

# Diffusion-Informed Probabilistic Contact Search for Multi-Finger Manipulation

Abhinav Kumar<sup>\*1</sup>, Thomas Power<sup>\*1</sup>, Fan Yang<sup>1</sup>, Sergio Aguilera Marinovic<sup>2</sup>,  
Soshi Iba<sup>2</sup>, Rana Soltani Zarrin<sup>2</sup>, Dmitry Berenson<sup>1</sup>

**Abstract**—Planning contact-rich interactions for multi-finger manipulation is challenging due to the high-dimensionality and hybrid nature of dynamics. Recent advances in data-driven methods have shown promise, but are sensitive to the quality of training data. Combining learning with classical methods like trajectory optimization and search adds additional structure to the problem and domain knowledge in the form of constraints, which can lead to outperforming the data on which models are trained. We present Diffusion-Informed Probabilistic Contact Search (DIPS), which uses an A\* search to plan a sequence of contact modes informed by a diffusion model. We train the diffusion model on a dataset of demonstrations consisting of contact modes and trajectories generated by a trajectory optimizer given those modes. In addition, we use a particle filter-inspired method to reason about variability in diffusion sampling arising from model error, estimating likelihoods of trajectories using a learned discriminator. We show that our method outperforms ablations that do not reason about variability and can plan contact sequences that outperform those found in training data across multiple tasks. We evaluate on simulated tabletop card sliding and screwdriver turning tasks, as well as the screwdriver task in hardware to show that our combined learning and planning approach transfers to the real world.

## I. INTRODUCTION

Multi-finger manipulation is challenging as there are many ways in which the hand can make or break contact with manipulated objects. The discrete modes in which the hand makes contact induce different dynamics by imposing different constraints on the system. Thus the system is hybrid and there is no clear way to sequence contact modes in order to accomplish a given task efficiently.

Recent advances in learning methods, specifically generative modeling [1], [2], can be used to learn manipulation policies without requiring strong domain knowledge of the task in the form of constraint modeling. However, in multi-finger manipulation, which has sensitive constraints relating to contact, that domain knowledge is valuable to improve task performance. In addition, learning methods generate policies similar to the demonstration data on which they are trained. In tasks with varied possible contact interactions, a planned contact mode sequence could yield better results than that used in data collection.

In contrast to learning-based methods, planning methods that use techniques such as trajectory optimization offer a way to reason about constraints. While these methods can be used to plan a trajectory given a specific contact

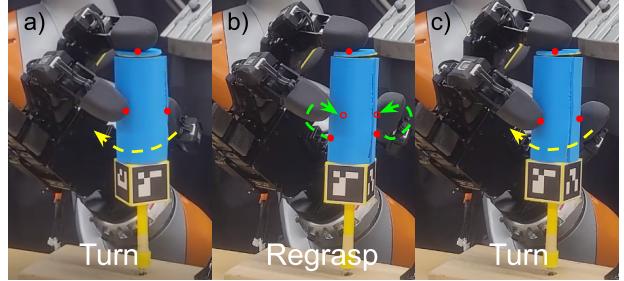


Fig. 1. DIPS plans contact interactions that turn the screwdriver from a) to b), where it is regrasped to allow for further turning in c). Contact points are shown in red, with empty circles for target contacts. The yellow arrows show screwdriver turning and green show finger motion.

mode [3] or can plan contact mode changes for lower-dimensional systems [4], [5], planning contact sequences in high-dimensional multi-finger manipulation problems can become intractably expensive to compute.

We approach multi-finger manipulation by combining the flexibility of learning with the constraint satisfaction of model-based methods. In our method, Diffusion-Informed Probabilistic Contact Search (DIPS), we train a diffusion model [6] on trajectories generated by a trajectory optimizer. We then use this model to generate trajectories corresponding to edges in an A\* search over contact modes.

We also propose a method to reason about the variability in the diffusion sampling, which can cause issues in contact sequence planning. This variability could come from low quality samples arising from the innate error of learned models or the quality of the training data. Reasoning about variability in the diffusion output can be helpful when planning sequences of multiple contact modes in which multiple diffusion samples would need to be concatenated over time.

To reason about the variability in diffusion sampling when planning contact sequences, we use a particle-based approximation of the distribution over trajectories modeled by the diffusion. We train an additional discriminator that assigns higher scores to more realistic trajectories to provide likelihood estimates of particles. Using the diffusion sampling and likelihood estimation, we construct a particle-based method to propagate variability estimates as edges are sequenced in the search. Our contributions are:

- A method for planning contact mode sequences using graph search informed by a diffusion model
- A method for explicitly reasoning about the variability in the diffusion model sampling when planning

We show that our method can be applied to challenging tool-use tasks and that DIPS outperforms ablations and

<sup>\*</sup> Equal contribution. <sup>1</sup>Robotics Department, University of Michigan, Ann Arbor, MI, USA [abhin, tpower, fanyangr, dmitryb]@umich.edu, <sup>2</sup> Honda Research Institute USA. This work was sponsored by Honda Research Institute USA.

baselines that do not plan contact sequences and that do not reason about the variability in the diffusion model output.

## II. RELATED WORK

*a) Planning for Contact-Rich Manipulation:* Due to the difficulty of planning through contact, recent work has optimized with a pre-defined contact mode [3], contact schedule [7], or sequence of constraints [8]. Other work has explicitly planned contact sequences in sample-based motion planning frameworks [9], [10]. Smooth contact models have also been used to simplify the planning problem. These have been used for trajectory optimization [11], [12] as well as sample-based planners [12].

*b) Combining Search & Trajectory Optimization:* Several recent works have combined search and trajectory optimization. Cheng et al. propose a hierarchical approach based on Monte-Carlo Tree Search to explore contact modes [13]. Other recent work has used tree-based planners to explore contact modes combined with contact-implicit trajectory optimization [14], [15]. There have been several recent works combining A\* search with trajectory optimization [4], [16]–[18]. Unlike previous work combining A\* and trajectory optimization, we accelerate our planning using a learned model to approximate trajectory optimization.

*c) Learning for Contact-Rich Manipulation:* In recent years learning-based methods have been increasingly popular for solving contact-rich tasks. These include methods that learn models for planning [19], [20], as well as methods that learn policies using reinforcement learning [21]–[24]. In general, these methods can solve complex tasks when given sufficient training data but often require large amounts of training data. Imitation learning has also been applied to dexterous manipulation, accelerating learning with demonstrations [25], [26]. These methods are effective but rely on collecting high-quality demonstrations. Efforts to combine learning and classical approaches have used reinforcement learning to learn mid-level policies to sequence primitives or controllers [27]–[30]. Our method similarly aims to combine learning with planning, but can automatically generate high-quality demonstrations from trajectory optimization.

## III. PROBLEM STATEMENT

In this paper, we consider the problem of contact-rich manipulation with a multi-fingered hand. Our goal is to find a sequence of robot configurations  $\mathbf{q}_{1:T}$ , where  $T$  is the number of execution timesteps, that successfully manipulate an object from start pose  $\mathbf{o}_0$  to goal pose  $\mathbf{o}_G$ . This task is challenging as successfully manipulating the object may require the making and breaking of contact, resulting in non-smooth dynamics. We define contact mode  $\mathbf{c} := \{0, 1\}^{n_f}$ , where  $n_f$  is the number of fingers.  $\mathbf{c}$  is a binary vector specifying which finger should be in contact. Given a contact mode, we formulate the following trajectory optimization problem for horizon  $H < T$ .

$$\begin{aligned} \mathbf{s}_{1:H}^*, \mathbf{u}_{1:H}^* = \arg \min J(\mathbf{s}_{1:H}, \mathbf{u}_{1:H}, \mathbf{c}) \\ \text{s.t. } h(\mathbf{s}_{1:H}, \mathbf{u}_{1:H}, \mathbf{c}) = 0 \\ g(\mathbf{s}_{1:H}, \mathbf{u}_{1:H}, \mathbf{c}) \leq 0, \end{aligned} \quad (1)$$

with state  $\mathbf{s}_t$ , action  $\mathbf{u}_t$  and cost and constraint functions  $J, h, g$ , which are all dependent on the contact mode. For shorthand, we denote the trajectory  $\tau := \{\mathbf{s}_{1:H}, \mathbf{u}_{1:H}\}$ . Once the contact mode is provided, this trajectory optimization problem can be efficiently solved. To make this problem tractable, we assume (1) access to the geometries of the hand and the object, (2) that the system is quasi-static, and (3) that the fingers begin the task in contact with the object.

To allow for the making and breaking of contact, we split the trajectory into  $K$  segments, i.e.  $H = \frac{T}{K}$ , and  $\tau = \{\tau_1, \dots, \tau_K\}$ . The aim is then to find a sequence of contact modes  $C = \mathbf{c}_{1:K}$ , with  $\tau_k$  being the solution to Problem (1) with contact mode  $\mathbf{c}_k$  for  $k \in [1, \dots, K]$ , such that the overall trajectory  $\tau$  results in completing the task.

To aid in solving this problem we assume we can generate a dataset of  $N$  trajectories,  $\mathcal{D} = \{\tau_i, \mathbf{c}_i\}_{i=1}^N$  which consists of solutions to Problem (1) for diverse initial configurations.

We evaluate our method on its ability to achieve the goal  $\mathbf{o}_G$ . For some tasks, we also consider the validity of the executed trajectories. For example, if the hand is manipulating a screwdriver, dropping the screwdriver would be the consequence of an invalid trajectory.

## IV. METHODS

In this section, we outline our method for in-hand contact-rich manipulation that combines search and trajectory optimization to search for a sequence of contact modes  $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$  and optimize trajectories given these modes.

Our method, shown in Fig. 2, uses A\* to search for the contact sequence. During planning, when expanding a potential contact mode  $\mathbf{c}_k$ , we must reason about the resulting trajectory  $\tau_k$ . Since solving the trajectory optimization problem in the inner loop of our planner would be prohibitively expensive, we instead train a diffusion model on a dataset of trajectories and sample from this as a proxy for solving the full optimization. Finally, we optimize and execute trajectories given the planned contact sequences using the trajectory optimizer. We also use the samples from the diffusion model to initialize the trajectory optimization.

### A. Trajectory Optimization

Our trajectory optimization formulation is based on prior work by Yang et al. [3]. The formulation in [3] assumes a fixed contact mode and only optimizes the motion for fingers in contact. We extend this formulation to be conditioned on the contact mode and additionally optimize the motion of specified fingers so they can “regrasp” i.e., make contact in a different location.

The state  $\mathbf{s}$  consists of finger configurations  $\{\mathbf{q}_i\}_{i=1}^{n_f}$  and object pose  $\mathbf{o}$ . The control vector  $\mathbf{u}$  is  $\{\{\Delta \mathbf{q}_i, \mathbf{f}_i\}_{i=1}^{n_f}, \mathbf{f}_e\}$ , where  $\mathbf{f}_i$  is the contact force for the  $i$ th finger, and  $\mathbf{f}_e$ , the contact force exerted by the environment. All contact forces are defined in the object frame. Given a contact mode  $\mathbf{c} \in \{0, 1\}^{n_f}$  we partition the state and control vectors into contact fingers  $\{\mathbf{s}_c, \mathbf{u}_c\} = \{\mathbf{q}_i, \Delta \mathbf{q}_i, \mathbf{f}_i : \mathbf{c}_i = 1\}$ , regrasping fingers  $\{\mathbf{s}_r, \mathbf{u}_r\} = \{\mathbf{q}_i, \Delta \mathbf{q}_i : \mathbf{c}_i = 0\}$ , and the object and environment  $\{\mathbf{s}_o, \mathbf{u}_o\} = \{\mathbf{o}, \mathbf{f}_e\}$ . There is no contact force for regrasping fingers, as they break contact. We similarly

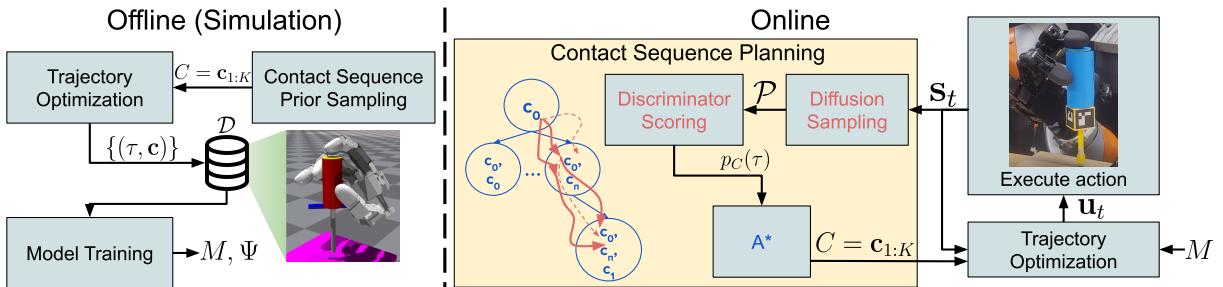


Fig. 2. **Offline**, we sample contact sequences from a designed prior. We generate a dataset  $\mathcal{D}$  of trajectories in simulation. We train a diffusion model  $M$  and discriminator  $\Psi$  on  $\mathcal{D}$ . **Online**, we plan a contact sequence  $C$  given a state  $s_t$ . We expand nodes in blue corresponding to contact mode sequences and inform the search using a distribution  $p_C(\tau)$  parameterized with a set of trajectories  $P$  in pink. We diffuse trajectories conditioned on the child node's contact mode and evaluate them with  $\Psi$ . The dotted lines are samples discarded in the resampling used to update  $p_C(\tau)$ . Given a single contact mode and  $s_t$ , we optimize a trajectory of length  $H$  initialized with samples from  $M$ . We rerun the trajectory optimization every timestep. After each contact mode, we replan  $C$ .

partition the trajectory into  $\tau = \{\tau_c, \tau_r, \tau_o\}$ . Our trajectory optimization problem is then written as

$$\begin{aligned} & \min_{\substack{s_1, s_2, \dots, s_H; \\ u_1, u_2, \dots, u_H}} J_g(\tau_o) + J_r(\tau_r, \tau_o) + J_{smooth}(\tau) \\ \text{s.t. } & q_{min} \leq q_t \leq q_{max} \\ & u_{min} \leq u_t \leq u_{max} \\ & f_{contact}(s_{c,t}, s_{o,t}) = 0 \\ & f_{kinematics}(s_{c,t}, s_{o,t}, s_{c,t+1}, s_{o,t+1}) = 0 \\ & f_{balance}(s_{c,t}, s_{o,t}, s_{c,t+1}, s_{o,t+1} u_{c,t}, u_{o,t}) = 0 \\ & f_{friction}(s_{c,t}, s_{o,t}, u_{c,t}) \leq 0 \\ & f_{contact}(s_{r,t}, s_{o,t}) \leq -\delta, t < H \\ & f_{contact}(s_{r,H}, s_{o,H}) = 0 \\ & q_{r,t} + \Delta q_{r,t} - q_{r,t+1} = 0. \end{aligned} \quad (2)$$

The cost term  $J_g$  encourages the object to reach the goal location,  $J_{smooth}$  incentivizes a smooth trajectory, and  $J_r$  is a cost on the distance to target contact points for the regrasping fingers. The constraints  $f_{kinematics}$ ,  $f_{contact}$ ,  $f_{balance}$ ,  $f_{friction}$  are unchanged from [3].  $f_{contact} \leq -\delta$  ensures that the regrasping fingers avoid contact with a threshold  $\delta$  up until the final time step. The final constraint ensures that configurations and actions are consistent for the regrasping fingers that move in freespace.

The target contact points for the regrasping can be defined based on the task. For example, if turning a screwdriver, we can set the targets to be the initial contact points of the fingers on the screwdriver to be able to reset fingers after turning. Other tasks may benefit from other specifications.

To solve the trajectory optimization we use Constrained Stein Variational Trajectory Optimization (CSVTO) [31]. This optimization formulation allows us to generate trajectories given a pre-specified contact mode. We will next discuss how we use this to generate high-quality demonstrations used to train a diffusion model for a variety of contact modes.

### B. Diffusion Model Training

To aid in contact sequence planning, we train a diffusion model  $M(c, s_0)$ . We can use  $M$  to sample from the distribution  $p(\tau|c, s_0)$ .  $p(\tau|c, s_0)$  is the distribution modeling trajectories computed by the trajectory optimizer given a contact mode and initial state. We train this model on a dataset  $\mathcal{D}$  of trajectories and corresponding contact modes.

When generating data, we use a high optimization budget for CSVTO to optimize high-quality trajectories. We can then

optionally use diffusion samples when executing the task to initialize CSVTO, potentially requiring a lower optimization budget at runtime due to the higher-quality initialization.

To obtain the contact sequences used to generate  $\mathcal{D}$ , we sample from a constructed prior  $p(C)$ .  $p(C)$  is designed to accomplish a specific task, for example, turning a screwdriver. While  $p(C)$  is useful for generating data and should represent a reasonable attempt to solve the task, we find that our method can plan contact sequences that outperform  $p(C)$ .

We adopt the 1-D U-Net architecture used in [1], [32] for the diffusion model. We diffuse a trajectory of dimension  $H \times (d_s + d_u)$ , where  $d_s$  is the dimensionality of the state, and  $d_u$  is the dimensionality of the action. We use classifier-free guidance [33] to condition on a specific contact mode  $c$ . To condition on  $s_0$ , we use the same inpainting approach as [1]. At training time, we randomly sample masks over the trajectory, emulating inpainting masks, which we find improves the inpainting performance when sampling.

As we are working with complex systems with high degrees of freedom and complex constraints,  $M$  may diffuse unrealistic trajectories. As we will discuss in Section IV-C, we compute weights of trajectories in our variability propagation method that represent their realism to account for this. One way to compute these weights could be using the likelihood of the trajectory under the diffusion model, as shown in [34]. However, we find using the likelihood to be intractable due to its high cost of approximation.

Instead, we train a discriminator  $\Psi(\tau, c)$  that takes in a trajectory and contact mode and outputs the probability that  $\tau$  is “real”, or similar to  $\mathcal{D}$ . To train the discriminator, we use a dataset consisting of  $\mathcal{D}$  and an equal number of trajectories sampled from the diffusion model. We use a U-Net with a Sigmoid activation output layer to model  $\Psi$ .

### C. Probabilistic Contact Sequence Search

While using  $C \sim p(C)$  can lead to task success, there are multiple reasons why planning contact modes can be beneficial. First, there may be redundancies in the contact sequences used offline. For example, finger positions may be reset more often than is necessary. In addition, planning contacts reduces the reliance on a strong prior for achieving good task performance.

To find the contact mode sequence, we construct an  $A^*$  tree search problem where each node  $n$  in the tree corresponds to a contact mode sequence  $C$ . The descendants

---

**Algorithm 1:** propagate\_variability

---

- 1 Given  $\mathbf{n}^p = (\mathcal{P}^p, S^p, C^p)$ ,  $\mathbf{c}'$ ,  $M$ ,  $\Psi$ ,  $k$ ,  $\gamma$
  - 2  $\bar{\tau} \leftarrow$  Diffuse  $k$  trajectories from  $M(\mathbf{c}', \mathcal{P}_H^p)$
  - 3  $d \leftarrow \text{depth}(\mathbf{n}^p)$
  - 4  $S \leftarrow \gamma^d \cdot \Psi(\bar{\tau}, \mathbf{c}') + S^p$
  - 5  $\bar{S} \leftarrow \{S_i / \sum_{S_i \in S} S_i \mid \forall S_i \in S\}$
  - 6  $\mathcal{P} \leftarrow k$  samples from  $\bar{\tau}$  given probabilities  $\bar{S}$
  - 7  $C \leftarrow C^p \cup \mathbf{c}'$
  - 8  $\mathbf{n} = (\mathcal{P}, S, C)$
  - 9 return  $\mathbf{n}$
- 

of a parent node  $\mathbf{n}^p$  are computed by appending an additional contact mode  $\mathbf{c}'$  to the parent node's sequence  $C^p$ .

We use trajectories conditioned on  $C$  to compute costs for A\* and check if  $\mathbf{o}_G$  has been achieved in the search. As opposed to prior work [4] that uses a single trajectory at each node, we model a distribution  $p_C(\tau)$  at each node to reason about variability in diffusion sampling. We parameterize  $p_C(\tau)$  with a set of particles  $\mathcal{P}$  where each particle is a trajectory diffused by  $M$ . Each particle has a weight, calculated by normalizing a score  $S$  computed by the discriminator  $\Psi(\tau, \mathbf{c})$ . The full definition of a node is  $\mathbf{n} = (\mathcal{P}, S, C)$ . By using this particle-based representation, we can approximate  $p(\tau|\mathbf{c}, \mathbf{s}_0)$  with  $p_C(\tau)$ .

To expand to new nodes in the A\* search and compute costs, we use our diffusion model. As diffusion models are learned and therefore can be unreliable, it is possible to diffuse trajectories that are unrealistic. To address this, we explicitly reason about the variability in the diffusion model output during the planning process.

As shown in Algorithm 1, we diffuse  $k$  trajectories to construct a population  $\bar{\tau}$  from which we sample new particles. We sample 1 trajectory for each particle from the parent node, conditioned on the child node's new contact mode  $\mathbf{c}'$  and the endpoints of the parent trajectories  $\mathcal{P}_H^p$  to enforce continuity. We compute the weights  $S$  for  $\bar{\tau}$  using  $\Psi$ . We accumulate the scores as we expand the tree, discounted by a factor  $\gamma^d$ , where  $d$  is the depth of the node in the search tree. We sample  $k$  trajectories from  $\bar{\tau}$  using the normalized scores.

Costs and goal evaluations for the search are calculated using expectations over the  $k$  particles. Combined with replanning after each contact mode is executed, explicitly reasoning about variability allows us to reduce the stochasticity of planned contact sequences. We seek to compute a contact sequence that leads to a minimum cost trajectory to the goal. We therefore design our cost-to-come  $g$  using the CSVTO cost  $J$  as shown in (3).

$$g(p_C(\tau), C) = \mathbb{E}_{\tau \sim p_C(\tau)}[J(\tau, C)] \quad (3)$$

We take an expectation over particles weighted by their scores and pass in  $C$  to account for mode-specific objectives.

A\* uses an additional heuristic  $h$ , to guide the search and improve search speed. We design a heuristic that focuses the search on contact sequences with high likelihood under the prior and is biased toward trajectories that have a lower goal cost. To compute  $h$ , we compute an approximation of the likelihood of a contact sequence under  $p(C)$  and

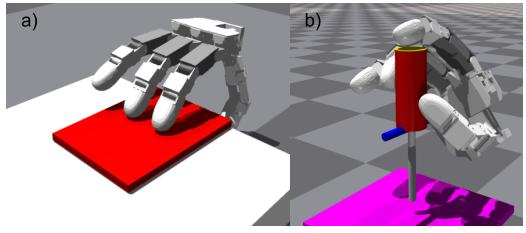


Fig. 3. a) Simulated card and b) Simulated screwdriver environments. The blue valve in b) is for visualization only and has no collision geometry.

also use a terminal cost  $\phi(\tau)$  as used in model predictive control methods. For example, by considering distance of the terminal state to the goal, we can encourage contact sequences that more quickly reach the goal.

We use a 1-step Markov approximation  $p(\mathbf{c}_n|\mathbf{c}_{n-1})$  of  $p(C)$  as a prior to guide the search. However, there will be contact mode transitions that may not be present in  $\mathcal{D}$  that we wish to consider when planning. To address this, we enforce a minimum probability  $p_{min}$  for transitions.

In our heuristic, shown in (4), we add the first term, the expected terminal cost, to the second term, the negative log-likelihood of  $C$  under the prior. The terms are weighted by  $\alpha, \beta \in \mathbb{R}$  respectively. While not admissible, this heuristic is useful to guide our search and improve its efficiency.

$$h(p_C(\tau), \mathbf{c}_{1:N}) = \alpha \cdot \mathbb{E}_{\tau \sim p_C(\tau)}[\phi(\tau)] - \beta \cdot \left[ \log(p(\mathbf{c}_0)) + \sum_{n=1}^N \log(p(\mathbf{c}_n|\mathbf{c}_{n-1})) \right] \quad (4)$$

## V. EXPERIMENTS AND RESULTS

We evaluate DIPS on 3 tasks using an Allegro multi-fingered hand: A simulated task in which the hand slides a card-like object along a table, a simulated task in which the hand turns a screwdriver, and the screwdriver-turning task in the real world. In all tasks, the pose of the base of the hand is fixed. Simulations are implemented in Isaac Gym [35].

For each task, we define contact modes where each mode specifies objectives and constraints for CSVTO.  $\mathbf{o}_G$  specifies the goal of the A\* planner, but the goal for a specific contact mode used in CSVTO will differ as we are attempting to achieve  $\mathbf{o}_G$  through a series of contact interactions. We define separate goals used with CSVTO for each contact mode. In addition, we use a timeout of 300 seconds when running the A\* search. If the search times out, we return the node that most closely reaches  $\mathbf{o}_G$ . For all tasks, we use  $\delta = 0.015 \text{ m}$ ,  $k = 16$ ,  $\gamma = .9$ ,  $\alpha = 1 \times 10^4$ ,  $\beta = 1 \times 10^3$ .

### A. Ablations and Baselines

We evaluate multiple ablations and baselines, running 10 trials for each method. Optimization budgets are the same for DIPS and all ablations. We run 5 ablations: (1) “CSVTO-Sampled Fixed Sequence”: We use  $C \sim p(C)$  and initialize the trajectory optimizer as in [3]; (2) “DIPS-Sampled Fixed Sequence”: We use  $C \sim p(C)$  and samples from  $M$  to initialize CSVTO; (3) “DIPS-No Contact Replanning”: We plan  $C$  once at the beginning of the task, executing without replanning. (4) “DIPS-No variability Propagation”: We use DIPS with  $k = 1$ , thus removing variability propagation in the A\* search; (5) “DIPS-Max likelihood”: We store a single

trajectory at each node but diffuse multiple samples, picking the highest-scoring when expanding.

We also baseline our method against Diffusion Policy [2], which uses a diffusion model to learn a receding horizon policy from demonstrations. We use the simulated demonstrations to train diffusion policy, concatenating trajectories from different contact modes. With this baseline, we seek to show the benefits of reasoning independently about different contact modes as well as the trajectory optimization.

While sample-based methods like Model Predictive Path Integral Control (MPPI) [36], might be considered, we do not include them here. MPPI does not strictly enforce constraints involving contact, and prior work [3] has shown it to struggle to perform dexterous tasks like screwdriver turning.

### B. Simulated Tabletop Card Manipulation

In this task, shown in Fig. 3a, the hand manipulates a card on a table. We use  $\mathbf{o} = [x, y, \theta]$ , where  $x, y$  are positions of the card in the world frame and  $\theta$  is the card’s yaw angle. The goal is to use the index and middle fingers to slide the card -6 cm along the world  $y$ -axis toward the palm to set up a grasp. We only plan the sliding behavior, not the grasping. We report distance to the goal, also used for  $\phi(\tau)$ .

We define 4 contact modes: (a) the index finger moves the card while the middle finger regrasps, (b) the middle finger moves the card while the index finger regrasps, (c) both fingers move the card along the table, and (d) where both fingers regrasp. The modes differ in the goal for CSVTO. For (a), (b), and (c), the CSVTO goal is to move the card -2 cm along the world  $y$  axis toward the palm. For (d), the CSVTO goal is to keep the card stationary. We use a uniform prior for  $p(C)$  in which all mode transitions are equally likely and generate 480 demonstrations, each with 5 contact modes.

We execute a maximum of 5 contact modes. We run the A\* planner before each contact mode, with a maximum depth that decreases as we execute contact modes to improve convergence to the goal.

As shown in Fig. 4(a,b), DIPS outperforms the baselines and ablations by avoiding unneeded regrasps. We come within .6 cm of  $\mathbf{o}_G$  while DIPS-Sampled Fixed Sequence comes within 2.6 cm and CSVTO-Sampled Fixed Sequence and Diffusion Policy come within 4 cm of  $\mathbf{o}_G$ . Even though the training data sequences are uniformly randomly sampled, DIPS is still able to consistently produce useful contact sequences through the use of planning. This shows the benefit of planning as a way to outperform training demonstrations.

Diffusing initializations for CSVTO takes 3.6 s on average and CSVTO takes 7 s per step, while Diffusion policy takes 1.1 s. Each A\* planning call takes 64.9 s on average for DIPS, 301.9 s for ablation (3), 52.7 s for ablation (4), and 13.8 s for ablation (5). Expanding an edge with CSVTO would take approximately 18x long as using  $M$ , motivating the use of  $M$  in the search.

### C. Simulated Screwdriver Turning

In this task, shown in Fig. 3b, the hand turns a screwdriver using the thumb, index, and middle fingers. The base of the screwdriver is attached to the table but can rotate, simulating

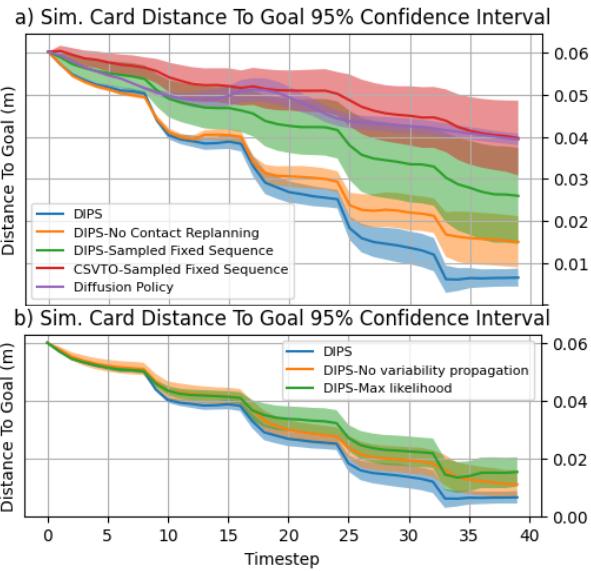


Fig. 4. Simulated Card results over 10 trials.

driving a screw in a slot. We define  $\mathbf{o}$  as the orientation of the screwdriver, parameterized by its roll, pitch, and yaw. The goal is to turn the screwdriver as far clockwise as possible. For the A\* goal  $\mathbf{o}_G$  and  $\phi(\tau)$ , we only consider the yaw angle. Additionally, because of the overall goal of turning the screwdriver as far as possible, we update  $\mathbf{o}_G$  before each planning call. Before planning, we set  $\mathbf{o}_G$  to be  $\frac{\pi}{3}$  less than the current yaw. This is based on expecting to turn approximately  $\frac{\pi}{2}$  across 7 modes in the prior, but a search over different values led to the optimal setting of  $\frac{\pi}{3}$ . For DIPS-No Contact Replanning, we performed a grid search to arrive at a goal of -1.7 rad.

We define 3 contact modes: (a) all 3 fingers are in contact and the hand is turning the screwdriver, (b) the thumb and middle finger are in contact and the index finger regrasps, and (c) the index finger is in contact and the thumb and middle fingers regrasp. For (a), the CSVTO goal is to maintain the same roll and pitch while reducing the yaw by  $\frac{\pi}{6}$ . For (b) and (c), the goal is to maintain the same screwdriver pose.

To sample from  $p(C)$ , we sequence (a), followed by (b) and (c) in random order, then repeat. This means we turn, then regrasp all fingers, randomly ordering the regrasp modes, then resume turning. We calculate a Markov approximation of the prior with  $p_{min} = .1$ :  $p(\mathbf{c}_0) = [.1 \ .1 \ .8], p(\mathbf{c}_n | \mathbf{c}_{n-1}) = .1$  if  $c_n = c_{n-1}, .45$  otherwise. We generate 240 training demonstrations, each with 7 modes.

Online, we execute a maximum of 7 modes. We use the same depth of 7 for A\* throughout the task to encourage turning as far as possible.

As shown in Fig. 5(a,b), DIPS outperforms the ablations and baselines, turning the screwdriver 12% further than DIPS-Sampled Fixed Sequence and 35% further than CSVTO-Sampled Fixed Sequence. This is because it is not always necessary to regrasp all fingers before executing another turn, as is done in the data. In addition, it is often possible to execute multiple turn modes in a row, even though this does not occur in the data. While the prior could be altered, what is significant is that we are able to generate

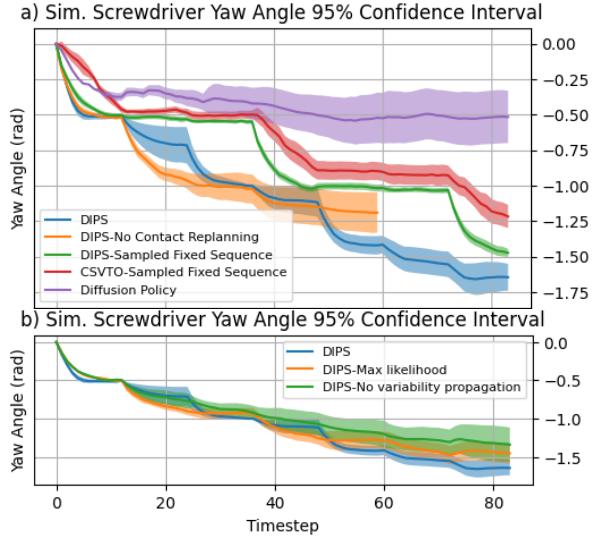


Fig. 5. Simulated Screwdriver results over 10 trials.

trajectories that outperform the data on which  $M$  is trained. We find the Diffusion Policy turns the screwdriver slower than DIPS, limiting how far it turns.

In addition, replanning the contact sequence is beneficial as error in the diffusion, even with variability modeling, can lead the planner to overestimate screwdriver turning and lead to a sub-optimal plan which does not reach the specified  $\mathbf{o}_G$ . Without replanning, as there is a fixed goal for the search, it is possible for A\* to terminate before expanding to the maximum depth, leading to a shorter overall trajectory.

We show that our variability propagation method leads to 23% further turning than the no variability propagation ablation with lower variance in the yaw angle. Without any variability propagation, low-quality samples from  $M$  can lead the planner to choose lower-quality contact mode sequences. We can see the utility of  $\Psi$  by comparing the max-likelihood and no variability propagation methods. Choosing the maximum likelihood sample outperforms choosing a random sample, indicating that  $\Psi$  helps select higher-quality trajectories. However, DIPS outperforms the maximum likelihood ablation. We believe this is due to overly trusting the  $\Psi$  output, which as a learned model also has error. In addition,  $\mathcal{D}$  can contain trajectories that do not satisfactorily perform the task due to failures of trajectory optimization. Diffused trajectories could therefore be similar to the dataset but low-quality. Storing a set of particles allows us to better account for low-quality samples from  $M$  and errors in  $\Psi$ .

Diffusing initializations for CSVTO takes 3.6 s on average and CSVTO takes 7.2 s per step, while Diffusion policy takes 1.1 s. Average A\* planning time is 46.8 s for DIPS, 13.7 s for ablation (3), 14 s for ablation (4), and 16.4 s for ablation (5). Expanding an edge with CSVTO would take approximately 20x long as using  $M$ , motivating the use of  $M$  in the search.

#### D. Real Screwdriver Turning

We also perform the screwdriver turning task in the real world (shown in Fig. 1), using the same specifications for the A\* search and using the same models for  $M, \Psi$ .  $\mathbf{o}_t$  is estimated using Aruco tags on the screwdriver.

For this task, we run DIPS and DIPS-Sampled Fixed

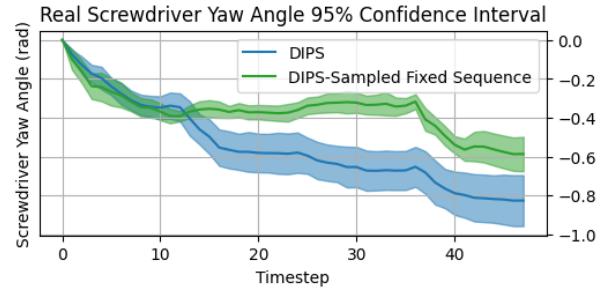


Fig. 6. Real screwdriver manipulation results for valid executions over 10 trials. DIPS (10/10 valid) outperforms the ablation (8/10 valid) by 41%.

Sequence for 10 trials, executing 4 modes, to demonstrate the utility of the contact planning over the prior in the real world. Due to imperfect modeling in our trajectory optimization and limitations of the hardware, we alter the force initializations sampled from  $M$ . Directly initializing with the diffused forces leads CSVTO to output forces that are too low to turn the screwdriver, as part of  $J_{smooth}$  is a regularization on force magnitude. We initialize as in [3] for the thumb and middle fingers in the turn mode. The diffused trajectories used in A\* are not altered. This sim-to-real gap can be addressed through more advanced modeling of the forces in the trajectory optimization.

As shown in Fig. 6, DIPS outperforms the ablation, turning 41% further. DIPS plans 25% more turning modes than the ablation, reducing unnecessary regrasps. However, due to perception and execution error and possibly the change in CSVTO initialization, we turn 74% as far as in simulation. These errors also lead to the ablation dropping the screwdriver twice while DIPS does not drop it. When regrasping the index finger, perception and execution errors can lead the thumb and middle fingers to be ill-positioned to support the screwdriver. We find DIPS plans index regrasps only twice across 10 trials as they are less likely to enable further turning than thumb/middle regrasps. Every ablation trial includes an index regrasp due to the structure of  $p(C)$ . We believe these errors in regrasping help explain the gap in performance between DIPS and the ablation beyond planning more turns.

Diffusing initializations for CSVTO takes 3.1 s on average, while CSVTO takes 16.1 s per step. We use a higher CSVTO budget on hardware. Each A\* planning call takes 10.1 s on average. A\* planning times are lower than in simulation as we only execute 4 contact modes. We find the planning time can be higher for later contact modes in simulation.

## VI. CONCLUSION

We presented DIPS, a planning method for contact-rich manipulation that combines generative modeling and search methods. We constructed an approximation to a trajectory optimizer by using a diffusion model trained on optimized trajectories. Then, using a particle-based representation, we reasoned about variability in the diffusion to plan contact sequences that outperformed those in the training set. DIPS outperformed ablations and baselines, including on a challenging hardware screwdriver turning task. As our method requires defining all considered contact modes in the data generation and search, future work could investigate methods to automatically generate task-relevant contact modes.

## REFERENCES

- [1] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, “Planning with diffusion for flexible behavior synthesis,” 2022.
- [2] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion,” 2024, arXiv:2303.04137 [cs].
- [3] F. Yang, T. Power, S. A. Marinovic, S. Iba, R. S. Zarrin, and D. Berenson, “Multi-finger manipulation via trajectory optimization with differentiable rolling and geometric constraints,” 2024.
- [4] R. Natarajan, G. L. Johnston, N. Simaan, M. Likhachev, and H. Choset, “Torque-Limited Manipulation Planning through Contact by Interleaving Graph Search and Trajectory Optimization,” in *ICRA*, 2023.
- [5] M. Posa, C. Cantu, and R. Tedrake, “A direct method for trajectory optimization of rigid bodies through contact,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.
- [6] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” 2020.
- [7] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, “A unified mpc framework for whole-body dynamic locomotion and manipulation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.
- [8] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Höning, “Sequence-of-constraints mpc: Reactive timing-optimal control of sequential manipulation,” 2022.
- [9] J. Xu, T.-K. Koo, and Z. Li, “Sampling-based finger gaits planning for multifingered robotic hand,” *Autonomofus Robots*, 05 2010.
- [10] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, “Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d,” in 2022 International Conference on Robotics and Automation (ICRA), 2022.
- [11] F. Rozzi, L. Roveda, and K. Haninger, “Combining sampling- and gradient-based planning for contact-rich manipulation,” 2024.
- [12] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake, “Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models,” *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4691–4711, 2023.
- [13] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason, “Enhancing dexterity in robotic manipulation via hierarchical contact exploration,” *RA-L*, vol. 9, no. 1, pp. 390–397, 2024.
- [14] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, “Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation,” *IROS*, pp. 8262–8268, 2021.
- [15] M. Zhang, D. K. Jha, A. U. Raghunathan, and K. Hauser, “Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning,” *arXiv preprint arXiv:2306.06465*, 2023.
- [16] R. Natarajan, H. Choset, and M. Likhachev, “Interleaving Graph Search and Trajectory Optimization for Aggressive Quadrotor Flight,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5357–5364, Jul. 2021, conference Name: IEEE Robotics and Automation Letters.
- [17] R. Natarajan, S. Mukherjee, H. Choset, and M. Likhachev, “PINSAT: Parallelized Interleaving of Graph Search and Trajectory Optimization for Kinodynamic Motion Planning,” Mar. 2024, arXiv:2401.08948 [cs].
- [18] S. Y. C. Chia, R. H. Jiang, B. P. Graesdal, L. P. Kaelbling, and R. Tedrake, “GCS\*: Forward Heuristic Search on Implicit Graphs of Convex Sets,” 2024.
- [19] V. Kumar, A. Gupta, E. Todorov, and S. Levine, “Learning Dexterous Manipulation Policies from Experience and Imitation,” 2016.
- [20] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, “Deep Dynamics ModeLearning Complexes for Learning Dexterous Manipulation,” in *Conference on Robot Learning (CoRL)*, 2019.
- [21] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, “Data-efficient deep reinforcement learning for dexterous manipulation,” 2017.
- [22] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, “Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost,” in *ICRA*, 2019.
- [23] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak, “Generalization in Dexterous Manipulation via Geometry-Aware Multi-Task Learning,” 2021.
- [24] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine, “Dexterous Manipulation from Images: Autonomous Real-World RL via Substep Guidance,” in *ICRA*. arXiv, 2023.
- [25] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, “Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations,” 2018.
- [26] K. Shaw, S. Bahl, A. Sivakumar, A. Kannan, and D. Pathak, “Learning dexterity from human hand motion in internet videos,” *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 513–532, 2024.
- [27] T. Li, K. Srinivasan, M. Q.-H. Meng, W. Yuan, and J. Bohg, “Learning hierarchical control for robust in-hand manipulation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8855–8862.
- [28] G. Khandate, C. P. Mehlman, X. Wei, and M. Ciocarlie, “Dexterous in-hand manipulation by guiding exploration with simple sub-skill controllers,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6551–6557.
- [29] R. S. Zarrin, R. Jitosho, and K. Yamane, “Hybrid learning-and model-based planning and control of in-hand manipulation,” in *IROS*. IEEE, 2023, pp. 8720–8726.
- [30] E. K. Gordon and R. S. Zarrin, “Online augmentation of learned grasp sequence policies for more adaptable and data-efficient in-hand manipulation,” in *ICRA*, 2023, pp. 5970–5976.
- [31] T. Power and D. Berenson, “Constrained stein variational trajectory optimization,” *IEEE Transactions on Robotics*, 2024.
- [32] T. Power, R. Soltani-Zarrin, S. Iba, and D. Berenson, “Sampling constrained trajectories using composable diffusion models,” in *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.
- [33] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” 2022.
- [34] S. Zhou, Y. Du, S. Zhang, M. Xu, Y. Shen, W. Xiao, D.-Y. Yeung, and C. Gan, “Adaptive online replanning with diffusion models,” 2023, [Online]. Available: <https://arxiv.org/abs/2310.09629>
- [35] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, “Isaac gym: High performance gpu-based physics simulation for robot learning,” 2021.
- [36] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic mpc for model-based reinforcement learning,” in *ICRA 2017*, 2017, pp. 1714–1721.