Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Sampling-based Exploration for Reinforcement Learning of Dexterous Manipulation

Gagan Khandate*†, Siqi Shang*†, Eric T. Chang‡, Tristan Luca Saidi†, Yang Liu‡,
Seth Matthew Dennis‡, Johnson Adams‡ and Matei Ciocarlie‡
†Dept. of Computer Science ‡Dept. of Mechanical Engineering *joint first authorship
Columbia University, New York, NY 10027, USA
Corresponding email: gagank@cs.columbia.edu

***Abstract*—In this paper, we present a novel method for achieving dexterous manipulation of complex objects, while simultaneously securing the object without the use of passive support surfaces. We posit that a key difficulty for training such policies in a Reinforcement Learning framework is the difficulty of exploring the problem state space, as the accessible regions of this space form a complex structure along manifolds of a high-dimensional space. To address this challenge, we use two versions of the non-holonomic Rapidly-Exploring Random Trees algorithm; one version is more general, but requires explicit use of the environment's transition function, while the second version uses manipulation-specific kinematic constraints to attain better sample efficiency. In both cases, we use states found via sampling-based exploration to generate reset distributions that enable training control policies under full dynamic constraints via model-free Reinforcement Learning. We show that these policies are effective at manipulation problems of higher difficulty than previously shown, and also transfer effectively to real robots. A number of example videos can also be found on the project website: sbrl.cs.columbia.edu**

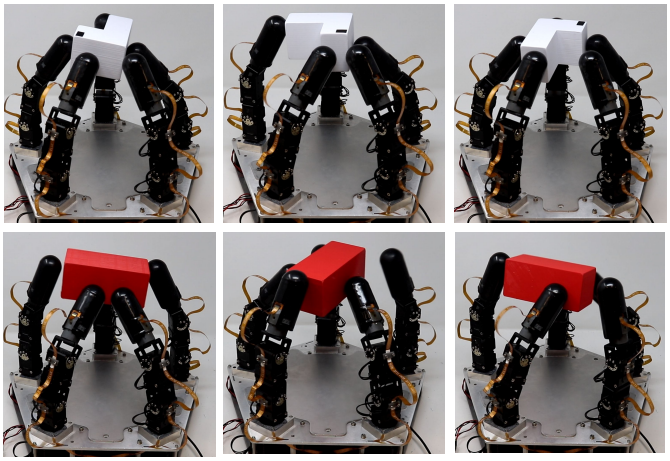

Fig. 1: Our method enables finger-gaiting manipulation of concave or elongated objects which require complex gaits. We demonstrate these gaits both in simulation and on a real robot hand (shown here), using only proprioceptive and tactile feedback intrinsic to the hand.

I. Introduction

Reinforcement Learning (RL) of robot sensorimotor control policies has seen great advances in recent years, demonstrated for a wide range of motor tasks. In the case of manipulation, this has translated in higher levels of dexterity than previously possible, typically demonstrated by the ability to re-orient a grasped object in-hand using complex finger movements [1–3].

However, training a sensorimotor policy is still a difficult process, particularly for problems where the underlying state space exhibits complex structure, such as "narrow passages" between parts of the space are accessible or useful. Manipulation is indeed such a problem: even when starting with the object secured between the digits, a random action can easily lead to a drop, and thus to an irrecoverable state. Finger-gaiting further implies transitions between different subsets of fingers used to hold the object, all while maintaining stability. This leads to difficulty in exploration during training, since random perturbations in the policy action space are unlikely to discover narrow passages in state space. Current studies address this difficult through a variety of means: using simple, convex objects to reduce the difficulty of the task, reliance on support surfaces to reduce the chances of a drop, object pose tracking through extrinsic sensing, etc.

The difficulty of exploring problems with labyrinthine state space structure is far from new in robotics. In fact, the large and highly effective family of Sampling-Based Planning (SBP) algorithms was developed in the field to address this exact problem. By expanding a known structure towards targets randomly sampled in the state space of the problem (as opposed to the action space of the agent), SBP methods can explore even very high-dimensional state spaces in ways that are probabilistically complete, or guaranteed to converge to optimal trajectories. However, SBP algorithms are traditionally designed to find trajectories rather than policies. For problems with computationally demanding dynamics, SBP can not be used on-line for previously unseen start states, or to quickly correct when unexpected perturbations are encountered along the way.

In this paper, we draw on the strength of both RL and SBP methods in order to train motor control policies for in-hand manipulation with finger gaiting. We aim to manipulate more difficult objects, including concave shapes, while securing them at all times without relying on support surfaces. Furthermore, we aim to achieve large re-orientation of the grasped object with purely intrinsic (tactile and proprioceptive) sensing. To achieve that, we explore multiple variants of the non-holonomic RRT algorithm with added constraints to find (approximate) trajectories that explore the useful parts of the

problem state space. Then, we use these trajectories as reset distributions to train complete RL policies based on the full dynamics of the problem. Overall, the main contributions of this work include:

- To the best of our knowledge, we are the first to show that reset distributions generated via SBP with kinematic constraints can enable more efficient training of RL control policies for dexterous in-hand manipulation.
- We show that SBP can explore useful parts of the manipulation state space by using either analytical approximations for contact and stability constraints, or by explicitly using the system's transition function (if available). When analytical approximations are used, RL later fills in the gaps by learning appropriate actions under more realistic dynamic constraints.
- The exploration boost from SBP allows us to train policies for dexterous skills not previously shown, such as in-hand manipulation of concave shapes, with only intrinsic sensing and no support surfaces. We demonstrate these skills both in simulation and on real hardware.

II. Related Work

Exploration methods for general RL operate under the strict assumption that the learning agent cannot teleport between states, mimicking the constraints of the real world. Under such constraints, proposed exploration methods include using intrinsic rewards [4, 5] or improving action consistency via temporally correlated noise in policy actions [6] or parameter space noise [7].

Fortunately, in cases where the policies are primarily trained in simulation, this requirement can be relaxed, and we can use our knowledge of the relevant state space to design effective exploration strategies. A number of these methods improve exploration by injecting useful states into the reset distribution during training. Nair et al. [8] use states from human demonstrations in a block stacking task, while Ecoffet et al. [9, 10] use states previously visited by the learning agent itself for problems such as Atari games and robot motion planning. Tavakoli et al. [11] evaluate various schemes for maintaining and resetting from the buffer of visited states. However, these schemes were evaluated only on benchmark continuous control tasks [12]. From a theoretical perspective, Agarwal et al. [13] show that a favorable reset state distribution provides a means to circumvent worst-case exploration issues, using sample complexity analysis of policy gradients.

Finding feasible trajectories through a complex state space is a well-studied motion planning problem. Of particular interest to us are sampling-based methods such as Rapidly exploring Random Trees (RRT) [14–16] and Probabilistic Road Maps (PRM) [17, 18]. These families of methods have proven highly effective, and are still being expanded. Stable Sparse-RRT (SST) and its optimal variant SST* [19] are examples of recent sampling-based methods for high-dimensional motion planning with physics. However, the goal of these methods is finding (kinodynamic) trajectories between known start and goal states, rather than closed-loop control policies which can handle deviations from the expected states.

Several approaches have tried to combine the exploratory ability of SBP with RL, leveraging planning for global exploration while learning a local control policy via RL [20–22]. These methods were primarily developed for and tested on navigation tasks, where nearby state space samples are generally easy to connect by an RL agent acting as a local planner. The LeaPER algorithm [23] also uses plans obtained by RRT as reset state distribution and learns policies for simple non-prehensile manipulation. However, the state space for the prehensile in-hand manipulation tasks we show here is highly constrained, with small useful regions and non-holonomic transitions. Other approaches use trajectories planned by SBP as expert demonstrations for RL [24], but this requires that planned trajectories also include the actions used to achieve transitions, which SBP does not always provide. Alternatively, Jurgenson et al. [25] and Ha et al. [26] use planned trajectories in the replay buffer of an off-policy RL agent for multi-arm motion planning. However, it is unclear how off-policy RL can be combined with the extensive physics parallelism that has been vital in the recent success of on-policy methods for learning manipulation [2, 27, 28].

Turning specifically to the problem of dexterous manipulation, a number of methods have been used to advance the state of the art, including planning, learning, and leveraging mechanical properties of the manipulator. Leveroni et al. [29] build a map of valid grasps and use search methods to generate gaits for planar reorientation, while Han et al. [30] consider finger-gaiting of a sphere and identify the non-holonomic nature of the problem. Some methods have also considered RRT for finger-gaiting in-hand manipulation [31, 32], but limited to simulation for a spherical object. More recently, Morgan et al. demonstrate robust finger-gaiting for object reorientation using actor-critic reinforcement learning [33] and multi-modal motion planning [34], both in conjunction with a compliant, highly underactuated hand designed explicitly for this task. Bhatt et al. [35] also demonstrate robust finger-gaiting finger-pivoting manipulation with a soft compliant hand, but these skills were not autonomously learned but rather hand-designed and executed in an open-loop fashion.

Model-free RL has also led to significant progress in dexterous manipulation, starting with OpenAI's demonstration of finger-gaiting and finger-pivoting [1], trained in simulation and translated to real hardware. However, this approach uses extensive extrinsic sensing infeasible outside the lab, and relies on support surfaces such as the palm underneath the object. Khandate et al. [36] show dexterous finger-gaiting and finger-pivoting skills using only precision fingertip grasps to enable both palm-up and palm-down operation, but only on a range of simple convex shapes and in a simulated environment. Makoviychuk et al. [28] showed that GPU physics could be used to accelerate learning skills similar to OpenAI's. Allshire et al. [27] used extensive domain randomization and sim-to-real transfer to re-orient a cube but used table top as an external support surface. Chen et al. [2, 37] demonstrated in-

hand re-orientation for a wide range of objects under palm-up and palm-down orientations of the hand with extrinsic sensing providing dense object feedback. Sievers et al. [38] and Pitz et al. [39] demonstrated in-hand cube reorientation to desired pose with purely tactile feedback. Qi et al. [3] used rapid motor adaptation to achieve effective sim-to-real transfer of in-hand manipulation skills for small cylindrical and cube-like objects. In our case, the exploration ability of SBP allows learning of policies for more difficult tasks, such as in-hand manipulation of non-convex and large shapes, with only intrinsic sensing. We also achieve successful, robust sim-to-real transfer without extensive domain randomization or domain adaptation, by closing the sim-to-real gap via tactile feedback.

III. Method

In this paper, we focus on the problem of achieving dexterous in-hand manipulation while simultaneously securing the manipulated object in a precision grasp. Keeping the object stable in the grasp during manipulation is needed in cases where a support surface is not available, or the skill must be performed under different directions for gravity (i.e. palm up or palm down). However, it also creates a difficult class of manipulation problems, combining movement of both the fingers and the object with a constant requirement of maintaining stability. In particular, we focus on the task of achieving large in-hand object rotation, which we, as others before [3], believe to be representative of this general class of problems, since it requires extensive finger gaiting and object re-orientation.

A. Problem Description

Formally, our goal is to obtain a policy for issuing finger motor commands, rewarded by achieving large object rotation around a given hand-centric axis. The state of our system at time t is denoted by $\boldsymbol{x}_t = (\boldsymbol{q}_t, \boldsymbol{p}_t)$, where $\boldsymbol{q} \in \mathcal{R}^d$ is a vector containing the positions of the hand's d degrees of freedom (joints), and $\boldsymbol{p} \in \mathcal{R}^6$ contains the position and orientation of the object with respect to the hand. An action (or command) is denoted by the vector $\boldsymbol{a} \in \mathcal{R}^d$ comprising new setpoints for the position controllers running at every joint.

For parts of our approach, we assume that a model of the forward dynamics of our environment (*i.e.* a physics simulator) is available for planning or training. We denote this model by $\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t, \boldsymbol{a}_t)$. We will show however that our results transfer to real robots using standard sim-to-real methods.

We chose to focus on the case where the only sensing available is hand-centric, either tactile or proprioceptive. Achieving dexterity with only proprioceptive sensing, as biological organisms are clearly capable of, can lead to skills that are robust to occlusion and lighting and can operate in very constrained settings. With this directional goal in mind, the observation available to our policy consists of tactile and proprioceptive data collected by the hand, and no global object pose information. Formally, the observation vector is

$$\boldsymbol{o}_t = [\boldsymbol{q}_t, \boldsymbol{q}_t^s, \boldsymbol{c}_t] \tag{1}$$

where $\boldsymbol{q}_t, \boldsymbol{q}_t^s \in \mathcal{R}^d$ are the current positions and setpoints of the joints, and $\boldsymbol{c}_t \in [0,1]^m$ is the vector representing binary (contact / no-contact) touch feedback for each of m fingers of the hand.

As discussed above, we also require that the hand maintain a stable precision grasp of the manipulated object at all times. Overall, this means that our problem is characterized by a high-dimensional state space, but only small parts of this state space are accessible for us: those where the hand is holding the object in a stable precision grasp. Furthermore, the transition function of our problem is non-holonomic: the subset of fingers that are tasked with holding the object at a specific moment, as well as the object itself, must move in concerted fashion. Conceptually, the hand-object system must evolve on the complex union of high-dimensional manifolds that form our accessible states. Still, the problem state space must be effectively explored if we are to achieve dexterous manipulation with large object re-orientation and finger gaiting.

B. Manipulation RRT

To effectively explore our high-dimensional state space characterized by non-holonomic transitions, we turn to the well-known Rapidly-Exploring Random Trees (RRT) algorithm. We leverage our knowledge of the manipulation domain to induce tree growth along the desired manifolds in state space. In particular, we expect two conditions to be met for any state: (1) the hand must maintain at least three fingers in contact with the object[1], and (2) the distribution of these contacts must be such that a stable grasp is possible. We note that these are necessary, but not sufficient conditions for stability; nevertheless, we found them sufficient for effective exploration.

Preservation of condition (1) during the transition between two states means that the object and the fingers that maintain contact with it must move in unison. Assume that we would like the system to evolve from state $\boldsymbol{x}_{start} = (\boldsymbol{q}_{start}, \boldsymbol{p}_{start})$ towards state $\boldsymbol{x}_{end}(\boldsymbol{q}_{end}, \boldsymbol{p}_{end})$, with a desired change in state of $\Delta\boldsymbol{x}_{des} = (\Delta\boldsymbol{q}_{des}, \Delta\boldsymbol{p}_{des}) = \boldsymbol{x}_{end} - \boldsymbol{x}_{start}$. Further assume that the set S comprises the indices of the fingers that are expected to maintain contact throughout the motion. The requirement of maintaining contact, linearized around $\boldsymbol{x}_{start}$, can be expressed as:

$$\boldsymbol{J}_S(\boldsymbol{q}_{start})\Delta\boldsymbol{q}_{des} = \boldsymbol{G}_S(\boldsymbol{p}_{start})\Delta\boldsymbol{p}_{des} \tag{2}$$

where $\boldsymbol{J}_S(\boldsymbol{q}_{start})$ is the Jacobian of contacts on fingers in set S computed at $\boldsymbol{q}_{start}$, and $\boldsymbol{G}_S(\boldsymbol{p}_{start})$ is the grasp map matrix of contacts on fingers in set S computed at $\boldsymbol{p}_{start}$. This is further equivalent to

$$\boldsymbol{N}_S(\boldsymbol{x}_{start})\Delta\boldsymbol{x}_{des} = 0 \tag{3}$$

[1]Three contacts are the fewest that can achieve stable grasps without relying on torsional friction, which is highly sensitive to the material properties of the objects in contact. In the future, we plan to also include two-contact conditions, which can allow a richer set of manipulation primitives, at the expense of more complex contact modeling.

where $\boldsymbol{N}_S(\boldsymbol{x}_{start}) = [\boldsymbol{J}_S(\boldsymbol{q}_{start}) \quad -\boldsymbol{G}_S(\boldsymbol{p}_{start})]^T$.

It follows that, if the desired direction of motion in state space $\Delta\boldsymbol{x}_{des}$ violates this constraint, we can still find a similar movement that does not violate the constraint by projecting the desired vector into the null space of the matrix $\boldsymbol{N}$ as defined above:

$$\Delta\boldsymbol{x}_{proj} = (\boldsymbol{I} - \boldsymbol{N}^T\boldsymbol{N})\Delta\boldsymbol{x}_{des} \tag{4}$$

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{start} + \alpha\Delta\boldsymbol{x}_{proj} \tag{5}$$

where α is a constant determining the size of the step we are willing to take in the projected direction.

We note that this simple projection linearizes the contact constraint around the starting state. Even for small α, small errors due to this linearization can accumulate over multiple steps leading the fingers to lose contact. Thus, in practice, we further modify $\boldsymbol{x}_{new}$ by bringing back into contact with the object any finger that is within a given distance threshold (in practice, we set this threshold to 5 mm).

Maintaining at least three contacts with the object does not in itself guarantee a stable grasp. We take further steps to ensure that the contact distribution is appropriate for stability. Assume a set of k contacts, where each contact i has a normal direction $\boldsymbol{n}_i$ expressed in the global coordinate frame. We require that, if at least one contact j applies a non-zero normal contact force of magnitude c_j, the other contacts must be able to approximately balance it via normal forces of their own, minimizing the resulting net wrench applied to the object. This is equivalent to requiring that the hand have the ability to create internal object forces by applying normal forces at the existing contacts. We formulate this problem as a Quadratic Program:

unknowns: normal force magnitudes $c_i,\ i = 1 \dots k$

minimize $\|\boldsymbol{w}\|$ subject to:

$$\boldsymbol{w} = \boldsymbol{G}^T [c_1\boldsymbol{n_1} \dots c_k\boldsymbol{n_k}]^T \tag{6}$$

$$c_i \geq 0 \ \forall i \tag{7}$$

$$\exists j \text{ such that } c_j = 1 \text{ (ensure non-zero solution)} \tag{8}$$

If the resulting minimization objective is below a chosen stability threshold, we deem the grasp to be stable:

$$\text{If } \|\boldsymbol{w}\| < \epsilon_{stab}\text{: grasp is stable} \tag{9}$$

We note that this measure is conservative in that it does not rely on friction forces. Furthermore, it ensure that the fingers are able to generate internal object forces using contact normal forces, but does not specify what are appropriate motor torques for doing so. Nevertheless, we have found it effective in pushing exploration towards useful parts of the state space.

We can now put together these constraints into the complete algorithm shown in Alg. 1 and referred to in the rest of this paper as M-RRT. The essence of this algorithm is the forward propagation in lines 7-11. Given a desired direction of movement in state space, we want to ensure that at least three fingers maintain contact with the object. We thus project the direction of motion onto each of the manifolds defined by the contact constraints of each possible set of three fingers that begin the transition in contact with the object. We then choose the projected motion that brings us closest to the desired state-space sample. Finally, we perform an analytical stability check on the new state via eqs. (6-9).

Algorithm 1 Manipulation RRT (M-RRT)

Require: Tree contains root node; $N \leftarrow 1$

1: **while** $N < N_{max}$ **do**
2: $\quad\boldsymbol{x}_{sample} \leftarrow$ random point in state space
3: $\quad\boldsymbol{x}_{node} \leftarrow$ node closest to $\boldsymbol{x}_{sample}$ currently in tree
4: $\quad\Delta\boldsymbol{x}_{des} \leftarrow \|\boldsymbol{x}_{sample} - \boldsymbol{x}_{node}\|$
5: $\quad\mathcal{S} \leftarrow$ all sets of three fingers contacting the object in state $\boldsymbol{x}_{node}$
6: $\quad d_{min} \leftarrow \infty; x_{new} \leftarrow$ NULL
7: $\quad$**for all** S_i in $\mathcal{S}$ **do**
8: $\quad\quad$Compute $\boldsymbol{x}_i$ by projecting $\Delta\boldsymbol{x}_{des}$ on the constraint manifold of contacts in S_i as in eqs. (4-5)
9: $\quad\quad$**if** Stable(x_i) **and** $dist(x_{sample}, x_i) < d_{min}$ **then**
10: $\quad\quad\quad d_{min} \leftarrow dist(x_{sample}, x_i)$
11: $\quad\quad\quad x_{new} \leftarrow x_i$
12: $\quad$**if** x_{new} is not NULL **then**
13: $\quad\quad$Add x_{new} to tree with x_{node} as parent
14: $\quad\quad N \leftarrow N + 1$

We note that M-RRT does not make use of the environment's transition function $F()$ (i.e. system dynamics). In fact, both the projection method in eqs. (4-5) and the stability check via eqs. (6-9) can be considered as approximations of the transition function, aiming to preserve movement constraints but without explicitly computing and checking the system's dynamics. As such, they are fast to compute but approximate in nature. It is possible that some of the transitions in the resulting RRT tree are in fact invalid under full system dynamics, or require complex sequences of motor actions. As we will see in Sec. III-D however, they are sufficient for helping learn a closed-loop control policy. Furthermore, for cases where the $F()$ is available and fast to evaluate, we also study a variant of our approach that makes explicit use of it in the next section.

C. General-purpose non-holonomic RRT

For problems where system dynamics $F()$ are available and fast to evaluate, we also investigate the general non-holonomic version of the RRT algorithm, which is able to determine an action that moves the agent towards a desired sample in state space via random sampling. We use the same version of this algorithm as described for example in [40], which we recapitulate here in Alg. 2 and refer to as G-RRT.

The essence of this algorithm is the **while** loop in line 5: it is able to grow the tree in a desired direction by sampling a number K_{max} of random actions, then using the transition function $F()$ of our problem to evaluate which of these produces a new node that is as close as possible to a sampled target.

Our only addition to the general-purpose algorithm is the stability check in line 8: a new node gets added to the tree only

Algorithm 2 General-purpose non-holonomic RRT (G-RRT)

Require: Tree contains root node; $N \leftarrow 1$
while $N < N_{max}$ **do**
  $\boldsymbol{x}_{sample} \leftarrow$ random point in state space
  $\boldsymbol{x}_{node} \leftarrow$ node closest to $\boldsymbol{x}_{sample}$ currently in tree
  $d_{min} \leftarrow \infty; \boldsymbol{x}_{new} \leftarrow$ NULL
  while $k < K_{max}$ **do**
    $\boldsymbol{a} \leftarrow$ random action
    $\boldsymbol{x}_a \leftarrow F(\boldsymbol{x}_{node}, \boldsymbol{a})$
    if Stable($\boldsymbol{x}_a$) **and** $dist(\boldsymbol{x}_{sample}, \boldsymbol{x}_a) < d_{min}$ **then**
      $d_{min} \leftarrow dist(\boldsymbol{x}_{sample}, \boldsymbol{x}_a)$
      $\boldsymbol{x}_{new} \leftarrow \boldsymbol{x}_a$
    $k \leftarrow k + 1$
  if $\boldsymbol{x}_{new}$ is not NULL **then**
    Add $\boldsymbol{x}_{new}$ to tree with $\boldsymbol{x}_{node}$ as parent
    $N \leftarrow N + 1$

if it passes a stability check. This check consists of advancing the simulation for an additional 1s with no change in the action; if, at the end of this interval, the object has not been dropped (i.e. the height of the object is above a threshold) the new node is deemed stable and added to the tree. Assuming a typical simulation step of 2 ms, this implies 500 additional calls to $F()$ for each sample; however, it does away with the need for domain-specific analytical stability methods as we used for M-RRT.

Overall, the great advantage of this algorithm lies in its simplicity and generality. The only manipulation-specific component is the aforementioned stability check. However, its performance can be dependent on K_{max} (i.e. number of action samples at each iteration), and each of these samples requires a call to the transition function. This problem can be alleviated by the advent of highly efficient and massively parallel physics engines implementing the transition function, which is an important research direction complementary to our study.

D. Reinforcement Learning

While the RRT algorithms we have discussed so far have excellent abilities to explore the complex state space of in-hand manipulation, and to identify (approximate) transitions that follow the complex manifold structure of this space, they do not provide directly usable policies. In fact, M-RRT does not provide actions to use, and the transitions might not be feasible under the true transition function. G-RRT does find transitions that are valid, and also identifies the associated actions, but provides no mechanism to act in states that are not part of the tree, or to act under slightly different transition functions.

In order to generate closed-loop policies able to handle variability in the encountered states, we turn to RL algorithms. Critically, we rely on the trees generated by our sampling-based algorithms to ensure effective exploration of the state space during policy training. The specific mechanism we use to transition information from the sampling based tree to the policy training method is via the reset distribution: we select relevant paths from the planned tree, then use the nodes therein as reset states for policy training.

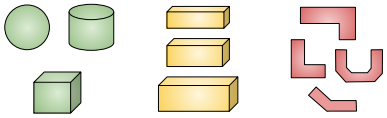

Fig. 2: The object shapes for which we learn finger-gaiting. From left to right: the easy, moderate and hard categories.

We note that the sampling-based trees as described here are task-agnostic. Their effectiveness lies in achieving good coverage of the state space (usually within pre-specified limits along each dimension). Once a specific task is prescribed (e.g. via a reward function), we must select the paths through the tree that are relevant for the task. For the concrete problem chosen in this paper (large in-hand object reorientation) we rely on the heuristic of selecting the top ten paths from the RRT tree that achieve the largest angular change for the object around the chosen rotation axis. (Other selection mechanisms are also possible; a promising and more general direction for future studies is to select tree branches that accumulate the highest reward.) After selecting the task-relevant set of states from the RRT tree, we use a uniform distribution over these states as a reset distribution for RL.

Our approach is compatible with RL methods that alternate between collecting episode rollouts and updating the policy, and restarting episode rollouts starting from a new set of states. Thus, both off-policy and on-policy RL are equally feasible. However, we use on-policy learning due to its its compatibility with GPU physics simulators and relative training stability.

IV. Experiments & Results

A. Experimental Setup

We use the robot hand shown in Fig. 1, consisting of five identical fingers. Each finger comprises a roll joint and two flexion joints, for a total of 15 fully actuated position-controlled joints. For the real hardware setup, each joint is powered by a Dynamixel XM430-210T servo motor. The distal link of each finger consists of an optics-based tactile fingertip as introduced by Piacenza et al. [41].

We test our methods on the object shapes illustrated in Fig. 2. We split this into categories: "easy" objects (sphere, cube cylinder), "moderate" objects (cuboids with elongated aspect ratios), and "hard" objects (either concave L- or U-shapes). We note that in-hand manipulation of the objects in the "hard" category has not been previously demonstrated in the literature.

1) Exploration Trees Setup: We run both G-RRT and M-RRT on the objects in our set. The first test consists, for both algorithms, in how effectively the tree explores its available state space given the number of iterations through the main loop (i.e. the number of attempted tree expansions towards a random sample). As a measure of tree growth, we look at the maximum object rotation achieved around our target axis. (We

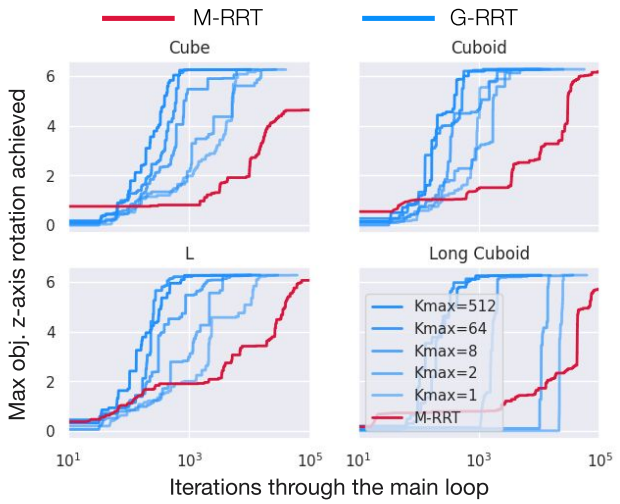

Fig. 3: Tree expansion performance for G-RRT and M-RRT. We plot the number of attempted tree expansions (i.e. iterations through the main loop, on a log scale) against the maximum object z-axis rotation achieved by any tree node so far. For G-RRT, we also plot performance for different values of K_{max}, the number of random actions tested at each iteration.

note that any rotation beyond approximately $\pi/4$ radians can not be done in-grasp, and requires finger repositioning.) Thus, for both algorithms, we compare maximum achieved object rotation vs. number of expansions attempted (on log scale). The results are shown in Fig. 3.

We found that both algorithms are able to effectively explore the state space. G-RRT is able to explore farther with fewer iterations, and its performance further increases with the number of actions tested at each iteration. We attribute this difference to the fact that M-RRT is constrained to taking small steps due to the linearization of constraints used in the extension projection. G-RRT, which uses the actual physics of the domain to expand the tree, is able to take larger steps at each iteration without the risk of violating the manipulation constraints.

As expected, the performance of G-RRT improves with the number K_{max} of actions tested at each iteration. Interestingly, the algorithm performs well even with $K_{max} = 1$; this is equivalent to a tree node growing in a completely random direction, without any bias towards the intended sample. However, we note that, at each iteration, the node that grows is the closest to the state-space sample taken at the beginning of the loop. This encourages growth at the periphery of the three and along the real constraint manifolds, and, as shown here, still leads to effective exploration.

Both these algorithms can be parallelized at the level of the main loop (line 1). However, the extensive sampling of possible actions, which is the main computational expense of G-RRT (line 5) also lends itself to parallelization. In practice, we use the IsaacGym [28] parallel simulator to parallelize this algorithm at both these levels (32 parallel evaluations of the main loop, and 512 parallel evaluations of the action sampling loop). This made both algorithms practical for testing in the context of RL. [2]

We then moved on to using paths from the planned trees in conjunction with RL training. Since our goal is finger gaiting for z-axis rotation, we planned additional trees with each method where object rotation around the x- and y-axes was restricted to 0.2 radians. Then, from each tree, we select 2×10^4 nodes from the paths which exhibit the most rotation around the z-axis, and extract their nodes. On average, each such path comprises 100-400 nodes. In the case of G-RRT, we recall that all tree nodes are subjected to an explicit stability check under full system dynamics before being added to the tree; we can thus use each of them as is. If using M-RRT, we also apply the same stability check to the nodes of the longest paths at this time, before using them as reset states for RL as described next.

2) Reinforcement Learning Setup: We train our policies using Asymmetric Actor Critic PPO [42, 43]; all training is done in the IsaacGym simulator. The critic uses object pose $\boldsymbol{p}$, object velocity $\dot{\boldsymbol{p}}$, and net contact force on each fingertip $\boldsymbol{t}_1 \ldots \boldsymbol{t}_m$ as feedback in addition to the feedback already as input to the policy network. Similar to Khandate et al. [36], we use a reward function that rewards object angular velocity about z-axis if the hand re-orienting the object with at least three fingertip contacts. In addition, we include penalties for the object's translational velocity and its deviation from the initial position [3]. We also use early termination to terminate the episode rollout if there are fewer than two contacts.

B. Experimental Conditions and Baselines

In our experiments, we compare the following approaches:

1) Ours, G-RRT: In this variant, we use the method presented in this paper, relying on exploratory reset states obtained by growing the tree via G-RRT. In all cases, we use a tree comprising 10^5 nodes as informed the ablation study in Fig 5.

2) Ours, M-RRT: This is also the method presented here, but using M-RRT for exploration trees. Again, we use trees comprising 10^5 nodes.

3) Stable Grasp Sampler (SGS): This baseline represents an alternative to the method presented in this paper: we use a reset distribution consisting of stable grasps generated by sampling random joint angles and varying object orientation about the rotation axis. This approach has been demonstrated precision in-hand manipulation with only intrinsic sensing [3, 36] for simple shapes.

4) Explored Restarts (ER): This method selects states explored by the policy itself during random exploration to use as reset states [11]. It is highly general, with no manipulation-specific component, and requires no additional step on top of RL training. We implement the "uniform restart" scheme as it was shown to have superior performance on high dimensional continuous control tasks. However, we have found it to be

[2]Given the advent of increasingly more powerful parallel architectures for general physics simulation, we expect that more general methods that are easier to parallelize might win out in the long term over more problem-specific solutions that are more sample efficient at the individual thread level.

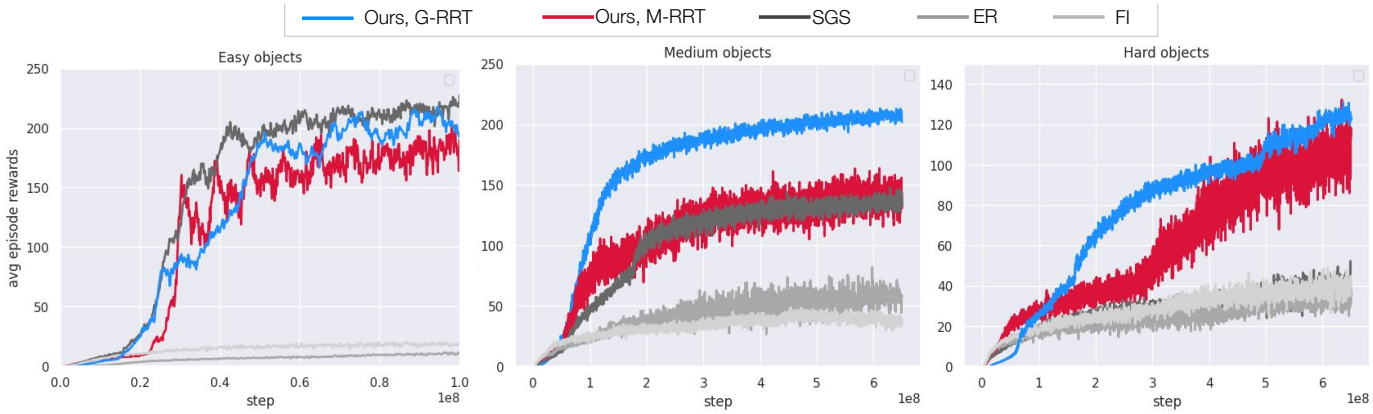

Fig. 4: Training performance for our methods (G-RRT and M-RRT) and a number of baselines on the object categories shown in Fig. 2.

insufficient for the complex state space of our problem: it fails to learn a viable policy even for simple objects.

5) Fixed Initialization (FI): For completeness, we also tried restarting training from a single fixed state. As expected, this method also failed to learn even in the simple cases. Additionally, we evaluated fixed initialization with gravity curriculum (zero to full). The policy only learned in-grasp manipulation, reorienting the object by the maximum possible amount without breaking contact before dropping. We found that the policy did not learn finger-gaiting even with zero gravity when using a fixed initialization. Thus, fixed initialization with or without gravity curriculum learning does not help with learning finger-gaiting. We hypothesize that curriculum learning has limited power to address exploration issues because policies tend to converge to sub-optimal behaviors that are hard to overcome later in training.

C. Results

Training results are summarized Fig. 4. The performance on easy objects confirms the results of previous studies, which showed that a reset distribution consisting of random grasps (SGS) enables learning of rotation gaits; sampling-based exploration (our methods) achieves similar performance. For medium objects, G-RRT, M-RRT, and SGS again all learn to gait, but the policies learned via G-RRT exploration are more effective. Finally, for complex problems (hard objects), a random grasp-based reset distribution is no longer workable. Only G-RRT and M-RRT are able to learn manipulation, and G-RRT does so more efficiently. We also note that none of the domain-agnostic methods (ER and FI) are able to learn in-hand manipulation on any object set, in the allotted training time.

We also studied the impact of size of the tree used in extracting reset states. Fig 5 summarizes our results for learning a policy for an L-shaped object using tree of different sizes grown via G-RRT. Qualitatively, we observe that, as the tree grows larger, the top 100-400 paths sampled from the tree contain increasingly more effective gaits, likely closer to the optimal policy. This suggests a strong correlation between the optimality of states used for reset distribution and sample efficiency of learning.

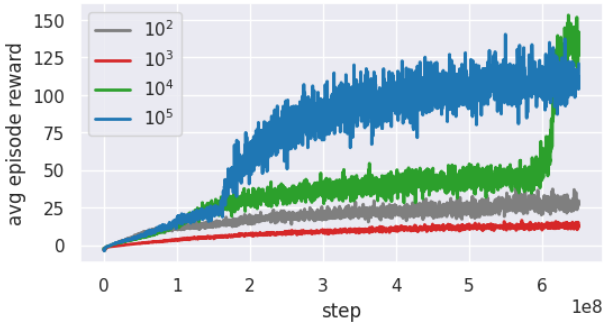

Fig. 5: Training performance with different tree sizes. The training curves shown are for different set of reset states obtained from trees of varying sizes. We see that we need a sufficiently large tree with at least 10^4 nodes to enable learning. However, training is most reliable with 10^5 nodes.

In addition, we performed an ablation study of policy feedback. Particularly, we aimed to compare intrinsically available tactile feedback vs. object pose feedback that would require external sensing. Fig. 6 summarizes these results for an L-shaped object. First, we found that touch feedback is essential for all moderate and hard objects in the absence of object pose feedback. For these objects, we also saw that replacing this tactile feedback with object pose feedback results in slower learning, underscoring the importance of touch feedback for in-hand manipulation skills. Richer tactile feedback such as contact position, normals, and force magnitude can be expected to provide even stronger improvements; we hope to explore this in future work.

D. Evaluation on real hand

To test the applicability of our method on real hardware, we attempted to transfer the learned policy for a subset of representative objects: cylinder, cube, cuboid & L-shape. We chose these objects to span the range from simpler to more difficult manipulation skills.

For sim-to-real transfer, we take a number of additional steps. We impose velocity and torque limits in the simulation, mirroring those used on the real motor (0.6 rad/s and 0.5 N-

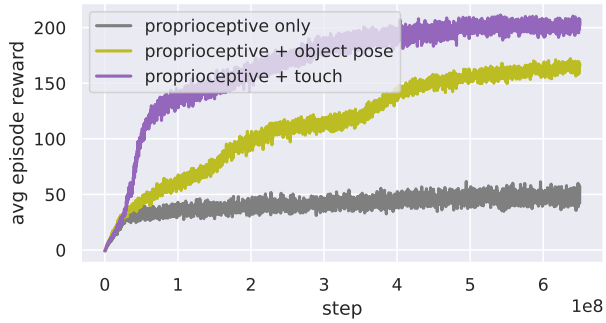

Fig. 6: Ablation of policy feedback components for L-shaped object. We note that touch feedback is essential in the absence of object pose feedback, and also leads to faster learning in comparison with object pose feedback.

m respectively). We found that our hardware has significant latency of 0.05s which we included in the simulation. In addition, we modified the angular velocity reward to maintain a desired velocity instead of maximizing the object's angular velocity. We also randomize joint origins ($0.1rad$), friction coefficient ($1 - 40$), and train with perturbation forces ($1N$). All these changes are introduced successively via a curriculum.

For sensing, we used the current position and setpoint from the motor controllers with no additional changes. For tactile data, we found that information from our tactile fingers is most reliable for contact forces above 1 N. We thus did not use reported contact data below this threshold, and imposed a similar cutoff in simulation. Overall, we believe that a key advantage of exclusively using proprioceptive data is a smaller sim-to-real gap compared to extrinsic sensors such as cameras.

For the set of representative objects, we ran the respective policy ten consecutive times, and counted the number of successful complete object revolutions achieved before a drop. In other words, five revolutions means the policy successfully rotated the object for $1,800°$ before dropping it. In addition, we also report the average object rotation speed observed during the trials. The results of these trials are summarized in Table I. Fig. 7 shows the keyframes of the finger-gaiting we achieved by the policy on the hand and also compared it with manipulation observed in simulation.

V. Discussion and Conclusions

The results we have presented show that sampling-based exploration methods make it possible to achieve difficult manipulation tasks via RL. In fact, these popular and widely used classes of algorithms are highly complementary in this case. RL is effective at learning closed-loop control policies that maintain the local stability needed for manipulation, and, thanks to training on large number of examples, are robust to variations in the encountered states. However, the standard RL exploration techniques (random perturbations in action space) are ineffective in the highly constrained state space with complex manifold structure of manipulation tasks. Conversely, SBP methods, which rely on a fundamentally different approach to exploration, can effectively discover relevant regions

TABLE I: Manipulation performance in simulation vs. real hardware. We report median number of object rotations achieved before dropping the object in ten consecutive trials, as well as the time needed to perform these rotations.

	Median revolutions	Mean rotation speed (rad/s)
Cylinder	5	0.42
Cube (s)	4.5	0.44
Cuboid	1.5	0.44
L-shape	1.5	0.24

of the state space, and convey this information to RL training algorithms, for example via an informed reset distribution.

Since sampling-based exploration methods are not expected to generate directly usable trajectories, exploration can also use approximate models of physical constraints, which can be informed by well-established analytical models of robotic manipulators. Interestingly, we found that using the general-purpose exploration algorithm using the full transition function of the environment is still more sample-efficient than using such analytical constraint models. Nevertheless, both are usable in practice, particularly with the advent of massively parallel physics simulators.

We use this approach to demonstrate finger gaiting precision manipulation of both convex and non-convex objects, using only tactile and proprioceptive sensing. Using only these types of intrinsic sensors makes manipulation skills insensitive to occlusion, illumination or distractors, and reduces the sim-to-real gap. We take advantage of this by demonstrating our approach both in simulation and on real hardware. We note that, while some applications naturally preclude the use of vision (e.g. extracting an object from a bag), we expect that in many real-life situations future robotic manipulators will achieve the best performance by combining touch, proprioception and vision.

Learning in-hand object reorientation to achieve a given desired pose may also potentially benefit from our approach, for example by leveraging information from the RRT tree to find appropriate trajectories for reaching a specific node. Another more general and promising direction for future work involves other mechanisms by which ideas from sampling-based exploration can facilitate RL, beyond reset distributions. Some SBP algorithms can also be used to suggest possible actions for transitions between regions of the state space, a feature that we do not take advantage of here (even though one of the exploration algorithms we use does indeed compute actions). Alternatively, sampling-based exploration techniques could be integrated directly into the policy training mechanisms, removing the need for two separate stages during training. We hope to explore all these ideas in future work.

Acknowledgements. This work was supported in part by the Office of Naval Research grant N00014-21-1-4010 and the National Science Foundation grant CMMI-2037101.

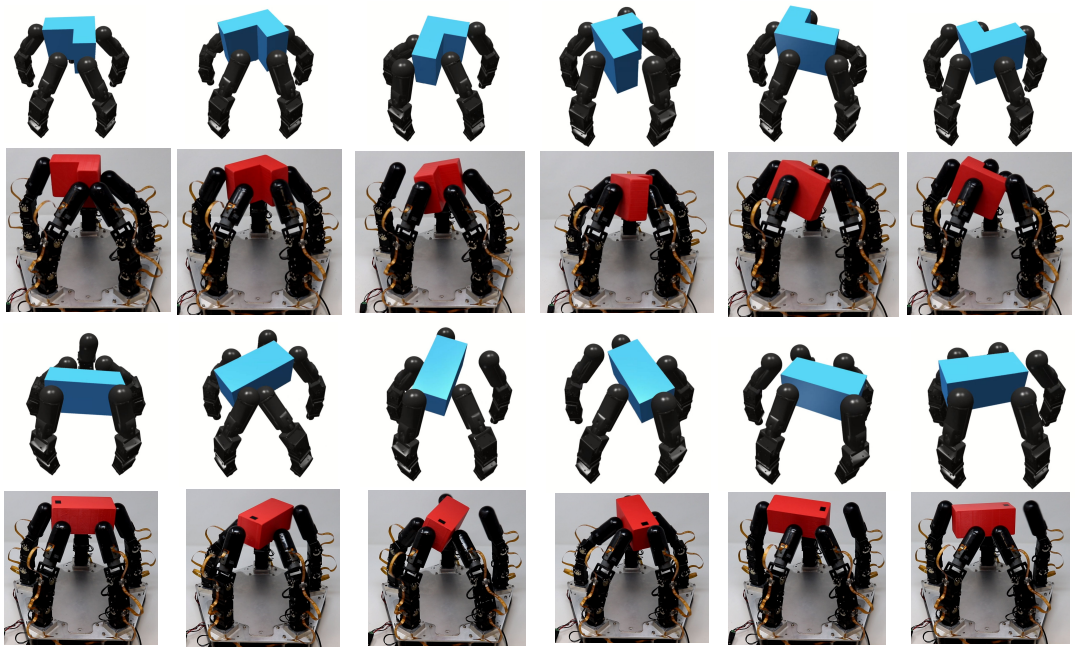

Fig. 7: Key frames of the finger-gaiting in simulation and on the real hand for representative objects in simulation and on real hand. Representative videos of these tasks can be found on our project website sbrl.cs.columbia.edu

References

[1] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. "Solving Rubik's Cube with a Robot Hand". In: (Oct. 2019). arXiv: 1910.07113 [cs.LG].

[2] Tao Chen, Jie Xu, and Pulkit Agrawal. "A System for General In-Hand Object Re-Orientation". Nov. 2021.

[3] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. "In-Hand Object Rotation via Rapid Motor Adaptation". In: (Oct. 2022). arXiv: 2210.04887 [cs.RO].

[4] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. "Curiosity-driven Exploration by Self-supervised Prediction". In: (May 2017). arXiv: 1705.05363 [cs.LG].

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: (Jan. 2018). arXiv: 1801.01290 [cs.LG].

[6] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. "A Survey of Exploration Methods in Reinforcement Learning". In: *arXiv:2109.00157 [cs]* (Sept. 2021).

[7] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. "Parameter Space Noise for Exploration". In: (June 2017).

[8] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: (Sept. 2017). arXiv: 1709.10089 [cs.LG].

[9] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. "Go-Explore: a New Approach for Hard-Exploration Problems". In: (Jan. 2019).

[10] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. "First return, then explore". In: *Nature* 590.7847 (Feb. 2021), pp. 580–586.

[11] Arash Tavakoli, Vitaly Levdik, Riashat Islam, Christopher M Smith, and Petar Kormushev. "Exploring Restart Distributions". In: (Nov. 2018).

[12] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. "Benchmarking Deep Reinforcement Learning for Continuous Control". In: (Apr. 2016). arXiv: 1604.06778 [cs.LG].

[13] *Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes*. PMLR, 2020.

[14] S LaValle. "Rapidly-exploring random trees : a new tool for path planning". en. In: *The annual research report* (1998).

[15] Sertac Karaman and Emilio Frazzoli. "Optimal kinodynamic motion planning using incremental sampling-based methods". In: *49th IEEE Conference on Decision and Control (CDC)*. Dec. 2010, pp. 7681–7687.

[16] Dustin J Webb and Jur van den Berg. "Kinodynamic RRT*: Asymptotically optimal motion planning for

robots with linear dynamics". In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 5054–5061.

[17] L E Kavraki, P Svestka, J-C Latombe, and M H Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Trans. Rob. Autom.* 12.4 (Aug. 1996), pp. 566–580.

[18] L E Kavraki, M N Kolountzakis, and J-C Latombe. "Analysis of probabilistic roadmaps for path planning". In: *IEEE Trans. Rob. Autom.* 14.1 (Feb. 1998), pp. 166–171.

[19] Linjun Li, Yinglong Miao, Ahmed H Qureshi, and Michael C Yip. "MPC-MPNet: Model-Predictive Motion Planning Networks for Fast, Near-Optimal Planning Under Kinodynamic Constraints". In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 4496–4503.

[20] Hao-Tien Lewis Chiang, Jasmine Hsu, Marek Fiser, Lydia Tapia, and Aleksandra Faust. "RL-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators From RL Policies". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 4298–4305.

[21] Anthony Francis, Aleksandra Faust, Hao-Tien Lewis Chiang, Jasmine Hsu, J Chase Kew, Marek Fiser, and Tsang-Wei Edward Lee. "Long-Range Indoor Navigation With PRM-RL". In: *IEEE Trans. Rob.* 36.4 (Aug. 2020), pp. 1115–1134.

[22] Liam Schramm and Abdeslam Boularias. "Learning-guided exploration for efficient sampling-based motion planning in high dimensions". In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022.

[23] Lerrel Pinto, Aditya Mandalika, Brian Hou, and Siddhartha Srinivasa. "Sample-Efficient Learning of Nonprehensile Manipulation Policies via Physics-Based Informed State Distributions". In: (Oct. 2018). arXiv: 1810.10654 [cs.RO].

[24] Philippe Morere, Gilad Francis, Tom Blau, and Fabio Ramos. "Reinforcement Learning with Probabilistically Complete Exploration". In: (Jan. 2020). arXiv: 2001.06940 [cs.LG].

[25] Tom Jurgenson and Aviv Tamar. "Harnessing Reinforcement Learning for Neural Motion Planning". In: (June 2019). arXiv: 1906.00214 [cs.RO].

[26] Huy Ha, Jingxi Xu, and Shuran Song. "Learning a Decentralized Multi-arm Motion Planner". In: (Nov. 2020). arXiv: 2011.02608 [cs.RO].

[27] Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. "Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger". In: (Aug. 2021). arXiv: 2108.09779 [cs.RO].

[28] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning". In: (Aug. 2021). arXiv: 2108.10470 [cs.RO].

[29] Susanna Leveroni and Kenneth Salisbury. "Reorienting Objects with a Robot Hand Using Grasp Gaits". In: *Robotics Research*. Springer London, 1996, pp. 39–51.

[30] L Han and J C Trinkle. "Dextrous manipulation by rolling and finger gaiting". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 1. May 1998, 730–735 vol.1.

[31] M Yashima, Y Shiina, and H Yamaguchi. "Randomized manipulation planning for a multi-fingered hand by switching contact modes". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. Sept. 2003, 2689–2694 vol.2.

[32] Jijie Xu, T John Koo, and Zexiang Li. "Finger gaits planning for multifingered manipulation". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2007, pp. 2932–2937.

[33] Andrew S Morgan, Daljeet Nandha, Georgia Chalvatzaki, Carlo D'Eramo, Aaron M Dollar, and Jan Peters. "Model Predictive Actor-Critic: Accelerating Robot Skill Acquisition with Deep Reinforcement Learning". In: (Mar. 2021). arXiv: 2103.13842 [cs.RO].

[34] Andrew S Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M Dollar. "Complex in-hand manipulation via compliance-enabled finger gaiting and multimodal planning". In: *IEEE Robot. Autom. Lett.* 7.2 (Apr. 2022), pp. 4821–4828.

[35] Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. "Surprisingly Robust In-Hand Manipulation: An Empirical Study". In: (Jan. 2022). arXiv: 2201.11503 [cs.RO].

[36] Gagan Khandate, Maximilian Haas-Heger, and Matei Ciocarlie. "On the Feasibility of Learning Finger-gaiting In-hand Manipulation with Intrinsic Sensing". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 2752–2758.

[37] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. "Visual Dexterity: In-hand Dexterous Manipulation from Depth". In: (Nov. 2022). arXiv: 2211.11744 [cs.RO].

[38] Leon Sievers, Johannes Pitz, and Berthold Bäuml. "Learning Purely Tactile In-Hand Manipulation with a Torque-Controlled Hand". In: *Proc. IEEE International Conference on Robotics and Automation*. 2022.

[39] Johannes Pitz, Lennart Röstel, Leon Sievers, and Berthold Bäuml. "Dextrous Tactile In-Hand Manipulation Using a Modular Reinforcement Learning Architecture". In: *Proc. IEEE International Conference on Robotics and Automation*. 2023.

[40] Jennifer E King, Marco Cognetti, and Siddhartha S Srinivasa. "Rearrangement planning using object-centric and robot-centric action spaces". In: *2016 IEEE*

International Conference on Robotics and Automation (ICRA). IEEE. 2016, pp. 3940–3947.

[41] Pedro Piacenza, Keith Behrman, Benedikt Schifferer, Ioannis Kymissis, and Matei Ciocarlie. "A Sensorized Multicurved Robot Finger With Data-Driven Touch Sensing via Overlapping Light Signals". In: *IEEE/ASME Trans. Mechatron.* 25.5 (Oct. 2020), pp. 2416–2427.

[42] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. "Asymmetric Actor Critic for Image-Based Robot Learning". In: (Oct. 2017). arXiv: 1710.06542 [cs.RO].

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal Policy Optimization Algorithms". In: (July 2017). arXiv: 1707.06347 [cs.LG].