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

Diffusing Trajectory Optimization Problems for Recovery During Multi-Finger Manipulation

Abhinav Kumar[1], Fan Yang[1], Sergio Aguilera Marinovic[2], Soshi Iba[2], Rana Soltani Zarrin[2], Dmitry Berenson[1]

***Abstract*— Multi-fingered hands are emerging as powerful platforms for performing fine manipulation tasks, including tool use. However, environmental perturbations or execution errors can impede task performance, motivating the use of recovery behaviors that enable normal task execution to resume. In this work, we take advantage of recent advances in diffusion models to construct a framework that autonomously identifies when recovery is necessary and optimizes contact-rich trajectories to recover. We use a diffusion model trained on the task to estimate when states are not conducive to task execution, framed as an out-of-distribution detection problem. We then use diffusion sampling to project these states in-distribution and use trajectory optimization to plan contact-rich recovery trajectories. We also propose a novel diffusion-based approach that distills this process to efficiently diffuse the full parameterization, including constraints, goal state, and initialization, of the recovery trajectory optimization problem, saving time during online execution. We compare our method to a reinforcement learning baseline and other methods that do not explicitly plan contact interactions, including on a hardware screwdriver-turning task where we show that recovering using our method improves task performance by 96% and that ours is the only method evaluated that can attempt recovery without causing catastrophic task failure. Videos can be found at https://dtourrecovery.github.io/.**

I. Introduction

As progress is made on techniques that enable fine multi-finger manipulation [1]–[7], it is important to consider scenarios in which these methods might perform poorly. These could occur due to execution error or external perturbations. For example, for precise tasks such as turning a screwdriver, perturbations to the object could push the system state out of the domain in which the task policy was trained. By detecting these scenarios, we can execute recovery behaviors after which the system state is conducive to normal task execution. Prior work has addressed learning recovery behavior [8]–[14] through reinforcement learning or behavior cloning. In contrast to prior work, we focus on learning recovery behaviors that are less likely to lead to catastrophic failure of precise, contact-rich tasks by imposing strict constraints via trajectory optimization, where we generate the trajectory optimization problem and initializations online using a learned model.

Our method, Diffused Trajectory Optimization for Use in Recovery (D-TOUR), uses a diffusion model [15], [16] to generate trajectory optimization problems solved to execute recovery behavior. The model samples the full parameterization of a trajectory optimization problem, including constraints, objectives, and initializations. Starting with a separate diffusion model trained on trajectories from normal task execution, we execute the task in simulation and apply perturbations. We use the task diffusion model to detect states from which it does not sample useful trajectories, taking advantage of recent work in estimating likelihoods of diffused trajectories [8] to structure this as an out-of-distribution (OOD) detection problem. We initiate recovery when we encounter OOD states. To inform generation of a training dataset of optimization problems, we propose a novel method to project OOD states in-distribution (ID) of the task diffusion model. This ID state serves as the target for recovery. We rely on a key insight, which is to use the natural likelihood-maximizing properties of diffusion sampling to perform the projection. To further structure the trajectory optimization, we define contact modes which encode contact behavior for the fingers of a robot hand. These contact modes define constraints and objective functions, where specific parameters of these functions are calculated from the projected ID state. The recovery diffusion is then trained to jointly sample contact modes and recovery trajectories, where the joint sampling allows us to reconstruct a recovery trajectory optimization problem online. Additionally, we use the sampled trajectories as initializations to the non-convex optimization. In this work, we use CSVTO [17] to solve the optimization, though other solvers could also be used.

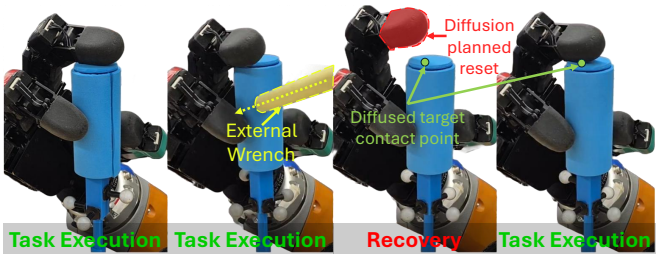

Fig. 1. During a screwdriver turning task, we apply external wrench perturbations. Our method detects that recovery is needed and diffuses a trajectory optimization problem that encodes the set of fingers to reset and corresponding target contact points. The index finger is reset in this example.

Our contributions are (1) A framework that uses diffusion models to generate recovery behavior and switch between task execution and recovery execution as needed; (2) A method, based on diffusion sampling, to project OOD states in-distribution and inform recovery trajectories; (3) A method to diffuse trajectory optimization problems that are solved to realize in-distribution projection of states.

We evaluate our method on a valve turning task and a screwdriver turning task in simulation and on hardware. Our results show that our method allows us to detect when recovery is necessary and execute recovery behavior in a way that improves task performance. We compare against

[1]Robotics Department, University of Michigan, Ann Arbor, MI, USA abhin@umich.edu, [2] Honda Research Institute USA. This work was sponsored by Honda Research Institute USA.

a safe reinforcement learning baseline and methods that do not diffuse the full optimization problem, or do not reason about specific contact interactions. We show these methods are more likely to lead to catastrophic task failure and less likely to recover in a way that aids task execution. We also compare against multiple methods that do not explicitly reason about the difference between task execution and recovery or explicitly reason about different contact modes, showing the benefit of explicit reasoning.

II. Related Work

Recovery: Prior work has examined how to detect and recover from problematic states [8]–[14]. Some works using reinforcement learning (RL) [9]–[11] learn classifiers that detect when recovery is required and recovery policies to compute the recovery behavior. The tasks we consider have contact constraints that induce low-dimensional constraint-satisfying trajectory manifolds and these methods can struggle to learn policies that avoid catastrophic failure, i.e. dropping a tool. We address this issue using a combination of diffusion, which has been shown to learn useful models of high-dimensional distributions, and trajectory optimization to compute recovery behaviors which allows us to better satisfy constraints that are important for enabling fine manipulation. We compare against [9] in our experiments to show the importance of explicitly considering constraints in recovery.

Other works use behavior cloning methods for task policies and use density estimation to decide when recovery is needed, using gradients of density estimates to compute recovery actions [12]. We also use a variant of density estimation through diffusion model likelihoods and train a recovery policy that is informed by these density estimates. However, our recovery policy is of a more general form than the one in [12] which defines actions only as changes in state and directly executes the gradient of the state density estimation network. In multi-finger manipulation tasks, it can be difficult to follow a direct state gradient to recover, requiring more complex recovery policies.

Implicit Methods: In contrast to recovery methods, other works approach the full manipulation problem with a single policy that implicitly reasons about contact mode changes. These include reinforcement learning methods such as [5], [6], [18], and contact-implicit trajectory optimization methods such as [3], [4], [7]. While these methods are attractive due to their ability to implicitly choose when to switch contact and react to disturbances, they provide less control and interpretability over when these behaviors occur, which is problematic for fine manipulation tasks. We compare against [18] and [7] to illustrate the benefits of our explicit reasoning over recovery and contact modes.

Diffusion and Planning: Diffusion models have recently emerged as powerful planners. Prior work that combines trajectory optimization or planning with diffusion usually focuses either on using diffusion models to initialize non-convex optimization problems [1], [19], [20] or using diffusion models to solve trajectory optimization or motion planning problems with *a priori* known constraints and objectives [21]–[29].

Additionally, prior work has explored using planning to generate trajectory data for diffusion models [30], [31] as we do, but we diffuse trajectory optimization problems along with trajectories to ensure constraint satisfaction of trajectories. While prior work has learned cost functions [32] and constraint representations for collision-checking [33], we focus on diffusing the entirety of the optimization problem.

[34] diffuses grasps for multi-finger hands. While this could compute ID states in our framework, it trains an additional diffusion model and classifier to perform the sampling unlike our method which only uses the task diffusion model.

III. Problem Statement

Within the broader problem of manipulation with multi-finger hands, we focus on detecting and recovering from states that, if recovery were not performed, would inhibit task progress. We define the state $\mathbf{s}_t$ at time t as $\{\mathbf{q}_t, \mathbf{o}_t\}$, where $\mathbf{q}_t$ is the robot configuration (joint angles) and $\mathbf{o}_t$ is the object's SE(3) pose. We assume the system is quasi-static, meaning that all velocities are 0 before each action is taken. We additionally assume access to a simulator where we can model the system and execute the task. To enable this, we assume that geometries of the object and robot are known. This would be the case in a factory setting, where robots would be manipulating a known set of objects.

As part of executing recovery behaviors, we choose how fingers make and break contact with the object. For example, we may want to reset the position of a specific finger that has slipped off a screwdriver while attempting to turn it. To help with this, we reason over contact modes that encode the contact behavior of each finger. We define contact mode $\mathbf{c} := \{0,1\}^{n_f}$, where n_f is the number of fingers. A value of 1 means the finger should remain in contact throughout the trajectory, whereas a value of 0 means the finger should break contact with the object before making contact again. We refer to this behavior as "resetting" the finger.

Contact modes define inequality constraints g, equality constraints h, and objectives J for a trajectory optimization problem that produces recovery behavior. The trajectory optimization problem, shown in (1) with more detail in Appendix A, optimizes trajectories $\boldsymbol{\tau} := \{\mathbf{s}_{0:H}, \mathbf{u}_{0:H-1}\}$ of length H, where $\mathbf{u}_t$ is a control input $\{\Delta\mathbf{q}_i, \mathbf{f}_i\}_{i=1}^{n_f}$. $\Delta\mathbf{q}_i$ is the commanded change in $\mathbf{q}$ and $\mathbf{f}_i$ is the force applied by the ith finger.

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

We assume access to a task diffusion model M trained on task executions, as done in [1], [8], [26], [27]. M can either be directly used during task execution or can model the distribution of trajectories output by the task policy. We assume M can be conditioned on $\mathbf{s}_0$ so that all sampled trajectories have $\mathbf{s}_0$ as the initial state. We use this diffusion model to calculate the likelihood of states encountered during task execution and inform recovery trajectory generation.

We evaluate our method on its ability to recover from detected errors as well as overall task performance to evaluate if recovery is beneficial for task execution.

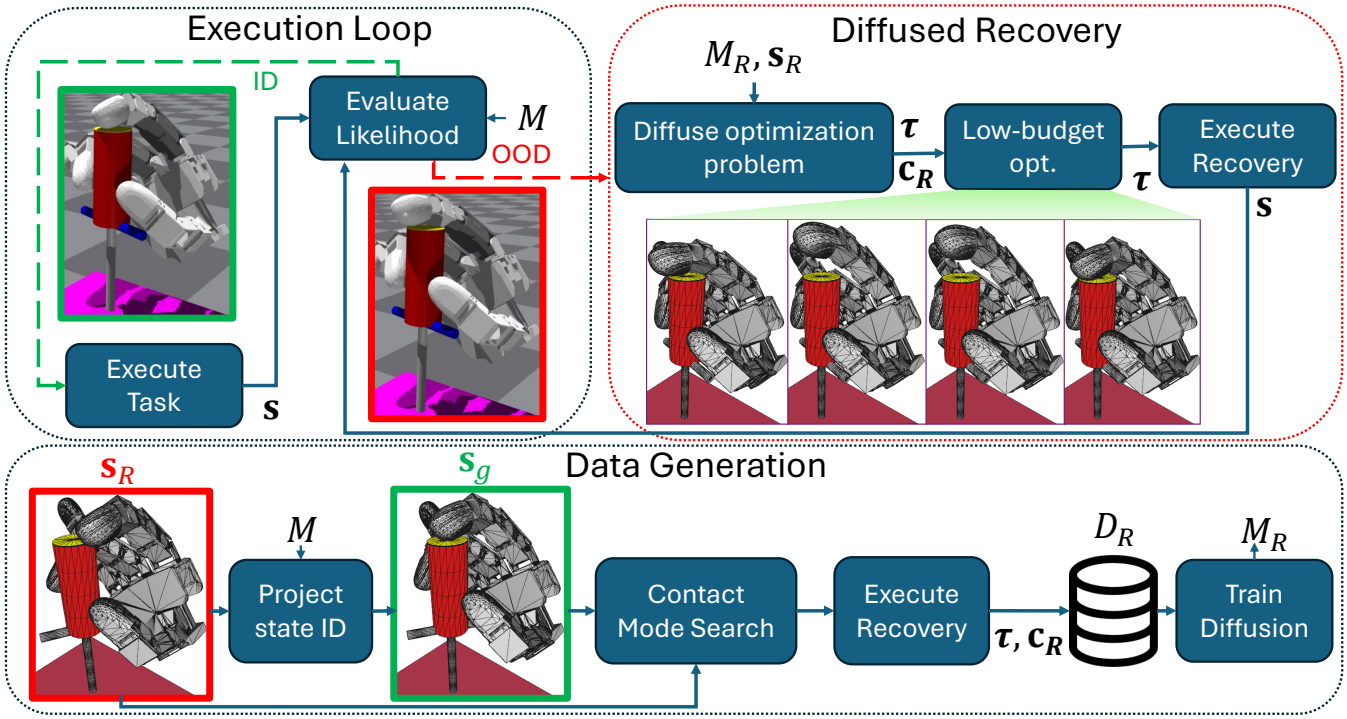

Fig. 2. While executing a task, our method uses a task diffusion model M to detect when the current state is out-of-distribution (OOD). We initiate recovery behavior in OOD states as indicated by the dashed red line. We train a diffusion model M_R to jointly diffuse trajectories $\boldsymbol{\tau}$ and contact modes $\mathbf{c}_R$ which together parameterize and initialize a trajectory optimization problem. We resume execution after recovery if the state is ID, indicated by the dashed green line. Otherwise, we retry recovery. To generate training data, we first project OOD states $\mathbf{s}_R$ to ID $\mathbf{s}_g$ using M. We then search over a set of contact modes, choosing the one that leads to the highest likelihood state. We add the planned trajectory and contact mode to the dataset D_R and train M_R on this dataset.

IV. Methods

We first explain how we detect when recovery is needed using M in Sec. IV-B. We then discuss our recovery pipeline in Sec. IV-C. Sec. IV-C.1 details how we use M to perform ID projection to compute a goal state for the recovery. In Sec. IV-C.2, we discuss a method to plan contact modes that provide constraints and objective functions for the trajectory optimization. Finally, Sec. IV-D explains how given a dataset of recovery trajectories, we can train a diffusion model to diffuse recovery optimization problems online.

A. Diffusion Preliminary

Diffusion models [15], [16] are trained to reverse a Markov process $q(x^k|x^{k-1})$ by which data x is iteratively corrupted by Gaussian noise. There are multiple ways to formulate this reverse diffusion process, one of which we discuss here to provide context for our method. We can interpret a diffusion model as learning a function $\epsilon(x^k, k)$, where x^k is a noisy input and k is the step in the denoising process. At each step of the denoising process, we update x^k as follows:

$$x^{k-1} = x^k + \alpha_k \cdot \epsilon(x^k, k) + \mathbf{z}\sqrt{2\alpha_k}, \quad \mathbf{z} \sim \mathcal{N}(0, I) \quad (2)$$

α is a learning-rate hyperparameter that follows a schedule over the reverse diffusion process. ϵ approximates the score function, or $\nabla_x \log\, p(x)$, and the reverse diffusion is treated as sampling using Langevin Dynamics. By performing the reverse diffusion, the learned diffusion model approximately samples from $p(x)$ by iteratively sampling from $p_\theta(x^{k-1}|x^k)$, with learned parameters θ.

B. OOD Detection

We use the task diffusion model M to detect when recovery behavior is required. Intuitively, we seek to detect states from which M is less likely to sample useful trajectories. As diffusion models approximate probability distributions over their training data, we can estimate likelihoods of trajectories under the model to use as an out-of-distribution, or OOD, metric. Given a state $\mathbf{s}$, we sample several trajectories from M conditioned on $\mathbf{s}$ and take the average of each trajectory's likelihood. With this, we compute a value $\tilde{p}(\mathbf{s}) \propto p(\boldsymbol{\tau}|\mathbf{s})$ that estimates M's ability to sample useful trajectories beginning at $\mathbf{s}$. We use the trajectory likelihood calculation method from [8], which quantifies the ability of the model to denoise a trajectory after it has been corrupted by various levels of Gaussian noise. It does this by computing $L(\boldsymbol{\tau}) = \frac{1}{|I|}\sum_{k\in K} KL(q(\boldsymbol{\tau}^{k-1}|\boldsymbol{\tau}^k, \boldsymbol{\tau})||p_\theta(\boldsymbol{\tau}^{k-1}|\boldsymbol{\tau}^k))$, where K is a subset of diffusion timesteps and KL is the Kullback-Leibler divergence. The choice of K is tunable, and we use $\{5, 10, 15\}$ as used in [8].

More formally, we compute:

$$\tilde{p}(\mathbf{s}) = \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\mathbf{s})}[L(\boldsymbol{\tau})] \quad (3)$$

where sampling from M approximates sampling from $p(\boldsymbol{\tau}|\mathbf{s})$ and $L(\boldsymbol{\tau})$ is the likelihood. We detect OOD if $\tilde{p}(\mathbf{s})$ is less than a threshold $p_{\min}$.

C. Offline Recovery Trajectory Generation

In this section, we explain how we generate a dataset D_R of recovery trajectories and contact modes, where $D_R = \{(\boldsymbol{\tau}_i, \mathbf{c}_i)\}_{i=1}^N$.

1) In-Distribution State Projection: We use M to project $\mathbf{s}_R$ from OOD to ID. Since M is trained on task executions, projecting $\mathbf{s}_R$ ID should result in a state from which task success is most likely, more so than possible alternatives such as attempting to recover to the most recently encountered

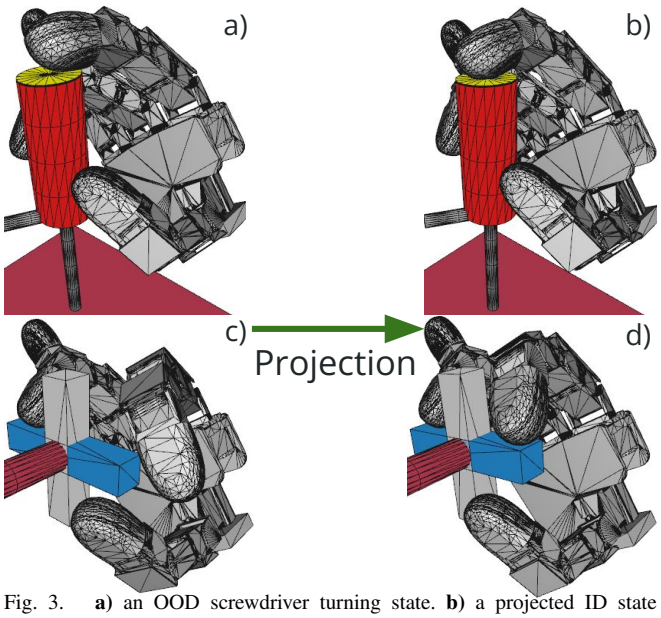

Fig. 3. **a)** an OOD screwdriver turning state. **b)** a projected ID state sampled from M. **c)** an OOD valve turning state. **d)** a projected ID state sampled from M.

state before recovery was initiated. As diffusion models can be interpreted as learning a score function, we can follow the learned likelihood gradient to compute an ID goal state $\mathbf{s}_g$. This is akin to sampling $\mathbf{s}_g$ from M by indexing the initial state of a trajectory sampled from M. M can be conditioned on $\mathbf{s}_R$, sampling from $p(\boldsymbol{\tau}|\mathbf{s}_R)$, but it can also be used to generate unconditioned trajectories, sampling from $p(\boldsymbol{\tau})$. If there is no state conditioning, we would expect M to sample high-likelihood trajectories, the initial states of which would be good ID targets. However, directly sampling from $p(\boldsymbol{\tau})$ could produce arbitrary ID states that may be difficult to reach from $\mathbf{s}_R$. We therefore modify the reverse diffusion process to sample an ID state informed by $\mathbf{s}_R$.

To do this, we begin the reverse diffusion process in (2) by sampling from $p(\boldsymbol{\tau}|\mathbf{s}_R)$, but remove the $\mathbf{s}_R$ conditioning partway through the reverse diffusion. We use the conditioned predictions for the first $T_\mathbf{s}$ diffusion steps, then remove the conditioning for the remaining steps. When $T_\mathbf{s} = 0$, the diffusion samples from the initial state distribution in M's training dataset with no influence from $\mathbf{s}_R$. When $T_\mathbf{s} = T_D$, we obtain $\mathbf{s}_R$. To compute $\mathbf{s}_g$, we sample N_g trajectories from M using the modified reverse diffusion and choose the initial state of the highest likelihood trajectory. We show example ID projections in Fig. 3, where the projection results in a state where the screwdriver is upright and the fingers are in more advantageous positions for turning. For the valve, the index finger configuration is more firmly placed on the valve.

2) Contact Mode Planning: We next need to choose a contact mode $\mathbf{c}_R$ that will determine the constraints and objective functions for the recovery trajectory optimization. We select $\mathbf{c}_R$ using a greedy search over a set of recovery modes C_R and choose the mode that results in the maximum likelihood state. As show in Alg. 1, we run trajectory optimization for each mode. We evaluate the terminal state of the optimized plan and select the contact mode that has the highest likelihood of reaching that state. While this computation is expensive as we run the trajectory optimization for multiple modes, we can afford to spend more computation time offline when generating data.

Algorithm 1 Offline Recovery Trajectory Generation

Require: OOD state $\mathbf{s}_R$, Recovery contact mode set C_R, Task diffusion model M, Recovery dataset D_R, OOD threshold $p_{\min}$, Trajectory optimizer `TrajOpt`

while $\tilde{p}(\mathbf{s}_R) < p_{\min}$ **do**
  Sample $\mathbf{s}_g$ from M, informed by $\mathbf{s}_R$
  $L_\mathbf{c} = \varnothing$
  for each $\mathbf{c} \in C_R$ **do**
    $\boldsymbol{\tau} = \texttt{TrajOpt}(\mathbf{s}_R, \mathbf{s}_g, \mathbf{c})$
    $L_\mathbf{c} = L_\mathbf{c} \cup \tilde{p}(\mathbf{s}_H)$ `//Final state`
  $\mathbf{c}_R = \underset{\mathbf{c}}{\text{argmax}}\ L_\mathbf{c}$
  $\mathbf{s}'_R =$ state after executing $\boldsymbol{\tau}$
  Add $(\boldsymbol{\tau}, \mathbf{c}_R)$ to D_R if $p(\mathbf{s}'_R) > p(\mathbf{s}_R)$
  $\mathbf{s}_R = \mathbf{s}'_R$

Once the contact mode is chosen, we execute the optimized trajectory. To account for error and disturbances, we run additional optimization steps after each action is executed. We do not perform receding horizon control, meaning the optimized trajectory reduces in length over the course of execution. To add trajectories to D_R, we concatenate the executed timesteps with the remaining plan at each timestep. This allows us to augment our data by adding a total of H, the horizon, trajectories to D_R per recovery mode. We only add these trajectories to D_R if, after complete execution of the recovery trajectory, the likelihood of the state has increased. This condition may not be met due to environmental perturbations. After the recovery trajectory is fully executed, we check if the state is now ID. If the state is not ID, we repeat the recovery process until the state is ID or until a maximum episode length is reached. Once we have reached an ID state, we continue with task execution, executing recovery behavior when necessary.

D. Trajectory Optimization Diffusion

The method proposed in Sec. IV-C can be expensive as it requires solving multiple trajectory optimization problems to select $\mathbf{c}_R$. To address this, we train a diffusion model that distills the recovery planning process by diffusing a recovery trajectory optimization problem. We train a second diffusion model M_R on D_R that models $p(\mathbf{c}, \boldsymbol{\tau}|\mathbf{s}_0)$, or the joint probability of trajectories and contact modes conditioned on an initial state. Learning this joint distribution allows us to generate the full parameterization of the trajectory optimization problem we solve to realize recovery behavior, including constraints, goal state, and initializations. We train M_R to approximate the following expression: $p(\mathbf{c}, \boldsymbol{\tau}|\mathbf{s}_0) = p(\boldsymbol{\tau}|\mathbf{s}_0)p(\mathbf{c}|\boldsymbol{\tau}, \mathbf{s}_0)$, derived using the chain rule of probability. We can interpret M_R as sampling a trajectory conditioned on the initial state while simultaneously estimating the contact mode that would induce the sampled trajectory.

Our model architecture is shown in Fig. 4. We use the U-Net Diffuser architecture [21] to predict $\epsilon_{\boldsymbol{\tau}}$, the denoising update for $\boldsymbol{\tau}$. We use a downsampling depth of 3 in the U-Net, doubling the feature dimension in each downsampling

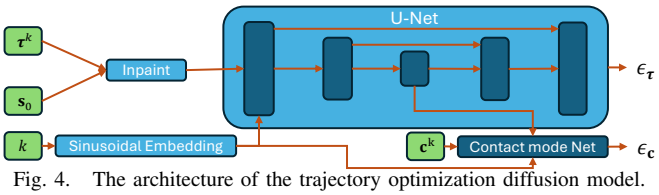

Fig. 4. The architecture of the trajectory optimization diffusion model.

block. We use a multi-layer-perceptron (MLP) that takes in the diffusion step k, noisy $\mathbf{c}^k$, and $\boldsymbol{\tau}^k$ encoded features from the U-Net to predict $\epsilon_{\mathbf{c}}$, the denoising update for $\mathbf{c}$. We use a Sinusoidal Positional Embedding [35] of k.

We condition the diffusion output on $\boldsymbol{\tau}, \mathbf{s}_0$ using multiple methods. We sample from $p(\boldsymbol{\tau}|\mathbf{s}_0)$ using in-painting, as done in [21]. In-painting is a method that directly overwrites part of the diffusion sample (trajectory) with the conditioning variable (initial state). We sample from $p(\mathbf{c}|\boldsymbol{\tau}, \mathbf{s}_0)$ using classifier-free guidance (CFG) [36]. which learns an additional score function $\tilde{\epsilon}(\mathbf{c}^k, k, \boldsymbol{\tau}, \mathbf{s}_0)$ and combines the predictions of $\tilde{\epsilon}$ with $\epsilon(\mathbf{c}^k, k)$ to compute $\epsilon_{\mathbf{c}}$. CFG learns $\tilde{\epsilon}$ and ϵ at training time and combines both in reverse diffusion:

$$\epsilon_{\mathbf{c}}(\mathbf{c}^k, k, \boldsymbol{\tau}, \mathbf{s}_0) = (1+w)\cdot\tilde{\epsilon}(\mathbf{c}^k, k, \boldsymbol{\tau}, \mathbf{s}_0) - w\cdot\epsilon(\mathbf{c}^k, k) \quad (4)$$

w is a weight that controls the strength of the conditioning. To learn both ϵ_c and ϵ, the conditioning input is randomly dropped out at training time with probability p_{drop}. As $\mathbf{s}_0$ is the first state of $\boldsymbol{\tau}$, we combine the two variables when implementing the model into a single conditioning on $\boldsymbol{\tau}$.

When recovering, we sample N_R pairs of $(\boldsymbol{\tau}, \mathbf{c})$ from M_R, conditioned on the OOD $\mathbf{s}_R$. We recover using the mode $\mathbf{c}_R$ with the highest likelihood sum. We use the final state of the highest likelihood trajectory with contact mode $\mathbf{c}_R$ as $\mathbf{s}_g$. As shown in Fig. 2, this effectively generates a trajectory optimization problem using M_R. Fig. 2 also includes an example recovery trajectory in the Diffused Recovery block. While we still run trajectory optimization online to improve constraint satisfaction, we avoid expensive contact mode selection and obtain initializations that allow us to use a smaller optimization budget.

V. Experimental Results

We evaluate on a valve turning task and a screwdriver turning task in simulation and on hardware. We use an Allegro hand with position-based PD control. We use CSVTO [17] for trajectory optimization. We use CSVTO to generate task model training data, as in [1], and use the same diffusion-initialized trajectory optimization for task execution. In all tasks, the pose of the hand is fixed. All simulations are implemented in Isaac Gym [37]. All methods are evaluated for 50 trials unless otherwise specified. Each trial corresponds to a full episode of task execution, which can include multiple recoveries and has a maximum length of 100 timesteps.

A. Ablations and Baselines

We compare D-TOUR against the following baselines and ablations to evaluate the quality of our recovery behavior and the effect recovery has on overall task performance. We baseline against Recovery RL [9], an existing safe reinforcement learning method. This method learns a function $Q_{risk}(\mathbf{s}, \mathbf{u})$ which predicts the probability that executing action $\mathbf{u}$ at state $\mathbf{s}$ will lead to violating a pre-defined constraint. Recovery is initiated when the prediction exceeds a threshold ϵ_{risk}, and is executed using a model predictive control (MPC) method with Q_{risk} as the cost, as done in [9]. We use MPPI [38], a sampling-based MPC method, as our controller for this baseline and use Isaac Gym as the dynamics model. We train Q_{risk} offline on a dataset generated by executing the task, with perturbation, as in [9], but with no recovery. We compare against the method, No Recovery, which generates this dataset to motivate the use of recovery behavior. We do not update Q_{risk} online to evaluate its ability to generalize in comparison to the diffusion likelihood.

For the first ablation, Likelihood MPPI, we detect recovery scenarios using (3) and execute recovery behaviors using MPPI with the likelihood as the cost. With this, we evaluate the efficacy of our trajectory optimization approach for generating recovery behavior, including our ID projection method. For the second ablation, Contact Mode MLP, we evaluate the utility of diffusing $\mathbf{c}_R$ with M_R. We train an MLP to predict $\mathbf{c}_R \in \mathbb{R}^{n_f}$ given $\mathbf{s}_0$, posed as multi-label binary classification. We sample recovery trajectories from a diffusion model conditioned on $\mathbf{s}_0$ and $\mathbf{c}_R$, as in [1]. This is a different diffusion model from M_R but is also trained on D_R as is the MLP.

We also evaluate an RL method and a contact-implicit MPC (CI-MPC) method designed to remove the need for explicit contact reasoning or reasoning about task execution vs. recovery modes. We evaluate HORA [18], which learns a model-free policy for in-hand object re-orientation, and IDTO [7], which performs CI-MPC on multiple platforms, including an Allegro hand. Importantly, to align closer to the original implementations and take advantage of their ability to quickly react to disturbances, we run these methods at a higher frequency than the other methods. We run these methods at 20x the frequency in simulation, meaning we step the simulation for 1/20 the time before re-planning.

B. Metrics

To evaluate recovery performance, we evaluate three metrics. The first, Recovery Success, measures how often the recovery returns the OOD $\mathbf{s}_R$ to an ID state, combining consecutive recovery trajectories. The second, Recovery Drop, measures how often the robot drops the object during recovery. The third, Recovery Timeout, measures how often the episode ends while recovery behavior is being executed. We also evaluate on task performance metrics, which are specific to each task. In addition, we compare the recovery planning time taken by our offline data generation and M_R.

C. Simulated Valve Turning

In this task, shown in Fig. 5a, the robot turns a simulated valve clockwise using the thumb, index, and middle fingers. The low friction small surface area of the valve necessitate precise finger placement to execute turning. The position of the valve is fixed and it rotates in 1 dimension. $\mathbf{o}$ is the valve's orientation. To execute the task, we sequence primitives which each attempt to turn the valve $-\frac{\pi}{4}$ radians while perturbing executed actions. A $-\frac{\pi}{4}$ rotation may not be achievable due to perturbations and execution error. The overall task goal is to turn the valve by $-\frac{\pi}{3}$ radians from the

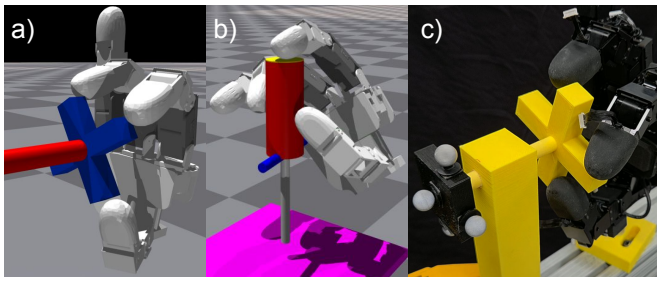

Fig. 5. **a)** Simulated valve. **b)** Simulated screwdriver. **c)** Hardware valve.

random initial configuration. During execution, we perturb the actions with Gaussian noise with standard deviation .03, further increasing the challenge of the fine manipulation task.

We use three recovery modes, where each mode resets 1 finger while the other 2 remain in contact with the valve. For this task, the task performance metric measures distance to the $\frac{\pi}{3}$ rad. goal. We run the No Recovery baseline for 179 trials. The Recovery RL constraint is violated if any finger breaks contact with the valve.

D. Simulated Screwdriver Turning

In this task, shown in Fig. 5b, the robot turns a simulated precision screwdriver clockwise using the thumb, index, and middle fingers. The base of the screwdriver is fixed to the table but is free to rotate in 3 dimensions, simulating driving a screw in a slot. $\mathbf{o}$ is the 3D orientation of the screwdriver, parameterized by roll, pitch, and yaw Euler angles. The blue stalk on the screwdriver is for visualization only, meaning the screwdriver is symmetric about its yaw-axis. The goal of the task is to turn the screwdriver clockwise as far as possible. To do this, we sequence primitives that each attempt to turn the screwdriver by $-\frac{\pi}{2}$ radians, though this may not be achieved due to environmental perturbations. We randomly initialize the object configuration and perturb the system during task execution by applying random wrenches to the screwdriver.

In simulation, we apply a random wrench at each action step with probability $\frac{1}{3}$. When a wrench is applied, we first randomly sample a point on the screwdriver body and apply a force perpendicular to the surface of the screwdriver, with a random rotation about the yaw axis of the screwdriver in the range of $[-\frac{\pi}{10}, \frac{\pi}{10}]$. Force magnitudes are uniformly randomly sampled from the range $(0, 1.5)$.

We use two recovery modes: (1) in which the thumb and middle fingers are reset, and (2) in which the index finger is reset. Non-resetting fingers remain in contact during recovery. These recovery modes are inspired by human use of precision screwdrivers. We account for the screwdriver's yaw-axis symmetry by enforcing that $\mathbf{s}_g$ has the same yaw as $\mathbf{s}_R$ before initiating recovery. This focuses recovery on the non-symmetric components of the state that have an effect on future task performance. For this task, we run the No Recovery baseline for 130 trials to generate data for the Recovery RL baseline. The constraint for Recovery RL is violated if the screwdriver is dropped.

E. Hardware Experiments

We run 10 trials of D-TOUR, Recovery RL, and Likelihood MPPI on hardware versions of the screwdriver task shown in Fig. 1. We use a Vicon motion capture system to estimate $\mathbf{o}$. To avoid issues with hardware overheating for the screwdriver task as a result of the higher forces needed for manipulation, we run each episode for a maximum of 50 steps. As in simulation, we randomly sample wrench perturbations with probability $\frac{1}{3}$ each timestep. If a perturbation is applied, the location of the perturbation is randomly chosen as either pushing the screwdriver toward the palm or perpendicular to the palm, with the applied force pointing toward the thumb. The perturbation is executed by a human (paper author) poking the screwdriver with a wooden rod. For the valve task, we run 20 trials of the same methods and do not add Gaussian perturbation to the actions.

	Method	Recovery Success ↑	Recovery Drop ↓	Recovery Timeout ↓	Task Metric (rad.) ↓
Sim.Valve	D-TOUR (Ours)	93.0%	-	7.0%	$\mathbf{.20 \pm .06}$
	D-TOUR-MLP	85.0%	-	15.0%	$.38 \pm .15$
	Recovery RL [9]	**94.0%**	-	**6.0%**	$1.57 \pm .20$
	Likelihood MPPI	0%	-	100%	$.75 \pm .15$
	No Recovery	-	-	-	$.67 \pm .14$
	HORA [18]	-	-	-	$.32 \pm .08$
	IDTO [7]	-	-	-	$.51 \pm .19$
Sim. Screwdriver	D-TOUR (Ours)	**78.9%**	16.8%	4.3%	$\mathbf{-1.51 \pm .27}$
	D-TOUR-MLP	78.5%	**16.3%**	5.1%	$-1.42 \pm .24$
	Recovery RL [9]	69.7%	27.3%	3.0%	$-1.19 \pm .09$
	Likelihood MPPI	3.9%	96.1%	**0.0%**	$-.46 \pm .05$
	No Recovery	-	-	-	$-1.15 \pm .08$
	HORA [18]	-	-	-	$-.46 \pm .24$
	IDTO [7]	-	-	-	$-.62 \pm .08$
Hardware Valve	D-TOUR (Ours)	89.3%	-	10.7%	$\mathbf{.18 \pm .10}$
	Recovery RL [9]	**93.2%**	-	**6.8%**	$1.3 \pm .17$
	Likelihood MPPI	0%	-	100%	$.77 \pm .20$
Hardware Screwdriver	D-TOUR (Ours)	**63.2%**	**36.8%**	**0%**	$\mathbf{-1.08 \pm .58}$
	Recovery RL [9]	-	-	-	$-.55 \pm .35$
	Likelihood MPPI	0%	100%	**0%**	$-.58 \pm .1$

TABLE I

95% CONFIDENCE INTERVAL FOR TASK METRIC. DROPPING IS NOT POSSIBLE FOR THE VALVE TASK. TASK METRIC: DISTANCE TO GOAL FOR VALVE TASK, RAD. TURNED FOR SCREWDRIVER.

F. Results

As shown in Table I, D-TOUR leads to better task performance than the baselines, with comparable to better performance to the ablation. Our results suggest recovery improves task performance as our method outperforms the No Recovery baseline on both tasks. We find that even though the MLP Ablation is capable of recovering, it can sometimes predict contact modes not seen in D_R. We find that the ablation predicts contact modes for the valve without any resetting fingers even though all contact modes in D_R have a finger being reset. D-TOUR only selects contact modes that appear in D_R and the task performance metrics suggest it learns a more useful recovery trajectory distribution.

The Recovery RL baseline has a high recovery success rate for the valve task, indicating that Q_{risk} can be used to guide MPPI. However, this high recovery rate does not translate to task performance. This is possibly due to the poor generalization of Q_{risk} or its failure to encode knowledge of task performance, unlike the likelihood. The sensitivity of Q_{risk} combined with the difficulty of sampling contact-constraint satisfying trajectories in MPPI leads it to drop the screwdriver more than D-TOUR.

We believe the likelihood, while useful to detect recovery,

is less useful as a cost. This is reasonable as diffusion models learn highly non-linear functions to maximize the likelihood. Sampling based MPC function can struggle to optimize given this cost landscape, leading to a high drop rate for the screwdriver task. This motivates the use of our trajectory optimization with ID projected goal.

We find that HORA learns a useful policy for the valve task, but struggles with the screwdriver task. This could be due to the increased complexity of the screwdriver task, as maintaining stable contact while experiencing external disturbances is important to prevent catastrophic task failure. As RL methods rely on random sampling for exploration, this can lead to breaking contact and dropping the object.

While IDTO is capable of completing the valve task, it can fall into local minima that cause task failure. This leads to the higher distance to goal and larger variance in the goal distance. While D-TOUR also uses non-convex trajectory optimization, we impose stricter constraints than IDTO that come from our diffused contact modes. This leads to a smaller search for the trajectory optimization, which can lead to overall better solutions. For the screwdriver task, we find that IDTO's performance, both quantitatively on task performance and qualitatively on how likely optimized trajectories are to contain contact switches, is sensitive to the tuning of parameters, and the parameter tuning that yielded the best overall task performance was observed to perform very little contact switching. This suggests that choosing contact switches when executing a sensitive task under perturbation is a challenging problem, which benefits from explicitly planning contact modes.

We also find that D-TOUR plans resets of the thumb and middle fingers without which the robot may eventually reach kinematic limits and be unable to turn further, allowing us to not only recover from perturbations but also naturally extend task execution for repetitive tasks. Along with recovering from execution error in the valve task, we find D-TOUR can improve execution in situations where the initial grasp of the object is not conducive to turning, as in Fig. 5a. These states have low likelihoods under the task model, triggering recovery behavior to states from which turning is easier.

D-TOUR turns the screwdriver further than the baselines for the hardware task. Recovery RL fails to detect that recovery is necessary, possibly due to poor generalization of Q_{risk} leading to poor sim-to-real transfer. In contrast, both the task model for detection and the trajectory optimization diffusion model for recovery generalize better to hardware. This is potentially due to the diffusion model being exposed to noised data in training, improving generalization. Likelihood MPPI detects recovery as it uses the same detection as D-TOUR, but fails to recover without dropping the screwdriver. We observe a higher drop percentage and lower recovery rate in hardware when compared to simulation, potentially due to the sim-to-real gap. Specifically, there could be error in the modeling of the robot and object in terms of their geometries and physical properties such as friction and joint damping. D-TOUR achieves lower distance to goal and better recovery metrics for the hardware valve task. We achieve comparable results between simulation and hardware for D-TOUR, showing less sensitivity to potential geometry or physical property mismatch for this task.

D-TOUR also enables faster planning. For the valve task, planning a recovery trajectory takes 90.4 s for the valve task and 22.7 s for the screwdriver task during offline data generation, with additional trajectory optimization time during task execution of 8.5 s for the valve and 9.5 s for the screwdriver. However, planning a recovery trajectory with M_R takes 7.0 s for the valve task and 8.2 s for the screwdriver task, with additional trajectory optimization time during task execution of 9.0 s for the valve and 11.1 s for the screwdriver. For both tasks we train M_R for approximately 2 hours using an Nvidia RTX 4090 GPU.

VI. Conclusion

We presented D-TOUR, a method to generate contact-rich recovery behaviors informed by diffusion model likelihood estimation. Our method uses a diffusion model trained on task data to detect when perturbations or execution error require recovery behaviors and generates a dataset of these recovery trajectories. We train a diffusion model on this dataset to more efficiently generate and solve recovery trajectory optimization problems online. We show our method outperforms a reinforcement learning baseline and methods that do not explicitly reason about contact interactions, including on a hardware screwdriver turning task. In future work, we plan on combining our work on planning and trajectory optimization with feedback control methods to improve execution time as well as relaxing the assumption of a prescribed set of contact modes by inferring the set from human demonstrations or attempting to learn a set of useful modes from object interaction.

Appendix

A. Trajectory Optimization

Our trajectory optimization formulation builds on prior work from [1], [2]. We partition the trajectory into $\boldsymbol{\tau} = \{\boldsymbol{\tau_c}, \boldsymbol{\tau_r}, \boldsymbol{\tau_o}\}$, where $\boldsymbol{\tau_c}$ is the trajectory for contact fingers, or fingers with contact mode 1, $\boldsymbol{\tau_r}$ is the trajectory for fingers being reset, or fingers with contact mode 0, and $\boldsymbol{\tau_o}$ is the trajectory for the object. Our full trajectory optimization problem is written as:

$$\min_{\substack{\mathbf{s}_1,\mathbf{s}_2,\cdots,\mathbf{s}_H;\\ \mathbf{u}_1,\mathbf{u}_2,\cdots,\mathbf{u}_H}} J_g(\boldsymbol{\tau_o}) + J_r(\boldsymbol{\tau_r}, \boldsymbol{\tau_o}) + J_{smooth}(\boldsymbol{\tau}) \tag{5}$$

$$\text{s.t.} \quad \mathbf{q}_{min} \leq \mathbf{q}_t \leq \mathbf{q}_{max} \tag{6}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_t \leq \mathbf{u}_{max} \tag{7}$$

$$f_{contact}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}) = 0 \tag{8}$$

$$f_{kinematics}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{s}_{c,t+1}, \mathbf{s}_{o,t+1}) = 0 \tag{9}$$

$$f_{balance}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{s}_{c,t+1}, \mathbf{s}_{o,t+1}, \mathbf{u}_{c,t}, \mathbf{u}_{o,t}) = 0 \tag{10}$$

$$f_{friction}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{u}_{c,t}) \leq 0 \tag{11}$$

$$f_{min\_f}(\mathbf{s}_{c,t}) \geq f_{min} \tag{12}$$

$$f_{contact}(\mathbf{s}_{r,t}, \mathbf{s}_{o,t}) \leq -\delta, \quad t < H \tag{13}$$

$$f_{contact}(\mathbf{s}_{r,H}, \mathbf{s}_{o,H}) = 0 \tag{14}$$

$$f_{contact\_patch}(\mathbf{s}_{r,H}, \mathbf{s}_{o,H}) \leq r \tag{15}$$

$$\mathbf{q}_{r,t} + \Delta\mathbf{q}_{r,t} - \mathbf{q}_{r,t+1} = 0 \tag{16}$$

The cost term J_g encourages the object configuration to reach $\mathbf{o}_g$, or the object state component of $\mathbf{s}_g$. $\mathbf{s}_g$ is the final state of the highest likelihood trajectory with diffused contact mode $\mathbf{c}_R$. J_r incentivizes trajectories to use the same contact point on the robot after resetting as used in $\mathbf{s}_g$. J_{smooth} incentivizes a smooth trajectory and is the same for all contact modes.

(6), (7) specify bounds on the robot configurations and actions respectively and are the same for all contact modes. (8), (9), (10), (11), (12) are the same as those in [2] and are used for the contact fingers. (8) ensures the finger stays in contact during the trajectory, (9) ensures the velocity of the contact point on the robot equals the velocity of the contact point on the object, (10) ensures the applied forces and gravity are balanced, (11) ensures applied forces lie within a Coulomb friction cone, and (12) specifies a minimum force magnitude applied by each finger which is useful for overcoming unmodeled friction or other reaction forces.

(13), (14), (15), (16) are used for the fingers being reset. (13), (14), (16) are the same as in [1]. (13) ensures that the fingers being reset avoid contact with a distance threshold δ until the final time step. (14) ensures that at the final step, the fingers are in contact. (16) ensures that configurations and actions are consistent for fingers not in contact. In this work we add an additional constraint (15), which specifies where on the object each finger should make contact when reset. We constrain the contact point to be within a radius r of the closest point on the object for each finger in $\mathbf{s}_g$.

REFERENCES

[1] A. Kumar, T. Power, F. Yang, S. A. Marinovic, S. Iba, R. S. Zarrin, and D. Berenson, "Diffusion-informed probabilistic contact search for multi-finger manipulation," *arXiv preprint arXiv:2410.00841*, 2024.

[2] 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," *IEEE RA-L*, 2025.

[3] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *T-RO*, 2023.

[4] Y. Jiang, M. Yu, X. Zhu, M. Tomizuka, and X. Li, "Contact-implicit model predictive control for dexterous in-hand manipulation: A long-horizon and robust approach," *IROS*, 2024.

[5] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, 2023.

[6] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang, "Lessons from learning to spin "pens"," *CoRL*, 2024.

[7] V. Kurtz, A. Castro, A. Ö. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *IJRR*, 2023.

[8] S. Zhou, Y. Du, S. Zhang, M. Xu, Y. Shen, W. Xiao, D.-Y. Yeung, and C. Gan, "Adaptive online replanning with diffusion models," *NeurIPS*, 2023.

[9] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *RA-L*, 2021.

[10] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," *ICRA*, 2019.

[11] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing robots with longer-term autonomy by recovering from external disturbances in manipulation through grounded anomaly classification and recovery policies," *Journal of Intelligent & Robotic Systems*, 2021.

[12] A. Reichlin, G. L. Marchetti, H. Yin, A. Ghadirzadeh, and D. Kragic, "Back to the manifold: Recovering from out-of-distribution states," *IROS*, 2022.

[13] T. Matsuoka, T. Hasegawa, and K. Honda, "A dexterous manipulation system with error detection and recovery by a multi-fingered robotic hand," *IROS*, 1999.

[14] S. Vats, D. K. Jha, M. Likhachev, O. Kroemer, and D. Romeres, "Recoverychaining: Learning local recovery policies for robust manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2410.13979

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, 2020.

[16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *ICLR*.

[17] T. Power and D. Berenson, "Constrained stein variational trajectory optimization," *T-RO*, 2024.

[18] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," *CoRL*, 2023.

[19] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint, "Structured deep generative models for sampling on constraint manifolds in sequential manipulation," *CoRL*, 2021.

[20] H. Huang, B. Sundaralingam, A. Mousavian, A. Murali, K. Goldberg, and D. Fox, "Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning," *CoRL*.

[21] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *ICML*, 2022.

[22] K. Kondo, A. Tagliabue, X. Cai, C. Tewari, O. Garcia, M. Espitia-Alvarez, and J. P. How, "Cgd: Constraint-guided diffusion policies for uav trajectory planning," *arXiv preprint arXiv:2405.01758*, 2024.

[23] J. Carvalho, A. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," *IROS*, 2023.

[24] C. Pan, Z. Yi, G. Shi, and G. Qu, "Model-based diffusion for trajectory optimization," *NeurIPS*, 2024.

[25] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, "Adaptdiffuser: diffusion models as adaptive self-evolving planners," *ICML*, 2023.

[26] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," *CoRL*, 2023.

[27] U. A. Mishra, Y. Chen, and D. Xu, "Generative factor chaining: Coordinated manipulation with diffusion-based factor graph," *CoRL*, 2024.

[28] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu, "Diffusion-based generation, optimization, and planning in 3d scenes," *CVPR*, 2023.

[29] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M 2 diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[30] X. Li, T. Zhao, X. Zhu, J. Wang, T. Pang, and K. Fang, "Planning-guided diffusion policy learning for generalizable contact-rich bimanual manipulation," *CoRR*, 2024.

[31] L. Yang, H. Suh, T. Zhao, B. P. Graesdal, T. Kelestemur, J. Wang, T. Pang, and R. Tedrake, "Physics-driven data generation for contact-rich manipulation via trajectory optimization," *arXiv preprint arXiv:2502.20382*, 2025.

[32] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," *ICRA*, 2023.

[33] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," *ICRA*, 2021.

[34] Z. Weng, H. Lu, D. Kragic, and J. Lundell, "Dexdiffuser: Generating dexterous grasps with diffusion models," *IEEE RA-L*, 2024.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[36] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[37] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[38] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *T-RO*, 2018.