}$	100

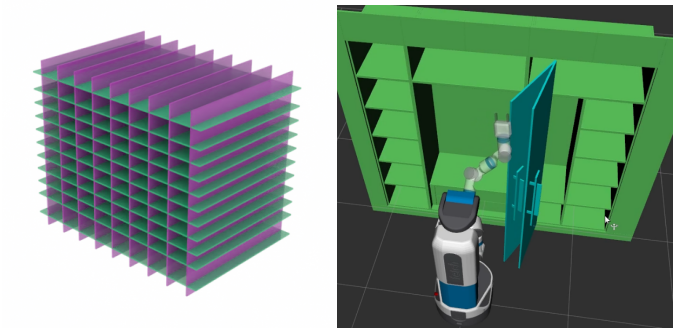
TABLE XIII: Configurations for Opening Drawer

The foliation configurations for the task are detailed in Table XIII, which includes the number of placements $\mathcal{P}$, orientation tolerance $\sigma_{\text{orient\_pull}}$, and position tolerance $\sigma_{\text{pose\_pull}}$. The reference poses $\mathcal{R}_{\text{pose\_pull}}$ and $\mathcal{R}_{\text{orient\_pull}}$ establish the desired

Intersection	Symbol
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{pull}}$	$\mathcal{I}_{\text{un-grasp\_pull}}$

TABLE XIV: Intersections for the Opening Drawer Task

end state for the pull action. The intersections are generated through a stochastic sampling process, ensuring that the robot can transition between grasping and pulling as required to achieve the drawer's opening. The detailed configurations of the foliations are provided in Table XIII, and the intersections are specified in Table XIV.



(a) Foliated structure (b) Experiment in RViz

Fig. 24: Visual representation of the open door task. (a) Foliated structure. (b) Experiment in RViz.

7) Opening Door: The task of opening a door is defined by a series of actions that include grasping the door handle, pulling it to initiate the opening, and re-grasp the handle as needed. The task is defined by a collection of $\mathcal{G}$ grasp manifolds and $\mathcal{P}$ un-grasp manifolds.

Foliation / Parameter	Symbol	Value
Un-grasp		
Number of Placements	$\mathcal{P}$	25
Pull	Ξ_{pull}	
Orientation Tolerance	$\sigma_{\text{orient\_pull}}$	$0.001, 0.001, 2\pi$ (rad)
Position Tolerance	$\sigma_{\text{pose\_pull}}$	$0.0002, 0.0002, 0.0002$ (m)
Reference Pose Position	$\mathcal{R}_{\text{pose\_pull}}$	$1.18, -0.60, -0.20$
Reference Pose Orientation	$\mathcal{R}_{\text{orient\_pull}}$	$0.0, 0.0, 0.0$
Num of Grasps	$\mathcal{G}_{\text{pull}}$	125

TABLE XV: Configurations for Opening Door

The foliation for the task are detailed in Table XV. The

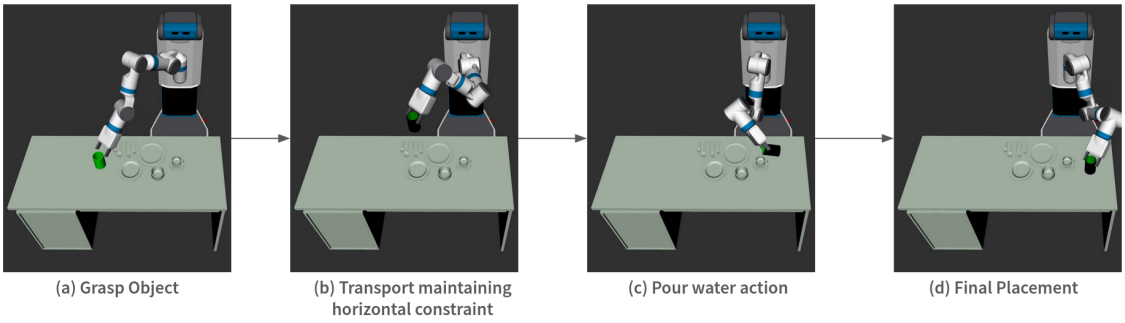
Intersection	Symbol
$\Xi_{\text{un-grasp}} \cap \Xi_{\text{pull}}$	$\mathcal{I}_{\text{un-grasp\_pull}}$

TABLE XVI: Intersections for the Opening Door Task

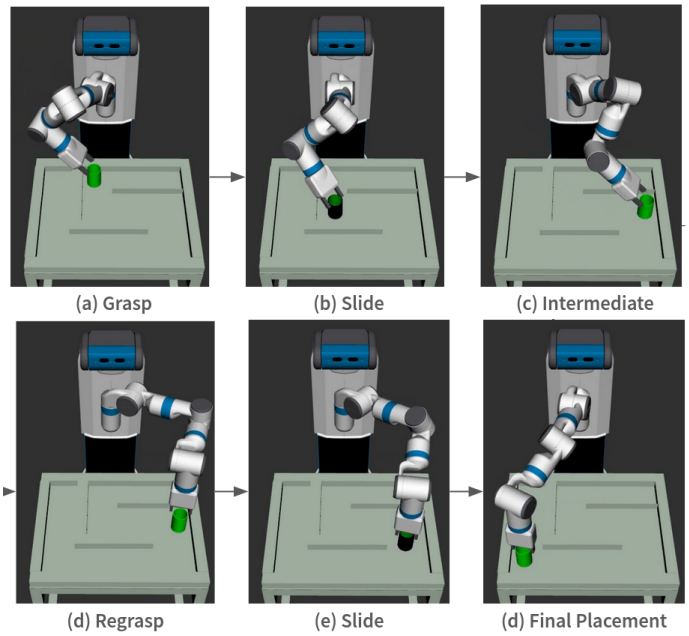
$\mathcal{I}_{\text{un-grasp\_pull}}$ is generated similarly to the previous task, paired with a specific manifold with a grasp pose and a manifold with door placement. The detailed configurations of the foliations are provided in Table XV, and the intersections are specified in Table XVI.

C. Examples of Task Sequences Generated

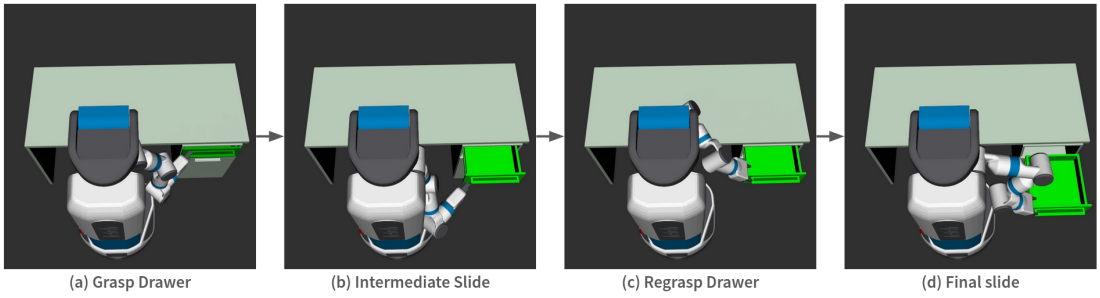
Figure 25 shows examples of task sequences generated and executed by the FoliatedRepMap with Atlas planner.



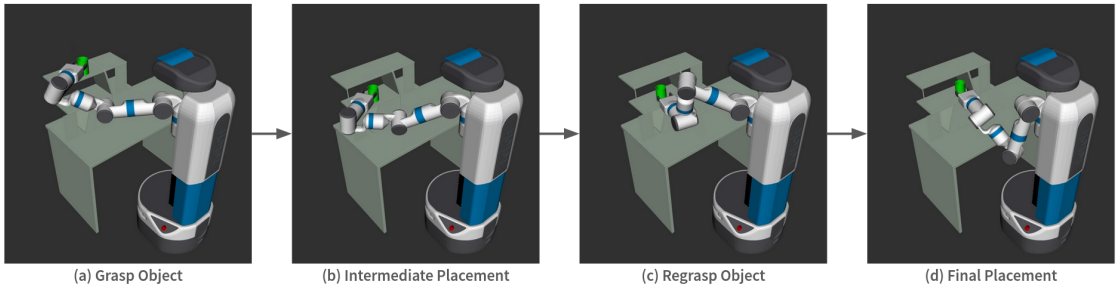
(a) Pour water



(b) Navigate Maze



(c) Open Drawer



(d) Rearrange Shelf

Fig. 25: Examples of successfully completed tasks