Supplementary Materials – Adaptive Online Replanning with Diffusion Models

Siyuan Zhou[1], Yilun Du[2], Shun Zhang[3], Mengdi Xu[4], Yikang Shen[3], Wei Xiao[2], Dit-Yan Yeung[1], Chuang Gan[3,5]

[1]Hong Kong University of Science and Technology
[2]Massachusetts Institute of Technology, [3]MIT-IBM Watson AI Lab
[4]Carnegie Mellon University, [5]UMass Amherst

In the supplementary, we first discuss the experimental details and hyperparameters in Section A. Next, we analyze the impact of different numbers of diffusion steps N on the replanning process in Section B, and further present the visualization in RLBench in Section C. Finally, we discuss how to compute the likelihood in Section D.

A Experimental Details

1. Our architecture is built based on Diffuser [2] and Decision Diffuser [1]. In detail, our architecture comprises a temporal U-Net structure with six repeated residual networks. Each network consists of two temporal convolutions followed by GroupNorm [6], and a final Mish nonlinearity [4]. Additionally, We incorporate timestep and conditions embeddings, which are both 128-dimensional vectors produced by MLP, within each block.
2. In RLBench, we exploit the image encoder from the pre-trained CLIP [5], followed by a 2-layered MLP with 512 hidden units and Mish activations.
3. The model is trained using Adam optimizer [3] with a learning rate of $2\mathrm{e}{-04}$ and a batch size of 64 for $1\mathrm{e}6$ training steps.
4. The planning horizon is set to 128 in Maze2D//Multi2D U-Maze, 256 in Maze2D//Multi2D Medium, 256 in Maze2D//Multi2D Large, 64 in Stochastic Environments, and 64 in RLBench.
5. We use a threshold of 0.7 for **Replan from scratch** and a threshold of 0.5 for **Replan with future**.
6. The probability ϵ of random actions is set to 0.03 in Stochastic Environments.
7. The diffusion steps i for computing likelihood is set to $\{5, 10, 15\}$ in Maze2D and Stochastic Environments and $\{10, 20, 30\}$ in RLBench.
8. The total number of diffusion steps, corresponding to the number of diffusion steps for **Replan from scratch** is set to 256 in Maze2D, 200 in Stochastic Environments, and 400 in RLBench. And the number of diffusion steps for **Replan with future** is set to 80 in Maze2D, 40 in Stochastic Environments, and 100 in RLBench.
9. We perform the whole experiment with a total of three Tesla V100 GPUs.

Environment		**Diffuser**	**RDM** ($N = 40$)	**RDM (Ours)**	**RDM** ($N = 256$)
Multi2D	Large	129.4	160.9	195.4	197.7

Table 1: **Replanning with different diffusion steps.** RDM with $N = 40$ performs better than Diffuser. Our method RDM with $N = 80$ further improves performance and is much more efficient than RDM with $N = 256$.

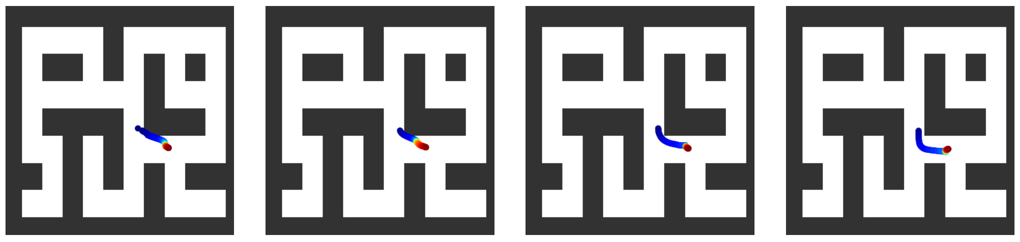

(a) Problematic Sample (b) $N = 40$ (c) $N = 80$ (d) $N = 256$

Figure 1: **Samples of replanning with different diffusion steps. (a)** demonstrates the problematic trajectory. **(b)** presents the sampled trajectory after replanning with $N = 40$. The sampled trajectory still leads to the collision. **(c)** presents the sampled trajectory after replanning with $N = 80$. The trajectory becomes feasible and avoids collision. **(d)** shows the sampled trajectory after replanning with $N = 256$. The trajectory improves quality but needs much more time.

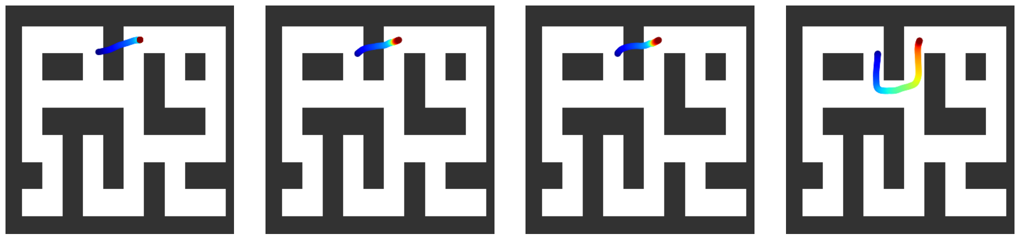

(a) Problematic Sample (b) $N = 40$ (c) $N = 80$ (d) $N = 256$

Figure 2: **Samples of replanning with different diffusion steps. (a)** demonstrates the problematic trajectory. **(b)(c)** presents the sampled trajectory after replanning with $N = 40$ and 80. The previously sampled trajectory is so troublesome that it can't recover from fast replanning. So the sampled trajectories still lead to the collision. **(d)** shows the sampled trajectory after replanning with $N = 256$. After **Replan from scratch**, the resampled trajectory becomes feasible.

B Different Diffusion Steps for Replanning

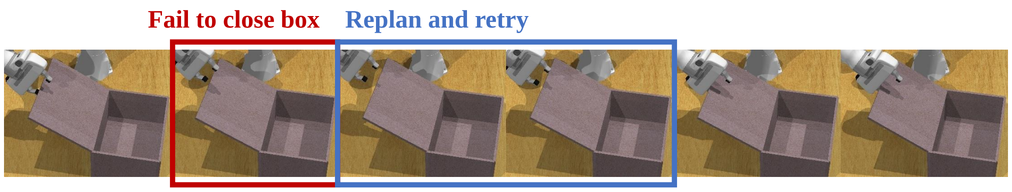
In this section, we visualize the impact of using different numbers of diffusion steps N to replan an existing trajectory. Figure 1 illustrates a problematic sampled trajectory after execution. When $N = 40$ of replanning, the resampled trajectory still leads to the collision with the wall. However, with $N = 80$, which aligns with the number of diffusion steps used in **Replan with future**, we observe that the sampled trajectory becomes feasible. For $N = 256$, corresponding to the number of diffusion steps used in **Replan from scratch**, the resampled trajectory shows additional improvement, but such a procedure is substantially more expensive than our replanning procedure.

We illustrate another example plan which based off likelihood would need to be planned from scratch in Figure 2. After replanning with $N = 40$ and $N = 80$, the trajectories still encounter issues with colliding with the wall. However, With $N = 256$, RDM successfully regenerates a completely different and feasible trajectory through replanning from scratch.

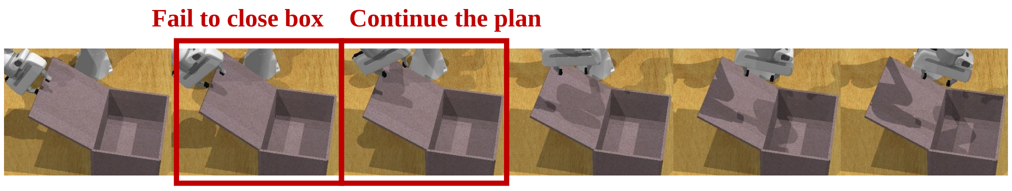
We further evaluate the performance with different replanning steps in Table 1. The results clearly demonstrate that that RDM consistently outperforms Diffuser in all cases. Remarkably, RDM with $N = 80$, corresponding to **Replan with future**, achieves performance close to RDM with $N = 256$, which corresponds to **Replan from scratch** which is substantially more expensive.

C Comparison between trajectories with and without replanning

In this section, we visualize the execution in the task `close box` in Figure 3. In the second frame of Figure 3, we observe that the robotic arm fails to close the box due to insufficient contact between the gripper and the lid. In Figure 3a, RDM successfully detects the failures and retries to establish proper contact with the lid. As a result, the gripper successfully closes the box this time. In contrast, Figure 3b demonstrates the scenario where the arm fails to notice the failures, making the arm persist

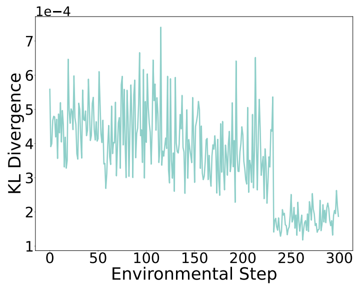


(a) With our replanning approach

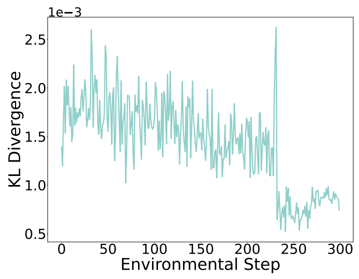


(b) Without our replanning approach

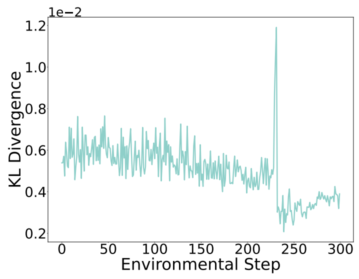
Figure 3: **Comparison between trajectories with and without replanning.** **(a)** Our replanning approach allows the agent to detect when the robotic arm fails to close the box and instructs it to retry. **(b)** Without our replanning approach, the failure goes unnoticed, leading the arm to persist in following the previous, unsuccessful plan.



(a) $i = 60$



(b) $i = 30$



(c) $i = 15$

Figure 4: **Different diffusion steps i for computing KL divergence.** We estimate the likelihood of trajectory by computing KL divergence between $q(\tau^{i-1} \mid \tau^{i}, \tau_{\rightarrow k})$ and $p_\theta(\tau^{i-1} \mid \tau^{i})$. Here, we illustrate the curve of KL divergence at different environmental steps in Maze2D where the number of total diffusion steps is 256. **(a)** When large amounts of noise are added at $i = 60$, the agent fails to detect any infeasibility as the KL divergence remains low and consistent. **(b)** When intermediate amounts of noise are added at $i = 30$, the agent has the ability to notice infeasibility but the value of KL divergence is similar to that of feasible trajectories. **(c)** When small amounts of noise are added at $i = 15$, the agent can effectively discern infeasibilities, as the KL divergence exhibits a clear boundary between feasible and infeasible trajectories.

in the unsuccessful plan. These results demonstrate the capability of RDM to detect failures and replan a successful trajectory in challenging robotic control tasks.

D How to Compute Likelihood

In this section, we analyze the number of noising steps we should use to estimate the likelihood. The likelihood estimation is performed by computing the KL divergence between $q(\tau^{i-1} \mid \tau^{i}, \tau_{\rightarrow k})$ and $p_\theta(\tau^{i-1} \mid \tau^{i})$ with different magnitudes of noise added at diffusion timesteps i. We analyze the impact of different diffusion steps i and the corresponding magnitude of noise added for computing KL divergence in Figure 4. We observe that (1) When large amounts of noise are added at $i = 60$ the agent fails to detect any infeasibility, as the KL divergence remains low and consistent. (2) When intermediate amounts of noise are added at $i = 30$, the agent has the ability to notice infeasibilities, but the KL divergence value is similar to that of a feasible trajectory. (3) When small amounts of noise are added at $i = 15$, the agent can effectively discern infeasibilities, as the KL divergence exhibits a clear boundary between feasible and infeasible trajectories. Based on these findings, we choose smaller values of i for computing the likelihood of trajectories and estimating when to replan.

E Additional Experimental Results

We show the additional experimental results in the following.

1. We show the results of the comparison between our RDM algorithm and the replanning-based baseline algorithms (SDM and PRDM) across more tasks. We run experiments using different intervals or thresholds and measure their computational costs by the total number of diffusion steps. Notably, RDM consistently outperforms all the baseline algorithms under the same total diffusion steps (that is, with the same computational budget).
2. We investigate different intervals I for replanning for Diffuser and Decision-Diffuser. We observe that as the interval decreases (that is replanning is done more frequently), the performance improves as we expect. However, when the interval is smaller than a certain value, the performance decreases significantly. For example, in the Maze2D Large domain shown in Figure VI, the return increases when the interval decreases from 250 to 100, while it drops when the interval decreases from 100 to 1. The ablation results confirm our statement that replanning with an interval that is too small (for example, replanning at every time step) may prevent successful task execution.
3. We also analyze the impact of different thresholds l for the baseline algorithms, SDM and RDM. As we expect, when l decreases (that is, when replanning is done more frequently), the performance of all the methods has improved. Despite the performance differences of the baseline algorithms, we want to emphasize again that under the same computational budget, our RDM algorithm outperforms all the baseline algorithms.

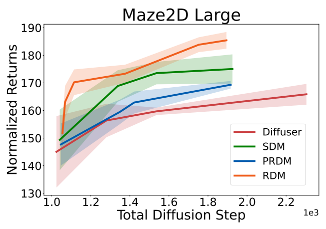


Figure 5: Performance vs. Total Diffusion Steps on Maze2D Large.

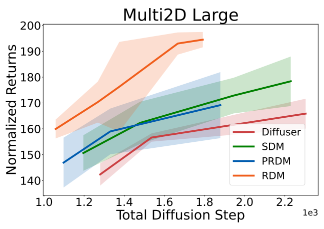


Figure 6: Performance vs. Total Diffusion Steps on Multi2D Large.

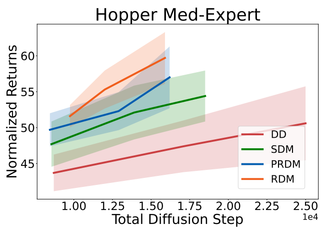


Figure 7: Performance vs. Total Diffusion Steps on Hopper Medium-Expert.

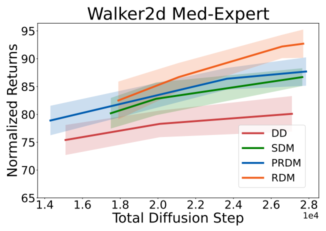


Figure 8: Performance vs. Total Diffusion Steps on Walker2d Medium-Expert.

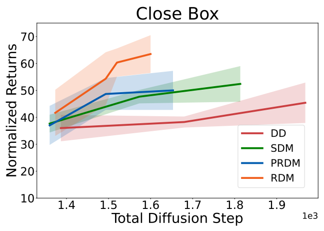


Figure 9: Performance vs. Total Diffusion Step on Close Box.

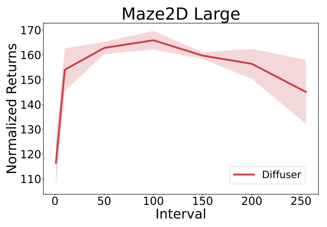


Figure 10: Different intervals for Diffuser on Maze2D Large.

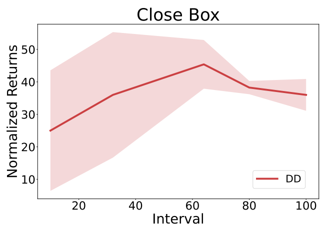


Figure 11: Different intervals for Decision Diffuser on Close Box.

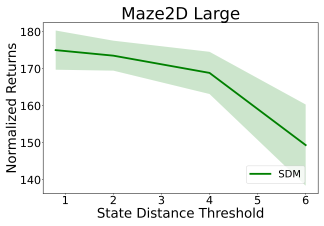


Figure 12: Different thresholds for SDM on Maze2D Large.

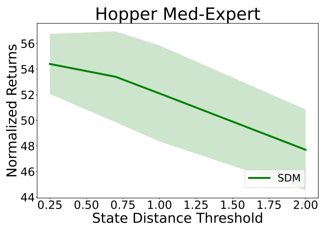


Figure 13: Different thresholds for SDM on Hopper Medium-Expert.

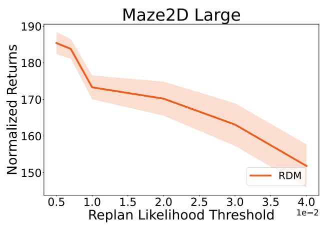


Figure 14: Different thresholds for RDM on Maze2D Large.

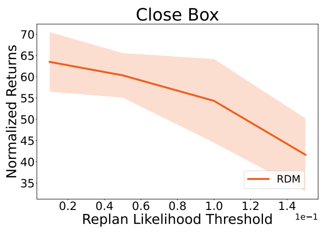


Figure 15: Different thresholds for RDM on Close Box.

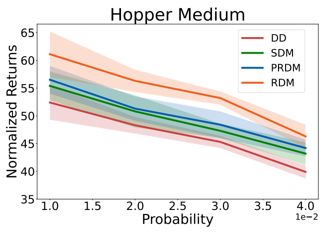


Figure 16: Different levels of stochasticity on Hopper Medium.

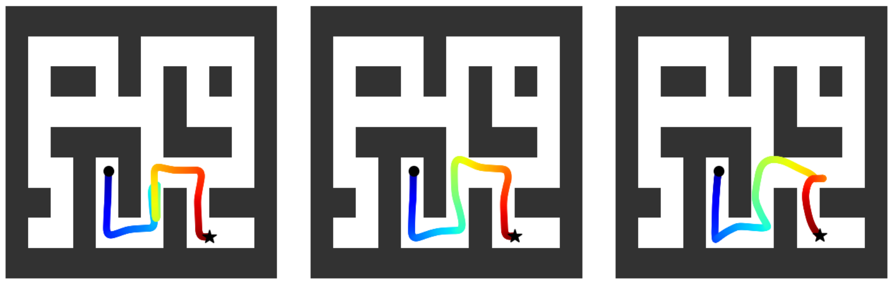

(a) interval=255 (b) interval=50 (c) interval=1

Figure 17: Visualization for Diffuser with different intervals. Replanning too frequently will cause some sub-optimal plans.

References

[1] 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.

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

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] D. Misra. Mish: A self regularized non-monotonic neural activation function. In *British Machine Vision Conference*, 2019.

[5] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[6] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision*, 2018.