Rolling Diffusion Policy for Robotic Action Prediction: Enhancing Efficiency and Temporal Awareness

Chanhyuk Jung[1*], Sangwon Kim[2*], Dasom Ahn[1], In-su Jang[2],
Kwang-Ju Kim[2], Sungkeun Yoo[1], and Byoung Chul Ko[1]

***Abstract*— Diffusion models have shown strong potential for generating high-quality robotic action sequences, yet their iterative nature often incurs substantial computational cost. In this paper, we propose a novel Rolling Diffusion Policy (RDP) that accelerates diffusion-based control by reducing the number of iterative steps required for action generation. Our approach introduces a dynamic rolling mechanism that incrementally refines action trajectories while effectively capturing the temporal dependencies inherent in robotic systems. Integrating this mechanism into the diffusion policy framework enables faster inference while maintaining high performance. Extensive experiments on simulation benchmarks reveal that RDP achieves comparable or improved performance compared to conventional diffusion-based methods, paving the way for real-time applications in complex robotic environments.**

I. INTRODUCTION

Recently, diffusion models [3], [12], [2], [4], [10] have garnered significant attention for their ability to generate high-quality samples across various domains, including image, video, and audio generation. For robotic trajectory generation, it is crucial to model multimodal distributions. Diffusion models are particularly powerful in this regard, capable of expressing complex multimodal distributions and thus well-suited for robotic trajectory generation. From an action trajectory prediction perspective, diffusion policies offer two primary advantages:

1) *Multimodal distribution modeling*: As implicit policies, diffusion models can distribute probability densities around various action basins in a multimodal action space.
2) *Training stability*: Compared to energy-based models, which are often used for implicit policies, diffusion models offer greater training stability.

Despite these advantages, diffusion-based methods suffer from a fundamental limitation: they require a large number of iterations, leading to high computational costs. To address this, ongoing research has explored more efficient sampling methods. In particular, approaches such as consistency distillation [7] and and streaming diffusion policies [5] have been proposed to accelerate the sampling process [6], [9], [1]. However, these methods often result in performance degradation compared to standard diffusion policies (e.g., Diffusion Policy (DP) [1]) due to distillation compromising the quality of generated trajectories.

To our knowledge, no prior work has incorporated temporal invariance into these models. To overcome this limitation, we introduce the Rolling Diffusion Policy (RDP). This innovative approach enhances diffusion-based control by incorporating natural temporal invariance and dramatically accelerating the sampling process. RDP achieves this by reducing the number of function evaluations to just one—the minimum required. Furthermore, RDP introduces additional noise to future actions, reflecting the inherent uncertainty in robotic behavior. This design aligns naturally with the principle that future robotic actions are increasingly uncertain. Our key contributions are summarized as follows:

- **Enhanced Generative Precision:** RDP integrates a sliding-window denoising mechanism, preserving the strengths of diffusion policies while effectively modeling temporal dependencies.
- **Improved Inference Efficiency:** By utilizing rolling diffusion, RDP significantly reduces computational cost, enabling real-time action prediction.
- **Comprehensive Evaluation:** Extensive experiments on the Push-T benchmark demonstrate that RDP achieves superior performance and lower latency compared to existing methods.

II. BACKGROUND

A. Diffusion Models

Recently, diffusion models have gained significant attention in generative modeling due to their strong capability in producing high-quality images and robust feature representations. Initially introduced as a class of probabilistic generative models based on iterative denoising processes, they have demonstrated state-of-the-art performance in various computer vision tasks, including image synthesis, style transfer, and data augmentation. Works such as Denoising Diffusion Probabilistic Models (DDPM) [3] and accelerated variants like Denoising Diffusion Implicit Models (DDIM) [12] have refined sampling efficiency. Furthermore, diffusion models have been successfully adapted for text-to-image generation [8] where text is used to guide the diffusion process towards text aligned images.

Recent advancements have extended diffusion models to sequential data generation. Rolling Diffusion [11] introduced a novel approach tailored for generating sequential data,

* These authors contributed equally to this work
[1] Chanhyuk Jung, Dasom Ahn, Sungkeun Yoo, and Byoung Chul Ko are with the Keimyung University/CE/Computer Vision and Pattern Recognition Laboratory (CVPR Lab), Daegu 42601, South Korea. Email: {seagullcjung, tommydasomahn}@gmail.com, {skyoo, niceko}@kmu.ac.kr.
[2] Sangwon Kim, In-su Jang, and Kwang-Ju Kim are with the Electronics and Telecommunications Research Institute (ETRI), Daegu 42994, South Korea. Email: {eddiekim, jef1015, kwangju}@etri.re.kr.

such as videos and fluid dynamics simulations. Unlike conventional diffusion models that apply equal noise scales across all frames, Rolling Diffusion Models can capture temporal dependencies and can rollout frames indefinitely by progressively increasing noise levels across time. This enables the model to learn relationships between past and future frames.

B. Diffusion Policies

DP [1] constructs an implicit policy that models the action distribution with a diffusion model. Just as in DDPM [3], DP is trained through a process in which noise is added to an action trajectory $A = \{a_1, a_2, ..., a_t\}$ where it is progressively denoised using a denoising model. Visual observations from the environment serves as conditioning inputs for the generation process. This process ultimately recovers actions from gaussian noise. Here, observation $O = \{o_1, o_2, ..., o_t\}$ represents the current and past states of the robot and the visual observations from the cameras of the environment. Formally, the denoising diffusion process is defined as follows:

$$A_{t-1}^k = \alpha(A_t^k - \gamma\epsilon_\theta(O, A_t^k, t)) + \mathcal{N}(0, \sigma^2 I), \quad (1)$$

where t denotes the denoising timestep and k denotes the action index. First, noise is added to the trajectory and ϵ_θ, a neural network is used to predict the noise in the trajectory. This is then used to sample the denoised action trajectory A_{t-1}^k. α and γ monotonically increases or decreases across time to enforce a progressively decreasing noise scale. While this approach is highly effective for performance, generating actions is extremely slow to the point where real-time inference is impossible.

To enhance the inference speed of DP, Consistency Policy (CP) [7] utilizes consistency distillation techniques to distill a pretrained diffusion model into a smaller policy, enabling low-latency decision-making suitable for resource-constrained robotic systems. Streaming Diffusion Policy (SDP) [5] employs a streaming-based approach to generate partially denoised action trajectories, maintaining high-quality policy synthesis while significantly boosting inference speed, making it suitable for tasks requiring rapid decision-making.

III. METHOD

We propose a new policy called Rolling Diffusion Policy (RDP), which efficiently samples from the diffusion process. First, we reframe the standard diffusion process into three distinct regions: clean, window, and noise. We show that the standard diffusion process is equivalent to denoising the window section of the trajectory. Since they are equivalent, we can apply a sliding window approach to denoise trajectories of infinite length. Next, we focus on the sliding window portion and explain the necessary constraints on the noise scale. To satisfy these constraints, we design two separate diffusion processes: one for denoising the sliding window, and one for initializing it.

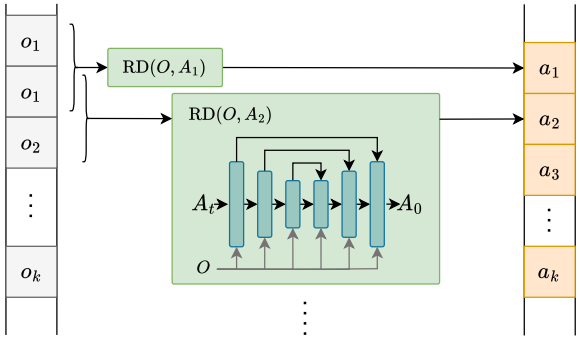

Fig. 1: The overall architecture of the proposed RDP (RD: Rolling Diffusion). Observation O and noisy action A are used as inputs to RD. In this process, RD generates one action per diffusion step.

A. Factorization

In the standard diffusion process (Equation (1), the same noise scale is applied uniformly across all actions. However, future actions are inherently more uncertain. To reflect this, the noise scale should monotonically increase as the action index increases. Thus, we can categorize regions of the trajectory into:

- Clean (no noise),
- Noise (fully noisy),
- Window (partially noisey, to be denoised).

This structure allows us to use a sliding window approach to denoise an infinite-length trajectory. As the window moves forward, it leaves behind clean actions that the robot can immediately execute. Formally, we define:

$$\begin{aligned} \text{clean}(s,t) &:= \{k | s_k = t_k = 0\}, \\ \text{noise}(s,t) &:= \{k | s_k = t_k = 1\}, \\ \text{window}(s,t) &:= \{k | s_k \in [0,1), t_k \in (s_k, 1]\}, \end{aligned} \quad (2)$$

where t is the starting diffusion timestep, s is the destination diffusion timestep, and s_k, t_k are the reparemeterized timesteps for each action index k. The overall denoising process can then be factorized as:

$$q(A_s|A_t) = q(A_s^{\text{clean}}|A_t)q(A_s^{\text{noise}}|A_t)q(x_s^{\text{window}}|A_t). \quad (3)$$

The denoising process $q(A_s|A_t)$ denoises A_t to A_s where $s < t$. This process is factorized into clean, noise, and window sections. the first section which is the clean section can be expressed as follows:

$$q(A_s^{\text{clean}}|A_t) = \prod_{k \in \text{clean}(s,t)} \delta(A_s^k | A_t^k). \quad (4)$$

If A_t^k is already clean, A_s^k which is supposed to be cleaner than A_t^k will also be clean, therefore the distribution can be expressed using a dirac-delta function. The second section which is the noise section is expressed as

TABLE I: Comparisons with state-of-the-art methods.

Models	Avg. score	Latency (ms)
Diffusion Policy (DP) [1]	0.91	110
Consistency Policy (CP) [7]	0.75	2
Streaming Diffusion Policy (SDP) [5]	0.84	7
Ours	**0.88**	**1**

$$q(A_s^{\text{noise}}|A_t) = \prod_{k \in \text{noise}(s,t)} \mathcal{N}(A_s^k|0, I). \tag{5}$$

As before, A_s^k is cleaner than A_t^k. If A_s^k is already noise, A_t^k must also be noise. Therefore, $q(A_s^{\text{noise}}|A_t)$ can be factorized into standard Gaussians. The final section is expressed as follows:

$$q(A_s^{\text{window}}|A_t) = \prod_{k \in \text{window}(s,t)} \mathcal{N}(A_s^k|\mu_\theta(A_t, t), \Sigma_\theta(A_t, t)). \tag{6}$$

The final section represents the most complex component among the three. Here, A_s^k must be cleaner than A_t^k , with no further simplifications or factorizations possible. Consequently, a model is required to learn this distribution directly. To achieve this, a standard diffusion process is employed, ensuring that A_s^k remains cleaner than A_t^k. throughout. Furthermore, since the factorization into clean, noise, and window components is derived under the condition $t_k \leq t_{k+1}$, this constraint must also be strictly maintained.

B. Sliding Window

Previously, we discussed which constraints must be met in order for the sliding window method can be used. First of all, $t_k \leq t_{k+1}$ must be met in order for the factorization to be valid. Second, A_s^k must be cleaner than A_t^k which is easier to satisfy as the standard diffusion process also satisfies this requirement. In order to satisfy the first constraint, monotonically increasing noise scale must be used for each action. Therefore, each action should have a different noise scale. In other words, each action has a different timestep which increases monotonically. Then the timesteps in the sliding window must be reparametrized to the sliding window index w:

$$t_k \to t_w = \frac{w + t}{W}, \tag{7}$$

where W denotes the window size and the reparametrization t_w is chosen so that the window can be shifted one index at a time. More specifically, as t changes from $t = 1$ to $t = 0$, the sliding window's timesteps changes from $[1/W, 2/W, ..., 1]$ to $[0, 1/W, ..., (W - 1)/W]$. Since the sliding window is partially denoised and only the first action is generated, we don't have to run the whole diffusion process. Therefore, we reduce the sampling process to the minimum required, which is a single evaluation. Using this reparametrization scheme, actions can be generated at each diffusion iteration which drastically improves inference time since we are only sampling from the diffusion process once instead of hundreds of steps.

Even though this reparametrization results in a much faster sampling process, a crucial problem exists: a partially denoised sliding window is required for this scheme to work. during the sliding window phase, this is not much of a problem but in the initial state, there is only noise. Therefore, a method is required to initialize the sliding window.

To initialize the sliding window we use the following reparametrization:

$$t_w \coloneqq \text{clip}(\frac{w}{W} + t). \tag{8}$$

Here, the $\text{clip}(\cdot)$ function clips values to $[0, 1]$. This reparametrization ensures that we can construct the sliding window from complete noise. We can see that as time t goes from $t = 1$ to $t = 0$, the sliding window goes from $[0, 0, ..., 0]$ to $[1/W, 2/W, ..., (W-1)/W]$ which connects to the sliding window section. Thus, we can continue denoising using the sliding window reparametrization scheme.

IV. EXPERIMENTS

In this section, we conduct experiments on various tasks to verify the performance of the proposed RDP. First, we evaluate the effectiveness of RDP by comparing it with existing diffusion-based policies. To this end, we evaluate the performance of RDP across various trajectory generation and control tasks, comparing latency and the quality of generated actions as the principal evaluation metrics.

A. Implementation Details

All code of RDP is implemented in PyTorch. We used the AdamW optimizer with a learning rate of 1×10^{-4} and a batch size of 128. Training and testing were conducted on a single NVIDIA Tesla V100 32GB GPU.

B. Evaluation Method

We evaluated the proposed RDP on the Push-T task, a widely-used benchmark for imitation learning and behavior cloning. The task consists of pushing a T-shaped block into a fixed T-shaped target. The goal of the task is to learn how to push the T-shaped block into the target. As the task is underactuated, handling various physical constraints such as friction and object dynamics is necessary. 200 expert demonstrations are given to train a policy to learn how to push the T-shaped block. While the Push-T environment offers both image-based and state-based settings, we focus solely on the image-based environment, reflecting our interest in visuomotor policies.

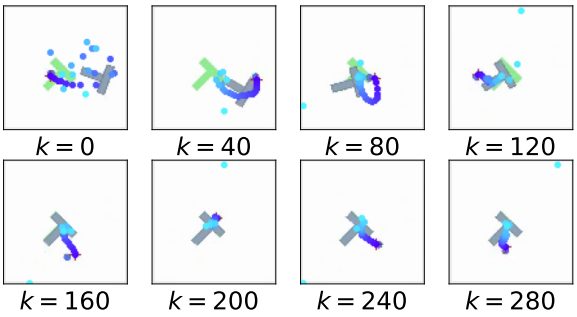

Fig. 2: Trajectory Quality of RDP. We visualize 300 step rollouts from the best checkpoint. Here, k denotes the action index starting from 0.

C. Comparisons with state-of-the-art methods

As shown in Table I, we ran comparative experiments with existing diffusion-based methods. The results demonstrate that the proposed RDP achieves competitive performance in the push-T task. This is due to RDP using a better sampling method that utilizes the sliding window method which denoises a partially noisy sliding window that is applied equally across time. Although the noise scale varies per action, our sampling process ensures that each action undergoes the entire diffusion process to generate high quality action trajectories.

Additionally, to further validate the efficiency of RDP, we compared latency against existing diffusion-based policies. As shown in Table I, RDP achieves the lowest latency among all tested methods. While SDP and CP demonstrated improved inference speeds compared to traditional DP, RDP surpasses all comparison methods in terms of speed. This superiority is attributed to RDP's ability to eliminate the iterative nature of diffusion sampling, resulting in significantly lower latency. Other methods require multiple function evaluations to produce a single action. Once initialized, RDP requires only a single function evaluation to generate an action, thereby enhancing its efficiency and practicality in robotic action prediction.

D. Visualization

In Fig. 2, the dots represent the trajectory predicted by our model with the blue color gradually lightening over time. When $k = 0$ the actions are generally noisy except for the first few actions which are clean. This can be attributed to the initalization process where the first few actions are trained to be clean and the rest are trained to have increasing levels of noise. Also, compared to the sliding window phase shown in $k > 0$, the actions are much noisier. The actions in $k > 0$ are almost clean where only the noise added at the end is noisy. During training only one-step rollouts are used, so this behavior is not explicitly encouraged. However, the model appears to converge to a noiseless sliding window. This is helpful for the model as a cleaner sliding window is much more easier to denoise as there is less noise to denoise. Furthermore, by leveraging temporal invariance, RDP smoothly predicts future trajectories, enhancing the realism and accuracy of the simulation.

V. CONCLUSION

In this study, we proposed the Rolling Diffusion Policy (RDP) to enhance the efficiency and performance of robotic trajectory generation. To address the high computational cost and lack of temporal invariance in conventional diffusion policies, we introduced a sliding-window denoising mechanism combined with a rolling diffusion process. RDP generates actions in a single step, effectively eliminating the expensive iterative function evaluations typically required by diffusion models, and significantly improving inference speed. Evaluations on the Push-T benchmark demonstrate that RDP outperforms existing diffusion-based policies in both action quality and latency. This work presents a novel approach that enhances the practicality and reliability of diffusion-based trajectory generation, contributing meaningfully to future developments in robot motion planning and autonomous systems. However, while RDP improves inference efficiency, its performance in more complex or dynamically changing environments remains an open question. Additionally, further investigations are needed to verify the consistency and stability of very long rollouts beyond the scale of the current experiments.

ACKNOWLEDGMENT

This work was supported by internal fund of Electronics and Telecommunications Research Institute (ETRI) [24BD1300, Development and Improvement of LLM/VLM-Based Humanoid Robot Interaction for Medical Assistance in Hospitals], ETRI grant funded by the Korean government [25ZD1120, Development of ICT Convergence Technology for Daegu-GyeongBuk Regional Industry], and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2022R1I1A3072904)

REFERENCES

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Robotics: Science and systems*, 2023.

[2] W. Harvey, S. Naderiparizi, V. Masrani, C. Weilbach, and F. Wood, "Flexible diffusion modeling of long videos," in *Advances in neural information processing systems*, 2022, pp. 27 953–27 965.

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

[4] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," in *Advances in neural information processing systems*, 2022, pp. 8633–8646.

[5] S. H. Høeg, Y. Du, and O. Egeland, "Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models," *arXiv preprint arXiv:2406.04806*, 2024.

[6] 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," in *Advances in neural information processing systems*, 2022.

[7] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency policy: Accelerated visuomotor policies via consistency distillation," *arXiv preprint arXiv:2405.07503*, 2024.

[8] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.
[9] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.
[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Computer vision and pattern recognition*, 2022, pp. 10 684–10 695.
[11] D. Ruhe, J. Heek, T. Salimans, and E. Hoogeboom, "Rolling diffusion models," in *International conference on machine learning*, 2024.
[12] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International conference on learning representations*, 2021.