2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Spectral Decomposition of Inverse Dynamics for Fast Exploration in Model-Based Manipulation

Solvin Sigurdson[1], Benjamin Riviere[2], and Joel Burdick[1]

***Abstract*— Planning long duration robotic manipulation sequences is challenging because of the complexity of exploring feasible trajectories through nonlinear contact dynamics and many contact modes. Moreover, this complexity grows with the problem's horizon length. We propose a search tree method that generates trajectories using the spectral decomposition of the inverse dynamics equation. This equation maps actuator displacement to object displacement, and its spectrum is efficient for exploration because its components are orthogonal and they approximate the reachable set of the object while remaining dynamically feasible. These trajectories can be combined with any search based method, such as Rapidly-Exploring Random Trees (RRT), for long-horizon planning. Our method performs similarly to recent work in model-based planning for short-horizon tasks, and differentiates itself with its ability to solve long-horizon tasks: whereas existing methods fail, ours can generate 45 second duration, 10+ contact mode plans using 15 seconds of computation, demonstrating real-time capability in highly complex domains.**

I. Introduction

Manipulation is a robot's primary means to interact with and change its environment. However, planning for manipulation is challenging because such interactions have complex dynamics and require creating long-horizon plans that change the state of surrounding objects through a variety of contact modes. In many cases, leveraging an understanding of how the object and the robot can interact with the environment is necessary for successful plans. Consider for instance rearranging objects on a desk where the shape or weight of the objects do not allow for pick and place manipulation, or sliding heavy objects around a construction site.

Model-based optimization has been widely successful in generating real-time trajectory plans for a variety of nonlinear dynamical systems such as drones [1], spacecraft [2], and locomotion [3]. However, applying these methods to robotic manipulation remains a challenge: the change in contact modes during manipulation creates a discontinuity in the gradient of the dynamics, making local linearization and optimization inaccurate. In addition, the number of possible contact modes between manipulators, objects, and the environment grows combinatorially in complex manipulation tasks, inhibiting a direct hierarchical approach of planning discrete contact sequences and then continuous optimization.

[1]SS and JB are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA solvin.sigurdson@caltech.edu, jburdick@caltech.edu

[2]BR is with the Department of Mechanical and Aerospace Engineering and the Department of Computer Science, New York University, New York City, NY 10012, USA riviere.b@nyu.edu

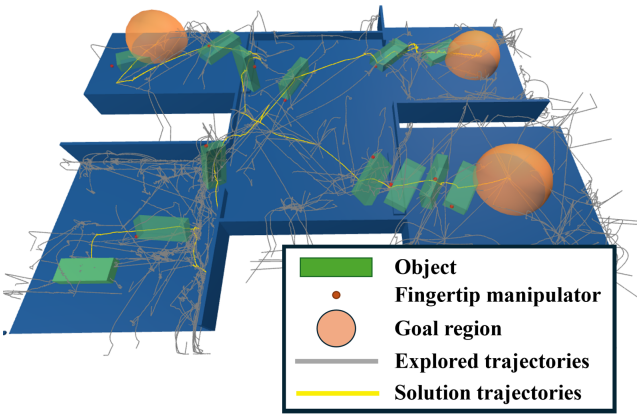


Fig. 1: Our method efficiently plans manipulation trajectories through several contact modes, discovering behavior like pushing, sliding, rolling, tumbling, and pivoting. In the physics-based simulator, an object (green prism) is moved with non-prehensile manipulation (red dot) through a contact-rich environment to a goal configuration.

To address these challenges, we propose a new hierarchy: tree search over trajectories generated from the spectrum of the inverse dynamics. The algorithm inherits global exploration from search and reduces reliance on inaccurate linearizations and susceptibility to local minima that are both inherent in applying first-order optimization methods to the entire trajectory. At the same time, the algorithm uses local dynamics information from the spectrum of the inverse dynamics to reduce search complexity: the local linearization's eigenvectors are efficient for exploration because they are orthogonal and approximate the reachable set of the object while remaining dynamically feasible. This spectral representation can be formalized as a Markov Decision Process (MDP), and paired with Rapidly-Exploring Random Trees (RRT) [4], [5] (or other search algorithms) to chain branches together and efficiently construct long-horizon trajectories.

We verify this approach in simulation, see Fig. 1, on non-prehensile manipulation tasks that slide, pivot, and rotate a rectangular prism through a highly varied environment involving many contact modes. Interacting with the environment via non-trivial extrinsic dexterity is necessary in order to complete the task. On a desktop computer, our method uses $\sim$15 seconds of computation time to generate plans of 45 seconds physical duration, demonstrating long-horizon planning capability while remaining fast enough for real-time deployment and re-planning.

979-8-3315-8160-2/26/$31.00 ©2026 IEEE

Overall, our method produces real-time complex plans involving contact between objects, manipulators, and the surrounding environment. The method is model-based and can be applied zero-shot without pre-training, fine-tuning or hyperparameter tuning in a variety of scenarios. Although the focus of this work is extrinsic dexterity and non-prehensile manipulation, it could be extended in the future to dexterous manipulation and loco-manipulation.

II. Background and Related Works

Robot manipulation is a rich field that spans hybrid systems theory, optimization, search, and machine learning.

The problem of planning and controlling an object's motion via pushing, sliding, rolling, tumbling, pivoting, etc. was traditionally viewed as a hybrid system of discrete contact modes, where within each contact mode the model reduces to a continuous non-linear system to which standard optimization techniques can be applied [6]. Although this formulation can be solved with mixed-integer-programming methods [7] [8] or search over contact mode transitions [9] [10], the complexity of these algorithms scales with the number of contact modes, limiting this formulation for long-horizon manipulation tasks.

To avoid contact mode enumeration, contact-implicit methods use compliant contact models to generate contact forces based solely on the relative configuration of objects, leading to a single unified dynamics model for all contact modes [11] [12]. This approach enables model-based optimization methods to generate solution trajectories. However the approach is heavily susceptible to local minima, and it requires a good initial guess [13].

Sampling-based methods are a standard approach to overcome local minima. For example, one could sample the input space directly at each timestep using Predictive Sampling [14], but sampling in a high-dimensional space quickly becomes intractable for long-horizon plans whose intermediate goals do not lie directly between the start and goal.

It is possible to reduce the search complexity by focusing on the object's motion alone [15] [16] [17]. However, efficiently proposing dynamically realizable motions for the object is challenging: [15] uses a nested search structure to find dynamically feasible motions, which can lead to inefficiency if inner searches fail to produce solutions for outer layers, [16] does not address uni-directional property of friction and is limited to a convex relaxation of the contact dynamics, and [18] proposes dynamically feasible paths but requires time to first build a roadmap offline.

Our method falls into this category of search-based methods and achieves both low sample complexity and dynamic feasibility by using the spectral decomposition of the inverse dynamics equation: its local linearization's eigenvectors are efficient for exploration because they are orthogonal and approximate the reachable set of the object while remaining dynamically feasible. Both our approach and [19] use spectral decompositions to reduce search complexity. However, whereas they use the spectrum of the locally linearized system's Grammian, we use the spectrum of the inverse dynamics equation, which is a specialized transformation that focuses on actuator to object displacement.

Recent progress has been made using data-driven methods. End-to-end visuomotor policies grounded in reinforcement learning and behaviour cloning can directly map camera inputs to manipulator actions, removing the need to explicitly plan through challenging contact dynamics [20]. The main challenge for these methods is demonstrating appropriate generalization and reliability, as success on the task often depends on its proximity to training examples, although recent work in foundation models for robotics are attempting to overcome this difficulty [21]. An alternate line of work considers how to use generative models to produce trajectories directly [22] although this again suffers from canonical out-of-distribution (OOD) challenges if the test time data distribution is not similar enough to that of the training. In contrast, purely model-based methods are inherently general because they do not rely on a training distribution, and are more interpretable, which makes them more reliable in an industrial setting. However, they do still have challenges: model-based methods suffer when the dynamics model is inaccurate, requiring good feedback controllers or online adaptation to bridge the sim-to-real gap.

Some works attempt to balance these regimes by merging physics-driven models with data- and simulation-driven adaptation. For instance, model-based methods can generate inductive biases for learning based methods, achieving the best of both worlds [23], [24].

III. Method

A. Problem Definition

This work seeks to efficiently address the problem of planning the motions of a single rigid body over a complex geometric terrain, where a high number of contact interactions between the rigid body, the manipulator, and the surrounding environment may be needed to reach the goal configuration. The geometry and basic material properties of the body, manipulator, and environment are assumed to be known a priori. For the purpose of simplicity in exposition, we first consider the manipulator as a single spherical fingertip, although the algorithm extends to an

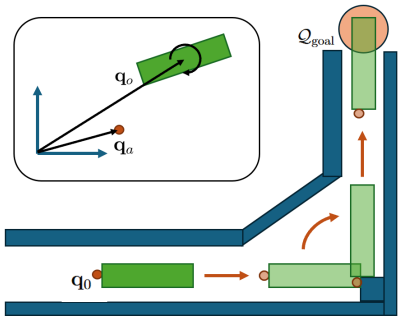


Fig. 2: Top left: Variables for the problem setup. Bottom right: Simple example of the fingertip exhibiting extrinsic dexterity in order to move an object through a maze.

arbitrary number of fingertips, which could be remapped to a dexterous hand or bimanual manipulator.

Taking $\mathbf{q} = \begin{bmatrix}\mathbf{q}_a & \mathbf{q}_o\end{bmatrix} \in \mathbb{R}^{n_a+n_o} = \mathbb{R}^n$ as the combined configuration of the n_a actuated degrees of freedom ($\mathbf{q}_a$) and n_o object degrees of freedom ($\mathbf{q}_o$), the problem is to find a dynamically feasible trajectory $\mathbf{q}_{0:H} \in \mathbb{R}^{n\times H}$ where $\mathbf{q}_0 = \mathbf{q}_{\text{init}}$ and $\mathbf{q}_H \in \mathcal{Q}_{\text{goal}}$ for a given $\mathbf{q}_{\text{init}}$ and $\mathcal{Q}_{\text{goal}}$, and H is the length of the trajectory (see Fig. 2). A trajectory is dynamically feasible if it is generated by a physics-based simulator – we use Drake [25], which implements hydroelastic contact models and is considered a close approximation of real world contact dynamics.

This problem setting is particularly applicable to cases where the manipulator does not have complete actuation authority over the object's degrees of freedom (e.g. non-prehensile manipulation). In this setting, the process of changing the object's state often relies on interactions between the object and the environment. Note that the framework is not limited to this case, and in fully actuated circumstances (i.e. where classical pick and place grasping is available), the method can discover grasping solutions. In this work, $n_a = 3$ and $n_o = 6$ for a single fingertip and a single free rigid body with Euler angle parameterization.

B. Preliminaries

We model the dynamics of the object and manipulator using the standard Euler-Lagrange equations, with a term for the contact forces λ_i between the i^{th} object pair.

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{k}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{B}\mathbf{u} + \sum_i \mathbf{J}_i(\mathbf{q})^\top \lambda_i \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the mass matrix, $\mathbf{k}(\mathbf{q},\dot{\mathbf{q}})$ accounts for gravity and Coriolis terms, $\mathbf{B} \in \mathbb{R}^{n\times n_a}$ is the actuation matrix, $\mathbf{u} \in \mathbb{R}^{n_a}$ are the forces applied to the actuated joints, and $\mathbf{J}_i(\mathbf{q}) \in \mathbb{R}^{3\times n}$ are the Jacobians mapping the contact forces into the generalized variables. We use the compliant contact model from [13][26], which uniquely specifies the contact force $\lambda_i \in \mathbb{R}^3$ between objects as a function of configuration $\mathbf{q}$. Drake is used to calculate queries for relative separation and normal velocity in the contact model.

Similar to [13], given $\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}$ we can invert the dynamics to find the instantaneous external forces $\tau_{\text{ext}} \in \mathbb{R}^n$ that would result in $\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}$ assuming full actuation authority:

$$\tau_{\text{ext}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{k}(\mathbf{q},\dot{\mathbf{q}}) - \sum_i \mathbf{J}_i(\mathbf{q})^\top \lambda_i(\mathbf{q}) \tag{2}$$

However, our system is underactuated and cannot apply forces to all of the degrees of freedom, therefore a given $\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}$ is only feasible if the necessary external forces on the unactuated degrees of freedom are zero:

$$\mathbf{h}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \mathbf{H}\tau_{\text{ext}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \mathbf{0} \tag{3}$$

where $\mathbf{H} \in \mathbb{R}^{n_o\times n}$ selects the unactuated elements of τ_{ext}.

C. Spectral Decomposition of Inverse Dynamics Equation

Given an initial configuration, $\mathbf{q}_0$, we want to compute a set of trajectories that are representative of the possible motions of the object. These trajectories will be used later in the search methods.

First, we approximate $\mathbf{h}$ with a linearization $\hat{\mathbf{h}}$, about a nominal point: $(\bar{\mathbf{q}},\dot{\bar{\mathbf{q}}},\ddot{\bar{\mathbf{q}}})$:

$$\begin{aligned}\hat{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \mathbf{h}(\bar{\mathbf{q}},\dot{\bar{\mathbf{q}}},\ddot{\bar{\mathbf{q}}}) + \frac{\partial \mathbf{h}}{\partial \mathbf{q}}(\mathbf{q}-\bar{\mathbf{q}}) \\ + \frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}}(\dot{\mathbf{q}}-\dot{\bar{\mathbf{q}}}) + \frac{\partial \mathbf{h}}{\partial \ddot{\mathbf{q}}}(\ddot{\mathbf{q}}-\ddot{\bar{\mathbf{q}}})\end{aligned} \tag{4}$$

We simplify this function by only considering a subset of possible trajectories. In particular, we fix the linearization trajectory at the current position with zero velocity and zero acceleration: $(\bar{\mathbf{q}},\dot{\bar{\mathbf{q}}},\ddot{\bar{\mathbf{q}}}) = (\mathbf{q}_0,\mathbf{0},\mathbf{0})$, and we assume that a constant acceleration occurs over timestep Δt to achieve velocity $\dot{\mathbf{q}}$: $(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = (\mathbf{q}_0,\dot{\mathbf{q}},\dot{\mathbf{q}}/\Delta t)$. Although these assumptions limit the set of trajectories we can consider, they are reasonable for small Δt in cases when the state of the system comes to rest when changing manipulator positions.

Substituting in these assumptions, $\hat{\mathbf{h}}(\cdot,\cdot,\cdot)$ is simplified to:

$$\hat{\mathbf{h}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \left(\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}} + \frac{\partial \mathbf{h}}{\partial \ddot{\mathbf{q}}}\frac{1}{\Delta t}\right)\dot{\mathbf{q}} \tag{5}$$

where we assume $\mathbf{h}(\mathbf{q}_0,\mathbf{0},\mathbf{0}) = \mathbf{0}$ because the system is linearized about a dynamically feasible point.

Setting our approximation of $\mathbf{h}$ to zero and applying chain rule for $\frac{\partial \mathbf{h}}{\partial \ddot{\mathbf{q}}}$ leads to the following equation:

$$\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}}\dot{\mathbf{q}} = \begin{bmatrix}\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_a} & \frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_o}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{q}}_a \\ \dot{\mathbf{q}}_o\end{bmatrix} = \mathbf{0} \tag{6}$$

which can be manipulated into the desired form:

$$\dot{\mathbf{q}}_o = -\left(\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_o}\right)^\dagger \frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_a}\dot{\mathbf{q}}_a = \mathbf{A}(\mathbf{q}_a,\mathbf{q}_o)\dot{\mathbf{q}}_a \tag{7}$$

where $\dagger$ is the pseudoinverse.

The configuration-dependent matrix $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{n_o\times n_a}$ is a mapping between the motion of the actuated degrees of freedom and the object motion. The range space of $\mathbf{A}$ is the set of possible velocities for the object given the current state $\mathbf{q}_a,\mathbf{q}_o$, which cover the one step reachable set when integrated. We can approximate this set with a finite number of vectors constructed from the eigenvectors $\mathbf{v}_i$ of $\mathbf{A}\mathbf{A}^\top$:

$$\sigma_i^2, \mathbf{v}_i = \text{Eig}(\mathbf{A}\mathbf{A}^\top) \tag{8}$$

In implementation, we calculate derivatives $\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_a}$ and $\frac{\partial \mathbf{h}}{\partial \dot{\mathbf{q}}_o}$ around nominal point $(\bar{\mathbf{q}},\dot{\bar{\mathbf{q}}},\ddot{\bar{\mathbf{q}}})$ by finite differences, and ignore eigenpairs with eigenvalues such that $\frac{\sigma_i^2}{\sigma_{\max}^2} < C_{\text{eig}}$.

D. Reachable Set Approximation

We use the spectral decomposition from (8) to construct a set of candidate trajectories with Algorithm 1, which is outlined in the following steps.

First, we sample N_{grasps}-many initial configurations for the same object state $\mathbf{q}_o$ by sampling fingertip locations on

the object (ObjGeometry) and using inverse kinematics to determine associated configurations $\mathbf{q}_j$ (Lines 1 and 2 of Algorithm 1). The linearized object motion directions $\mathbf{v}_i$ are computed using the spectral decomposition from (8) for each initial configuration (Line 3 of Algorithm 1).

Some of these motions will be infeasible for the nonlinear system because of the linearization error. For example, linearized dynamics allow the fingertips to pull on the object instead of push, and for environment penetration by the object. We filter out the former infeasible proposals by comparing the fingertip velocity vector with the fingertip-to-object normal vectors (removed if inner product below threshold $C_{\text{fingertip}}$), and the latter by comparing any significant object center of mass velocity with environment-to-object normals (removed if inner product below C_{env} and translational to rotational motion ratio over C_{ratio}). This procedure is denoted as FILTERPROPOSALS (Line 4 of Algorithm 1).

At this point, the number of representative motions is proportional to N_{grasps}. However, these motions could be similar to each other, and each additional motion costs computation by increasing the branching factor of the resulting tree. To maintain an efficient representation, a representative set of motions is chosen using kmeans clustering with N_{clusters} (Line 5 of Algorithm 1). The cluster centers in $\mathcal{V}_{\text{cc}}$ are then normalized by matrix $\mathbf{W} \in \mathbb{R}^{n_o \times n_o}$ to balance translational and rotational degrees of freedom (Line 6 of Algorithm 1).

Given the linearized object velocity from the prior steps, we now compute a trajectory $\mathbf{q}^i_{\text{traj}} \in \mathbb{R}^{n \times N_i}$ using the exact nonlinear dynamics, encapsulated in PDROLLOUT (Line 7 of Algorithm 1). To achieve this, we use a proportional derivative (PD) controller that actuates the fingertips and tracks a desired position trajectory that keeps the fingertip stationary relative to the object's surface, while the object moves with velocity $\mathbf{v}_i$, see Fig. 3. The desired fingertip position setpoint for the PD controller can be calculated at any time t_0 using Equations 9-11 evaluated at time $t_0 + T_{\text{proj}}$.

$$\mathbf{q}_o(t) = \mathbf{q}_o(t_0) \ + \mathbf{v}_i(t - t_0) \tag{9}$$

$$\bar{\mathbf{p}}_w(t) = \mathbf{T}_{wo}(q_o, t)\bar{\mathbf{p}}_o \tag{10}$$

$$\mathbf{u} = \mathbf{K}_d\dot{\mathbf{q}}_a + \mathbf{K}_p(\mathbf{q}_a - \mathbf{p}_w) \tag{11}$$

where T_{proj} is the duration of the forward projection in seconds, $\bar{\mathbf{p}}_o \in \mathbb{R}^4$ is the homogeneous vector associated with $\mathbf{p}_o \in \mathbb{R}^3$ the contact location in the object body frame, $\mathbf{T}_{wo} \in \mathbb{R}^{4\times4}$ is the transform to the world frame, and $\bar{\mathbf{p}}_w \in \mathbb{R}^4$ is the homogeneous vector associated with $\mathbf{p}_w \in \mathbb{R}^3$ the contact location in the world frame. The setpoint is updated every T_{track} seconds during the simulation rollout using the current object position and fingertip contact location, which may have shifted if the finger slips during the rollout. This controller explores object motion near $\mathbf{v}_i$ while accounting for non-linear contact dynamics via the simulator.

While the trajectory is being rolled out, we check if the object loses contact with the fingertip over threshold d_{contact}, whether motion has stopped below threshold v_{stopped} relative to the initial velocity, whether the object has rotated by ϕ_{max}, or whether time T_{max} has elapsed, in which case we stop tracking the proposed motion $\mathbf{v}_i$. The approximate reachable set $\mathcal{R}$ is the union of the trajectories produced by the rollouts, see Fig. 4. We specify the hyperparameters of the subroutines PDROLLOUT, FILTERPROPOSALS, and KMEANS in Sec. IV.

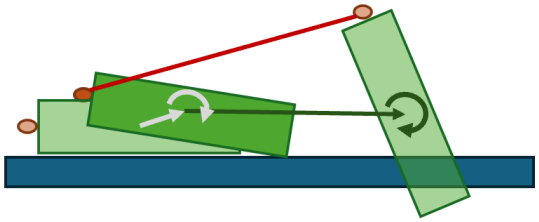


Fig. 3: An illustration of the fingertip controller. The dark green arrows are the linearized motion $\mathbf{v}_i$ projected forward T_{proj} seconds, the gray arrows are the nonlinear motion, and the end of the red line is the fingertip reference point.

Algorithm 1

Input: Configuration $\mathbf{q}_o$
Output: Reachable set $\mathcal{R}$
Hyperparameters N_{grasps}, $\mathbf{W}$

1: $\mathcal{P} \leftarrow \{\mathbf{p}^o_j \in \mathbb{R}^3 | \mathbf{p}^o_j \sim \text{ObjGeometry}, j = 1 \dots N_{\text{grasps}}\}$
2: $\mathcal{Q} \leftarrow \{\mathbf{q}_j = \text{InverseKinematics}(\mathbf{p}^o_j), \forall j = 1 \dots N_{\text{grasps}}\}$
3: $\mathcal{V}_{\text{full}} \leftarrow \{\mathbf{v}_i | \ \_, \mathbf{v}_i = \text{Eig}(\mathbf{A}_j\mathbf{A}_j^\top), \ j = 1 ... N_{\text{grasps}}\}$
4: $\mathcal{V}_{\text{feasible}} = \text{FILTERPROPOSALS}(\mathcal{V}_{\text{full}})$
5: $\mathcal{V}_{\text{cc}} \leftarrow \text{KMEANS}(\mathcal{V}_{\text{feasible}})$
6: $\mathcal{V}_{\text{norm}} \leftarrow \{\mathbf{v}'_i \mid \mathbf{v}'_i = \frac{\mathbf{v}_i}{\sqrt{\mathbf{v}_i^T\mathbf{W}\mathbf{v}_i}} \ \forall \mathbf{v}_i \in \mathcal{V}_{cc}\}$
7: $\mathcal{R} \leftarrow \{\mathbf{q}^i_{\text{traj}} | \mathbf{q}^i_{\text{traj}} = \text{PDROLLOUT}(\mathbf{q}_i, \mathbf{v}_i) \ \forall \mathbf{v}_i \in \mathcal{V}_{\text{norm}}\}$
return $\mathcal{R}$

E. Reachable Sets in Search-Based Planners

We use the reachable set to construct a Markov Decision Process (MDP) that can be used in search-based planners, see Fig. 5, and we specify one possible algorithmic variant using Rapidly-Exploring Random Trees (RRT).

We define the elements of the MDP, the novelty of which is the action set generator that uses our previously defined reachable set $\mathcal{R}$:

$$\mathcal{S} : \{\mathbf{q}_{0:H} \in \mathbb{R}^{n\times H}, \ H \in \mathbb{N}\} \tag{12}$$

$$\mathcal{A}(\mathbf{s} \in \mathcal{S}) : \{\mathbf{q}^i_{0:N_i} | \mathbf{q}^i_{0:N_i} \in \mathcal{R}(\mathbf{q}_{o,H})\} \tag{13}$$

$$\mathcal{T}(\mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}) : \mathbf{s}' = \mathbf{s} \oplus \mathbf{a} \tag{14}$$

$$\mathcal{R}(\mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}, \mathbf{s}' \in \mathcal{S}) : \{1 \text{ if } \mathbf{s}'_{H'} \in \mathcal{Q}_{\text{goal}}\} \tag{15}$$

where the $\mathbf{q}_{0:H}$ are discretized system trajectories over a horizon H, which grows as new path segments are added, N_i is the length of the i^{th} trajectory in the reachable set, $\oplus$ denotes concatenation of the trajectories in time, $\mathcal{Q}_{\text{goal}}$ is the set of states that are considered sufficiently close to the goal state to end the search (with radius R_{terminal}), and the reward is not used in every search method (e.g. RRT), but is included here for completeness.

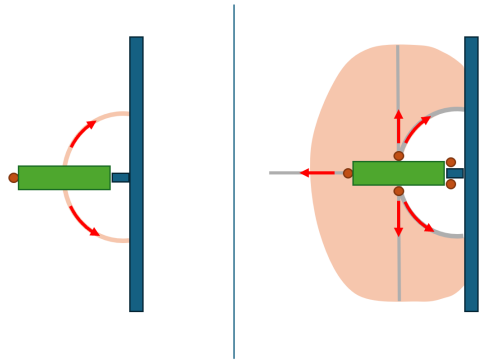


Fig. 4: Left: For a single configuration where the object is pinned between the fingertip and the environment, proposed motion directions $\mathbf{v}_i$ are shows with red arrows, while the true reachable set for the center of the box is shown in pink. Right: Same as the left image except for a set of 5 different initial configurations to capture sampling different contacts between the object and manipulator. The approximate reachable set for the object's COM is shown in gray, and efficiently captures a distinct subset of all possible motions in pink.

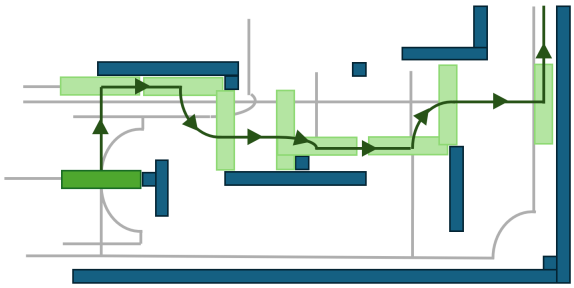


Fig. 5: Pictoral representation of the MDP tree for a simple maze problem. In grey are the possible motion paths for the center of mass, while in dark green is a possible path chosen by the RRT algorithm.

We implement a standard RRT algorithm [4], [5] with the following choices of **Sample**, **Select**, and **Steer**. The choices of **Sample** and **Select** are standard, our novelty is in the construction of $\mathcal{A}(s^*)$ that is used to **Steer**.

Sample: Randomly sample an object configuration within a predefined problem-dependent bounding box: $\mathbf{q}^o_{\text{sub-goal}} \sim [a_i, b_i]^{n_o}$. We use standard "goal biasing": we sample the center of the problem goal region $\mathcal{Q}_{\text{goal}}$ with frequency $\alpha \in [0, 1]$.

Select: We select node $\mathbf{s}^*$ for expansion, which has the smallest Euclidean distance between the trajectory's end state $\mathbf{q}_H$ and the randomly sampled configuration. While the distance calculation can include both translation and orientation factors, we used only a translation-based metric.

Steer: Compute the transition for a random action $\mathbf{a} \in \mathcal{A}(\mathbf{s}^*)$ from Algorithm 1, and add N_{nodes} new states to the tree, evenly spaced along the new trajectory segment $\mathbf{a}$.

On Algorithmic Completeness - Due to environment contact constraints, solution trajectories may exist in narrow passages of configuration space. We suspect our algorithm is resolution complete through these passages as individual rollout length decreases, although this increases computational cost and we do not include a formal proof.

IV. Experimental Results and Discussion

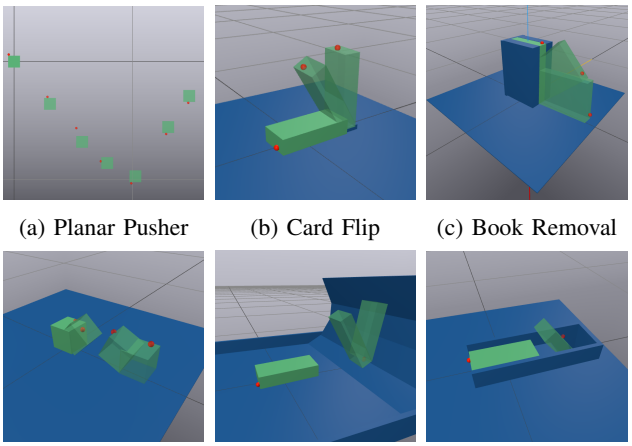
We evaluate our method's efficiency and accuracy on several simulations, focusing on non-prehensile manipulation and extrinsic dexterity. The computation was entirely CPU-based (Intel i9-14900K 3.20 GHz CPU) since forward rollouts are done in Drake. The forward rollouts were parallelized on all 24 cores. In order to avoid instantaneous changes of fingertip position (recalling each segment in $\mathcal{A}$ in (13) begins with the single fingertip in a new position), we additionally append $\mathbf{q}_{\text{settle}}$ to each transition in (14), which is a short forward simulation with the fingertip removed. Forward simulation is done with discrete timestep Δt_{action} during computation of elements of $\mathcal{R}$, and Δt_{settle} during fingertip removal.

We use the same hyperparameters for all experiments: $N_{\text{grasps}} = 100$, $N_{\max} = 50$, $T_{\text{proj}} = 2.0$ [s], $T_{\text{track}} = 0.4$ [s], $\alpha = 0.2$, $K_p = 5000$, $K_d = 500$, $\Delta t_{\text{action}} = 10^{-2}$ [s], $\Delta t_{\text{settle}} = 10^{-3}$ [s], $\Delta t = 10^{-3}$ [s], $d_{\text{contact}} = 10^{-3}$ [m], $v_{\text{stopped}} = 10^{-2}$, $\phi_{\max} = 2.5$ [rad], $T_{\max} = 10$ [s], $N_{\text{clusters}} = 9$, $C_{\text{fingertip}} = -0.1$, $C_{\text{env}} = -0.3$, $C_{\text{ratio}} = 0.05$, $C_{\text{eig}} = 10^{-6}$, $R_{\text{terminal}} = 0.2$ [m], $N_{\text{nodes}} = 5$, $\mathbf{W}$ has diagonal elements 40.5 for translational d.o.fs and 0.405 for rotational d.o.fs - accounting for an object length of roughly 0.1m and normalizing a unit of rotation to $\pi/2$. Note that if the object properties varied more significantly, the impedance gains and length scales could be automatically chosen from the object's inertia and size.

A. Short-Horizon Extrinsic Dexterity Tasks

We first consider single fingertip variants of the extrinsic dexterity tasks proposed in [15] (picking up a card, removing a book from a bookshelf, flipping a box) alongside some additional tasks (planar pushing, flipping via a half-pipe, sliding a prism into a slot), see Fig. 6. To pick up the card with a single fingertip, we added a small lip in the environment for the finger to push the card against. In all cases, the finger leverages interaction with the environment to achieve its goal - for instance, in the half-pipe scenario, the finger must slide the prism up the half-pipe to let gravity tip it over backwards.

We present the results of this experiment in Table I. The table columns correspond to each task: planar pushing (PP), card flipping (CF), book removal (BR), box flipping (BF), half-pipe (HP), prism slot (PS). The table rows correspond to each metric: Success is a binary value indicating that the goal configuration is reached, Time is the wall clock time until an exit condition is reached, Modes is the final plan's number of fingertip contact mode changes, and Branches is an indication of the search tree size. We collect statistics from



(a) Planar Pusher (b) Card Flip (c) Book Removal

(d) Box Flip (e) Half-pipe Flip (f) Prism Slot

Fig. 6: Short-Horizon Extrinsic Dexterity Tasks

Method	Metric	PP	CF	BR	BF	HP	PS
Ours	Success	1.0	1.0	1.0	1.0	1.0	1.0
	Time [s]	6.04	2.06	0.75	3.02	3.28	2.56
	Modes	2.6	4.7	2.5	6.1	6.7	6.0
	Branches	73.5	60.1	11.7	82.4	90.5	54.7
Bsl. 1 [16]	Success	0.9	0.05	0.0	0.25	–	–
	Time [s]	3.1	17.9	>30.3	10.3	–	–
Bsl. 2 [15]	Success	–	1.0	1.0	1.0	–	–
	Time [s]	–	5.1	1.2	4.6	–	–

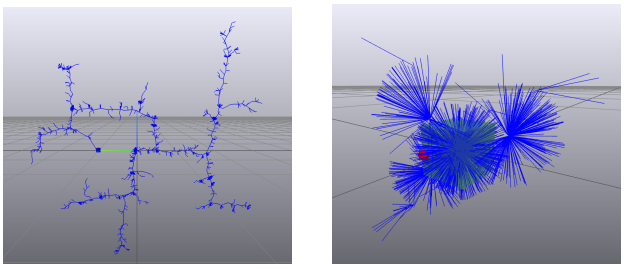
TABLE I: Performance on Short-Horizon Tasks

running each experiment 20 times with different random seeds and report the average across trials.

We compare against two state of the art baselines for contact rich global planning from Pang et al. [16] (Baseline 1) and Cheng et al. [15] (Baseline 2), using their open-source code. We re-implement the first four tasks for Baseline 1, and indicate the results reported in [15] for Baseline 2.

Despite being successful on 2D Planar Pushing, the baseline from [16] struggles with short horizon extrinsic dexterity tasks for 3D objects because the tree search collapses to a breadth first search regime, see Fig. 7. We suspect this is because their method relies on a distance metric that becomes poorly conditioned in higher dimensions: we sampled the distance metric over thousands of states and found that the eccentricity of their distance metric is above 0.9 (i.e. highly anisotropic) in 99% of the samples in 3D, while only 1% are above 0.9 in 2D. Our method avoids this challenge because our exploration is guided by the spectral decomposition.

Our results indicate that we achieve comparable timing to the $\sim$5 seconds reported in the original experiments from [15]. However, whereas their method separates the process of proposing object motions from the process of determining motion actuation via contact, our method jointly proposes distinct, actuatable motions without the need for nested tree search. This allows our method to scale to long-horizon tasks, which we present next.



(a) 2D Planar Pusher (b) 3D Rectilinear Pusher

Fig. 7: Plots of tree expansion using Baseline 1 [16] for 2D and 3D pusher tasks. Tree branches shown in blue. 3D tasks collapse to breadth first search.

B. *Long-horizon Extrinsic Dexterity Maze Task*

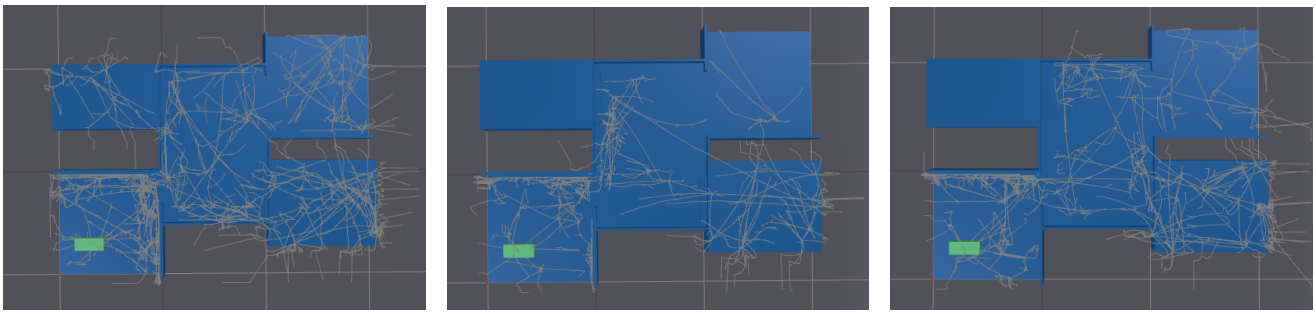
To determine our methods's function and computational timing on a longer horizon problem, we consider a hand-crafted maze-environment formed by 5 shelves of varying heights (see Figs. 1 and 8). This environment has rises, drops, protruding boxes, walls, and doorways for the object to move around and through. The 5 shelves are arranged in an X-shaped configuration, and we randomly sample each of the four non-central shelves as starting and ending locations for the planner. We run 50 random trials and report the average results in Table II, where metrics other than Success Rate are only calculated for the trials that are successful. Trials are given 180 seconds of computer time to find a solution.

Additionally, we ablate key elements of our algorithm to determine their influence on performance. We also include Baseline 2 since it was successful on the short-horizon tasks. This baseline can fail in less than 180 seconds since there is a computational limit on the tree size, which we made as large as possible. We reported the average failure time as a lower bound on the solution time in Table II.

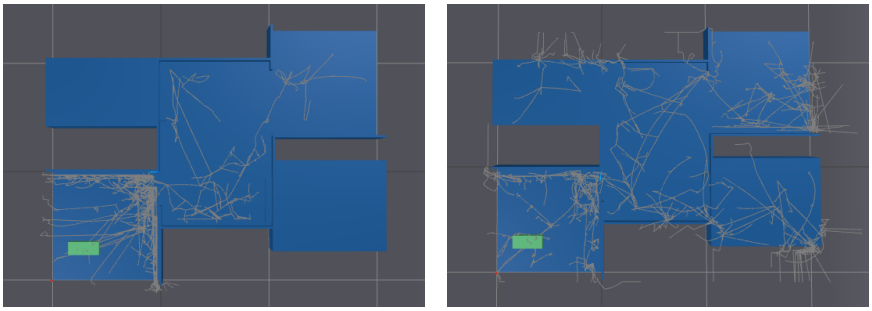
- *Ablation 1*: During RRT Expansion, expand every trajectory from $\mathcal{A}(\mathbf{s}^*)$ instead of selecting a random one.
- *Ablation 2*: Remove the KMEANS clustering step.
- *Ablation 3*: Remove the FILTERPROPOSALS step.
- *Ablation 4*: During construction of $\mathcal{A}(\mathbf{s}^*)$, replace the proposed $\mathbf{v}_i$ from Equation 8 with a random vector.
- *Baseline 2*: Solving the task using [15].

A qualitative view of tree expansion under each ablation can be seen in Fig. 8, which illustrates the discovered tree paths for the maze problem beginning on the bottom left shelf. This is a challenging initial condition since progress can only be made by tipping the object onto a box near the doorway to the central shelf. Non-terminal trajectories were not available for plotting from Baseline 2.

As shown in Table II, the algorithm's perfect success rate is indicative of its ability to plan through multiple contact modes across long task horizons. This success happens in spite of the fact that many motions plans must pass through very specific sequences of object motions in order to be successful, or through small gaps, such as when going from the bottom-left and upper-right platforms to the central platform. The tree efficiently considers a wide variety of



(a) Our Method (b) Ablation 1 (c) Ablation 2



(d) Ablation 3 (e) Ablation 4

Fig. 8: Plots of tree expansion after 60s of computation in the 3D maze environment

Method	Success	Time [s]	Modes	Branches	Length [m]
Ours	1.0	15.0	9.34	436	3.90
Abl. 1	1.0	35.1	13.8	389	4.46
Abl. 2	0.96	20.55	10.8	492	3.32
Abl. 3	0.74	20.0	11.9	591	3.04
Abl. 4	0.88	13.3	31.1	634	3.15
Bsl. 2*	0.0	>32.5	∼	∼	∼

TABLE II: Ablation (Abl.) and Baseline (Bsl.) Performance. *Ablations 1–4 have 50 trials and Baseline 2 has 10 trials.

diverse possible motions in order to quickly cover the entire area, as evidenced by the coverage of the maze in Fig. 8a.

The results in Table II also demonstrate that the wall clock planning time is less than the generated plan's execution time, assuming that the plans are followed with an object speed of 0.05-0.2 m/s (depending upon the local complexity along the planned trajectory). These long-horizon motions take upwards of 30-45 seconds to execute. This means that future plans can begin to be generated during the original object motion, allowing real-time re-planning.

The relative performance of our algorithm and Baseline 2 clearly demonstrates improvements over existing work [15]. We first associate this performance gap with issues in dynamic feasibility of the exploration: whereas [15] proposes object motions before determining whether they can be achieved via contact, our proposed motion uses the range space of $\mathbf{A}(q)$ to inform whether the fingertip is capable of actuating those motions. Secondly, their nested search structure coupled with the long-horizon duration limits the number of available rollouts to build the search tree.

The remaining ablations knock out key subroutines in our algorithm and reveal our method's most critical features. Ablation 1 favors a breadth first exploration by expanding all the children of a node before continuing the exploration process. Although this ablation leads to a smaller average number of branches to reach a solution, it requires more total computation time. The timing gap with Ablation 2 demonstrates that planning duration and tree expansion efficiency arises from clustering the proposed actions. This result is expected given that clustering reduces the number of exploratory choices to a few effective ones. The performance gap with Ablation 3 demonstrates the importance of proposal filtering, which prevents non-physical behavior like pulling on objects or pushing into the solid surroundings. Ablation 4 follows random vectors rather than informed principle directions, resulting in lower success rate, higher number of contacts, and higher number of explored branches.

Finally, we present a visual analysis of the methods and ablations in Fig. 8. We notice that our full method is able to cover the entire space within the allotted time frame, while most of the ablations fail to find at least one of the platforms. Ablation 4's relatively high success rate demonstrates that exploratory rollouts in *any* direction in a high fidelity simulator form a key part of reaching the goal. However, the shorter choppier motion segments and increased contact changes that come from ignoring the inverse dynamics and dynamically actuatable motions cover the space less thoroughly and are likely far less robust in deployment.

V. Conclusion and Future Work

This work proposed a model-based planning algorithm that is specialized for manipulation tasks involving long-horizons and multiple contact modes requiring physical interaction with the environment. The core of our proposed method is a new hierarchical decomposition: tree search over trajectories generated from the spectrum of the inverse dynamics. This combination inherits global exploration from search and efficient exploration from local information in the inverse dynamics equation. The method's effectiveness was demonstrated in high-fidelity simulations of long-horizon tasks, where our method generated complete solutions faster than state-of-the-art baselines and achieved a perfect success rate. Our simulations also demonstrated planning times that are less than plan execution time, demonstrating a capability for real-time deployment and online re-planning. In future work, we will continue to develop this decomposition with experimental demonstrations and algorithmic improvements focusing on transient dynamic contact motions and robustness to model uncertainty.

References

[1] D. Hanover, A. Loquercio, *et al.*, "Autonomous drone racing: A survey," *IEEE Trans. on Robotics*, vol. 40, pp. 3044–3067, 2024.

[2] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *J. Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[3] A. Hereid, S. Kolathaya, and A. D. Ames, "Online optimal gait generation for bipedal walking robots using legendre pseudospectral optimization," in *IEEE Conf. Decision and Control*, 2016, pp. 6173–6179.

[4] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *int. J. Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[6] E. R. Westervelt, J. W. Grizzle, *et al.*, *Feedback Control of Dynamic Bipedal Robot Locomotion*. 2007.

[7] T. Marcucci and R. Tedrake, "Mixed-integer formulations for optimal control of piecewise-affine systems," Apr. 2019.

[8] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Trans. on Automatic Control*, vol. PP, pp. 1–1, Jul. 2020.

[9] M. Toussaint, K. R. Allen, *et al.*, "Differentiable physics and stable modes for tool-use and manipulation planning - extended abtract," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 6231–6235.

[10] Z. Kingston and L. E. Kavraki, "Scaling multimodal planning: Using experience and informing discrete search," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 128–146, 2023.

[11] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[12] S. Le Cleac'h, T. A. Howell, *et al.*, "Fast contact-implicit model predictive control," *IEEE Trans. on Robotics*, vol. 40, pp. 1617–1629, 2024.

[13] V. Kurtz, A. Castro, *et al.*, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *Int. J. Robotics Research*, 2025.

[14] T. Howell, N. Gileadi, *et al.*, *Predictive sampling: Real-time behaviour synthesis with mujoco*, 2022.

[15] X. Cheng, S. Patil, *et al.*, "Enhancing dexterity in robotic manipulation via hierarchical contact exploration," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 390–397, 2024.

[16] T. Pang, H. J. T. Suh, *et al.*, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," 2022.

[17] H. Zhu, A. Meduri, and L. Righetti, "Efficient object manipulation planning with monte carlo tree search," in *IEEE/RSJ Int. Conf. intelligent robots and systems*, 2023, pp. 10 628–10 635.

[18] H. J. T. Suh, T. Pang, *et al.*, *Dexterous contact-rich manipulation via the contact trust region*, 2025.

[19] B. Rivière, J. Lathrop, and S.-J. Chung, "Monte carlo tree search with spectral expansion for planning with dynamical systems," *Science Robotics*, vol. 9, no. 97, eado1010, 2024.

[20] S. Levine, C. Finn, *et al.*, *End-to-end training of deep visuomotor policies*, 2016.

[21] K. Black, N. Brown, *et al.*, *π^0: A vision–language–action flow model for general robot control*, 2024.

[22] V. Kurtz and J. W. Burdick, *Generative predictive control: Flow matching policies for dynamic and difficult-to-demonstrate tasks*, 2025.

[23] H. Zhu, T. Zhao, *et al.*, "Should we learn contact-rich manipulation policies from sampling-based planners?" *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6248–6255, 2025.

[24] B. Riviere, W. Hönig, *et al.*, "Neural tree expansion for multi-robot planning in non-cooperative environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6868–6875, 2021.

[25] R. Tedrake and the Drake Development Team, *Drake: Model-based design and verification for robotics*, 2019.

[26] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *J. of Applied Mechanics*, vol. 42, no. 2, pp. 440–445, 1975.