Subgoal Diffuser: Coarse-to-fine Subgoal Generation to Guide Model Predictive Control for Robot Manipulation

Zixuan Huang[1], Yating Lin[1], Fan Yang[1], Dmitry Berenson[1]

***Abstract*— Manipulation of articulated and deformable objects can be difficult due to their compliant and under-actuated nature. Unexpected disturbances can cause the object to deviate from a predicted state, making it necessary to use Model-Predictive Control (MPC) methods to plan motion. However, these methods need a short planning horizon to be practical. Thus, MPC is ill-suited for long-horizon manipulation tasks due to local minima. In this paper, we present a diffusion-based method that guides an MPC method to accomplish long-horizon manipulation tasks by dynamically specifying sequences of subgoals for the MPC to follow. Our method, called Subgoal Diffuser, generates subgoals in a coarse-to-fine manner, producing sparse subgoals when the task is easily accomplished by MPC and more dense subgoals when the MPC method needs more guidance. The density of subgoals is determined dynamically based on a learned estimate of reachability, and subgoals are distributed to focus on challenging parts of the task. We evaluate our method on two robot manipulation tasks and find it improves the planning performance of an MPC method, and also outperforms prior diffusion-based methods.**

More visualizations and results can be found at https://sites.google.com/view/subgoal-diffuser-mpc

I. Introduction

Robotic manipulation of articulated and deformable objects is challenging in part because they are compliant and under-actuated. External forces from the environment, e.g. friction from contact, or other disturbances can cause the actual state of the object to deviate from that predicted by a simulator. Thus, it is necessary to adapt to disturbances quickly when manipulating these objects. Sample-based Model-predictive Control (MPC) methods [1], [2] are a good choice for this application due to their flexibility, as they do not impose stringent requirements on the form of the cost function and dynamics.

However, these methods trade off horizon length in favor of speed, making them ill-suited for long horizon manipulation tasks. This paper presents an approach to robotic manipulation of articulated and deformable objects that overcomes this limitation by using a learned conditional generative model to dynamically predict sequences of subgoals. These subgoals are then used as guidance by the MPC method.

Recent work on learning conditional generative models for manipulation has produced powerful methods based on diffusion [3]. These methods demonstrate the capacity to capture the distribution of states and actions required to produce trajectories for certain manipulation tasks [4], [5], [6]. However, they either output the full trajectory directly [4], [5], [6], or adopt a fixed hierarchical structure [7]. Also, they use a learned policy for low-level control, which can be data-inefficient and does not generalize well to new situations.

In this paper we present a method to generate subgoals using a diffusion model and delegate the task of finding a sequence of controls to move between subgoals to an MPC method. For these subgoals to be effective, we require the ability to produce subgoals at different resolutions (in terms of the number of steps needed to move between them). This is crucial for accomplishing difficult manipulation tasks, as varying levels of guidance are needed at different times. For example, consider the task of picking up an open notebook from the floor and placing it, closed, on a table (Fig. 4). Transporting the notebook to the table may require sparse subgoals and minimal guidance for MPC, whereas placing the notebook down and folding it requires more careful manipulation. In order to generate a sequence of subgoals at an appropriate resolution, we propose a diffusion-based architecture called Subgoal Diffuser. This methodgenerates subgoals in a coarse-to-fine manner. It initially outlines a coarse high-level plan with subgoals spaced far apart and subsequently fills in more subgoals as necessary.

We introduce a reachability-based measure to determine when it is necessary to add more subgoals. The main idea is that more subgoals should be used if adjacent subgoals are not reachable given the low-level MPC controller. Reachability is learned from the same dataset that is used to train the diffusion model. Furthermore, our method uses this learned distance metric to dynamically redistribute the subgoals to focus on the challenging parts of the task. Thus, the contributions of this paper are:

- A diffusion-based framework to generate subgoals in a coarse-to-fine manner.
- A strategy based on an estimate of reachability to determine a suitable subgoal resolution for the task.
- A system that integrates subgoal generation and MPC for robot manipulation.

Our experiments on notebook and rope manipulation show that the generated subgoals effectively prevent the myopic MPC controller from falling into local minima. Our method also compares favorably to existing diffusion-based methods.

II. Related Work

A. Diffusion models for Robotics

Diffusion models have shown great premise in generative modeling, such as images [8] and videos [9]. Recently, researchers have applied diffusion models to various robotics

This work was supported in part by the Office of Naval Research Grant N00014-21-1-2118 and NSF grants IIS-1750489, IIS-2113401, and IIS-2220876. [1] Department of Robotics, University of Michigan, Ann Arbor

applications, such as data augmentation [10], [11], grasp synthesis [12], text-conditioned scene rearrangement [13], [14], constrained trajectory generation [15], and motion planning [16]. In this paper, we focus on robot manipulation. Diffuser [4] and SceneDiffuser [17] propose to jointly model the dense trajectory of states and actions, and draw the connection with standard trajectory optimization techniques. Another line of work [18], [6], [19] explores using diffusion models in the context of imitation learning and only models the distribution of demonstrated actions. Ajay et al. [5] and Li et al. [7] train a state-based diffusion model to predict desirable future states and a low-level policy to reach the predicted states. In contrast, we introduce a hierarchical framework for modeling the distribution of states (subgoals), and a procedure to automatically determine subgoal resolution required for the task. Also, we leverage MPC for low-level control. In our experiments, we show that dynamically deciding the subgoal resolution is critical for the task performance.

B. Subgoal generation for long horizon planning

Prior work has used reachability to decompose long-horizon tasks. Hierarchical Visual Forsights (HVF) [20] proposes to estimate the reachability between adjacent subgoals by explicitly running an MPC method to plan. However, it requires running MPC on multiple start-goal pairs, which is computationally expensive. Other methods [21], [22], [23] leverage learning to estimate reachability. They first train a goal-conditioned policy using reinforcement learning, then they frame subgoal generation as online optimization over the value function of the policy. While effective, like other RL methods, they are sample-inefficient and require online interaction. DiffSkill [24] follows a similar strategy but avoids the caveats of RL by training the policy with demonstrations obtained by trajectory optimization. We propose a way to evaluate the reachability of an MPC controller that does not require demonstrations or online interaction.

C. MPC with a learned prior

Learning a prior distribution of actions and subgoals has been used to speed up MPC and accomplish complex tasks. Power and Berenson [25] leverage normalizing flow for modeling the action distributions. Wang and Ba [26] use a policy network to initialize the action sequences for MPC. Sacks and Boots [27] introduce a framework with a learned optimizer with imitation learning, which makes better use of the expert samples. Similar to us, Li et al. [28] propose an MPC framework with a generator for intermediate waypoints and a discriminator to choose the best waypoint candidate. However, they all require expert demonstration. While some prior works [29], [30], [31] do not require expert demonstrations, they cannot conduct global reasoning. Our proposed method is able to generate subgoals to guide MPC to alleviate local minima, while only using a low-quality offline dataset.

III. Preliminaries

A. Problem Statement

We consider the problem of discrete-time optimal control with state denoted by $\mathbf{x}_t \in \mathbb{R}^{d_\mathbf{x}}$ and control by $\mathbf{u}_t \in \mathbb{R}^{d_\mathbf{u}}$. The state consists of two components: robot state $\mathbf{r}_t \in \boldsymbol{R}$ and object state $\mathbf{o}_t \in \boldsymbol{O}$. After applying the control, the system will transition to the next state with a known transition probability function represented as $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. A trajectory of states is defined as $\boldsymbol{\tau}_\mathbf{x} \triangleq [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}]$, and a trajectory of controls is defined as $\boldsymbol{\tau}_\mathbf{u} \triangleq [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}]$. Thus, the full trajectory is denoted by $\boldsymbol{\tau} = [\boldsymbol{\tau}_\mathbf{x}; \boldsymbol{\tau}_\mathbf{u}]$. Given a cost function $\boldsymbol{J}$, MPC seeks to find a sequence of controls $\boldsymbol{\tau}_\mathbf{u}$ of length H (the planning horizon) that minimizes $\boldsymbol{J}$.

In this paper, we mostly consider the manipulation problem of under-actuated objects, such as rope. Our goal is to find a sequence of robot controls $\boldsymbol{\tau}_\mathbf{u}$ to move the object from the current state $\mathbf{o}_{cur}$ to a the desired configuration $\mathbf{o}_G$. The cost function $\boldsymbol{J}$ is a function that measures distance to the goal state, e.g., Euclidean distance.

We assume the structure of the environment is provided in the form of Signed Distance Function (SDF), and the 3D model of the object is known.

This type of manipulation problem is challenging since the state space is high-dimensional and the object is under-actuated. We do not assume that high-quality demonstrations of the task are available. To tackle this problem, we resort to a data-driven approach where we assume access to an offline dataset $\mathcal{D} \triangleq \{\boldsymbol{\tau}^i\}_{0 \le i < N}$, which contains robot interactions with the target object. The dataset is collected by a random policy. Our goal is to learn a generative model that is able to produce a sequence of subgoals for the object:

$$\boldsymbol{\tau}_\mathcal{G} = [\mathbf{o}_{cur}, \mathbf{o}_{g_1}, \ldots, \mathbf{o}_{g_{M-2}}, \mathbf{o}_G] \tag{1}$$

given current state, goal state, and (optionally) a scene representation. The subgoals will be used to guide a sampling-based MPC method to complete the task. The number of subgoal M is variable and will be automatically estimated based by our method.

B. Diffusion Models

Diffusion models [32], [3] are a powerful class of generative models designed to approximate the data distribution $q(\boldsymbol{\tau}_0)$ from the dataset $\mathcal{D} \triangleq \{\boldsymbol{\tau}^i\}_{0 \le i < M}$. Diffusion models frame the data generation as a K-step iterative denoising procedure, with a predefined forward noising process $q(\boldsymbol{\tau}_{k+1}|\boldsymbol{\tau}_k) = \mathcal{N}(\boldsymbol{\tau}_{k+1}; \sqrt{\alpha_k}\boldsymbol{\tau}_k, (1-\alpha_k)\boldsymbol{I})$, and a learnable reverse denoising process $p_\theta(\boldsymbol{\tau}_k|\boldsymbol{\tau}_{k+1})$. The forward diffusion process can be seen as gradually fusing data with noise, and K and α are hyperparameters that define this noise schedule. The data distribution $p_\theta(\boldsymbol{\tau}_0)$ is expressed as:

$$p_\theta(\boldsymbol{\tau}_0) = \int p(\boldsymbol{\tau}_K) \prod_{k=1}^{K} p_\theta(\boldsymbol{\tau}_{k-1}|\boldsymbol{\tau}_k) d\boldsymbol{\tau}^{1:K} \tag{2}$$

where $p(\boldsymbol{\tau}_K)$ is a unit Gaussian prior. In practice, the data generation process is usually implemented via stochastic Langevin Dynamics [33] starting from Gaussian noise.

While diffusion models can by trained by optimizing the variational lower-bound on $\log p_\theta(\boldsymbol{\tau})$, like prior work [4], [5], [6], we use the simplified objective from DPPM [3]:

$$\mathcal{L}_{denoise}(\theta) = \mathbb{E}_{k \sim [1,K], \boldsymbol{\tau}_0 \sim q, \epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} ||\epsilon - \epsilon_\theta(\boldsymbol{\tau}_k, k)||^2 \tag{3}$$

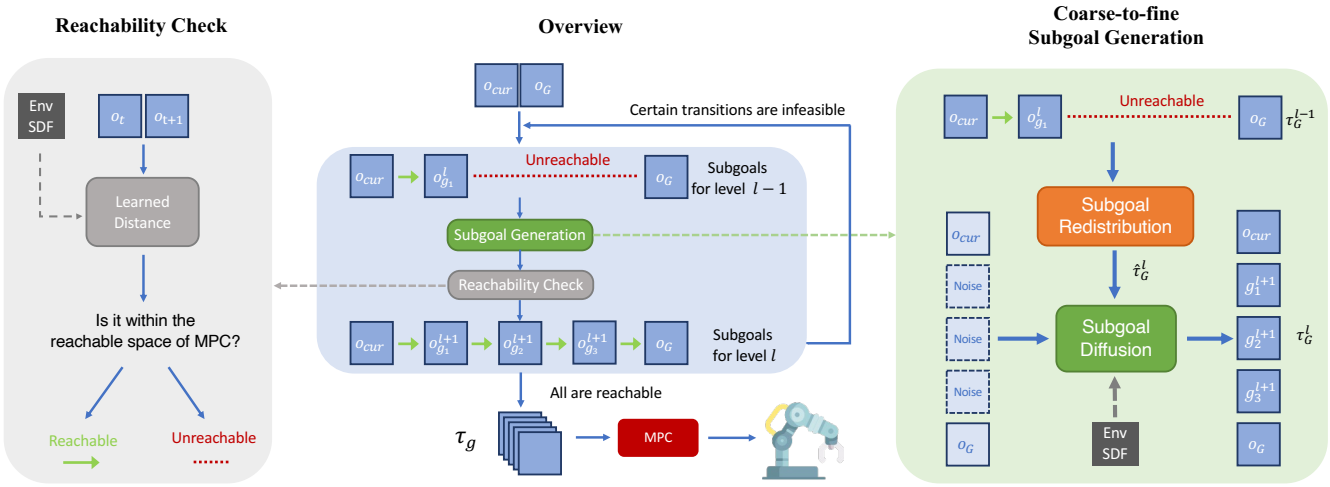

Fig. 1: Middle: Our system consists of a diffusion model that generates subgoals in a coarse-to-fine manner, and a low-level MPC controller that tracks the subgoals. The diffusion model generates subgoals recursively until all subgoals are reachable from their predecessors. **Left**: To estimate the reachability, we learn a function that estimates the number of steps required to move between the two subgoals. If the prediction is smaller than the horizon of the MPC, we assume it is reachable. **Right**: Our Subgoal Diffuser is conditioned on current state, goal state, subgoals from the previous level, and (optionally) an SDF of the environment. The subgoals from the previous level will be redistributed so that they are equally spaced in terms of execution steps.

where ϵ_θ is parameterized by a neural network to estimate the noise that can be used to recover the original data.

IV. Method

We propose to generate a sequence of subgoals $\boldsymbol{\tau}_\mathcal{G}$ to guide a sampling-based MPC method to perform a manipulation task. In Sec. IV-A, we describe *Subgoal Diffuser*, which generates a sequence of subgoals recursively from coarse to fine. With *Subgoal Diffuser*, we can generate an arbitrary number of subgoals. However, it is not clear how to determine how much guidance should come from the subgoals and how much should be left up to the MPC method. For example, tasks that are temporally extended or sensitive to error may require finer reasoning and thus more subgoals. To address this problem, we introduce a reachability-based method to dynamically determine the number of subgoals required for the task (Sec. IV-B). Then we discuss how we integrate it with an MPC controller (Sec. IV-C) as well as the implementation details (Sec. IV-D).

A. Coarse-to-fine Subgoal Generation using Diffusion

For challenging problems such as rope manipulation, generating a full sequence of locally and globally coherent subgoals in one shot is difficult. In this section, we introduce a diffusion architecture that is able to generate a chain of subgoals in a coarse-to-fine manner to enable planning at different temporal resolutions.

First, let Δt be the *temporal resolution*, which is the number of time steps between two consecutive states in a trajectory. An object trajectory $\tau_o = [\mathbf{o}_0, \mathbf{o}_1, \dots, \mathbf{o}_{T-1}]$ of length T has temporal resolution $\Delta t = 1$ between each pair of consecutive object states. When we set $\Delta t > 1$, we are able to extract a sequence of $\mathbf{M}$ subgoals $\boldsymbol{\tau}_\mathcal{G} = [\mathbf{o}_{g_0}, \mathbf{o}_{g_1}, \dots, \mathbf{o}_{g_{M-1}}]$, such that $\mathbf{o}_{g_i} = \mathbf{o}_{i*\Delta t}$ and $(\mathbf{M}-1)*\Delta t = T$. We define $\mathbf{o}_{cur} = \mathbf{o}_{g_0}, \mathbf{o}_G = \mathbf{o}_{g_{M-1}}$ and the hierarchy starts at $M_0 = 2$. To enable coarse-to-fine subgoal generation, we define a hierarchy of $L+1$ levels of subgoals $\boldsymbol{\tau}_\mathcal{G}^{0:L}$. Level l contains M_l subgoals, and $\mathbf{M}_{l+1} = 2\times\mathbf{M}_l - 1$.

To model $p(\boldsymbol{\tau}_\mathcal{G}|\mathbf{o}_{cur}, \mathbf{o}_G)$ at different resolutions, we propose a novel architecture - *Subgoal Diffuser*, which generates subgoals in a coarse-to-fine manner. As shown on the right side of Fig. 1, *Subgoal Diffuser* is a conditional generative model $p_\theta(\boldsymbol{\tau}_\mathcal{G}^l|\mathbf{o}_{cur}, \mathbf{o}_G, \boldsymbol{\tau}_\mathcal{G}^{l-1})$ that predicts finer subgoals given current state, goal state and subgoals generated from the previous level. $\boldsymbol{\tau}_\mathcal{G}^L$ can be predicted in a recursive manner:

$$p(\boldsymbol{\tau}_\mathcal{G}^L|\mathbf{o}_{cur}, \mathbf{o}_G) = p(\boldsymbol{\tau}_\mathcal{G}^0|\mathbf{o}_{cur}, \mathbf{o}_G)\prod_{l=1}^{L} p_\theta(\boldsymbol{\tau}_\mathcal{G}^l|\mathbf{o}_{cur}, \mathbf{o}_G, \boldsymbol{\tau}_\mathcal{G}^{l-1})$$

At each level, the subgoals are predicted in an in-painting manner [4], [32]. As shown on the right side of Fig. 1, the subgoal chain is initialized with Gaussian noise and gradually denoised into plausible subgoals during the reverse diffusion process. The first and final subgoals are $\mathbf{o}_{cur}$ and $\mathbf{o}_G$, which serve as conditioning. They are kept fixed throughout the diffusion process.

Subgoal Diffuser is also conditioned on the predicted subgoals of the previous level $\boldsymbol{\tau}_\mathcal{G}^{l-1}$. Since higher-level subgoals $\boldsymbol{\tau}_\mathcal{G}^{l-1}$ are coarser than $\boldsymbol{\tau}_\mathcal{G}^l$, we need a way to "upsample" them to a higher resolution. A straightforward way is to upsample subgoals via linear interpolation under the assumption that *the generated subgoals are equally spaced* (top figure in Fig. 2). However, this assumption does not always hold, especially for long-horizon problems. MPC may require more steps between certain pairs of subgoals than others, thus requiring more subgoals to be generated in-between.

To account for this, we propose to upsample $\boldsymbol{\tau}_\mathcal{G}^{l-1}$ according to the pairwise reachability between adjacent subgoals. Reachability is estimated by a learned function that approximates the number of steps the MPC will take to reach the next subgoal (described in Sec. IV-B). By doing so, the model focuses more on connecting the distant subgoals, which reduces the chance that a myopic MPC method

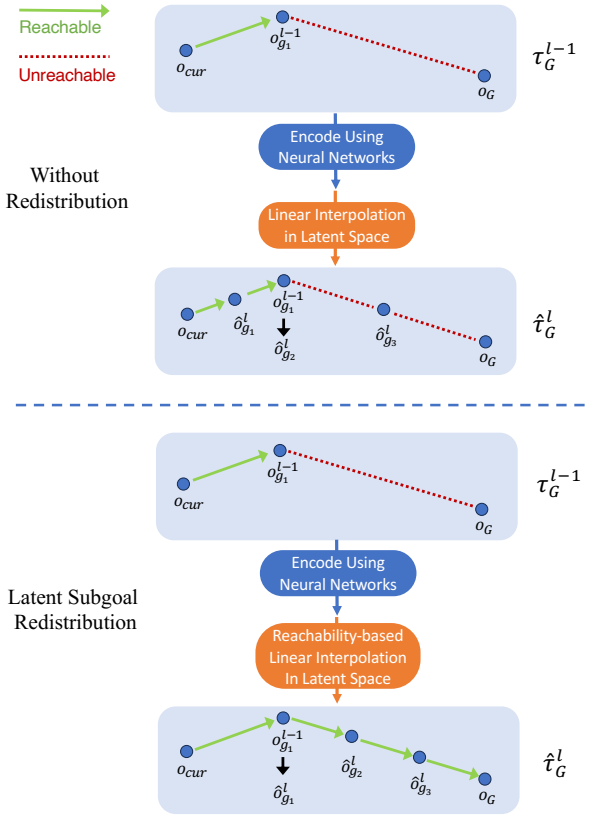

Fig. 2: Prediction of finer subgoals $\boldsymbol{\tau}_{\mathcal{G}}^{l}$ are generated contitioned on the coarse subgoals $\boldsymbol{\tau}_{\mathcal{G}}^{l-1}$. To compute the conditioning $\hat{\boldsymbol{\tau}}_{\mathcal{G}}^{l}$, we encode $\boldsymbol{\tau}_{\mathcal{G}}^{l-1}$ into latent space and upsample it using linear interpolation. **Top**: Without redistribution, the new subgoals are evenly distributed, ignoring the relative distance between consecutive subgoals. Thus, the subgoals that are far apart remain unreachable. **Bottom**: $\boldsymbol{\tau}_{\mathcal{G}}^{l-1}$ are redistributed in latent space using the estimated pairwise distance. By doing so, more subgoals will be filled in to the unreachable segment.

becomes stuck in a local minima (illustrated in bottom plot of Fig. 2). The assumption is that the MPC method is more likely to be stuck when subgoals are farther away. Also, since linear interpolation in the object state space $\boldsymbol{O}$ could be problematic, e.g. creating unrealistic states, we encode $\boldsymbol{\tau}_{\mathcal{G}}^{l-1}$ into a latent space using a neural network f_ϕ, then interpolate $f_\phi(\boldsymbol{\tau}_{\mathcal{G}}^{l-1})$ according to the reachability estimate and obtain a higher resolution subgoal chain $\hat{\boldsymbol{\tau}}_{\mathcal{G}}^{l}$.

We use a diffusion model with a temporal U-Net architecture, similar to [4], [16]. Temporal U-net applies a 1D convolution over the time dimension and allows for generating different number of subgoals using a single model.

Optionally, our method can also be conditioned on an SDF of the environment. We use two approaches for extracting information from the SDF: 1. *Global conditioning*. We process the SDF using a 3D convolution neural network to condense the information of the entire scene into a single feature vector. 2. *Local conditioning*. We render the point cloud of the object and compute the SDF value of each point. Then we process the point cloud with a PointNet ++ [34] model to extract local contact information.

B. Adaptive Subgoal Resolution Selection

Now we have a subgoal generator that is able to generate an arbitrary number of subgoals, but the method still needs to decide how many subgoals are necessary to complete the task. When the number of subgoals is insufficient, the low-level MPC controller will fail to follow the subgoals and get trapped. When there are too many subgoals, the distances between subgoals become small, and the MPC controller will make limited progress when tracking each subgoal. In our experiments, (Sec. V-A) we found that this is the reason why the baselines [5], [6] got stuck in later stages of the task.

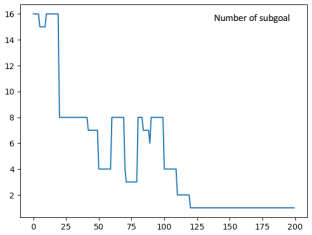

Fig. 3: Number of subgoals as the robot progresses.

Our insight is that the number of subgoals is sufficient when the MPC controller can travel between subgoals successfully, i.e., all subgoals are reachable from their predecessors. Since we focus on an offline learning setting, we cannot learn a reachability function for an MPC controller through online trial-and-error. Instead, we assume that our MPC controller is able to reach $\mathbf{o}_b$ from $\mathbf{o}_a$, if the random policy that we use to collect the dataset can. Put another way, $\mathbf{o}_b$ is reachable from $\mathbf{o}_a$, if in the offline dataset there exists a path $\boldsymbol{\tau}_{a\to b}$ that goes from $\mathbf{o}_a$ to $\mathbf{o}_b$ with steps $k < H$, where H is the horizon of the MPC controller.

To estimate the least number of steps k required to travel between $\mathbf{o}_a$ and $\mathbf{o}_b$ in the dataset, we follow [35]. First, we model the whole distribution of travel time between states $p_\psi(k|\mathbf{o}_a, \mathbf{o}_b)$. Then, we select the smallest k such that $p_\psi(k|\mathbf{o}_a, \mathbf{o}_b) > 0$. To learn $p_\psi(k|\mathbf{o}_a, \mathbf{o}_b)$, we discretize the travel time into 40 bins, where the i-th bin represents i steps, and the distances greater than 40 will be classified to be in the last bin. We frame this minimum distance estimation problem as classification, since regression will converge to the mean. To be robust to the error of function approximation, we use LogSumExp over the distribution to obtain a soft estimate of the least number of steps:

$$\hat{d}(\mathbf{o}_a, \mathbf{o}_b) = -\alpha \log \mathbb{E}_{k\sim p_\psi(\cdot|\mathbf{o}_a,\mathbf{o}_b)} \left[e^{-k/\alpha} \right], \qquad (4)$$

where α is a temperature parameter that controls the softness of this estimate. We say that $\mathbf{o}_b$ is reachable from $\mathbf{o}_a$ if $\hat{d}(\mathbf{o}_a, \mathbf{o}_b) < H$. During test time, we use this learned metric to compute the length of all segments in the subgoal chain. If the maximum $\hat{d}$ is larger than H, we increase the temporal resolution and generate more subgoals unless we have reached the maximum number of subgoals. $\hat{d}$ is also used during subgoal redistribution (Sec. IV-A).

C. Model Predictive Control (MPC)

As shown in the middle figure of Fig. 1, *Subgoal Diffuser* is integrated with an MPC controller to complete a robot manipulation task. We chose a sampling-based MPC method, MPPI [1], due to its robustness and flexibility. For our manipulation tasks, $\mathbf{u}$ is the change in gripper position. Given a τ_u produced by MPPI, we roll it out in a simulator

to produce the corresponding trajectory of object states τ_o, which is used to evaluate cost.

Usually, MPPI plans with a terminal cost for a single goal, yet simply picking the next subgoal predicted by diffusion does not perform well. This is because subgoal diffuser is trained on a random dataset, so the predicted subgoals can be sub-optimal (e.g. taking a detour), and it is safe to skip some intermediate subgoals. Therefore, we adopt the strategy of planning with goal sets, where MPPI considers all predicted subgoals simultaneously to compute the cost for a $\mathbf{o}_t \in \tau_o$. The optimization problem that MPPI seeks to solve is then

$$\arg\min_{\tau_u} \sum_{t=0}^{H-1} \Bigg(\min_{\mathbf{o}_{g_i} \in \boldsymbol{\tau}_\mathcal{G}} \left[||\mathbf{o}_t - \mathbf{o}_{g_i}||^2 - \lambda_{remote} \frac{1-\gamma^i}{1-\gamma} \right] + \lambda_{col} \max(-SDF(\mathbf{r}_t), 0) + \lambda_{smooth} ||\mathbf{u}_t - \mathbf{u}_{t-1}||^2 \Bigg).$$

The first term computes the distance to all subgoals with an incentive to encourage later subgoals in the chain, controlled by γ. The second term penalizes robot collisions (the simulator resolves object collisions since the objects are compliant), and the third term encourages smoothness of controls. A subgoal will be removed from the goal chain once it is reached. We regenerate the subgoals $\boldsymbol{\tau}_\mathcal{G}$ every 10 steps. We cannot guarantee that MPC will not become stuck when following the subgoal chains we predict. However, our results show that our method outperforms a baseline MPC method and two learning-based methods, suggesting that the subgoals we produce are indeed effective at guiding MPC.

D. Implementation

The training dataset $\mathcal{D} \triangleq \{\boldsymbol{\tau}^i\}_{0 \le i < N}$ is collected using a random policy and contains 10,000 trajectories with a length of 100. The random policy will first sample a random reachable location in the free space and plan a collision-free trajectory to it using MPPI. We define a subgoal hierarchy by specifying the number of subgoals at each layer $[M_0, M_1, \ldots, M_{L-1}]$. We use [2, 3, 5, 7, 9, 17] in our experiments. In each training iteration, we sample a truncation of the trajectory $\hat{\boldsymbol{\tau}}$ as well as the number of subgoals M_l. Then we subsample M_l equally spaced states $\boldsymbol{\tau}_\mathcal{G}^l$ as ground-truth subgoals and also $\boldsymbol{\tau}_\mathcal{G}^{l-1}$ as conditioning. When training the diffusion model $p_\theta(\boldsymbol{\tau}_\mathcal{G}^l | \mathbf{o}_{cur}, \mathbf{o}_G, \boldsymbol{\tau}_\mathcal{G}^{l-1})$, we add Gaussian noise to $\boldsymbol{\tau}_\mathcal{G}^{l-1}$ to approximate the prediction errors at test time. The model is trained according to Eq. 3.

During planning, MPPI samples 80 trajectories with a horizon of 10. We use a noise scale of 0.001 for action sampling and a temperature of 0.02 when computing the weights of sampled trajectories. We set $\lambda_{remote} = 0.02$ and $\gamma = 0.6$ to encourage the planner to reach later subgoals in the chain when possible. We also set $\lambda_{col} = 10$ and $\lambda_{smooth} = 0.001$. We warm-start the planner by initializing the nominal trajectory with results from the previous timestep. We run 5 iterations of refinement in the first step, and 2 iterations in the later steps. We use Mujoco [36] as the dynamics model for simulated and real-world experiments.

Method	Rope ↓	Notebook ↓
Ours	$\mathbf{2.2 \pm 0.9}$	$\mathbf{1.6 \pm 2.4}$
Diffusion Policy [6]	10 ± 6	7.9 ± 3.5
Decision Diffuser [5]	7.6 ± 1.7	56 ± 9
MPPI	6.3 ± 5.4	3.1 ± 4.0
Ours w/ fixed # subgoals + receding horizon	3.4 ± 1.8	3.1 ± 4.0
Ours w/ fixed # subgoals + fixed horizon	3.6 ± 1.8	8.7 ± 12
Ours w/o coarse-to-fine	2.5 ± 0.7	9.5 ± 9.6
Ours w/o subgoal redistribution	2.6 ± 0.8	2.4 ± 3.4

TABLE I: Mean and std. dev. of the minimum distance to the goal over 10 test cases for each task.

V. Experiments

A. Simulated experiments

Our experiments seek to answer the following questions: (1) Can our method outperform the state-of-the-art diffusion-based methods? (2) What are the most important design choices in our method? We consider two difficult manipulation tasks in simulation.

Rope reconfiguration. Rope reconfiguration is a challenging manipulation task due to high-dimensional state space and complex dynamics. The goal of this task is to make certain shapes with rope, such as a circle or S shape. We attach one end of the rope to a fixed point and the other to a robot gripper. The rope is modeled as a 10-link linkage in Mujoco.

Notebook manipulation. To investigate how well the method generalizes to novel environments, we create an environment with randomized obstacles (as shown in Fig. 4). The goal is to pick up the notebook from the ground, lay it on the table, and close it while avoiding all the obstacles. This is a task that would intuitively be separated into several stages, yet it is unclear where the intermediate subgoals should be. The robot grasps the notebook in the middle of its edge.

For both tasks, we define the object state space $\mathbf{o}$ using 10 keypoints on the object. All methods are evaluated on 10 start/goal pairs, and we use the euclidean distance to the goal as the evaluation metric. The maximum execution times are 350 (notebook) and 200 (rope) steps.

B. Baselines

We compare our method to the following baselines and ablations:

- Decision Diffuser [5]: Decision Diffuser is the state-of-the-art offline reinforcement learning method. It models the dense trajectory of states by diffusion and extracts actions via an inverse dynamics model. During test time, it predicts a fixed-length trajectory.
- Diffusion Policy [6]: Diffusion Policy is the state-of-the-art imitation learning method that directly models the action distribution of the dataset. Since our offline dataset contains low-quality data, we adapt the original implementation with hindsight relabeling [37], and add goal conditioning.
- Our method with fixed number of subgoals and receding horizon: In this baseline, we trained the model to always predict finest level of subgoals. During planning, a sequence of history states is used as conditioning, and the actual horizon is reduced as the history increases.

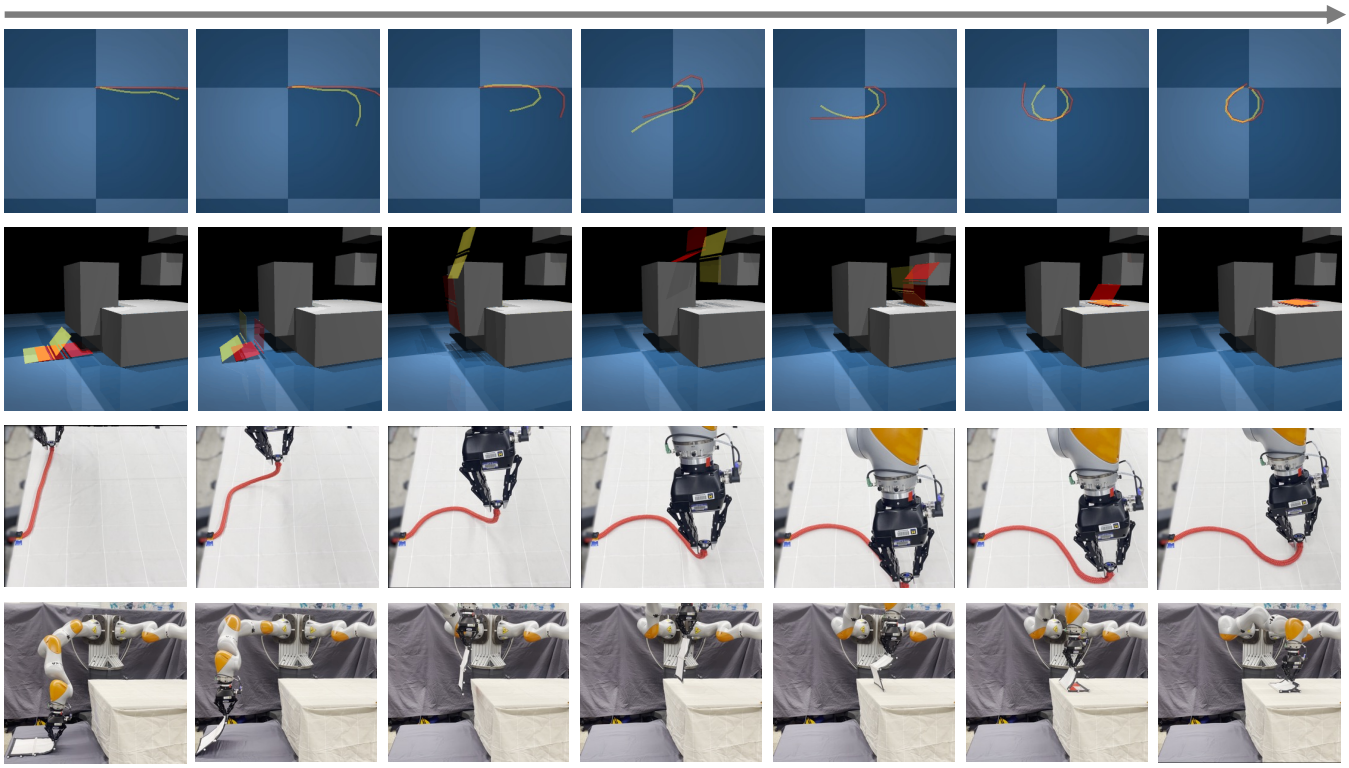

Fig. 4: Snapshots of rope and notebook manipulation tasks in simulation (top two rows) and the real world (bottom two rows). The subgoals are visualized in yellow for simulation experiments. More visualizations can be found on our website

- Our method with fixed number of subgoals and fixed horizon: Similar to above, this variant also always predicts the finest level of subgoals. However, during planning, it only uses current state as conditioning so that the planning horizon is fixed.
- Our method without coarse-to-fine generation: In this baseline, the model is trained to predict $\boldsymbol{\tau}_{\mathcal{G}}^{l}$ without being conditioned on $\boldsymbol{\tau}_{\mathcal{G}}^{l-1}$. During planning, it also uses adaptive subgoal resolution selection.
- Our method without subgoal redistribution: For this baseline, we upsample $\boldsymbol{\tau}_{\mathcal{G}}^{l}$ using linear interpolation and assume the subgoals are equally spaced.

For all methods, we record the minimum cost attained throughout the episode and compute their mean and standard deviation. As shown in Table I, our method outperforms all baselines for both tasks. Diffusion Policy [6] doesn't work very well in our setting. This may be due to its' sensitivity to the quality of the dataset, even with Hindsight Relabeling.

Decision Diffuser [5] works slightly better but is still worse than our method. Since it predicts long, dense trajectories (H = 100), we find that it predicts very small actions when it is close to the goal and becomes stuck.

MPPI alone is unable to solve complex manipulation tasks that contain local minima, thus obtaining sub-optimal performance. With the aid of our proposed subgoal generation method, the performance of MPPI is significantly improved.

We believe that part of the performance improvement over the Decision Diffuser and Diffusion Policy baselines comes from the fact that our method and MPPI use ground-truth dynamics models for planning. It is difficult to make a fair comparison, as diffusion policy and decision diffuser cannot be easily adapted to incorporate a ground-truth dynamics model. In fact, we consider the ability to leverage existing dynamics models to be an advantage of our method.

Regarding ablations, compared against the two ablations with a fixed number of subgoals, we see a 30% performance gain by using adaptive resolution selection. The coarse-to-fine generation scheme and subgoal redistribution also help with the performance, especially on the Notebook task.

C. Physical Experiments

To validate whether our method can be transferred to the real world, we replicated both the notebook manipulation and rope manipulation experiments using a 7 DoF Kuka LBR iiwa arm. To estimate the state of the object in the real world, we used motion capture for the notebook and CDCPD [38] for the rope. It is important to note that the dynamics model we use for physical experiments (Mujoco simulation) is only a rough approximation of real-world dynamics. Although the dynamics model we use is not accurate, i.e., we model the notebook as a rigid hinge while in the real world it is deformable, our method is able to reach the goal state reliably by regenerating the subgoals and replanning. See the accompanying video for executions.

VI. Conclusion

We introduce the Subgoal Diffuser, a novel architecture that generates subgoals recursively to guide Model Predictive Control. We also propose a reachability-based subgoal resolution selection scheme to dynamically determine the number of subgoals based on the difficulty of the task. Our experiments show that these methods effectively guide MPC to perform difficult long-horizon manipulation tasks.

REFERENCES

[1] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[2] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004, vol. 133.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[4] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[5] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" *arXiv preprint arXiv:2211.15657*, 2022.

[6] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[7] W. Li, X. Wang, B. Jin, and H. Zha, "Hierarchical diffusion for offline decision making," 2023.

[8] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[9] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, *et al.*, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[10] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, *et al.*, "Scaling robot learning with semantically imagined experience," *arXiv preprint arXiv:2302.11550*, 2023.

[11] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," *arXiv preprint arXiv:2309.01918*, 2023.

[12] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5923–5930.

[13] I. Kapelyukh, V. Vosylius, and E. Johns, "Dall-e-bot: Introducing web-scale diffusion models to robotics," *IEEE Robotics and Automation Letters*, 2023.

[14] W. Liu, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects," *arXiv preprint arXiv:2211.04604*, 2022.

[15] 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.

[16] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," *arXiv preprint arXiv:2308.01557*, 2023.

[17] 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," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 750–16 761.

[18] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, *et al.*, "Imitating human behaviour with diffusion models," *arXiv preprint arXiv:2301.10677*, 2023.

[19] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.

[20] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," *arXiv preprint arXiv:1909.05829*, 2019.

[21] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[22] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] K. Fang, P. Yin, A. Nair, and S. Levine, "Planning to practice: Efficient online fine-tuning by composing goals in latent space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4076–4083.

[24] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan, "Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools," *arXiv preprint arXiv:2203.17275*, 2022.

[25] T. Power and D. Berenson, "Variational inference mpc using normalizing flows and out-of-distribution projection," *arXiv preprint arXiv:2205.04667*, 2022.

[26] T. Wang and J. Ba, "Exploring model-based planning with policy networks," *arXiv preprint arXiv:1906.08649*, 2019.

[27] J. Sacks and B. Boots, "Learning to optimize in model predictive control," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 549–10 556.

[28] L. Li, Y. Miao, A. H. Qureshi, and M. C. Yip, "Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4496–4503, 2021.

[29] J. Carius, R. Ranftl, F. Farshidian, and M. Hutter, "Constrained stochastic optimal control with learned importance sampling: A path integral approach," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 189–209, 2022.

[30] T. Lai, W. Zhi, T. Hermans, and F. Ramos, "Parallelised diffeomorphic sampling-based motion planning," in *Conference on Robot Learning*. PMLR, 2022, pp. 81–90.

[31] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[32] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[33] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 681–688.

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[35] J. Hejna, J. Gao, and D. Sadigh, "Distance weighted supervised learning for offline interaction data," *arXiv preprint arXiv:2304.13774*, 2023.

[36] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[37] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[38] Y. Wang, D. McConachie, and D. Berenson, "Tracking partially-occluded deformable objects while enforcing geometric constraints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 199–14 205.