Diffusion-SSM based Policy Learning for Multi-Granularity Action Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract: We aim to solve the problem of multi-granularity action learning from demonstrations (LfD). To scale precision, traditional LfD approaches often rely on extensive fine-grained demonstrations to produce fixed-resolution actions. For memory-efficient learning and convenient granularity modulation, we propose a novel diffusion-SSM based policy (DiSPo) that learns from diverse coarse demonstrations and varying action scales using a state-space model, Mamba. Our evaluations show that the adoption of Mamba and the proposed step-scaling method enable DiSPo to outperform in three coarse-to-fine benchmark tests with a maximum 81% higher success rate than baselines. In addition, DiSPo improves inference efficiency by generating coarse motions in less critical regions. We finally demonstrate the scalability through two real-world manipulation tasks.

Keywords: multi-granularity learning, imitation learning, state-space model

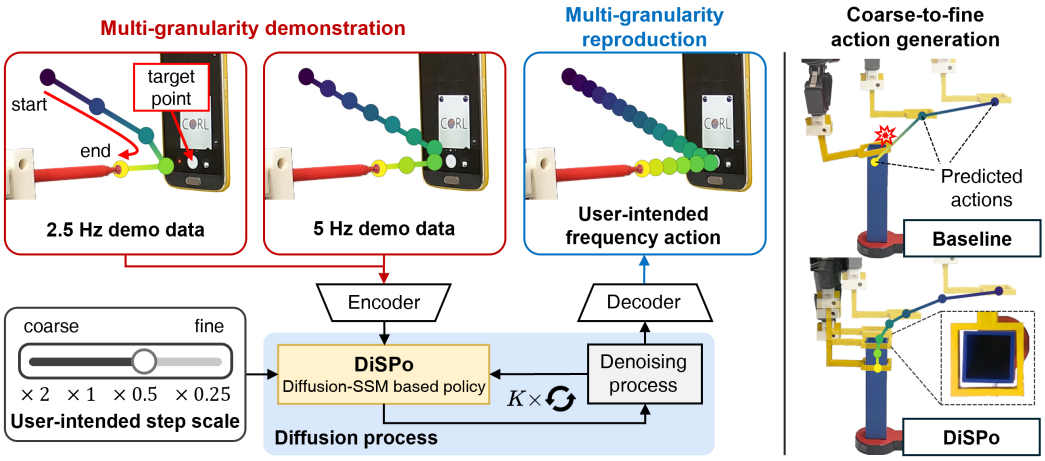

Figure 1: Overview of DiSPo: a diffusion-SSM based policy for coarse-to-fine imitation learning. Leveraging the representation power of diffusion policy and the flexible discretization capabilities of Mamba architecture, DiSPo learns from multi-granularity demonstrations (e.g., 2.5 Hz and 5 Hz) and generates actions at user-intended frequencies. DiSPo demonstrates improved accuracy and inference efficiency in fine-grained manipulation tasks compared to baseline methods.

1 Introduction

Researchers have increasingly focused on endowing robots with dexterous, generalizable policies such as human manipulations. These manipulations are often a mixture of coarse to fine actions [1], we call *multi-granularity* actions. These involve large positioning movements alongside precise maneuvers critical for tasks such as screwing, welding, and insertion. Learning these locally precise behaviors is crucial to task success [2, 3, 4].

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

In this context, we aim to solve the problem of generating manipulation skills at multiple levels of granularity through imitation learning (IL), a process we call *multi-granularity learning* as shown in Fig. 1. This requires models to learn from both fine-grained and general coarse demonstrations. Further, the models need to generate precise actions across varying control scales according to user needs, understanding the temporal structure of demonstrations. We term it as *multi-granularity reproduction*.

Traditional IL methods, such as dynamic movement primitives [5], learn complex trajectories [6]. By adopting dynamics models, these methods allow for frequency adjustments in output, learning from a specific frequency of input trajectories. In the line of research, state-space models (SSMs), such as Mamba [7], offer memory-efficient, powerful encoding. However, their fixed action representations struggle to capture complex or multi-modal behaviors across diverse task conditions or modalities.

Alternatively, neural IL methods, such as behavior transformers [8, 9, 10] and diffusion-based policies [11, 12, 13, 14], are increasingly acquiring attention with expressive power and robustness. These approaches are capable of learning from diverse, high-dimensional multi-modal datasets [15, 16, 17, 18, 19]. However, most approaches learn from a specific frequency of trajectories [20] or an unspecified timescale of state-action pairs [15], without understanding *multi-granularity*. Further, modeling fine-grained skills typically requires high-frequency demonstrations causing storage and computational overhead.

We propose a novel coarse-to-fine imitation learning algorithm, diffusion-SSM based policy (DiSPo), combining the representation power of diffusion policy with the flexible discretization power of SSM. We particularly adopt a state-of-the-art SSM, Mamba, to enable DiSPo to learn and reproduce trajectories at *multi-granularity* through data-efficient training strategies. We show that DiSPo is capable of producing varying scales of behavior, not only learning from multiple rates of coarse demonstrations but also modulating the discretization level of trajectories through a granularity predictor online. To the best of our knowledge, this is the first attempt to modulate Mamba's discrete model for fine-grained manipulations. We introduce novel coarse-to-fine IL benchmarks evaluating our method against state-of-the-art visuomotor policy learning methods. The evaluation shows the modulation of step size in DiSPo generates finer movements with expert-like behaviors.

2 Preliminaries

An SSM describes a dynamic system that accepts inputs $\mathbf{u} \in \mathbb{C}^D$, produces outputs $\mathbf{y} \in \mathbb{C}^D$, and updates a set of internal states $\mathbf{h} \in \mathbb{C}^N$, where D and N denote the dimensions of the input and state, respectively. The system consists of first-order differential equations, known as state and output equations: $\dot{\mathbf{h}}(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{h}(t)$, where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times D}$, and $\mathbf{C} \in \mathbb{R}^{D\times N}$ are the state, input, and output parameters, respectively.

For discrete computations, the SSM transforms the continuous-time system into a discrete-time system, defined over a discrete input sequence $\mathbf{u}_t \in \mathbb{R}^{L\times D}$ and output sequence $\mathbf{y}_t \in \mathbb{R}^{L\times D}$ at each time step t, where L denotes the sequence length. Given a step size $\mathbf{\Delta} \in \mathbb{R}^{L\times D}$, the discrete-time system is $\mathbf{h}_t = \bar{\mathbf{A}}\mathbf{h}_{t-1} + \bar{\mathbf{B}}\mathbf{u}_t, \quad \mathbf{y}_t = \mathbf{C}\mathbf{h}_t$, where the discrete parameters are

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}), \;\; \bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}, \tag{1}$$

following the zero-order hold (ZOH) discretization rule. In this work, the discrete parameters are updated to $\bar{\mathbf{A}} \in \mathbb{R}^{L\times N\times N}$, $\bar{\mathbf{B}} \in \mathbb{R}^{L\times N\times D}$, and $\mathbf{C} \in \mathbb{R}^{L\times D\times N}$. In contrast to S4 [21] with fixed step sizes, Mamba makes parameters $(\mathbf{B}, \mathbf{C}, \mathbf{\Delta})$ as a function of the input $\mathbf{u}_t$,

$$\mathbf{B}_t = f_B(\mathbf{u}_t), \; \mathbf{C}_t = f_C(\mathbf{u}_t), \; \mathbf{\Delta}_t = \text{SoftPlus}(f_{\mathbf{\Delta}}(\mathbf{u}_t)), \tag{2}$$

where f_B, f_C, and $f_{\mathbf{\Delta}}$ are trainable linear layers, and SoftPlus is an activation function.

3 Diffusion-SSM based Policy Model

Fig. 2 illustrates our proposed model architecture, which incorporates a denoising diffusion probabilistic model (DDPM) [22] with $N_{\mathcal{M}}$ stacked DiSPo blocks $\{\mathcal{M}_i\}_{i=1}^{N_{\mathcal{M}}}$. Each DiSPo block is a

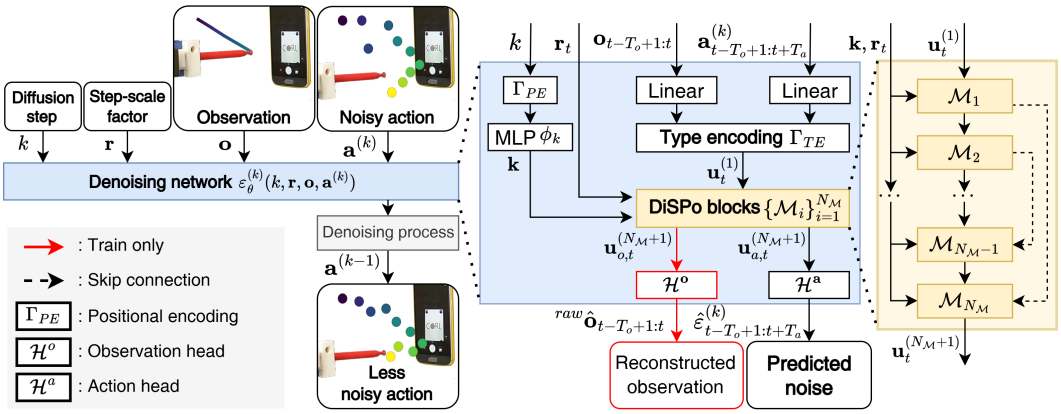

Figure 2: Illustration of the DiSPo architecture. DiSPo takes diffusion step k, step-scale factors $\mathbf{r}_t$, encoded observations $\mathbf{o}_{t-T_o+1:t}$, and noisy actions $\mathbf{a}^{(k)}_{t-T_o+1:t+T_a}$. The model identifies the noise $\hat{\varepsilon}^{(k)}_{t-T_o+1:t+T_a}$ within the input noisy actions through stacked DiSPo blocks and utilizes the identified noise to generate the less noisy action $\mathbf{a}^{(k-1)}_{t-T_o+1:t+T_a}$ from the previous noisy action.

variant of the Mamba block. Inspired by the decoder-only Mamba (D-Ma) [23], we design the architecture to learn denoising networks $\varepsilon_\theta^{(k)}$, parameterized by θ, generating a less noisy sequence of actions $\mathbf{a}^{(k-1)}$ conditioned on a history of observations $\mathbf{o}$, noisy actions $\mathbf{a}^{(k)}$, and step-scale factors $\mathbf{r}$ at the k-th denoising step ($k \in [1, \dots, K]$):

$$\mathbf{a}^{(k-1)} = \alpha \left(\mathbf{a}^{(k)} - \gamma \varepsilon_\theta^{(k)}(k, \mathbf{r}, \mathbf{o}, \mathbf{a}^{(k)}) + \mathcal{N}(0, \sigma^2 I) \right), \tag{3}$$

where α, γ, and σ are the noise schedule parameters following the DDPM formulation [22]. For notational simplicity, we omit the time index t. Starting from an initial Gaussian noise sample, $\mathbf{a}^{(K)}$, DiSPo recursively applies the denoising process to generate an imitated action sequence.

A distinct feature of DiSPo is the integration of step-scale factors $\mathbf{r}_t$ into Mamba blocks, inspired by manual adjustment of rates in time-invariant SSMs [21, 24]. This allows DiSPo to learn from multiple rates of demonstrations and to adjust step sizes for discrete-time SSM parameters. We describe the details below.

3.1 Mamba-based denoising process

Consider an input sequence $\mathbf{u}_t^{(1)} \in \mathbb{R}^{L \times D}$ in the k-th diffusion step and the time step t, where L and D are the length and dimension of the input sequence, respectively. Note that, to simplify the notation, we omit k and retain i for the variables defined in the k-th step below (e.g. $\mathbf{u}_t^{(k,1)} = \mathbf{u}_t^{(1)}$). The Mamba-based denoising network predicts the action noise $\hat{\varepsilon}^{(k)}$ by updating the sequences $\mathbf{u}_t^{(i)}$ with noise-relevant features through the $\{\mathcal{M}_i\}_{i=1}^{N_\mathcal{M}}$ blocks. Then it transforms the action component of the last updated sequence $\mathbf{u}_{a,t}^{(N_\mathcal{M}+1)}$ into the action noise through an output action head $\mathcal{H}^a$,

$$\mathbf{u}_t^{(i+1)} = \mathcal{M}_i(\mathbf{k}, \mathbf{r}_t, \mathbf{u}_t^{(i)}) \quad \text{and} \quad \hat{\varepsilon}^{(k)} = \mathcal{H}^a(\mathbf{u}_{a,t}^{(N_\mathcal{M}+1)}), \tag{4}$$

where $i \in [1, \dots, N_\mathcal{M}]$, $\mathbf{k} \in \mathbb{R}^D$ is an embedding for the diffusion step k, and $\mathbf{r}_t \in \mathbb{R}^L$. Each $\mathcal{M}_i$ block processes input sequences with the same size, $\mathbf{u}_t^{(i)} \in \mathbb{R}^{L \times D}$. The denoising process consists of three parts: input encoding, diffusion process, and noise prediction.

Input encoding: The first DiSPo block takes the input sequence $\mathbf{u}_t^{(1)}$, a diffusion step embedding $\mathbf{k}$, and step-scale factors $\mathbf{r}_t$ at each k-th step. The input sequence $\mathbf{u}_t^{(1)}$ consists of observation and noisy-action embeddings over lengths T_o and $T_o + T_a$, respectively. We represent it as

$$\mathbf{u}_t^{(1)} = [\Gamma_{\text{TE}}(\mathbf{f}_{o,t-T_o+1}), ..., \Gamma_{\text{TE}}(\mathbf{f}_{o,t}), \Gamma_{\text{TE}}(\mathbf{f}_{a,t-T_o+1}), ..., \Gamma_{\text{TE}}(\mathbf{f}_{a,t+T_a})], \tag{5}$$

where $\mathbf{f}_{o,t}$ and $\mathbf{f}_{a,t}$ are observation and action features, respectively. $\Gamma_{TE} : \mathbb{R}^D \to \mathbb{R}^D$ represent *type encoding*, which injects a learnable vector to the input ($\in \mathbb{R}^D$). Note that $L = 2T_o + T_a$.

An observation feature $\mathbf{f}_{o,t} \in \mathbb{R}^D$ is an embedding vector of the observation $\mathbf{o}_t \in \mathbb{R}^{D_o}$ preprocessed from raw sensory observations ${}^{\text{raw}}\mathbf{o}_t$ at a timestep t. The embedding process is a linear projection by $\mathbf{f}_{o,t} = \mathbf{w}_o \mathbf{o}_t + \mathbf{b}_o$ with a weight matrix $\mathbf{w}_o \in \mathbb{R}^{D \times D_o}$ and a bias $\mathbf{b}_o \in \mathbb{R}^D$. In this work, we use $\mathbf{o}_t$ as a concatenated vector of an image encoding from ResNet18 [25] with attentional pooling [26] and a proprioception vector (e.g., end-effector positions) normalized in the range of $[-1, 1]$.

An action feature $\mathbf{f}_{a,t} \in \mathbb{R}^D$ is an embedding vector of the action $\mathbf{a}_t \in \mathbb{R}^{D_a}$, obtained either by normalizing the raw action command ${}^{\text{raw}}\mathbf{a}_t$ with noise during training or by denoising the noisy action from the previous diffusion step during inference. The embedding process is a linear projection by $\mathbf{f}_{a,t} = \mathbf{w}_a \mathbf{a}_t + \mathbf{b}_a$ with a weight matrix $\mathbf{w}_a \in \mathbb{R}^{D \times D_a}$ and a bias $\mathbf{b}_a \in \mathbb{R}^D$. In this work, we use a pose vector as a command, normalizing in the range of $[-1, 1]$.

Lastly, as a part of input conditions, we embed the diffusion step k into a D-dimensional vector $\mathbf{k} = \phi_k(\Gamma_{\text{PE}}(k))$ by sinusoidal *positional encoding* $\Gamma_{PE} : \mathbb{R} \to \mathbb{R}^D$ followed by a multi-layer perceptron $\phi_k : \mathbb{R}^D \to \mathbb{R}^D$. We describe the step-scale factors $\mathbf{r}_t$ in Sec. 3.2.

Diffusion process: At each diffusion step, we update the sequence $\mathbf{u}_t^{(i)}$ with noise-relevant features through stacked DiSPo blocks $\{\mathcal{M}_i\}_{i=1}^{N_\mathcal{M}}$ with skip connections. Fig. 3 shows a DiSPo block that is a step-scaled Mamba block with adaptive layer normalization (adaLN) [27], performing dimension-wise scaling and shifting $\mathbf{u}_t^{(i)}$ into ${}^\dagger\mathbf{u}_t^{(i)}$ conditioned on the diffusion step embedding $\mathbf{k}$. Taking ${}^\dagger\mathbf{u}_t^{(i)}$, the step-scaled Mamba block adjusts the parameters of discrete-time SSM, according to user needs, i.e., a vector of step-scale factors $\mathbf{r}_t \in \mathbb{R}^L_{>0}$, and then updates the input sequence into ${}^\dagger\mathbf{u}_t^{(i+1)}$ via the internal step-scaled SSM. In contrast to conventional Mamba blocks, we exclude convolutional layers that limit handling diverse granularity of input sequences due to fixed-size receptive fields.

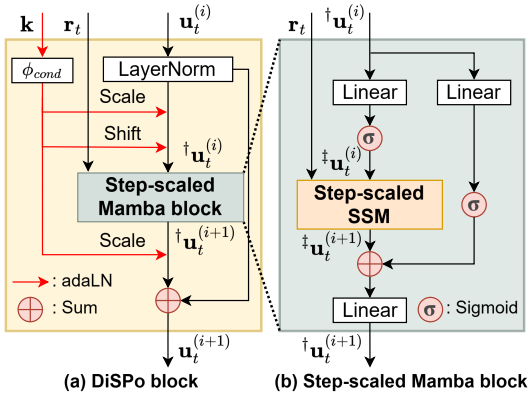

Figure 3: (a) A DiSPo block $\mathcal{M}_i$ refines noise-related features in the type encoded sequence $\mathbf{u}_t^{(i)}$ using adaLN conditioned on the diffusion step embedding $\mathbf{k}$. (b) A step-scaled Mamba block takes $\mathbf{r}_t$ and ${}^\dagger\mathbf{u}_t^{(i)}$.

Fig. 4 shows the proposed step-scaled SSM for *multi granularity*. Our SSM predicts the appropriate step size $\mathbf{\Delta}_t^{(i)} \in \mathbb{R}^{L \times D}_{>0}$ with respect to the input sequence ${}^\ddagger\mathbf{u}_t^{(i)}$, a non-linear projection of ${}^\dagger\mathbf{u}_t^{(i)}$, and the user-intended scales $\mathbf{r}_t$,

$$\mathbf{\Delta}_t^{(i)} = \mathbf{r}_t \cdot \text{SoftPlus}\left(f_{\mathbf{\Delta}}^{(i)}\left({}^\ddagger\mathbf{u}_t^{(i)}\right)\right), \quad (6)$$

where $f_{\mathbf{\Delta}}^{(i)}$ is a block-wise trainable linear layer used in Eq. (2). We use $\mathbf{\Delta}_t^{(i)}$ to calculate $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ following Eq. (1).

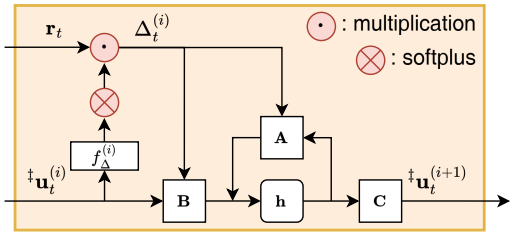

Figure 4: A step-scaled SSM takes input sequence ${}^\ddagger\mathbf{u}_t^{(i)}$ and $\mathbf{r}_t$ to scale $\Delta_t^{(i)}$, and discretizes the learned SSM parameters using the step sizes.

Noise prediction: After $N_\mathcal{M}$ times of feature updates, the action head $\mathcal{H}^a$ predicts the action noise $\hat{\varepsilon}^{(k)}_{t-T_o+1:t+T_a}$ with respect to $\mathbf{u}_{a,t}^{(N_\mathcal{M}+1)}$ that corresponds to the noisy action input $\mathbf{a}^{(k)}_{t-T_o+1:t+T_a}$ for the k-th denoising process. We then use the predicted noise to find the denoised action input $\mathbf{a}^{(k-1)}_{t-T_o+1:t+T_a}$ for the next diffusion step $k-1$, following Eq. (3) in inference.

In addition, during training, we enable our model to reconstruct the given raw observation ${}^{\text{raw}}\mathbf{o}_t$ decoding the updated sequence $\mathbf{u}_{o,t}^{(N_\mathcal{M}+1)}$ through an observation head $\mathcal{H}^o$. The reconstruction helps the model to keep capturing fine details in observations across layers. Here, the decoder consists of a linear layer for low-dimensional observation and a ResNet18 decoder for image-based observation.

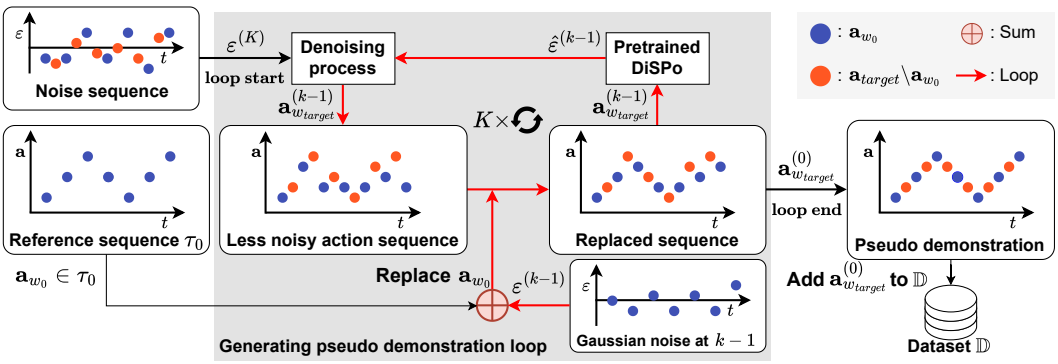

Figure 5: Generating a pseudo demonstration for fine-tuning. Starting from Gaussian noise $\varepsilon^{(K)}$ and a reference sequence τ_0, the model iteratively denoises and replaces w_0 frequency actions in the less noisy action sequence with noise added $\mathbf{a}_{w_0} \in \tau_0$. We repeat this process until the model generates a noise-less action sequence at target frequency $\mathbf{a}^{(0)}_{w_{\text{target}}}$, which we refer to as a pseudo demonstration.

3.2 Multi-granularity reproduction

To control the granularity of generated actions, DiSPo takes a vector of step-scale factors, $\mathbf{r}_t = [r^o_{t-T_o+1}, ..., r^o_t, r^a_{t-T_o+1}, ..., r^a_t, ..., r^a_{t+T_a}]$, where r^o_t and r^a_t represent the step size scales, we call factors, of the observation $\mathbf{o}_t$ and the action $\mathbf{a}_t$ relative to those of a reference sequence. We apply identical scales to the observation and past action sequences, such that $\mathbf{r}^o_{t-T_o+1:t-1} = \mathbf{r}^a_{t-T_o+1:t-1}$.

To define the reference step size, we use a mode selection approach that chooses the most frequently observed step size in demonstrations. DiSPo then allows for manual selection of the desired step-scale factors. For example, we set $\mathbf{r}_t = \mathbf{1}_{t-T_o+1:t} \mathbin{+\!\!+} \mathbf{1}_{t-T_o+1:t-1} \mathbin{+\!\!+} \mathbf{0.5}_{t:t+T_a}$ when we want to achieve twice finer actions and $\mathbf{r}_t = \mathbf{1}_{t-T_o+1:t} \mathbin{+\!\!+} \mathbf{1}_{t-T_o+1:t-1} \mathbin{+\!\!+} \mathbf{2}_{t:t+T_a}$ for twice coarser actions. In addition, DiSPo includes a step-scale factor predictor ϕ_r, implemented as an MLP, which predicts a factor $\mathbf{r}_t$ to accomplish the task given the observation $\mathbf{o}_t$.

4 Multi-Granularity Policy Learning

We introduce a *multi-granularity learning* scheme to improve the prediction performance of high-frequency actions that are not available in the demonstration dataset $\mathbb{D}$. Our scheme consists of two steps: 1) pretraining with sample-rate augmentation and 2) fine-tuning with pseudo actions.

In pretraining, to handle various granularities, we first augment the dataset $\mathbb{D}$ with random step-scale factors. We randomly draw a reference sequence $\tau_0 = [({}^{\text{raw}}\mathbf{o}_1, {}^{\text{raw}}\mathbf{a}_1), ..., ({}^{\text{raw}}\mathbf{o}_T, {}^{\text{raw}}\mathbf{a}_T)] \in \mathbb{D}$ with length T and sample frequency ω_0. By selecting a frequency $\omega_j \leq \omega_0$, we resample a sequence τ_j with step-scale factors $\mathbf{r}_j = \frac{\mathbf{1}_L}{(\omega_j/\omega_0)}$ from τ_0. Repetition of these enhancements creates the N_ω number of random frequency sequences: $\boldsymbol{\tau} = \{\tau_1, ..., \tau_{N_\omega}\}$. We then introduce a total loss $\mathcal{L} = \mathcal{L}^\varepsilon_{MSE} + \lambda \cdot \mathcal{L}^o_{MSE}$, where $\mathcal{L}^\varepsilon_{MSE}$, $\mathcal{L}^o_{MSE}$, and λ are a noise prediction error loss, an observation reconstruction loss, and a weighting coefficient ($\in \mathbb{R}_{>0}$), respectively. In detail, $\mathcal{L}^\varepsilon_{MSE}$ uses the mean squared error (MSE) to minimize a variational bound on the KL divergence between the true denoising process and that modeled by DiSPo:

$$\mathcal{L}^\varepsilon_{MSE} = MSE(\varepsilon^{(k)}, \varepsilon_\theta(k, \mathbf{r}_t, \mathbf{o}_t, \mathbf{a}^{(0)}_t + \varepsilon^{(k)})). \tag{7}$$

where $k \in [0, \dots, K]$. Likewise, $\mathcal{L}^o_{MSE}$ is the MSE between an input observation ${}^{\text{raw}}\mathbf{o}_t$ and its reconstruction from the observation head $\mathcal{H}^o$.

In fine-tuning, we co-finetune DiSPo on original and pseudo demonstration dataset to produce high-frequency actions not available in the dataset $\mathbb{D}$. Fig. 5 shows the process of generating fine-grained pseudo demonstrations. We randomly draw a reference sequence τ_0 with its frequency w_0 and generate a fine-grained sequence using the pretrained DiSPo by selecting a target frequency $\omega_{\text{target}} > \omega_0$ with $\mathbf{r} = \frac{\mathbf{1}_L}{(\omega_{\text{target}}/\omega_0)}$. In practice, starting from Gaussian noise, we perform the diffusion process

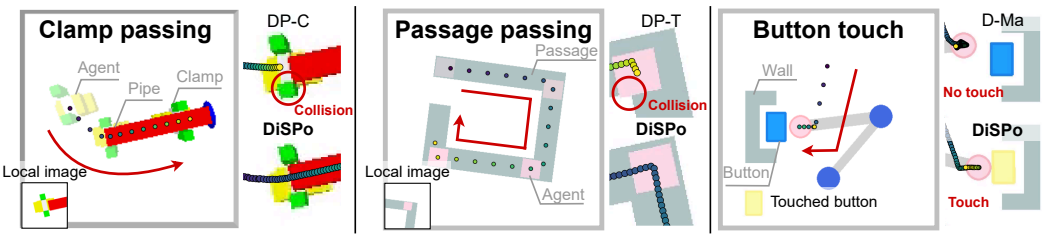

Figure 6: Illustrations of three simulation benchmarks, *clamp passing*, *passage passing*, and *button touch*. Dots denote either demonstrations at 2.5 Hz or predicted actions from DiSPo and baselines.

K times to generate a noise-less action sequence $\mathbf{a}^{(0)}_{1:T'}$, where $T' = T \cdot \omega_{\text{target}}/\omega_0$. However, the pretrained DiSPo is not sufficient to accurately produce high-frequency actions yet.

To figure it out, we decompose the predicted high-frequency actions $\mathbf{a}^{(k)}_{1:T'}$ into a subset with ω_0 frequency of actions $\mathbf{a}^{(k)}_{w_0}$ and its complement, $\mathbf{a}^{(k)}_{1:T'} \setminus \mathbf{a}^{(k)}_{w_0}$. At each k-th denoising process, we replace $\mathbf{a}^{(k)}_{w_0}$ with the demonstration actions in τ_0 with noise, $\mathbf{a}_{1:T}+\varepsilon^{(k)}$. This replacement helps in producing fine-grained pseudo actions that remain close to the demonstrations. In addition, as DiSPo predicts an action chunk, it produces multiple actions at a timestep. We aggregate these repeated predictions by weighted averaging, following the temporal ensemble in ACT [28], to obtain the final fine-grained action sequence. In contrast, the generation of fine-grained observations remains challenging. Thus, we retain the original frequency ω_0 of the observations by setting $r^o_t = 1$ and $r^a_t = \omega_0/\omega_{\text{target}}$ in fine-tuning. We call each outcome sequence a pseudo demonstration. We fine-tune DiSPo using both pseudo demonstrations and original demonstrations. In practice, we repeat the generation of pseudo demonstrations and fine-tuning, gradually increasing the target frequency ω_{target}. Note that we fine-tune the model with the loss $\mathcal{L}$ corresponding to $\mathbf{a}^{(k)}_{w_0}$ only since the predicted actions $\mathbf{a}^{(k)}_{1:T'} \setminus \mathbf{a}^{(k)}_{w_0}$ are not reliable as original demonstrations. However, sequential prediction with finer step-scale factors helps fine-tuning it as SSM internal state propagates through a sequence.

5 Experimental Setup

We conduct quantitative and qualitative evaluations using three simulated benchmarks and two real-world manipulation tasks. The benchmarks statistically assess the ability to generate fine-grained actions from coarse demonstrations. Below, we describe each benchmark in detail.

Clamp passing: A gripper agent (yellow) manipulates a clamp (green) to precisely approach and pass through a 2D pipe (red) without collision, as shown in Fig. 6 (Left). The raw observation ${}^{\text{raw}}\mathbf{o}_t$ comprises the agent's pose ($\in \mathbb{R}^3$) and two RGB images ($\in \mathbb{Z}^{96\times96\times3}$), one focusing on the agent (Fig. 6 left local image) and the other capturing the entire scene. The raw action ${}^{\text{raw}}\mathbf{a}_t$ is the agent's target pose ($\in \mathbb{R}^3$). We randomize the initial agent pose and vary the pipes' geometric properties (length and thickness) and spatial pose (position and orientation) using Pybullet [29].

Passage passing: A rectangular agent (pink) precisely maneuvers through a narrow 2D passage (gray) navigating corners without collision, as shown in Fig. 6 (Middle). The observation ${}^{\text{raw}}\mathbf{o}_t$ includes the agent's pose ($\in \mathbb{R}^2$) and two RGB images as in the *clamp passing* benchmark. The action is the 2D target position aligning the agent with the passage boundary. We randomize the passage's shape, width, and orientation using Pymunk [30] and Pygame [31].

Button touch: A two-link planar arm precisely touches a button (blue) without causing a collision between the button and wall, as shown in Fig. 6 (Right). The observation ${}^{\text{raw}}\mathbf{o}_t$ consists of the end-effector position ($\in \mathbb{R}^2$) and an RGB image. The action ${}^{\text{raw}}\mathbf{a}_t$ is the desired end-effector position ($\in \mathbb{R}^2$). We randomize the initial arm configuration and button placement, using Pymunk and Pygame.

For evaluation, we collect 90 high-frequency demonstrations at 20 Hz for each benchmark using the toppra path planning library [32]. We train our method and four baselines on coarsely sampled versions of demonstrations, selecting the best checkpoints based on performance over 50 random

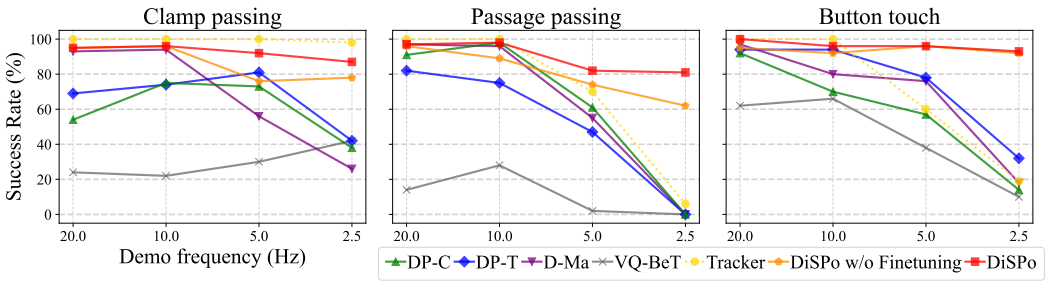

Figure 7: Comparison of task success rates [%] across four frequencies of demonstrations per simulated benchmark. We train each method with a source frequency (x-axis) of demonstrations and test a 20 Hz target frequency of actions in new environments. Note that tracker is a complexity indicator.

validation environments. We finally evaluate the approaches on 100 unseen test environments. The four baselines are as follows:

- DiffusionPolicy-C (DP-C) and DiffusionPolicy-T (DP-T) [12]: CNN- and Transformer-based diffusion policies, respectively.
- D-Ma [23]: A decoder-only variant of the Mamba-based diffusion model, MaIL.
- VQ-BeT [9]: A vector-quantized behavior Transformer (BeT) tokenizing continuous actions.

As baselines require fixed step sizes, we linearly interpolate their action sequences. In contrast, DiSPo generates fine-grained actions on demand using user-intended or predicted step-scale factors from the learned predictor ϕ_r. For comparison, we compute step-scale factors based on the demonstration and required frequency. We also use relative poses as desired actions when advantageous for baselines; baselines adopt relative poses as action representations for the *clamp passing* and *passage passing* tasks except DP-C, known to perform better with absolute positions [12]. We also report the performance of tracker, following downsampled ground-truth motion, as a task complexity indicator.

Finally, we demonstrate our method and a baseline, D-Ma, on real-world *clamp passing* and *button touch* tasks using a UR5e manipulator. Unlike the simulated benchmarks, we extend the action space to 3D translation and horizontal rotation ($\in \mathbb{R}^4$) for *clamp passing* and 3D translation for *button touch*. Each task uses three RealSense cameras: two for local views and one for a fixed global view. We collect 95 human demonstrations at 10 Hz and train both methods on coarsely sampled demonstrations: 2.5 Hz for *clamp passing* and a mixture of 2.5 Hz and 5 Hz for *button touch*. We compare two methods in 10 random environmental setup for each task. For real-time control, we use the denoising diffusion implicit model (DDIM) [33].

6 Evaluation

We first evaluate coarse-to-fine IL performance across three benchmarks using demonstrations at various frequencies. As shown in Fig. 7, DiSPo consistently achieves the highest success rates of over 81% across all frequencies, whereas baseline performances significantly drop given 2.5 Hz and 5 Hz demonstrations. For example, baseline methods usually fail at the corner of *passage passing* where DiSPo generates sharp motion as shown in Fig. 6. Occasionally, DiSPo without fine-tuning underperforms compared to baselines at tasks with fine-grained demonstrations, since the tasks are still solvable with low-frequencies demonstrations as the tracker's 100% performances. Nevertheless, our fine-tuning method improves performance by up to 19%, with an average gain of 6%, without additional data collection. In contrast, the tracker and baseline performances drop to near zero at 2.5 Hz, failing to reproduce abrupt corner maneuvering. These results highlight DiSPo's data efficiency and its ability to accurately learn feature spaces from coarse datasets.

We also evaluate *multi-granularity learning* by training methods with demonstrations at mixed frequencies, 2.5 Hz and 5 Hz. As shown in Fig. 8, DiSPo achieves the highest success rate of 93% on the *button touch* task, outperforming all baselines. DiSPo distinguishes sample-wise frequency differences, enabling effective *multi-granularity learning* without performance degradation. In con-

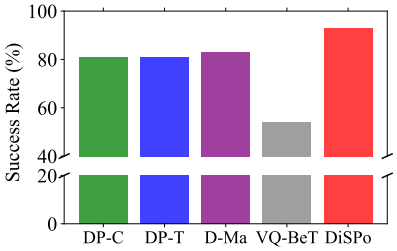

Figure 8: Task success rates under a mixed-frequency (2.5 and 5 Hz) training dataset in the *Button touch* task.

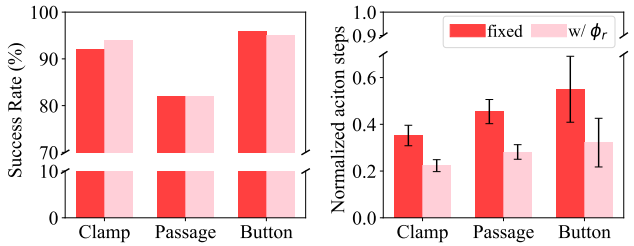

Figure 9: Comparison of DiSPo using fixed versus data-driven step-scaling factors from the predictor ϕ_r. We normalize the number of action steps by the maximum number.

trast, baselines naively model heterogeneous frequency of state-action pairs, producing actions at inappropriate speed that cause repetitive small back-and-force motions near the button.

Using DiSPo trained on 5 Hz demonstrations, we further evaluate the *multi-granularity reproduction* capability of DiSPo applying adaptive step scaling guided by the proposed predictor ϕ_r. As shown in Fig. 9, adopted scaling reduces the number of required steps by 39% with only a minor drop in task success rate on the *button touch* task, still significantly outperforming all baselines. These results demonstrate that DiSPo effectively modulates action discretization levels online, producing coarse motions in less critical regions (e.g., free space) to reduce inference overhead while maintaining fine-grained control in critical areas. Finally, we evaluate DiSPo and D-Ma on a UR5e manipulator in two real-world tasks: *clamp passing* and *button touch*. As shown in Fig. 10, DiSPo successfully inserts a square ring clamp with radial clearance 2.5 mm from random initial positions and precisely touches the shutter button by generating fine-grained, collision-free actions. Table 1 shows DiSPo achieves higher success rates than D-Ma in both setups. While D-Ma captures rough motions well, it often causes pipe scratching or stops near the button.

Table 1: Real world result success rate (%)

Method	***clamp passing***	***button touch***
D-Ma	20	20
DiSPo	**70**	**90**

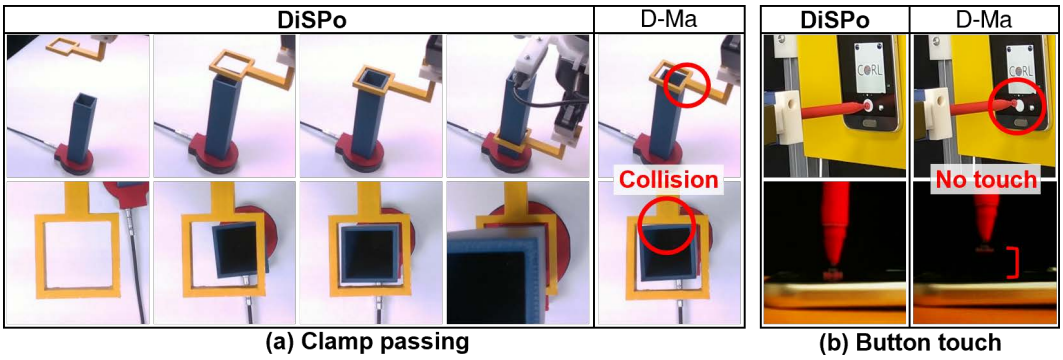

Figure 10: Representative samples showing the UR5e manipulator performing *clamp passing* and *button touch* from random initial and target positions in a real-world environment.

7 Conclusion

We proposed DiSPo, a novel diffusion-SSM based policy for *multi-granularity learning* and *reproduction* of coarse-to-fine demonstrations. DiSPo adaptively modulates the action step size via a step-scaling factor, enabling learning from various frequencies of coarse demonstrations and generating behaviors with varying scale of details. Furthermore, by integrating pseudo demonstration generation and step-scale prediction, our method shows potential for reducing storage and computational overhead. Experimental results on new benchmarks demonstrate that DiSPo produces smoother and more accurate motions than state-of-the-art methods. We successfully demonstrate the applicability and superiority of DiSPo through real-world experiments.

8 Limitations

Although DiSPo significantly outperforms baseline methods in tasks where the demonstration frequency is lower than that of the action execution, the performance gap between DiSPo and baselines becomes smaller when demonstrations are already fine-grained. We consider the exploration of high-frequency demonstration scenarios to be outside the primary scope of this work. Second, we use a fixed image resolution for all observation inputs, without mechanisms to selectively focus on specific regions. In our real-world experiments, we find consistent performance improvements when we zoom into task-relevant regions. Future works can integrate recent advances in task-relevant region detection or visual attention mechanism. Additionally, we primarily focus on using 2D RGB images as observation, which lack explicit depth or geometric context. We acknowledge that incorporating richer modalities such as RGB-D images or point clouds may enhance the capacity of model for finer action generation and spatial reasoning. Investigating the integration of such multimodal sensory inputs remains a direction of future work.

References

[1] E. Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619. IEEE, 2021.

[2] W. Lian, T. Kelch, D. Holz, A. Norton, and S. Schaal. Benchmarking off-the-shelf solutions to robotic assembly tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1046–1053. IEEE, 2021.

[3] X. Li, M. Baum, and O. Brock. Augmentation enables one-shot generalization in learning from demonstration for contact-rich manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3656–3663. IEEE, 2023.

[4] G. Papagiannis and E. Johns. Miles: Making imitation learning easy with self-supervision. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 270, pages 810–829. PMLR, 2025.

[5] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1398–1403. IEEE, 2002.

[6] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[7] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[8] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 22955–22968, 2022.

[9] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 235, pages 26991–27008. PMLR, 2024.

[10] Z. Gong, P. Ding, S. Lyu, S. Huang, M. Sun, W. Zhao, Z. Fan, and D. Wang. Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction. *arXiv preprint arXiv:2412.06782*, 2024.

[11] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 9902–9915. PMLR, 2022.

[12] 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. *International Journal of Robotics Research*, 2023.

[13] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 2905–2925. PMLR, 2023.

[14] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[15] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[16] A. Brohan, N. Brown, and a. Justice Carbajal et. Rt-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 1–22, 2023.

[17] B. Zitkovich, T. Yu, S. Xu, and a. Xu, Peng et. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 229, pages 2165–2183, 2023.

[18] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 785–799. PMLR, 2023.

[19] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[20] A. Khazatsky, K. Pertsch, S. Nair, and A. e. a. Balakrishna. Droid: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[21] A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2022.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

[23] X. Jia, Q. Wang, A. Donat, B. Xing, G. Li, H. Zhou, O. Celik, D. Blessing, R. Lioutikov, and G. Neumann. Mail: Improving imitation learning with selective state space models. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 270, pages 3888–3907. PMLR, 2025.

[24] A. Mondal, S. Alletto, and D. Tome. Hummuss: Human motion understanding using state space models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2318–2330, 2024.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[26] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[27] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[28] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[29] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016.

[30] V. Blomqvist. Pymunk: A easy-to-use pythonic rigid body 2d physics library. https://www.pymunk.org, 2007.

[31] pygame. https://www.pygame.org/, 2007.

[32] H. Pham and Q.-C. Pham. A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34(3):645–659, 2018.

[33] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *Proceedings of the International Conference on Learning Representation (ICLR)*, 2021.