Probabilistic Contact Mode Planning for Multi-Finger Manipulation Using Diffusion Models

Thomas Power[*1], **Abhinav Kumar**[*1], **Fan Yang**[1], **Sergio Aguilera Marinovic**[2], **Soshi Iba**[2], **Rana Soltani Zarrin**[2], **Dmitry Berenson**[1]
[1] University of Michigan, [2] Honda Research Institute USA.

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. Our method uses 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.

1 Introduction

Multi-finger manipulation is challenging as there are many ways in which the hand can make or break contact with manipulated objects. 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. However, domain knowledge is valuable in improving task performance in multi-finger manipulation, which has sensitive constraints relating to contact. In addition, learning methods generate policies similar to the 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.

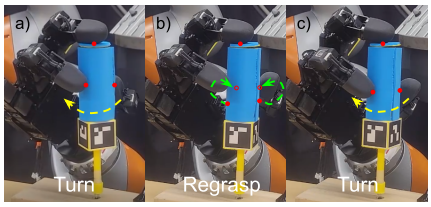


Figure 1: DIPS plans contact interactions that turn the screwdriver from **a)** to **b)**, where it is regrasped to allow for further turning in **c)**. Contact points are shown in red, with empty circles for target contacts. The yellow arrows show screwdriver turning and green show finger motion.

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 [3] 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. 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.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

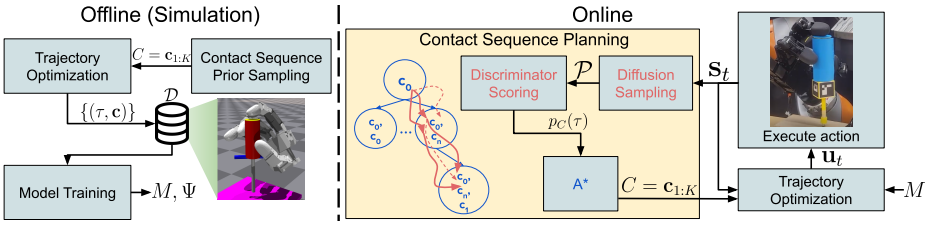


Figure 2: **Offline**, we train a diffusion model M and discriminator Ψ on dataset $\mathcal{D}$ collected in simulation. **Online**, we plan a contact sequence C given a state $\mathbf{s}_t$. We expand nodes in blue corresponding to contact mode sequences and inform the search using a distribution $p_C(\boldsymbol{\tau})$ parameterized with a set of trajectories $\mathcal{P}$ in pink. We diffuse trajectories conditioned on the child node's contact mode and evaluate them with Ψ. The dotted lines are samples discarded in the resampling used to update $p_C(\boldsymbol{\tau})$. Given a single contact mode and $\mathbf{s}_t$, 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.

Our contributions are twofold. First, we present a method for planning contact mode sequences using graph search informed by a diffusion model. Secondly, we propose an approach for 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 baselines that do not plan contact sequences and that do not reason about the variability in the diffusion model output.

2 Related Work

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 [4]. Other recent work has used tree-based planners to explore contact modes combined with contact-implicit trajectory optimization [5, 6]. There have been several recent works combining A* search with trajectory optimization [7, 8, 9, 10]. Unlike previous work combining A* and trajectory optimization, we accelerate our planning using a learned model to approximate trajectory optimization.

In recent years learning-based methods have been increasingly popular for solving contact-rich tasks. These include methods that learn models for planning [11, 12], as well as methods that learn policies using reinforcement learning [13, 14, 15, 16]. 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 [17, 18]. 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 [19, 20, 21, 22].

3 Problem Statement

We consider the problem of contact-rich manipulation with a multi-fingered hand. Our goal is to find a sequence of robot configurations $\mathbf{q}_{1:T}$, that successfully manipulate an object from start $\mathbf{o}_0$ to goal $\mathbf{o}_G$. 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 $\mathbf{c}$, we solve a trajectory optimization problem for horizon $H < T$.

We denote 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 $\boldsymbol{\tau} := \{\mathbf{s}_{1:H}, \mathbf{u}_{1:H}\}$. To allow for the making and breaking of contact, we split the trajectory into K segments, i.e. $H = \frac{T}{K}$, and $\boldsymbol{\tau} = \{\boldsymbol{\tau}_1, ..., \boldsymbol{\tau}_K\}$. The aim is then to find a sequence of contact modes $C = \mathbf{c}_{1:K}$, with $\boldsymbol{\tau}_k$ being the solution to the trajectory optimization with contact mode $\mathbf{c}_k$ for $k \in [1, ..., K]$, such that the overall trajectory $\boldsymbol{\tau}$ results in completing the task. We assume we can generate a dataset of N trajectories, $\mathcal{D} = \{\boldsymbol{\tau}_i, \mathbf{c}_i\}_{i=1}^N$ consisting of solutions to the trajectory optimization problem for diverse initial configurations.

4 Methods

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 $\boldsymbol{\tau}_k$. 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.

Trajectory Optimization: Our trajectory optimization formulation is based on prior work by Yang et al. [23]. The formulation in [23] 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. More details of the trajectory optimization are given in Appendix A. To solve the trajectory optimization we use Constrained Stein Variational Trajectory Optimization (CSVTO) [24]. 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.

Diffusion Model Training: To aid in contact sequence planning, we train a diffusion model $M(\mathbf{c}, \mathbf{s}_0)$. We can use M to sample from the distribution $p(\boldsymbol{\tau}|\mathbf{c}, \mathbf{s}_0)$. $p(\boldsymbol{\tau}|\mathbf{c}, \mathbf{s}_0)$ 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. Further details of the diffusion model are given in Appendix B.

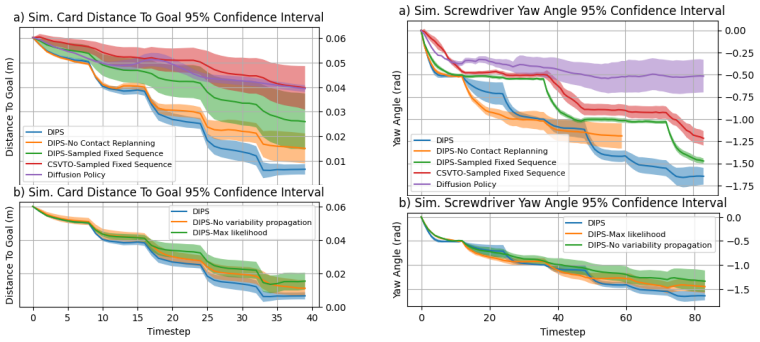
As we are working with complex systems with high degrees of freedom and complex constraints, M may diffuse unrealistic trajectories. We compute weights of trajectories in our variability propagation method that represent their realism to account for this. We train a discriminator $\Psi(\boldsymbol{\tau}, \mathbf{c})$ that takes in a trajectory and contact mode and outputs the probability that $\boldsymbol{\tau}$ 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.

Probabilistic Contact Sequence Search: To find the contact mode sequence, we construct an A* tree search problem where each node $\mathbf{n}$ in the tree corresponds to a contact mode sequence C. The descendants 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 a set of trajectories conditioned on C to compute costs for A* and check if $\mathbf{o}_G$ has been achieved in the search. More details are given in Appendix C.

5 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 [25].

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 [23]; (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 pick the highest-scoring sampled trajectory when expanding. We also baseline our method against Diffusion Policy [2], trained using the same demonstrations. More details on the tasks and results are given in Appendix D.



(i) Simulated Card (ii) Simulated Screwdriver

Figure 4: Simulation results over 10 trials for screwdriver turning and card experiments

Simulated Tabletop Card Manipulation: As shown in Fig. 4i(a,b), DIPS outperforms the baselines and ablations by avoiding unneeded regrasps. We come within .6 cm of $\mathbf{o}_G$ compared with 2.6 cm for DIPS-Sampled Fixed Sequence and 4 cm for both CSVTO-Sampled Fixed Sequence and Diffusion Policy. 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.

Simulated Screwdriver Turning: As shown in Fig. 4ii(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 our method can perform plan contact sequences not seen in the data, such as executing multiple turns in a row, or not performing unnecessary regrasps. Our method can generate trajectories that outperform the data on which M is trained.

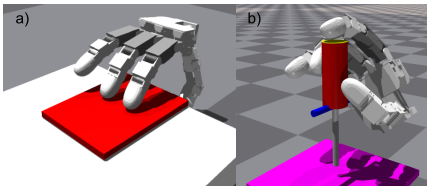


Figure 3: **a)** Simulated card and **b)** Simulated screwdriver environments. The blue valve in b) is for visualization only and has no collision geometry.

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, Ψ. $\mathbf{o}_t$ is estimated using Aruco tags on the screwdriver. As shown in Fig. 5, 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.

6 Conclusion

We presented DIPS, a planning method for contact-rich manipulation that combines a learned diffusion model with A* search that accounts for variability in the diffusion sampling process.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.

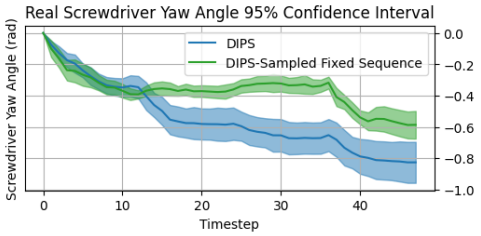


Figure 5: Real screwdriver manipulation results for valid executions over 10 trials. DIPS (10/10 valid) outperforms the ablation (8/10 valid) by 41%.

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] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.

[4] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason. Enhancing dexterity in robotic manipulation via hierarchical contact exploration. *RA-L*, 9(1):390–397, 2024.

[5] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg. Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. *IROS*, pages 8262–8268, 2021.

[6] 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.

[7] R. Natarajan, H. Choset, and M. Likhachev. Interleaving Graph Search and Trajectory Optimization for Aggressive Quadrotor Flight. *IEEE Robotics and Automation Letters*, 6(3):5357–5364, July 2021. ISSN 2377-3766. doi:10.1109/LRA.2021.3067298. Conference Name: IEEE Robotics and Automation Letters.

[8] 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.

[9] 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].

[10] 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, July 2024. arXiv:2407.08848 [cs].

[11] V. Kumar, A. Gupta, E. Todorov, and S. Levine. Learning Dexterous Manipulation Policies from Experience and Imitation, Nov. 2016. arXiv:1611.05095 [cs].

[12] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep Dynamics ModeLearning Complexs for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

[13] 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.

[14] 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.

[15] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak. Generalization in Dexterous Manipulation via Geometry-Aware Multi-Task Learning, 2021.

[16] 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*, 2023.

[17] 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, June 2018. arXiv:1709.10087 [cs].

[18] 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*, 43(4):513–532, 2024.

[19] 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)*, pages 8855–8862, 2020.

[20] 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)*, pages 6551–6557, 2024. doi:10.1109/ICRA57147.2024.10611300.

[21] R. S. Zarrin, R. Jitosho, and K. Yamane. Hybrid learning-and model-based planning and control of in-hand manipulation. In *IROS*, pages 8720–8726. IEEE, 2023.

[22] 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*, pages 5970–5976, 2023.

[23] 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.

[24] T. Power and D. Berenson. Constrained stein variational trajectory optimization. *IEEE Transactions on Robotics*, 2024.

[25] 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.

[26] 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.

[27] J. Ho and T. Salimans. Classifier-free diffusion guidance, 2022.

A Trajectory Optimization with a Given Contact Mode

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 ith 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 partition the trajectory into $\boldsymbol{\tau} = \{\boldsymbol{\tau}_c, \boldsymbol{\tau}_r, \boldsymbol{\tau}_o\}$. Our trajectory optimization problem is then written as

$$
\begin{aligned}
&\min_{\substack{\mathbf{s}_1,\mathbf{s}_2,\cdots,\mathbf{s}_H;\\ \mathbf{u}_1,\mathbf{u}_1,\cdots,\mathbf{u}_H}} J_g(\boldsymbol{\tau_o}) + + J_r(\boldsymbol{\tau_r}, \boldsymbol{\tau_o}) + J_{smooth}(\boldsymbol{\tau}) \\
&\text{s.t.} \quad \mathbf{q}_{min} \leq \mathbf{q}_t \leq \mathbf{q}_{max} \\
&\mathbf{u}_{min} \leq \mathbf{u}_t \leq \mathbf{u}_{max} \\
&f_{contact}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}) = 0 \\
&f_{kinematics}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{s}_{c,t+1}, \mathbf{s}_{o,t+1}) = 0 \\
&f_{balance}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{s}_{c,t+1}, \mathbf{s}_{o,t+1}\mathbf{u}_{c,t}, \mathbf{u}_{o,t}) = 0 \\
&f_{friction}(\mathbf{s}_{c,t}, \mathbf{s}_{o,t}, \mathbf{u}_{c,t}) \leq 0 \\
&f_{contact}(\mathbf{s}_{r,t}, \mathbf{s}_{o,t}) \leq -\delta, t < H \\
&f_{contact}(\mathbf{s}_{r,H}, \mathbf{s}_{o,H}) = 0 \\
&\mathbf{q}_{r,t} + \Delta\mathbf{q}_{r,t} - \mathbf{q}_{r,t+1} = 0.
\end{aligned} \tag{1}
$$

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 [23]. $f_{contact} \leq -\delta$ ensures that the regrasping fingers avoid contact with a threshold δ 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.

B Diffusion Model Details

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, 26] for the diffusion model. We use a U-Net with a Sigmoid activation output layer for the discriminator Ψ. 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 [27] to condition on a specific contact mode $\mathbf{c}$. To condition on $\mathbf{s}_0$, 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.

C Probabilistic Contact Sequence Search

Algorithm 1: `propagate_variability`

Given $\mathbf{n}^p = (\mathcal{P}^p, S^p, C^p)$, $\mathbf{c}'$, M, Ψ, k, γ
$\bar{\boldsymbol{\tau}} \leftarrow$ Diffuse k trajectories from $M(\mathbf{c}', \mathcal{P}^p_H)$
$d \leftarrow \texttt{depth}(\mathbf{n}^p)$
$S \leftarrow \gamma^d \cdot \Psi(\bar{\boldsymbol{\tau}}, \mathbf{c}') + S^p$
$\bar{S} \leftarrow \{S_i / \sum_{S_i \in S} S_i \mid \forall S_i \in S\}$
$\mathcal{P} \leftarrow k$ samples from $\bar{\boldsymbol{\tau}}$ given probabilities $\bar{S}$
$C \leftarrow C^p \cup \mathbf{c}'$
$\mathbf{n} = (\mathcal{P}, S, C)$
return $\mathbf{n}$

As opposed to prior work [8] that uses a single trajectory at each node, we model a distribution $p_C(\boldsymbol{\tau})$ at each node to reason about variability in diffusion sampling. We parameterize $p_C(\boldsymbol{\tau})$ 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 $\Psi(\boldsymbol{\tau}, \boldsymbol{c})$. The full definition of a node is $\mathbf{n} = (\mathcal{P}, S, C)$. By using this particle-based representation, we can approximate $p(\boldsymbol{\tau}|\mathbf{c}, \mathbf{s}_0)$ with $p_C(\boldsymbol{\tau})$.

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 $\bar{\boldsymbol{\tau}}$ 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 $\mathcal{P}^p_H$ to enforce continuity. We compute the weights S for $\bar{\boldsymbol{\tau}}$ using Ψ. We accumulate the scores as we expand the tree, discounted by a factor γ^d, where d is the depth of the node in the search tree. We sample k trajectories from $\bar{\boldsymbol{\tau}}$ 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 define our cost-to-come using the CSVTO cost J as $g(p_C(\boldsymbol{\tau}), C) = \mathbb{E}_{\boldsymbol{\tau} \sim p_C(\boldsymbol{\tau})}[J(\boldsymbol{\tau}, C)]$. 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 also use a terminal cost $\phi(\boldsymbol{\tau})$ 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. We enforce a minimum probability p_{min} for transitions to address this.

In our heuristic, shown in (2), 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.

$$h(p_C(\boldsymbol{\tau}), \mathbf{c}_{1:N}) = \alpha \cdot \mathbb{E}_{\boldsymbol{\tau} \sim p_C(\boldsymbol{\tau})}[\phi(\boldsymbol{\tau})] - \beta \cdot \left[\log(p(\mathbf{c}_0)) + \sum_{n=1}^{N} \log(p(\mathbf{c}_n|\mathbf{c}_{n-1})) \right] \tag{2}$$

D Experimental Details

For each task, we define contact modes, which specify CSVTO objectives and constraints. $\mathbf{o}_G$ 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 $\delta = 0.015\,m, k = 16, \gamma = .9, \alpha = 1 \times 10^4, \beta = 1 \times 10^3$.

D.1 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 θ 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 $\phi(\boldsymbol{\tau})$.

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.

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.

D.2 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 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 $\phi(\boldsymbol{\tau})$, 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 $p_{min} = .1$: $p(\mathbf{c}_0) = [.1 \quad .1 \quad .8]$, $p(\mathbf{c}_n|\mathbf{c}_{n-1}) = .1$ 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.

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.3 Real Screwdriver Turning

For this task, we run DIPS and DIPS-Sampled Fixed 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 [23] 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.

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.