

A Details of Datasets

We evaluate our method on the OGBench benchmark (Park et al., 2024a).¹ Since our primary goal is to assess trajectory stitching capability and long-horizon reasoning, we specifically utilize the Stitch and Explore datasets. As shown in Figure 5, the Stitch dataset is explicitly designed to challenge trajectory stitching ability, comprising short, goal-reaching trajectories limited to a maximum length of four cell units. Consequently, agents must effectively stitch together multiple short segments (up to eight) to successfully complete long-horizon tasks. In contrast, the Explore dataset is designed to test navigation skills learned from extensive yet low-quality trajectories. These trajectories are generated by commanding a low-level policy with random movement directions re-sampled every ten steps, along with significant action noise. Each demonstration trajectory typically spans only two to three blocks, resulting in noisy and clustered paths that pose additional challenges for evaluating the ability to learn effective policies from highly suboptimal data.

Table 4: Dataset specifications.

Env	Type	Size	# Transitions	# Episodes	Data Episode Length
PointMaze	Stitch	Medium	1M	5,000	200
		Large	1M	5,000	200
		Giant	1M	5,000	200
AntMaze	Stitch	Medium	1M	5,000	200
		Large	1M	5,000	200
		Giant	1M	5,000	200
	Explore	Medium	5M	10,000	500
		Large	5M	10,000	500

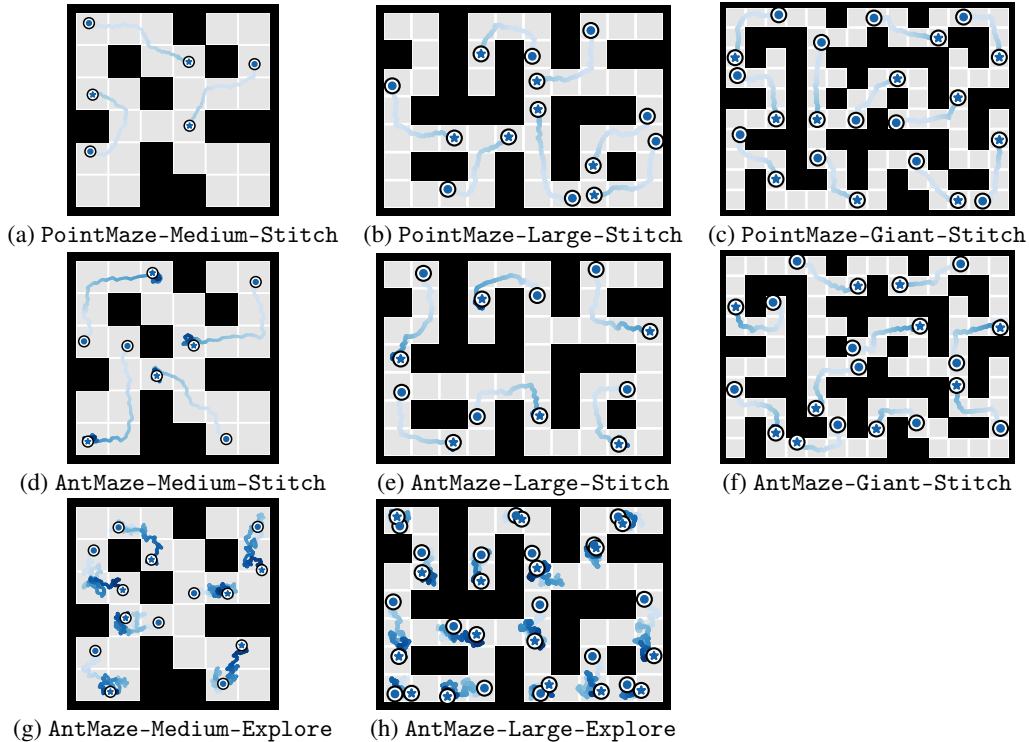


Figure 5: **Visualization of trajectories from OGBench datasets.** Each sub-figure illustrates example trajectories from different combinations of environments and datasets used in our experiments.

B Implementation Details

Network architecture. We utilize DiT1D (Peebles & Xie, 2023) as the neural network backbone for both the diffusion planner and the stitcher, due to its large receptive field and effectiveness in modeling trajectory-level dependencies. Following prior studies (Dong et al., 2023; Lu et al., 2025), we employ a DiT1D architecture with a hidden dimension of 256, a head dimension of 32, and a total of 8 DiT blocks consistently across all environments.

¹<https://github.com/seohongpark/ogbench>

Details of the low-level controller. A key challenge in diffusion-based planning is balancing global trajectory coherence with effective low-level control in high-dimensional state-action spaces (Chen et al., 2024a,b; Yoon et al., 2025). Previous approaches, such as PlanDQ (Chen et al., 2024b) and MCTD (Yoon et al., 2025), address this issue by integrating high-level diffusion planners with separately trained low-level controllers. Similarly, we adopt a hierarchical strategy, where the diffusion planner generates plans based primarily on compact, lower-dimensional state representations (e.g., positions of the agent itself), delegating the fine-grained, low-level action execution to a dedicated low-level controller. In our experiments, we specifically employ GCIQL (Kostrikov et al., 2022) as the learned low-level policy in the PointMaze environments and CRL (Eysenbach et al., 2022) in the AntMaze environments. A detailed visualization of generated subgoals and their corresponding execution rollouts can be seen in Figure 9. Furthermore, an ablation study examining the impact of the horizon length of the low-level controller is presented in Figure 6.

Implementation details for SCoTS and diffusion planning. In the temporal distance-preserving search stage of SCoTS, we retrieve the top $k = 10$ candidate segments based on their proximity in the learned latent embedding space during each stitching step. For computing the novelty score, we utilize a density estimator parameter $k_{\text{density}} = 30$ and set the novelty weighting factor $\beta = 2.0$ consistently across all tested environments. The horizon length for the diffusion-based stitcher is uniformly set to $H_{\text{stitcher}} = 26$.

To generate the augmented dataset \mathcal{D}_{aug} , we perform the stitching procedure N_{stitch} iterations per trajectory, creating a total of N_{traj} trajectories, thus ensuring the augmented dataset comprises approximately 5 million transitions. Specifically, in the AntMaze-Large-Stitch environment, we set $N_{\text{stitch}} = 40$ and $N_{\text{traj}} = 5000$.

For configuring the Hierarchical Diffusion (HD) planner (Chen et al., 2024c), parameters are adapted according to the properties of the training data. When training on the original `Stitch` and `Explore` datasets, which contain inherently shorter trajectories (as detailed in Table 4, column "Data Episode Length"), we set the high-level planning horizon to 101 steps for `Stitch` and 401 steps for `Explore`, both with temporal jumps of 26 steps between waypoints. However, when utilizing SCoTS-augmented datasets that feature longer and more diverse trajectories, we extend this planning horizon to 501 steps for `Medium` and `Large` environments, and to 1001 steps for `Giant` environments, maintaining the temporal jump of 26 steps. Similarly, for SCoTS-augmented `Explore` datasets, we also use a planning horizon of 1001 steps with 26-step jumps.

We apply jumpy denoising with DDIM sampling (Song et al., 2020) using 20 denoising steps across all environments. Additionally, we tune the replanning interval from the set $\{50, 100, 200\}$ steps and tune the horizon for the low-level controller from $\{5, 10, 15, 20, 25\}$. A full list of the hyperparameters is reported in Table 5.

Practical implementation of temporal distance-preserving search. Our SCoTS framework relies on a learned latent space \mathcal{Z} where the L_2 distance, $\|\phi(\mathbf{s}) - \phi(\mathbf{g})\|_2$, approximates the optimal temporal distance $d^*(\mathbf{s}, \mathbf{g})$ between states (as detailed in Section 3.1). A critical step in SCoTS is the efficient identification of suitable candidate trajectory segments from a large offline dataset \mathcal{D} . This requires a fast nearest neighbor search mechanism within the learned latent space \mathcal{Z} . To achieve this, we employ an Inverted File (IVF) index from the Faiss library (Douze et al., 2024), which is specifically designed for large-scale similarity searches.

The practical implementation of this search mechanism involves several stages. First, we prepare the data for indexing. This consists of computing the latent embeddings $\phi(\mathbf{s}_{\text{init}})$ for the initial states \mathbf{s}_{init} of all trajectories within the offline dataset \mathcal{D} . Let d denote the dimensionality of these latent embeddings. An IVF index is then constructed upon this collection of d -dimensional vectors. The construction process begins by partitioning the latent vectors into n_{list} clusters using the k -means algorithm. Each cluster is represented by a centroid $\mathbf{c}_j \in \{\mathbf{c}_1, \dots, \mathbf{c}_{n_{\text{list}}}\}$. Subsequently, each latent vector $\phi(\mathbf{s}_{\text{init}})$ in our collection is assigned to its nearest centroid, and for each centroid, an inverted list is maintained, storing references to the vectors belonging to its cluster.

During the temporal distance-preserving search phase of SCoTS (detailed in Algorithm 1, line 10), the latent embedding of the current composed trajectory endpoint, $\phi(\text{end}(\tau_{\text{comp}}))$, serves as the query vector \mathbf{q} . To find the k nearest neighbors for \mathbf{q} , the IVF index first identifies a limited set of clusters whose centroids $\{\mathbf{c}_j\}$ are closest to the query vector \mathbf{q} . The search for neighbors is then confined to

the latent vectors stored within the inverted lists corresponding to these selected clusters. This targeted approach significantly prunes the search space compared to an exhaustive search. Furthermore, the Faiss library provides support for GPU acceleration, which can further expedite this search process and enable efficient candidate retrieval. Once the k nearest latent embeddings corresponding to initial states of segments are identified, we retrieve the full original trajectory segments from \mathcal{D} to form the candidate set for the stitching process.

Computational resources and runtimes. All experiments were conducted using a single NVIDIA A10 GPU. The approximate execution times for each component of our method are as follows:

- Temporal distance-preserving embedding training: 1.5 hours
- Inverse dynamics model training: 0.25 hours
- Low-level controller training: 2.5 hours
- Diffusion-based stitcher training: 7 hours
- Trajectory augmentation via SCoTS: 0.5 hours
- Diffusion planner training: 18 hours

These times are per model training instance or data generation run and may vary slightly depending on the specific environment and dataset characteristics.

C Additional Results

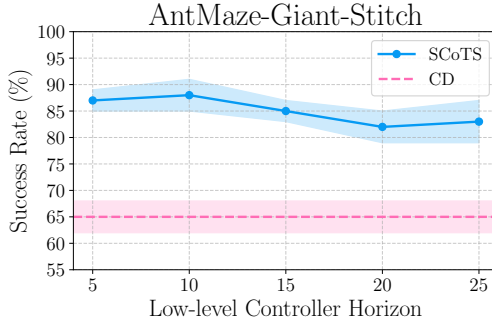


Figure 6: **Ablation study on low-level controller horizon.** Success rates in the AntMaze-Giant-Stitch environment comparing SCoTS against Compositional Diffuser (CD) (Luo et al., 2025), across various low-level controller horizon lengths.

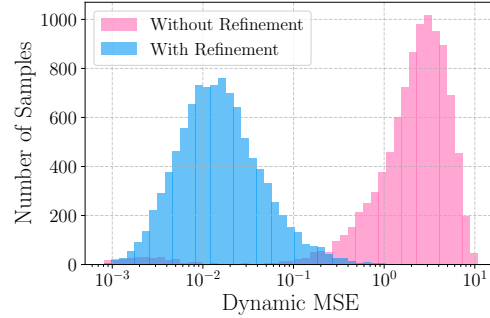


Figure 7: **Dynamic MSE comparison at stitching points.** Histograms showing the distributions of Dynamic MSE at trajectory stitching points in the AntMaze-Giant-Stitch environment, comparing results with and without the diffusion-based stitching refinement step.

Ablation study on low-level controller horizon. We investigate how the performance of our approach (SCoTS) is influenced by varying the horizon length of the low-level controller in the AntMaze-Giant-Stitch environment. As shown in Figure 6, SCoTS achieves consistently strong performance across different horizon lengths $H \in \{5, 10, 15, 20, 25\}$, outperforming the Compositional Diffuser (CD) (Luo et al., 2025). These results demonstrate that the diffusion planner trained with SCoTS generates highly feasible subgoals, maintaining robustness and effectiveness regardless of the chosen low-level execution horizon.

Effectiveness of diffusion-based stitching refinement. To further illustrate the effectiveness of the diffusion-based stitching refinement step in our SCoTS framework, we quantitatively evaluate its impact on dynamic consistency at stitching points. Specifically, we compute the *Dynamic Mean Squared Error (Dynamic MSE)* (Lu et al., 2023), defined as:

$$\text{Dynamic MSE} = \|f^*(s, a) - s'\|_2^2,$$

Table 5: **Hyperparameters for SCoTS.**

Component	Hyperparameter	Value	Tuning Choices
<i>SCoTS: Temporal Distance-Preserving Embedding (ϕ)</i>			
	Learning Rate	3×10^{-4}	-
	Latent Dimension	32	-
	Batch Size	1024	-
	Training Steps	1,000,000	-
	Network Backbone	MLP	-
	MLP Dimensions	(512, 512, 512)	-
	Expectile (ξ for ℓ_ξ^2)	0.95	-
<i>SCoTS: Inverse Dynamics Model (for actions in \mathcal{D}_{aug})</i>			
	Network Backbone	MLP	-
	MLP Dimensions	(256, 256, 256)	-
	Training Steps	200,000	-
<i>SCoTS: Stitching Process Parameters</i>			
	Top- k Candidates (Search)	10	-
	k_{density} (Novelty Score)	30	-
	Novelty Weight (β)	2.0	-
	Augmented Dataset Size	$\sim 5\text{M}$ transitions	-
	N_{stitch} (Stitches per Traj.)	Task-dependent (e.g., 40)	-
	N_{traj} (Generated Traj.)	Task-dependent (e.g., 5,000)	-
<i>SCoTS: Diffusion-based Stitcher ($p_\theta^{\text{stitcher}}$)</i>			
	Network Backbone	DiT1D	-
	Learning Rate	2×10^{-4}	-
	Weight Decay	1×10^{-5}	-
	Batch Size	64	-
	Training Steps	1,000,000	-
	Solver	DDIM	-
	Sampling Steps (DDIM)	20	-
	Horizon (H_{stitcher})	26	-
<i>Hierarchical Diffusion Planner (HD)</i>			
	Network Backbone	DiT1D	-
	Learning Rate	2×10^{-4}	-
	Weight Decay	1×10^{-5}	-
	Batch Size	64	-
	Training Steps	1,000,000	-
	Solver	DDIM	-
	Sampling Steps (DDIM)	20	-
	Plan Horizon (on original data)	101 (Stitch), 401 (Explore)	-
	Plan Horizon (on \mathcal{D}_{aug})	501 (M/L), 1001 (G/Explore)	-
	Temporal Jump	26	-
<i>Execution Parameters</i>			
	Low-level Controller Horizon	Tuned	{5, 10, 15, 20, 25}
	Replanning Interval	Tuned	{50, 100, 200}

591 which measures how closely the generated transitions adhere to the true environment dynamics f^* .
592 Figure 7 compares the distribution of Dynamic MSE at stitching points before and after applying
593 refinement on a logarithmic scale. Results clearly show that diffusion-based refinement substantially
594 reduces dynamic inconsistencies, highlighting its critical role in generating dynamically feasible and
595 coherent trajectories.

596 **Visualization of temporal distance-preserving latent representations.** We train temporal
597 distance-preserving latent representations with dimension 32 across all environments. To visu-
598 alize these learned representations, we apply a t -distributed stochastic neighbor embedding (t-SNE)
599 to project the 32-dimensional latent vectors onto a 2-dimensional plane, as shown in Figure 8. Recall
600 from Equation 6 that we parameterize a goal-conditioned value function $V(s, g)$ following (Park
601 et al., 2024b):

$$V(s, g) := -\|\phi(s) - \phi(g)\|_2, \quad (13)$$

602 which approximates the optimal goal-conditioned value function, defined as the maximum possible
603 return (cumulative sum of rewards) for sparse-reward settings. Specifically, an agent receives a
604 reward of 0 if the l_2 distance between states s and g is within a small threshold δ_g , and -1 otherwise.
605 The embedding function ϕ is trained using a temporal-difference objective inspired by implicit
606 Q-learning (Kostrikov et al., 2022) on the offline dataset \mathcal{D} . As illustrated in Figure 8, the learned
607 representations effectively capture the temporal proximity between states, resulting in latent spaces
608 where states that are temporally close in the environment are also clustered closely in the embedding
609 space.

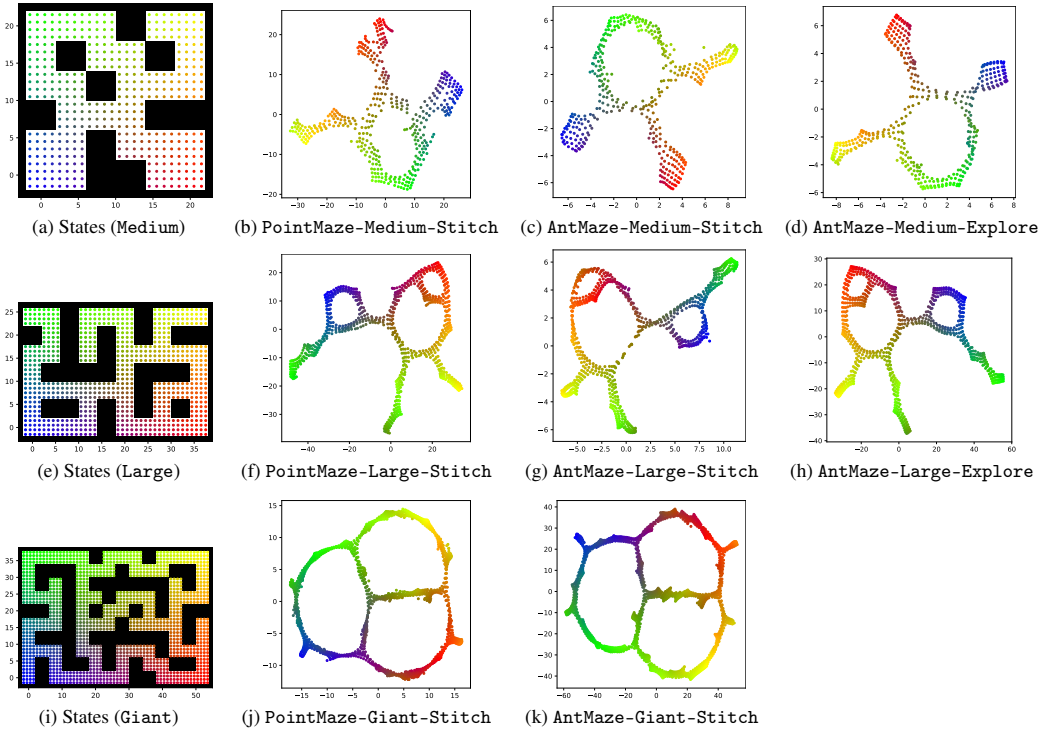


Figure 8: **Visualization of learned temporal distance-preserving latent representations.** The left-most column shows original states from maze environments of varying sizes (Medium, Large, Giant). Subsequent columns illustrate t-SNE projections of latent embeddings $\phi(s)$ for corresponding OG-Bench datasets, maintaining the same color scheme for consistency. This visualization demonstrates how spatial proximity and structure in the original state space are preserved and reflected in the learned latent representations.

610 **Visualization of rollout execution.** We visualize a generated plan by the diffusion plan-
611 ner trained on SCoTS-augmented data, along with its corresponding rollout execution in the
612 AntMaze-Giant-Stitch environment, as illustrated in Figure 9. The initial image (top-left) shows

the overall planned trajectory generated by the diffusion planner, with subgoals marked by green spheres. Subsequent images provide sequential snapshots from the rollout execution, demonstrating the agent actively pursuing and reaching these subgoals. This visualization highlights how effectively the generated high-level plan guides the low-level controller during task execution.

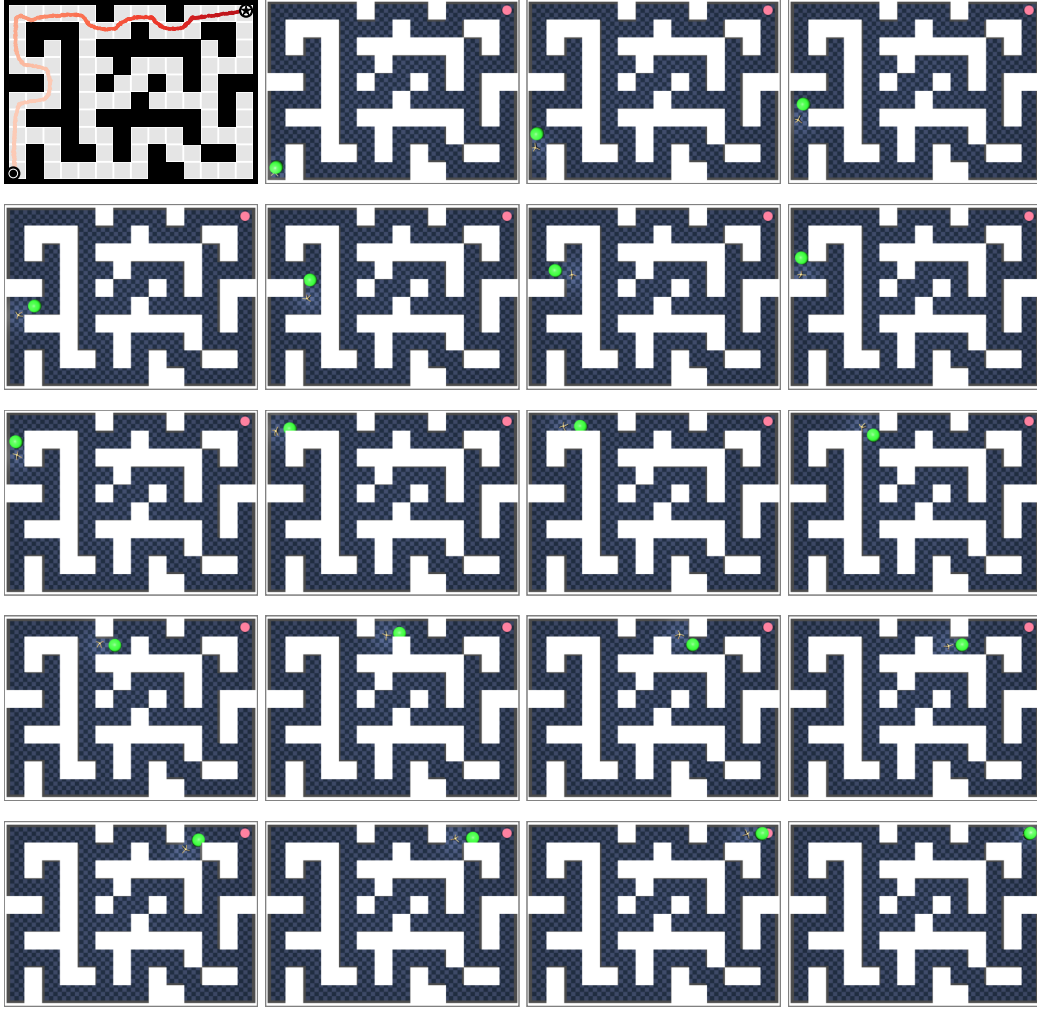


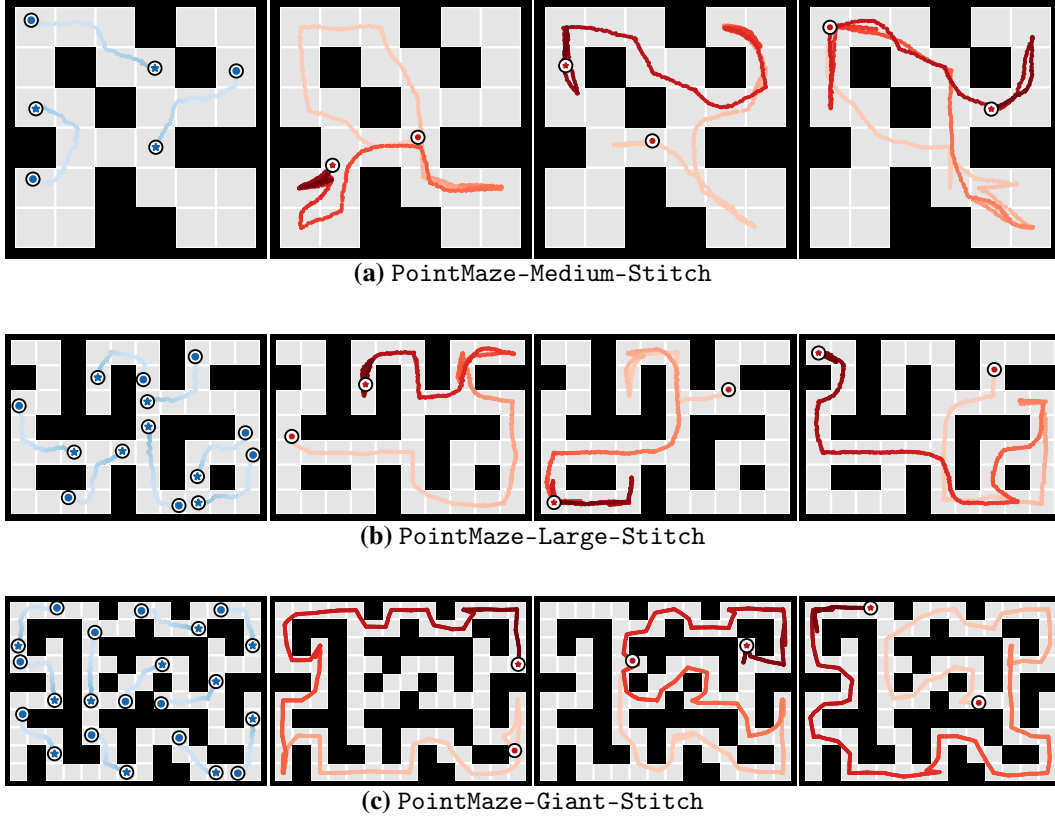
Figure 9: **Visualization of diffusion Planner rollout execution.** The top left image shows the planned trajectory generated by the diffusion planner, with subgoals marked by green spheres. Subsequent images sequentially illustrate the agent progressing toward these subgoals in the AntMaze-Giant-Stitch environment, demonstrating effective guidance provided by the generated plan.

Visualization of trajectories generated by SCoTS. In Figure 10, 11, and 12 we present representative examples of trajectories synthesized by our SCoTS framework across all considered environments and dataset types. Compared to the original trajectories provided in Figure 5, the SCoTS-generated trajectories clearly demonstrate extended coverage, illustrating the effectiveness of our method in augmenting the original offline datasets.

D Baseline Performance Sources

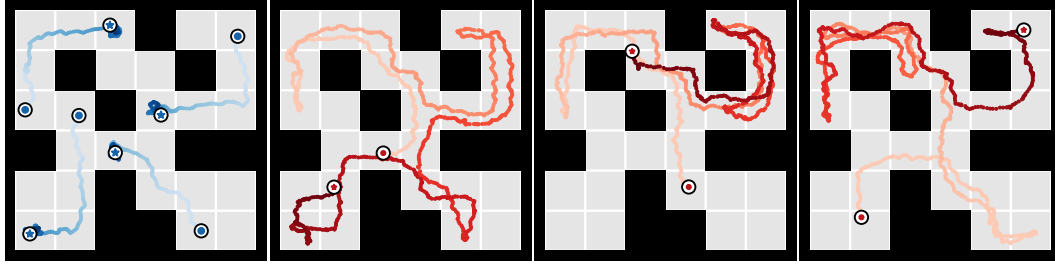
Performance scores reported for offline goal-conditioned reinforcement learning (GCRL) methods, including Goal-Conditioned Implicit Q-Learning (GCIQL) (Kostrikov et al., 2022), Quasimetric RL (QRL) (Wang et al., 2023), Contrastive RL (CRL) (Eysenbach et al., 2022), and Hierarchical Implicit

Figure 10: **SCoTS-augmented trajectories for PointMaze Stitch datasets.** For each PointMaze Stitch dataset, the leftmost column shows trajectories from the original OGBench dataset. The subsequent columns are examples of SCoTS-generated trajectories.

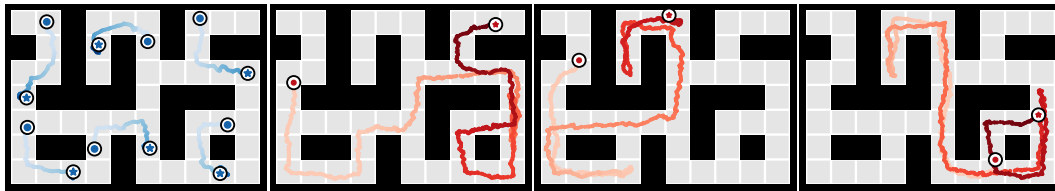


626 Q-Learning (HIQL) (Park et al., 2023a), are sourced from Table 2 in Park et al. (2024a). Scores for
 627 diffusion-based generative planning methods explicitly designed for long-horizon generalization,
 628 including Generative Skill Chaining (GSC) (Mishra et al., 2023) and Compositional Diffuser (CD)
 629 (Luo et al., 2025), are sourced from Tables 1 and 2 in Luo et al. (2025).

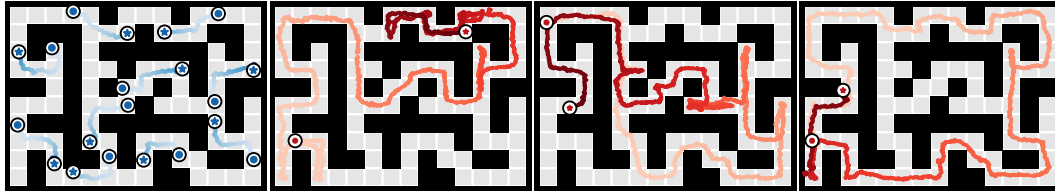
Figure 11: **SCoTS-augmented trajectories for AntMaze Stitch datasets.** For each AntMaze Stitch dataset, the leftmost column shows trajectories from the original OGBench dataset. The subsequent columns are examples of SCoTS-generated trajectories.



(d) AntMaze-Medium-Stitch



(e) AntMaze-Large-Stitch

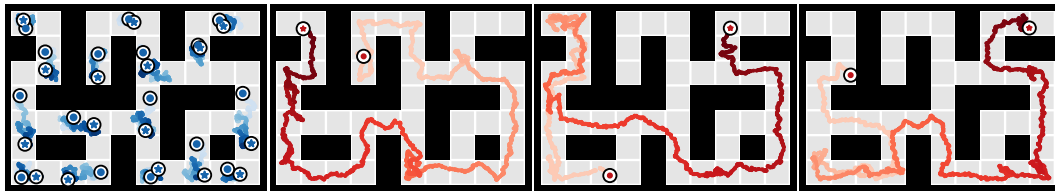


(f) AntMaze-Giant-Stitch

Figure 12: **SCoTS-augmented trajectories for AntMaze Explore datasets.** For each AntMaze Explore dataset, the leftmost column shows trajectories from the original OGBench dataset. The subsequent columns are examples of SCoTS-generated trajectories.



(g) AntMaze-Medium-Explore



(h) AntMaze-Large-Explore