A Further Details on the Experimental Setup

A.1 Task Descriptions

We consider a total of 9 continuous control tasks from 3 benchmarks: ManiSkill2 (Gu et al., 2023), ManiSkill3 (Tao et al., 2024), and Adroit (Rajeswaran et al., 2017). This section provides detailed task descriptions on overall information, task difficulty, object sets, state space, and action space. Some task details are listed in Table 8.

Table 8: We consider 9 continuous tasks from 3 benchmarks. We list important task details below.

Task	**State Obs Dim** C_{state}	**Act Dim** C_a	**Max Episode Step**
ManiSkill3: PushT	31	7	100
ManiSkill3: RollBall	44	4	80
ManiSkill2: StackCube	55	4	200
ManiSkill2: TurnFaucet	43	7	200
ManiSkill2: PushChair	131	20	200
Adroit: Door	39	28	300
Adroit: Pen	46	24	200
Adroit: Hammer	46	26	400
Adroit: Relocate	39	30	400

A.1.1 ManiSkill2 Tasks

StackCube

- Overall Description: Pick up a red cube and place it onto a green one. See Figure 6 for episode visualization.
- Task Difficulty: This task requires precise control. The gripper needs to firmly grasp the red cube and accurately place it onto the green one.
- Object Variations: No object variations
- Action Space: Delta position of the end-effector and joint positions of the gripper.
- State Observation Space: Proprioceptive robot state information, such as joint angles and velocities of the robot arm, and task-specific goal information
- Visual Observation Space: one 64x64 RGBD image from a base camera and one 64x64 RGBD image from a hand camera.

Figure 6: StackCube Episode Visualization.

TurnFaucet

- Overall Description: Turn on a faucet by rotating its handle.
- Task Difficulty: This task needs to handle object variations. See Figure 7 for episode visualization.

- Object Variations: We have a source environment containing 10 faucets, and the dataset is collected in the source environment. w/o g means the agent directly interacts with the source environment online; w/ g means the agent interacts with the target environment online, which contains 4 novel faucets.
- Action Space: Delta pose of the end-effector and joint positions of the gripper.
- State Observation Space: Proprioceptive robot state information, such as joint angles and velocities of the robot arm, the mobile base, and task-specific goal information.

Figure 7: TurnFaucet Episode Visualization.

PushChair

- Overall Description: A dual-arm mobile robot needs to push a swivel chair to a target location on the ground (indicated by a red hemisphere) and prevent it from falling over. The friction and damping parameters for the chair joints are randomized. See Figure 8 for episode visualization.
- Task Difficulty: This task needs to handle object variations.
- Object Variations: We have a source environment containing 5 chairs, and the dataset is collected in the source environment. w/o g means the agent directly interacts with the source environment online; w/ g means the agent interacts with the target environment online, which contains 3 novel faucets.
- State Observation Space: Proprioceptive robot state information, such as joint angles and velocities of the robot arm, task-specific goal information.
- Visual Observation Space: three 50x125 RGBD images from three cameras 120° apart from each other mounted on the robot.

Figure 8: PushChair Episode Visualization.

A.1.2 ManiSkill3 Tasks

PushT

- Overall Description: It is a simulated version of the real-world push-T task from Diffusion Policy: https://diffusion-policy.cs.columbia.edu/. In this task, the robot needs to precisely push the T-shaped block into the target region, and move the end-effector to the end-zone which terminates the episodes. The success condition is that the T block covers 90% of the 2D goal T's area. See Figure 9 for episode visualization.
- Task Difficulty: The task involves manipulating a dynamic T-shaped object, which introduces non-linear dynamics, friction, and contact forces.
- Object Variations: No object variations.

- Action Space: Delta pose of the end-effector and joint positions of the gripper.
- State Observation Space: Proprioceptive robot state information, such as joint angles and velocities of the robot arm, and task-specific goal information.
- Visual Observation Space: one 64x64 RGBD image from a base camera.

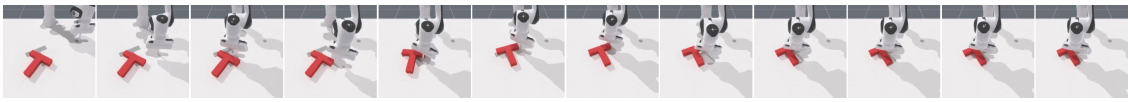

Figure 9: PushT Episode Visualization. The T-Block is pushed from sampled initial configuration to the goal area.

RollBall

- Overall Description: A task where the objective is to push and roll a ball to a goal region at the other end of the table. The success condition is that The ball's xy position is within goal radius (default 0.1) of the target's xy position by euclidean distance. See Figure 10 for episode visualization.
- Task Difficulty: The task involves manipulating a dynamic ball, which introduces non-linear dynamics, friction, and contact forces.
- Object Variations: No object variations.
- Action Space: Delta position of the end-effector and joint positions of the gripper.
- State Observation Space: Proprioceptive robot state information, such as joint angles and velocities of the robot arm, and task-specific goal information.
- Visual Observation Space: one 64x64 RGBD image from a base camera.

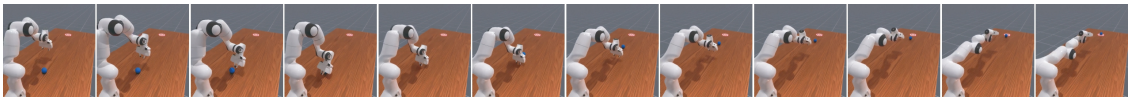

Figure 10: Rollball Episode Visualization. The blue ball is pushed and rolled from sampled initial configuration to the target red circle.

A.1.3 Adroit Tasks

Adroit Door

- Overall Description: The environment is based on the Adroit manipulation platform, a 28 degree of freedom system which consists of a 24 degrees of freedom ShadowHand and a 4 degree of freedom arm. The task to be completed consists on undoing the latch and swing the door open. See Figure 11 for episode visualization.
- Task Difficulty: The latch has significant dry friction and a biass torque that forces the door to stay closed. No information about the latcch is explicitly provided. The position of the door is randomized.
- Object Variations: No object variations.
- Action Space: Absolute angular positions of the Adoit hand joints.

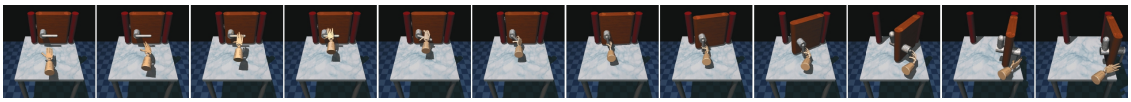

Figure 11: Door Episode Visualization.

- State Observation Space: The angular position of the finger joints, the pose of the palm of the hand, as well as state of the latch and door.
- Visual Observation Space: one 128x128 RGB image from a third-person view camera.

Adroit Pen

- Overall Description: The environment is based on the Adroit manipulation platform, a 28 degree of freedom system which consists of a 24 degrees of freedom ShadowHand and a 4 degree of freedom arm. The task to be completed consists on repositioning the blue pen to match the orientation of the green target. See Figure 12 for episode visualization.
- Task Difficulty: The target is also randomized to cover all configurations.
- Object Variations: No object variations.
- Action Space: Absolute angular positions of the Adroit hand joints.
- State Observation Space: The angular position of the finger joints, the pose of the palm of the hand, as well as the pose of the real pen and target goal.
- Visual Observation Space: one 128x128 RGB image from a third-person view camera.

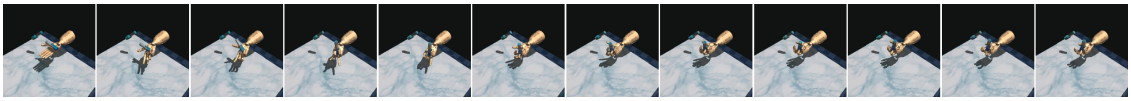

Figure 12: Pen Episode Visualization.

Adroit Hammer

- Overall Description: The environment is based on the Adroit manipulation platform, a 28 degree of freedom ShadowHand and a 4 degree of freedom arm. The task to be completed consists on picking up a hammer with and drive a nail into a board. See Figure 13 for episode visualization.
- Task Difficulty: The nail position is randomized and has dry friction capable of absorbing up to 15N force.
- Object Variations: No object variation.
- Action Space: Absolute angular positions of the Adroit hand joints.
- State Observation Space: The angular position of the finger joints, the pose of the palm of the hand, the pose of the hammer and nail, and external forces on the nail.
- Visual Observation Space: one 128x128 RGB image from a third-person view camera.

Adroit Relocate

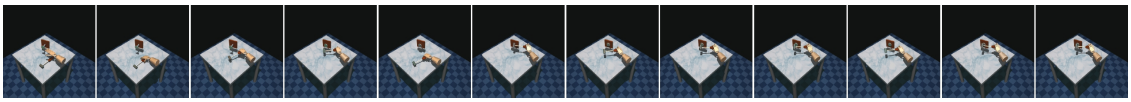

Figure 13: Hammer Episode Visualization.

- Overall Description: The environment is based on the Adroit manipulation platform, a 30 degree of freedom system which consists of a 24 degrees of freedom ShadowHand and a 6 degree of freedom arm. The task to be completed consists on moving the blue ball to the green target. See Figure 14 for episode visualization.
- Task Difficulty: The positions of the ball and target are randomized over the entire workspace.
- Object Variations: No object variations.
- Action Space: Absolute angular positions of the Adroit hand joints.
- State Observation Space: The angular position of the finger joints, the pose of the palm of the hand, as well as kinematic information about the ball and target.
- Visual Observation Space: one 128x128 RGB image from a third-person view camera.

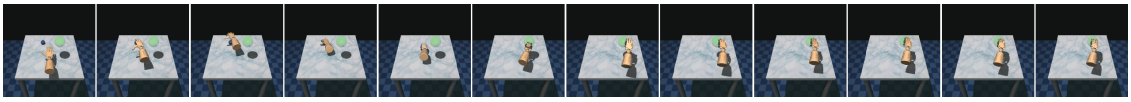

Figure 14: Relocate Episode Visualization.

A.2 Demonstrations

This subsection provides the details of demonstrations used in our experiments. See Table 9. ManiSkill2 and ManiSkill3 demonstrations are provided in Gu et al. (2023) and Tao et al. (2024), and Adroit demonstrations are provided in Rajeswaran et al. (2017).

Table 9: Demonstration sources, numbers and generation methods.

Task	Traj Num for Training	Generation Method
ManiSkill3: PushT	1000	Reinforcement Learning
ManiSkill3: RollBall	1000	Reinforcement Learning
ManiSkill2: StackCube	1000	Task & Motion Planning
ManiSkill2: TurnFaucet	1000	Model Predictive Control
ManiSkill2: PushChair	1000	Reinforcement Learning
Adroit: Door	25	Human Demonstration
Adroit: Pen	25	Human Demonstration
Adroit: Hammer	25	Human Demonstration
Adroit: Relocate	25	Human Demonstration

B Implementation Details

B.1 Noise-Relaying Diffusion Policy Inference

We summarize the inference pseudo-code of our RNR-DP in Algorithm 1.

Algorithm 1 Noise-relaying Diffusion Policy Inference

Require: denoising model, ε_θ; observation, $\mathbf{O}_t$; noise-relaying buffer, $\tilde{\mathbf{Q}}_t$; buffer capacity f;
while task execution **do**
  $\mathbf{Q}_t \leftarrow \varepsilon_\theta(\tilde{\mathbf{Q}}_t; \mathbf{O}_t, \{1, \cdots, f\})$ ▷ ε_θ is trained using f noise levels
  $\mathbf{a}_t^{(0)} \leftarrow \mathbf{Q}_t.\text{pop}(0)$ ▷ $\mathbf{a}_t^{(0)}$ is a clean action (fully denoised)
  $\tilde{\mathbf{Q}}_t \leftarrow \mathbf{Q}_t.\text{push}(\mathbf{z})$ ▷ $\mathbf{z}$ is a random noisy action sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{O}_t \leftarrow \text{env.step}(\mathbf{a}_t^{(0)})$ ▷ executes $\mathbf{a}_t^{(0)}$ and envrionment updates observation
  $t \leftarrow t + 1$ ▷ update timestep for the next iteration

B.2 Noise-Relaying Diffusion Policy Training

We summarize the training pseudo-code of our RNR-DP in Algorithm 2.

Algorithm 2 Responsive Noise-Relaying Diffusion Policy Training

Require: demonstration dataset, $\mathcal{D} = \{(\mathbf{O}_i, \mathbf{A}_i)\}_{i=1}^N$; denoising model, ε_θ; number of diffusion steps, f
repeat
  Sample $(\mathbf{O}, \mathbf{A}) \sim \mathcal{D}$
  Sample $p \sim \text{Unif}(0, 1)$; Sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and reshape to $\mathbb{R}^{C_a \times f}$
  if $p \leq p_{\text{linear}}$:
    $\mathbf{k} = \{k_1 = 1, \cdots, k_f = f\}$ ▷ linear schedule
  else
    $\mathbf{k} = \{k_1 \sim \text{Unif}(\{1, \cdots, f\}), \cdots, k_f \sim \text{Unif}(\{1, \cdots, f\})\}$ ▷ random schedule
  for all $\mathbf{a}_j \in \mathbf{A}$ indexed by frame index j **do**
    $\hat{\mathbf{a}}_j = \sqrt{\bar{\alpha}_{t_j}}\mathbf{a}_j + \sqrt{1 - \bar{\alpha}_{t_j}}\boldsymbol{\epsilon}_j$ ▷ perturbe each $\mathbf{a}_j$ independently
  $\hat{\mathbf{A}} = \{\hat{\mathbf{a}}_0, \cdots, \hat{\mathbf{a}}_{f-1}\}$
  Take gradient descent step to update θ on
    $\nabla_\theta \|\boldsymbol{\epsilon} - \varepsilon_\theta(\hat{\mathbf{A}}; \mathbf{O}, \mathbf{k})\|$ ▷ noise-aware conditioning
until converged

B.3 Policy Architecture

We build our RNR-DP on top of the UNet-based architecture of Diffusion Policy (Chi et al., 2023). The model includes 2 downsampling modules and 2 upsampling modules with each module containing 2 residual blocks. The residual block consists of 1D temporal convolutions (Conv1d), group normalizations (GN), and Mish activation layers. The encoded noise-aware conditioning data (Section 5.2) is fused into each residual block through the FiLM transformation (Perez et al., 2018). The raw conditioning data is of shape $R^{f \times (C_{\text{emb}} + C_{\text{state}})}$ for state policies and of shape $R^{f \times (C_{\text{emb}} + C_{\text{visual}} + C_{\text{state}})}$ for visual policies. See Figure 15 for the visualization of a visual policy. We follow the UNet denoiser design for the observation that transformer-based policies are more sensitive to hyperparameters and often require more tuning (Chi et al., 2023). The choice of policy architecture is orthogonal to our method and we believe our design would also improve this policy class.

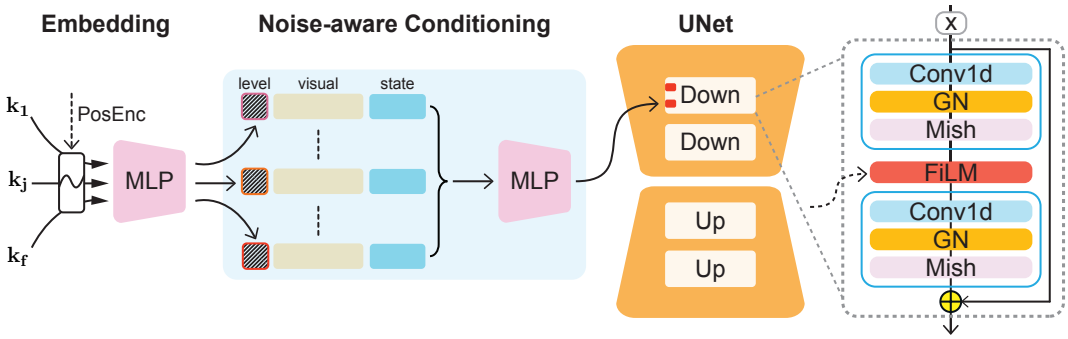

Figure 15: The detailed policy architecture for our RNR-DP. We only extract visual features for visual policies.

B.4 Important Hyperparameters

B.4.1 Key Hyperparameters of RNR-DP

We summarize the key hyperparameters of RNR-DP in Table 10. The observation horizon T_o and noise-relaying buffer capacity f for each task is listed in Table 11. The number of trainable parameters for each task is listed in Table 12.

Table 10: We list the key hyperparameters of RNR-DP used in our experiments.

Hyperparameter	Value
RNR-DP Noise Scheduling Scheme	Mixture Sampling(p_{linear}) $p_{\text{linear}} = 0.4$
RNR-DP Model Prediction Type	Noise
Diffusion Step Embedding Dimension	64
UNet Downsampling Dimensions	[64, 128, 256]
Optimizer	AdamW
Weight Decay	1e-6
Learning Rate	1e-4
Learning Rate Scheduler	Cosine
EMA Model Update	0.9999
Online Evaluation Episodes	1000

B.4.2 Key Hyperparameters of Diffusion Policy

We summarize the key hyperparameters of Diffusion Policy in Table 13. The observation horizon T_o, action executation horizon T_a and action prediction horizon T_p for each task are listed in Table 14.

B.5 Training Details

We train our models and baselines with cluster assigned GPUs (NVIDIA 2080Ti & A10). We use AdamW optimizer with an initial learning rate of 1e-4, applying 500 warmup steps followed by cosine decay. We use batch size of 1024 for state policies and 256 for visual policies for both ManiSkill and Adroit benchmarks. We evaluate DP, CP and RNR-DP model checkpoints using EMA weights every 10K training iterations for ManiSkill tasks and every 5K for Adroit tasks. DDIMs are evaluated using the best checkpoints of DDPMs in an offline manner. CPs are trained using the best checkpoints of EDMs.

Table 11: The observation horizon and noise-relaying buffer capacity of our RNR-DP for each task.

Task	**Obs** T_o	**Capacity** f
ManiSkill3: PushT (Visual)	2	48
ManiSkill3: RollBall (Visual)	2	64
ManiSkill2: StackCube (Visual)	2	84
Adroit: Pen (Visual)	2	4
Adroit: Hammer (Visual)	2	64
Adroit: Door (Visual)	2	56
ManiSkill3: PushT (State)	2	32
ManiSkill3: RollBall (State)	2	4
ManiSkill2: StackCube (State)	2	84
ManiSkill2: TurnFaucet w/g (State)	2	64
ManiSkill2: TurnFaucet w/o g (State)	2	72
ManiSkill2: PushChair w/g (State)	2	56
ManiSkill2: PushChair w/o g (State)	2	48
Adroit: Door (State)	2	74
Adroit: Pen (State)	2	4
Adroit: Hammer (State)	2	32
Adroit: Relocate (State)	2	84

Table 12: The number of our RNR-DP trainable parameters for each task. Noise-relaying buffer size doesn't affect the number of trainable parameters for each task.

Task	**Trainable Params**
ManiSkill3: PushT (Visual)	11.42M
ManiSkill3: RollBall (Visual)	11.59M
ManiSkill2: StackCube (Visual)	10.85M
Adroit: Pen (Visual)	14.67M
Adroit: Hammer (Visual)	14.72M
Adroit: Door (Visual)	14.41M
ManiSkill3: PushT (State)	4.53M
ManiSkill3: RollBall (State)	4.73M
ManiSkill2: StackCube (State)	4.91M
ManiSkill2: TurnFaucet (State)	4.71M
Adroit: Door (State)	4.66M
Adroit: Pen (State)	4.75M
Adroit: Hammer (State)	4.77M
Adroit: Relocate (State)	4.66M

C Additional Results

C.1 Empirical Comparison with Acceleration Methods on Visual Observations

We summarize the results of vision-based experiments in Table 15. As shown in Table 15, our RNR-DP ourperforms all DDIM variations and CP variations and particularly has an overall improvement over 8-step DDIM by 6.9%, over 8-step-chaining CP by 3.4%.

Table 13: We list the key hyperparameters of Diffusion Policy baseline used in our experiments.

Hyperparameter	**Value**
Diffusion Step Embedding Dimension	64
UNet Downsampling Dimensions	[64, 128, 256]
Optimizer	AdamW
Weight Decay	1e-6
Learning Rate	1e-4
Learning Rate Scheduler	Cosine
EMA Model Update	0.9999
Online Evaluation Episodes	1000

Table 14: We list the observation horizon, action executation horizon and action prediction horizon of Diffusion Policy baseline for each task.

Task	**Obs T_o**	**Act Exec T_a**	**Act Pred T_p**
ManiSkill3: PushT (Visual)	2	2	16
ManiSkill3: RollBall (Visual)	2	4	16
ManiSkill2: StackCube (Visual)	2	8	16
Adroit: Pen (Visual)	2	8	16
Adroit: Hammer (Visual)	2	8	16
Adroit: Door (Visual)	2	8	16
ManiSkill3: PushT (State)	2	1	16
ManiSkill3: RollBall (State)	2	4	16
ManiSkill2: StackCube (State)	2	8	16
ManiSkill2: TurnFaucet (State)	2	8	16
Adroit: Door (State)	2	8	16
Adroit: Pen (State)	2	8	16
Adroit: Hammer (State)	2	8	16
Adroit: Relocate (State)	2	8	16

Table 15: Evaluation on simpler tasks (Regular Group) not requiring responsive control from ManiSkill and Adroit benchmarks (Visual Observations). We follow our evaluation metric and report values under the same settings as in Table 4. Tasks in which none of the methods achieve a reasonable success rate under visual observations are omitted.

Method	**Steps (S)**	**NFEs/a**	**StackCube**	**Pen**	**Hammer**	**Avg. SR of tasks**
DDPM	**100**	**12.5**	**0.958**	**0.133**	**0.123**	**0.404**
DDIM	1	0.125	0.000	0.000	0.000	0.000
	2	0.25	0.946	0.042	0.000	0.329
	4	0.5	**0.947**	0.125	0.000	0.357
	8	1	0.946	0.139	0.009	0.365
EDM	80	20	0.930	0.156	0.067	0.384
CP	1	0.125	0.615	0.127	0.088	0.277
	8	1	0.910	**0.161**	0.077	0.383
RNR-DP	1	1	0.924	0.154	**0.110**	**0.396**

C.2 Comparsion with Streaming Diffusion Policy (SDP)

Streaming Diffusion Policy (SDP) (Høeg et al., 2024) is a recent advancement over Diffusion Policy that stays close to our approach. In this section, we compare our method with SDP in terms of motivation (Appendix C.2.1), methodology (Appendix C.2.2), and empirical results (Appendix C.2.3).

C.2.1 Motivation Comparison

SDP accelerates Diffusion Policy inference by reducing the number of denoising steps required to generate an action sequence. While improving diffusion inference speed is a relevant research topic, its impact in robotics is less compelling, as DDIM and Consistency Policy already provide reasonable speedups with strong performance. In contrast, our method addresses a fundamental limitation of Diffusion Policy—its lack of responsiveness—which significantly hinders performance in rapidly changing environments (e.g., contact-rich dynamic object manipulation). This challenge is far more critical to advancing robotic control. Although our approach also serves as an effective acceleration method, we view this as a secondary benefit compared to its primary advantage of enabling more responsive control.

C.2.2 Method Comparison

Rollout Method SDP also employs an action buffer structure but partitions the prediction horizon T_p into multiple action chunks, ensuring that (1) each chunk maintains the same noise level and (2) noise levels increase across chunks. This chunk-wise design focuses solely on reducing denoising steps and accelerating inference. However, it does not address responsiveness and thus retains the limitations of Diffusion Policy. In contrast, our sequential denoising scheme conditions all actions on the latest observations, enabling responsive control while leveraging the noise-relaying buffer to maintain efficiency.

Policy Architecture SDP fuses all time embeddings along the temporal dimension into a single embedding. In contrast, our architecture retains multiple time embeddings, ensuring that noisy actions within the noise-relaying buffer can perceive time step changes based on the latest observation features. This design preserves temporal dynamics, allowing each action to adapt to varying time steps, thereby improving responsiveness and consistency in action generation.

C.2.3 Empirical Results Comparison

To demonstrate that SDP inherits the same limitations as Diffusion Policy and lacks responsive control, we conduct experiments on the Adroit Relocate task (see Table 16). As the empirical results indicate, Streaming Diffusion Policy performs similarly to Diffusion Policy on the Adroit Relocate task, whereas our method achieves significantly more responsive control than both.

Table 16: We compare Streaming Diffusion Policy with Diffusion Policy and our method on Adroit Relocate task.

Task	DP	SDP	RNR-DP
Relocate (Adroit)	0.422	0.436	**0.585**