Structured-Diffuser: Diffusion with Task-Conditioned Structured Priors for Motion Planning

anonymous author(s)

***Abstract*— We propose Structured-Diffuser, a diffusion planner that embeds task and motion structure directly into the noise model. Unlike standard diffusion-based planners that rely on zero-mean, isotropic Gaussian corruption, we introduce task-conditioned structured Gaussians whose means and covariances are derived from Gaussian Process Motion Planning (GPMP), explicitly encoding trajectory smoothness and task semantics in the prior. We first formulate diffusion under a task-conditioned, non-isotropic Gaussian prior with closed-form forward and posterior expressions. Building on this formulation, our hierarchical design separates prior instantiation from trajectory denoising. At the upper level, sparse task-centric key states and timings are obtained, which instantiate a structured Gaussian prior (mean and covariance). At the lower level, the full trajectory is denoised under this prior, treating the upper-level outputs as noisy observations. Experiments across three motion-planning tasks show improved task success and training efficiency, with additional gains in data efficiency, trajectory smoothness, and position–velocity consistency where evaluated. Ablation studies further show that explicitly structuring the corruption process provides benefits beyond neurally conditioning the denoising network alone. Overall, our approach concentrates the prior's probability mass around task-relevant, temporally structured trajectories. We additionally demonstrate deployment on a physical G1 humanoid. Project page: https://structured-diffuser.github.io.**

I. Introduction

Diffusion models have garnered increasing attention in robotic motion planning and shown strong flexibility across tasks [1], [2], [3], [4]. However, most existing approaches adopt a default corruption process using zero-mean, isotropic Gaussian noise, overlooking the inherent structure of robotic trajectories. In contrast to image or text data, robotic motion possesses intrinsic properties such as temporal smoothness (e.g., consistent evolution across time steps for dynamic feasibility) and can also be shaped by structured external information, such as task-imposed conditions (e.g., start and goal states or phase-specific constraints). These structured sources of information are typically incorporated through conditioning or guidance, rather than by shaping the corruption process itself [1], [3], [5], [6], [7].

In this work, we propose to encode such structure in the form of a task-aware noise model. This design enables our diffusion planner to encode smoothness and task constraints directly in the noise prior. We realize this idea with a two-level hierarchical diffusion framework. The upper-level planner produces sparse, task-centric key states (or key constraints). We treat these key states as soft observations with explicit uncertainty, allowing imperfect predictions to guide sampling without hard constraints or rejection. The lower-level diffusion then generates the full trajectory under a task-conditioned Gaussian, obtained by conditioning a linear time-varying Gaussian process (LTV-GP) motion prior on the key states, inspired by Gaussian process motion planning (GPMP) [8]. This yields a temporally correlated Gaussian whose mean is anchored by task-relevant key states and whose covariance encodes smooth motion between them. Accordingly, this framework produces temporally consistent and constraint-aware plans.

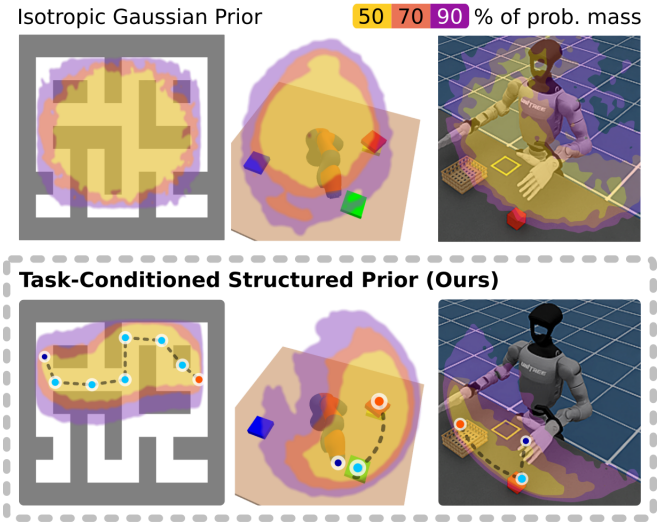

Fig. 1: **Prior probability mass comparison for Maze2D, KUKA block stacking, and G1 object-in-box tasks.** The yellow, orange, and purple regions contain the densest 50%, 70%, and 90% of the sampled probability mass, respectively. Isotropic Gaussian prior (top) spreads probability mass broadly, while our structured prior (bottom) concentrates it around task-relevant trajectory regions. Sparse, potentially imperfect key states (circles) from a lightweight upper-level module instantiate the structured prior.

Previous diffusion works typically keep the corruption isotropic and inject structural constraints through conditioning or guidance during inference [1], [2], [7]. For instance, conditioning has been used to incorporate visual context from observations or to inject goal information [3], [5]. Guidance mechanisms and auxiliary objectives incorporate dynamics, task rewards, or safety specifications into the generative process [9], [10]. In addition to conditioning and guidance, task-centric representations have been leveraged to encode constraints, for example object-centric demonstrations [11] or SE(3)-based cost fields that couple grasp and motion optimization [12]. Projection-based methods have

also been explored, where diffusion samples are mapped into feasible sets such as dynamically admissible trajectories [13] or collision-free multi-robot paths [14]. Beyond these categories, diffusion models have also been integrated with model-based plan [15] or with neural dynamics models [16].

Structured corruptions and source distributions have been explored more extensively outside robotics, while their use in robotic generative models remains comparatively underexplored. Examples include non-Gaussian but zero-mean choices [17], per-pixel variance schedules without temporal or robotics priors [18], and non-isotropic formulations that capture joint dependencies in human pose but still assume zero-mean noise and omit temporal correlations along the horizon, limiting their use for trajectory state conditioning [19]. Mixture priors have also been proposed for multi-modality and controllability, where Gaussian mixture parameters may be predefined through clustering [20] or heuristic structure [21], learned end-to-end to adaptively cover target support [22], or dynamically adjusted during inference by solver-based approaches [23].

Recent robotics works have begun to explore this design axis through structured or informative diffusion priors. Examples include coarse trajectory predictions for diffusion-based refinement [24], diffusion bridges initialized from informative prior actions [25], and learned diagonal Gaussian source distributions over action chunks [26]. Other approaches perform diffusion in continuous function space with temporally correlated noise [27]. While these works demonstrate the benefit of structured priors, they do not jointly encode task-specific key constraints and temporal motion structure in a full-trajectory prior. In contrast, our method constructs a structured Gaussian prior over the full trajectory by conditioning an LTV-GP on uncertain key states and timings, yielding a task-aware mean and temporally coupled covariance.

Hierarchical diffusion has been used to separate coarse-level decisions from finer-level motion generation through subgoals, contact plans, kinematic targets, or coarse-to-fine refinement [28], [29], [30], [31], [32]. These approaches typically pass high-level outputs as neural conditioning while retaining isotropic low-level corruption. Our hierarchy instead uses the upper-level outputs to reparameterize the low-level diffusion prior itself. This yields a trajectory posterior whose mean interpolates the key states and whose covariance couples time steps, so reverse diffusion operates in a Mahalanobis geometry that biases sampling to trajectories consistent between key states while modeling key-state uncertainty. This improves feasibility without post hoc constraints or rejection.

We evaluate our approach on three domains with increasing task and embodiment complexity: Maze2D goal reaching and KUKA block stacking from the Diffuser benchmarks [2], as well as an object-in-box manipulation task with the Unitree G1 humanoid.

Our contributions are summarized as follows:

- A two-level planning framework that decomposes motion generation into sparse task-structure prediction and full-trajectory generation. The upper level provides task-centric key states and timings, which are treated as uncertain soft observations to construct a smooth trajectory prior for the lower level.
- A task-conditioned diffusion formulation that uses the resulting GP posterior $(\xi, \mathcal{K})$ to parameterize the corruption and reverse processes, rather than using key states only as neural conditioning. The formulation admits closed-form forward marginals and posteriors and induces a Mahalanobis training objective.
- Experiments across three tasks show consistent gains over vanilla Diffuser, competitive or improved performance over conditioning- and guidance-based alternatives, and improved training and data efficiency.
- A real-hardware demonstration on Unitree G1, using a diffusion planner trained entirely on simulated data, with only lightweight real-world adaptation of the upper-level key-state predictor.

II. Preliminary

A. DDPM for motion planning

Let a robot trajectory be a stacked sequence of states $\tau = [x_1; \dots; x_H] \in \mathbb{R}^{H \cdot d}$, where $x_t \in \mathbb{R}^d$ is the robot state at time step t and H is the planning horizon. We denote by τ^i the trajectory after i forward diffusion steps.

Diffusion models learn to denoise trajectories that are gradually corrupted by Gaussian noise. Under the standard DDPM corruption [33], the forward process is

$$\tau^i = \sqrt{\alpha_i}\,\tau^{i-1} + \sqrt{1-\alpha_i}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I), \tag{1}$$

where $\alpha_i := 1 - \beta_i$ for $i = 1, \dots, N$, $\{\beta_i\}_{i=1}^N \subset (0,1)$ is a scalar variance schedule, and N is the total number of diffusion steps. Let $\bar{\alpha}_i := \prod_{k=1}^{i} \alpha_k$. Then the closed-form marginal is

$$\tau^i \mid \tau^0 \sim \mathcal{N}\big(\sqrt{\bar{\alpha}_i}\,\tau^0,\ (1-\bar{\alpha}_i)I\big).$$

The one-step reverse posterior is also Gaussian,

$$q(\tau^{i-1} \mid \tau^i, \tau^0) = \mathcal{N}\big(\tau^{i-1};\ \hat{\mu}_i(\tau^i, \tau^0),\ \hat{\beta}_i I\big), \tag{2}$$

where $\hat{\mu}_i$ and $\hat{\beta}_i$ are the posterior mean and variance coefficient at the i^{th} diffusion step (closed forms omitted for simplicity).

A denoising network $\mu_\theta(\tau^i, i, \texttt{cond})$ is trained to predict the posterior mean with the objective

$$\mathcal{L}(\theta) = \mathbb{E}_q \left[\left\| \hat{\mu}_i - \mu_\theta(\tau^i, i, \texttt{cond}) \right\|_2^2 \right], \tag{3}$$

where `cond` may include initial and goal states, maps, waypoints, or other task variables. At test time, sampling runs the reverse Markov chain starting from $\tau^N \sim \mathcal{N}(0, I)$.

In Sec. III, we extend (1) by biasing the mean and allowing a non-isotropic covariance in the corruption process.

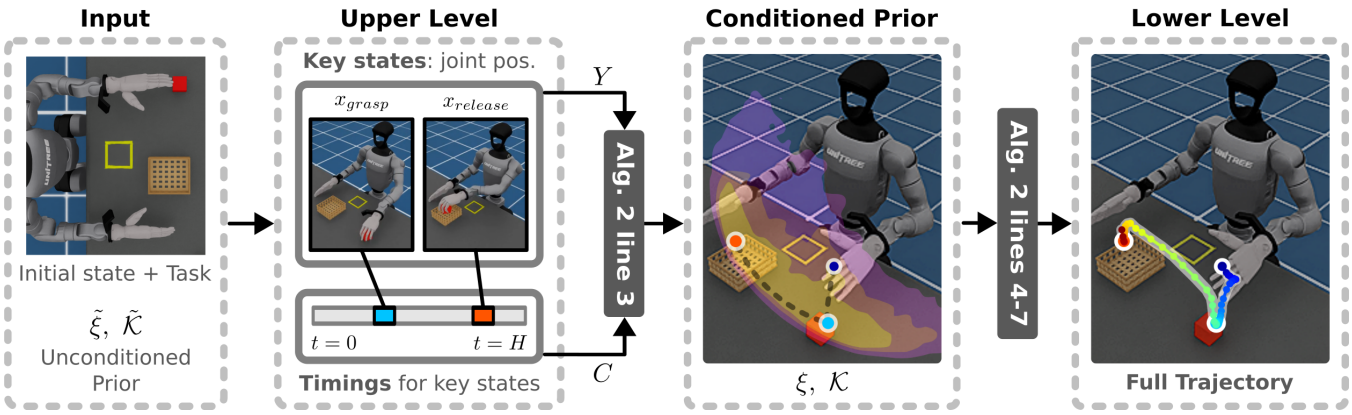

Fig. 2: **Trajectory generation pipeline.** Input: task goal and initial conditions. Upper level produces Y and C for the sparse key constraints $C \cdot \tau = Y$, using task-specific learned predictors, model-based components, and/or simple rules. (e.g., if the constraints are simple sparse waypoints, Y is a stack of key states and C is a selection matrix, which corresponds to designation of timings for the key states.) With (Y, C), we construct a Gaussian prior by conditioning a GP over trajectories, yielding $(\xi, \mathcal{K})$ via (8)–(9). The lower level then generates the full trajectory by reverse diffusion from $\mathcal{N}(\xi, \mathcal{K})$.

B. Gaussian Process Motion Planning

Gaussian Process Motion Planning (GPMP) models a continuous time trajectory as a Gaussian process (GP), i.e.,

$$x(t) \sim \mathcal{GP}(\tilde{\xi}(t), \tilde{\mathcal{K}}(t, t')), \tag{4}$$

whose mean and kernel are induced by a linear time varying (LTV) dynamics prior [8]. Consider the stochastic LTV model $\dot{x}(t) = A(t)x(t) + u(t) + w(t)$ where $u(t)$ is control input and $w(t)$ is zero mean white noise with spectral density $Q_C \succeq 0$. Let the initial mean and covariance at time t_0 be ξ_0 and $\mathcal{K}_0$, and let $\Phi(t, s)$ denote the state transition matrix of $A(\cdot)$. Then the GP has mean $\tilde{\xi}(t) = \Phi(t, t_0)\xi_0 + \int_{t_0}^{t} \Phi(t, s)u(s)\, ds$ and kernel $\tilde{\mathcal{K}}(t, t') = \Phi(t, t_0)\mathcal{K}_0\Phi(t', t_0)^\top + \int_{t_0}^{\min\{t,t'\}} \Phi(t, s)Q_C\Phi(t', s)^\top ds$, which admit sparse and exactly computable discrete counterparts.

Following [1], our work uses a discrete-time version of (4) derived from a constant velocity or constant acceleration model with step Δt. Writing the state at timestep t as x_t, the one-step evolution is $x_{t+1} = \Phi_{t,t+1}x_t + q_t$ with $q_t \sim \mathcal{N}(0, Q_{t,t+1})$, where $\Phi_{t,t+1} = \begin{bmatrix} I & \Delta t\, I \\ 0 & I \end{bmatrix}$ and $Q_{t,t+1} = \begin{bmatrix} \frac{1}{3}\Delta t^3 Q_C & \frac{1}{2}\Delta t^2 Q_C \\ \frac{1}{2}\Delta t^2 Q_C & \Delta t\, Q_C \end{bmatrix}$. The same structure applies, no matter if the state includes positions only or positions and velocities; velocities can be included explicitly or marginalized to obtain an equivalent GP prior over positions. In our experiments, Maze2D uses positions and velocities, whereas KUKA and G1 use joint positions only.

III. Method

We first generalize DDPM to a structured Gaussian corruption process with a nonzero mean and non-isotropic covariance (Sec. III-A). We then describe how our hierarchical planner constructs the structured prior and uses it for trajectory generation (Secs. III-B–III-D).

A. DDPM with a Structured Gaussian Prior

Let $\mathcal{D}$ be a dataset of robot trajectories where each clean sample $\tau^0 \in \mathcal{D}$ is associated with task specification that may admit multiple feasible solutions. Rather than corrupting trajectories toward an isotropic Gaussian $\mathcal{N}(0, I)$, we formulate diffusion toward a task-conditioned structured Gaussian $\mathcal{N}(\xi, \mathcal{K})$, with mean $\xi \in \mathbb{R}^{H \cdot d}$ and covariance $\mathcal{K} \in \mathbb{R}^{Hd \times Hd}$, where $\mathcal{K} \succ 0$. We first derive the corresponding forward and reverse processes for arbitrary $(\xi, \mathcal{K})$, and then describe how our hierarchical planner instantiates these parameters from task structure in the following subsections.

Reusing the schedule notation from Sec. II-A, with $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{k=1}^{i} \alpha_k$, we define

$$\tau^i \sim \mathcal{N}\Big(\sqrt{\alpha_i}\,\tau^{i-1} + (1 - \sqrt{\alpha_i})\,\xi,\ (1 - \alpha_i)\,\mathcal{K}\Big), \tag{5}$$

For fixed $(\xi, \mathcal{K})$, this process can equivalently be viewed as standard DDPM after centering by ξ and whitening according to $\mathcal{K}$. (5) yields closed form conditionals with respect to τ^0:

$$\tau^i \mid \tau^0 \sim \mathcal{N}\Big(\sqrt{\bar{\alpha}_i}\,\tau^0 + (1 - \sqrt{\bar{\alpha}_i})\,\xi,\ (1 - \bar{\alpha}_i)\,\mathcal{K}\Big). \tag{6}$$

The one step backward posterior is also Gaussian,

$$q(\tau^{i-1} \mid \tau^i, \tau^0) = \mathcal{N}\big(\tau^{i-1};\, \tilde{\mu}_i(\tau^i, \tau^0, \xi),\, \tilde{\beta}_i\,\mathcal{K}\big), \tag{7}$$

where

$$\tilde{\mu}_i(\tau^i, \tau^0, \xi) := \frac{\sqrt{\bar{\alpha}_{i-1}}\,\beta_i}{1 - \bar{\alpha}_i}\,\tau^0 + \frac{\sqrt{\alpha_i}\,(1 - \bar{\alpha}_{i-1})}{1 - \bar{\alpha}_i}\,\tau^i + \eta_i \xi,$$

$$\eta_i := \frac{1}{1 - \bar{\alpha}_i}\left[\beta_i\big(1 - \sqrt{\bar{\alpha}_{i-1}}\big) - \sqrt{\alpha_i}\,(1 - \bar{\alpha}_{i-1})\big(1 - \sqrt{\alpha_i}\big)\right],$$

$$\tilde{\beta}_i = \left(\frac{\alpha_i}{1 - \alpha_i} + \frac{1}{1 - \bar{\alpha}_{i-1}}\right)^{-1}$$

These are generalized forms of the standard DDPM posterior mean and variance coefficient. (derivations are given in the Appendix). As $\bar{\alpha}_N \to 0$, the terminal distribution approaches

$$\tau^N \sim \mathcal{N}(\xi, \mathcal{K}).$$

In our framework, $(\xi, \mathcal{K})$ are not fixed globally, but are instantiated for each task instance using a two-level hierarchy based on GPMP. The upper level produces sparse task-relevant key states and their timings, which are used to condition a GPMP motion prior and obtain the task-conditioned mean ξ and covariance $\mathcal{K}$ (Sec. III-C). The resulting prior captures temporal smoothness while incorporating task-specific constraints with explicit uncertainty. The lower level then generates the full trajectory by reverse diffusion under this structured prior. The inputs are a task specification and an initial condition. An overview of the framework is shown in Fig. 2. The next subsections detail each level.

B. Upper level: key states and timings

Given a full clean trajectory $\tau = [x_1; \dots; x_H] \in \mathbb{R}^{Hd}$, which stacks d-dimensional states over a horizon of H, let $Y = [y_1; \dots; y_n] \in \mathbb{R}^{nd}$ denote a stack of n sparse states that are relevant for task completion, such as waypoints or states at contact events. We refer to these as *key states*.

The temporal locations of the key states are represented by a binary selection matrix $C \in \{0,1\}^{nd \times Hd}$, which extracts the sparse states from the full trajectory:

$$Y = C\tau.$$

Thus, C specifies the timings of the key states.

The upper level provides the pair (C, Y). How (C, Y) is obtained is task-dependent, as described in Sec. IV. When a learned predictor is used, it is trained with task-specific supervision on C, Y, or both.

C. GPMP prior construction

Given the upper-level output (C, Y), we construct a task-conditioned trajectory prior by conditioning the unconditioned GPMP prior introduced in Sec. II-B. After discretization over the planning horizon, this prior is $\tau \sim \mathcal{N}(\tilde{\xi}, \tilde{\mathcal{K}})$, where $\tilde{\xi} = [\tilde{\xi}_1; \dots; \tilde{\xi}_H] \in \mathbb{R}^{Hd}$ and $\tilde{\mathcal{K}} = [\tilde{\mathcal{K}}(t_i, t_j)]_{1 \le i,j \le H}$, with t_i denoting the ith discretized time step.

We treat the predicted key states Y as soft observations of the trajectory at the timings specified by C:

$$Y = C\tau + \epsilon_y, \qquad \epsilon_y \sim \mathcal{N}(0, \mathcal{K}_y),$$

where $\mathcal{K}_y \succeq 0$ represents the uncertainty of the key states.

Conditioning the unconditioned GPMP prior on the soft observations $(Y, C, \mathcal{K}_y)$ yields a *task conditioned* posterior over the full trajectory with a closed form mean and covariance [8]

$$\xi = \tilde{\xi} + \widetilde{\mathcal{K}}\, C^\top \left(C\, \widetilde{\mathcal{K}}\, C^\top + \mathcal{K}_y \right)^{-1} \left(Y - C\, \tilde{\xi} \right), \quad (8)$$

$$\mathcal{K} = \widetilde{\mathcal{K}} - \widetilde{\mathcal{K}}\, C^\top \left(C\, \widetilde{\mathcal{K}}\, C^\top + \mathcal{K}_y \right)^{-1} C\, \widetilde{\mathcal{K}}. \quad (9)$$

This posterior $\mathcal{N}(\xi, \mathcal{K})$ serves as the task-conditioned prior for the lower-level diffusion model. Small variances in $\mathcal{K}_y$ enforce tight adherence to key states (e.g., fixed initial or goal states), while larger variances allow flexibility (e.g., uncertain waypoints). Thus, Y anchors the trajectory near important states and $\mathcal{K}_y$ encodes confidence in those anchors, without imposing hard constraints.

D. Lower level: trajectory generation by denoising

The conditioned prior $\mathcal{N}(\xi, \mathcal{K})$, constructed as described in Secs. III-B–III-C, determines the task-conditioned noise geometry of the lower-level diffusion. At inference, the lower level initializes $\tau^N \sim \mathcal{N}(\xi, \mathcal{K})$ and iteratively samples from the learned reverse transitions to obtain the final trajectory τ^0. Following the standard DDPM parameterization, we model the learned reverse transition as

$$p_\theta(\tau^{i-1} \mid \tau^i, \mathtt{cond}) = \mathcal{N}\Big(\tau^{i-1}; \mu_\theta(\tau^i, i, \mathtt{cond}), \tilde{\beta}_i \mathcal{K}\Big),$$

where $\mathtt{cond}$ denotes the task information provided to the denoising network. We fix the reverse covariance to the analytical posterior covariance $\tilde{\beta}_i \mathcal{K}$ in (7) and learn the posterior mean with μ_θ. The corresponding variational objective [33] minimizes the KL divergence between the true and learned reverse distributions. Because both use the covariance $\tilde{\beta}_i \mathcal{K}$, their KL divergence reduces to

$$D_{\mathrm{KL}}(q \,\|\, p_\theta) = \frac{1}{2\tilde{\beta}_i} \left\| \tilde{\mu}_i - \mu_\theta(\tau^i, i, \mathtt{cond}) \right\|^2_{\mathcal{K}^{-1}}, \quad (10)$$

up to terms independent of θ, where $\|v\|_A^2 := v^\top A v$. Thus, omitting the scalar schedule-dependent factor, we train with the Mahalanobis MSE

$$L(\theta) = \mathbb{E}_q \left[\left\| \tilde{\mu}_i - \mu_\theta(\tau^i, i, \mathtt{cond}) \right\|^2_{\mathcal{K}^{-1}} \right]. \quad (11)$$

Because $\mathcal{K}$ is the covariance of the conditioned GP prior, this objective weights prediction errors according to its task-dependent uncertainty and temporal correlation structure.

During lower-level training, we use the ground-truth key states and timings (C, Y) as the nominal upper-level outputs and perturb them to account for prediction errors at test time. We perturb the key-state timings encoded by C and reconstruct the corresponding selection matrix C_{pert}, while perturbing the key states as $Y_{\mathrm{pert}} = Y + dY_\omega$, where dY_ω denotes a sampled key-state perturbation. The perturbed pair $(C_{\mathrm{pert}}, Y_{\mathrm{pert}})$ is then used to instantiate the task-conditioned prior $(\xi, \mathcal{K})$ via (8)–(9). We then update the lower-level diffusion parameters using the Mahalanobis MSE loss (11), as summarized in Alg. 1. The complete inference process, including both the upper and lower levels, is summarized in Alg. 2.

Remark 1: Computing covariance-related terms $\left(C\tilde{\mathcal{K}}C^\top + \mathcal{K}_y\right)^{-1}$, $\mathcal{K}$, and $\mathcal{K}^{-1}$ repeatedly can be costly. We therefore precompute the corresponding gain terms for all timing configurations used at training and inference. For example, in the KUKA manipulation task, we divide the horizon H into 10 bins and precompute gains for all grasp and release bin pairs $\binom{10}{2} = 45$, since only C changes with timing choices. This preserves time range conditioning while keeping online costs low.

Remark 2: Motion Planning Diffusion (MPD) [1] applies a GP as a guidance cost only during reverse sampling, while the forward corruption remains isotropic. We instead parameterize the noise with a GP: conditioning on key states sets the mean ξ and covariance $\mathcal{K}$ that govern both the forward and posterior distributions.

Algorithm 1 Lower-level training (for full trajectory)

Inputs: dataset $\mathcal{D}$, GP prior $(\tilde{\xi}, \widetilde{\mathcal{K}})$
Initialize lower-level diffusion module parameters θ
while not converged **do**
  Sample $\tau^0 \sim \mathcal{D}$ and extract (C, Y) from τ^0
  Perturb the key-state timings encoded by C and reconstruct C_{pert}; $Y_{\text{pert}} \leftarrow Y + dY_\omega$
  $(\xi, \mathcal{K}) \leftarrow$ GP-Post$(\tilde{\xi}, \widetilde{\mathcal{K}}, C_{\text{pert}}, Y_{\text{pert}}, \mathcal{K}_y)$ via (8)–(9)
  Sample $i \sim \text{Unif}\{1, \dots, N\}$
  $\tau^i \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_i}\tau^0 + (1 - \sqrt{\bar{\alpha}_i})\xi,\ (1 - \bar{\alpha}_i)\mathcal{K}\right)$
  Gradient descent $\nabla_\theta \left\| \tilde{\mu}_i(\tau^i, \tau^0, \xi) - \mu_\theta(\tau^i, i) \right\|^2_{\mathcal{K}^{-1}}$
end while

Algorithm 2 Test-time Planning (Inference)

Inputs: task specification, unconditioned GP prior $(\tilde{\xi}, \widetilde{\mathcal{K}})$
Upper level: produce (C, Y) ▷ using the task-specific key-state/timing predictor or heuristic
$(\xi, \mathcal{K}) \leftarrow$ GP-Post$(\tilde{\xi}, \widetilde{\mathcal{K}}, C, Y, \mathcal{K}_y)$ via (8)–(9)
Lower Level:
$\tau^N \sim \mathcal{N}(\xi, \mathcal{K})$
for $i = N, \dots, 1$ **do**
  $\tau^{i-1} \sim \mathcal{N}(\mu_\theta(\tau^i, i), \tilde{\beta}_i \mathcal{K})$
end for
return τ^0

IV. Experiments

A. Experimental Setup

We evaluate the proposed structured-noise formulation on three domains. We describe the benchmark tasks and how key states and timings are defined.

(i) Maze2D goal reaching. Each task specifies a start and goal state on a fixed maze. Key states are defined as evenly spaced temporal waypoints between them, so C is fixed by design because the waypoint timings are predetermined. The upper-level key-state predictor is a lightweight diffusion model trained to generate the waypoint set Y at the fixed timings C.

(ii) KUKA arm block stacking. The task is to grasp and stack blocks with a 7-DoF KUKA arm. The Diffuser benchmark provides joint-position trajectories, per-block binary contact flags, and block-position trajectories. Since block positions are determined from kinematics and contact, we train the models only on joint trajectories and contact flags. The key-state stack Y contains the initial, grasp, and release joint configurations, while their corresponding indices define C. A lightweight upper-level diffusion model predicts the contact-flag trajectory, from which grasp and release timings are extracted to construct C. For Y, we use the grasp and release joint configurations from the reference trajectory for each evaluation scene. Thus only the timings are predicted in this task, and Diffuser + KeyCond receives the same Y.

At test time, the grasp target is the currently grasped block if any, or otherwise the block nearest the initial end-effector position. The release target is the highest remaining block.

Method \Task	Maze2D	KUKA	G1
Diffuser	51.6 ± 6.2	23.6 ± 4.7	53.8 ± 6.3
Diffuser + KeyCond	59.4 ± 1.8	55.8 ± 4.2	**69.2 ± 4.3**
EDMP	**78.4 ± 2.9**	30.8 ± 4.5	59.0 ± 1.4
Structured-Diffuser	**74.4 ± 2.6**	**67.2 ± 2.5**	**69.6 ± 2.8**

TABLE I: Task success rate (%) in simulation (mean ± standard deviation over five seeds, with 100 trials per seed).

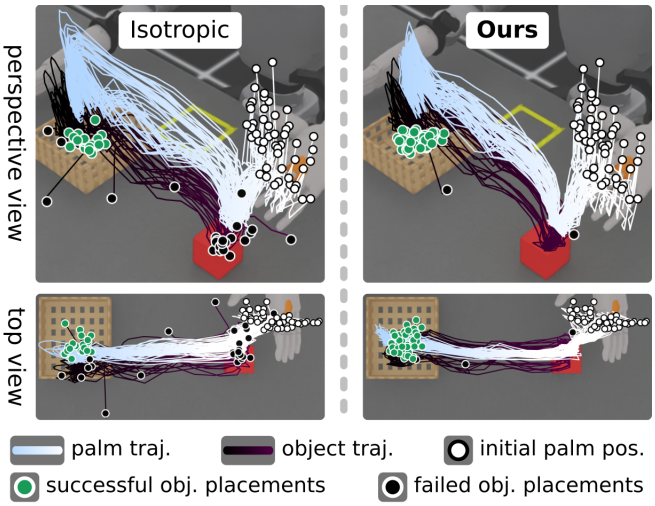

Fig. 3: **Distribution of end-effector trajectories for the G1 object-in-box task.** For a representative object–box configuration, 50 sampled plans are executed in Isaac Sim. Our method (right) concentrates trajectories around appropriate grasp and release regions, resulting in fewer failures.

(iii) Unitree G1 object-in-box manipulation. The G1 humanoid grasps an object and places it inside a target box. The trajectory state consists of 7-DoF left-arm joint positions and a hand grasping fraction. The key-state stack Y contains the initial, grasp, and release configurations. We train an MLP-based inverse-kinematics (IK) predictor for the grasp and release joint configurations, and another MLP to predict their timings for constructing C. The dataset consists of 219 successful Finite State Machine-generated trajectories collected in Isaac Sim.

Simulation evaluations randomize start–goal pairs for Maze2D, initial joint configurations and block layouts for KUKA, and object/box poses and initial arm configurations for G1.

B. Baselines and Ablations

We compare three planners in addition to ours that share the same UNet backbone and training schedule, differing only in how task structure is incorporated. All models are trained for the same number of optimization steps (Tasks from Diffuser benchmark – 100k steps with batch size 32, G1 – 3k steps with batch size 512) The detailed design of each planner is as follows:

(i) Diffuser. The baseline Diffuser [2], which uses isotropic Gaussian noise and conditions only on task inputs such as initial and goal states.

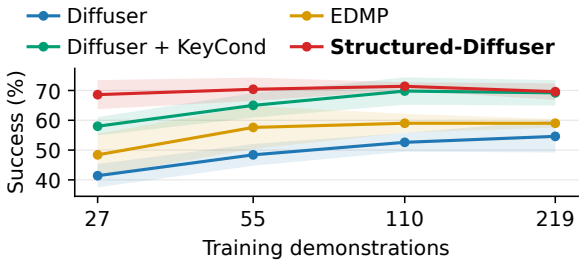

Fig. 4: **Data efficiency for G1 Put-In-Box.** Success rate is plotted over number of training demonstrations. Points are means over five evaluation seeds of 100 episodes each and shading shows standard deviation.

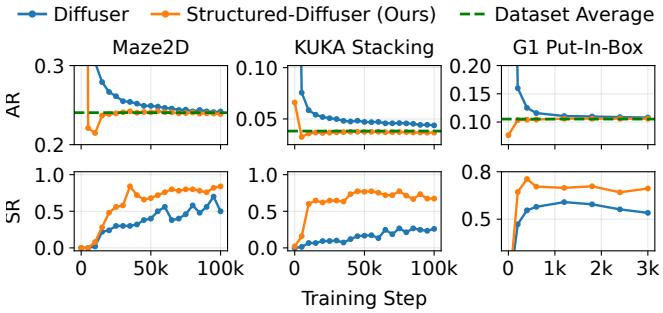

Fig. 5: **Average roughness (AR) and success rate (SR) over training for three tasks.** Metric calculated as average of 50 samples. The dashed line shows the dataset-average AR as a reference for reasonable smoothness.

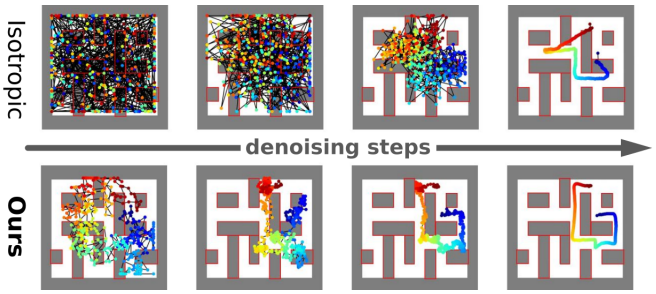

Fig. 6: **Reverse diffusion in Maze2D.** Intermediate denoising steps from left to right. Our prior rapidly converges to smooth, goal-reaching trajectories.

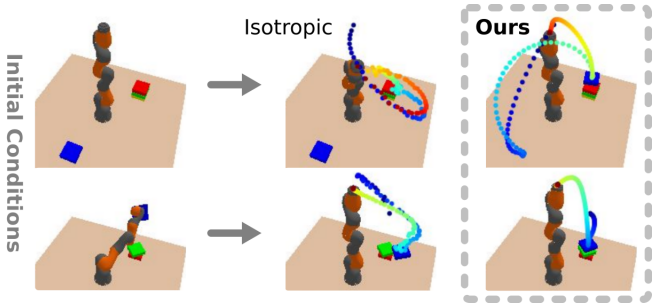

Fig. 7: **Comparison of generated trajectories for KUKA block stacking.** Each row shows a task with a random initial arm configuration and block positions. Columns: (left) initial condition, (middle) trajectories generated with an isotropic Gaussian prior, (right) trajectories generated with our model.

(ii) Diffuser + KeyCond. A hierarchical variant of Diffuser where the lower-level diffusion is conditioned on key states and their timings produced by the upper level, while still using isotropic Gaussian noise. This isolates the effect of neural task conditioning without structured noise.

(iii) EDMP [34]. A single-level Diffuser guided at inference time by model-based costs. Following the original formulation, we use task-specific costs for each benchmark. For Maze2D, the cost penalizes obstacle penetration at each state and at midpoints between adjacent states to approximate swept-volume collisions. This implementation requires access to the maze geometry at test time. For KUKA and G1 manipulation, since [34] does not specify a cost for grasp-and-place tasks, we design a simple heuristic cost based on the minimum distance between the end-effector and the target, rewarding configurations with a small distance.

(iv) Structured-Diffuser (Ours). Our two-level approach, where the upper level provides key states and timings, and the lower level denoises under a GPMP-conditioned prior.

Note that the choice of noise prior applies only to the lower-level diffusion. Whenever the upper level uses diffusion, it employs a standard DDPM with isotropic Gaussian.

C. Results and Analysis

Task success and data efficiency. In Maze2D, success requires collision-free goal reaching; in KUKA block stacking, a successful final block stack, verified by kinematic replay of the generated trajectory; and in G1 object-in-box, successfully grasping the object and placing it inside the target box in physics-based simulation (Isaac Sim). These binary criteria differ from the metrics in [2], so absolute numbers are not comparable. Table I shows that Structured-Diffuser outperforms vanilla Diffuser across all three tasks. We attribute these gains to the structured prior, which concentrates probability mass around task-critical regions through the soft GP conditioning in (8)–(9). This effect is visualized in Fig. 3, where sampled G1 trajectories concentrate near the object during grasping and near the box during release.

Among all methods, Structured-Diffuser achieves the highest mean success rate on KUKA and G1. On Maze2D, Structured-Diffuser achieves the second-highest success rate (74.4%), behind EDMP (78.4%). However, EDMP uses ground-truth map geometry at test time, whereas Structured-Diffuser does not. Diffuser + KeyCond performs similarly to Structured-Diffuser on full-data G1. With less training data, however, the gap widens: using 27 demonstrations, Structured-Diffuser retains 68.6% compared with 58.0% for KeyCond (Fig. 4).

Smoothness and inter-state consistency. We evaluate trajectory smoothness using the *Average Roughness* (AR) metric [34], defined as the average of $\|x_{t+1} - x_t\|_2$ along the trajectory. Because excessively low AR can correspond to little or no motion, we compare generated trajectories with the dataset-average AR in Fig. 5. Representative trajectories and denoising processes are shown in Figs. 6 and 7. These

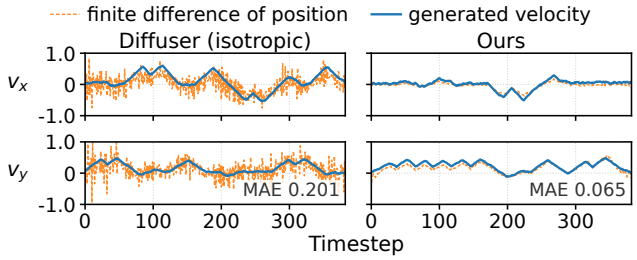

Fig. 8: **Maze2D velocity consistency.** Left: isotropic Gaussian prior; Right: our model. Orange: numerical derivatives of generated positions; Blue: generated velocities. Closer alignment indicates better inter-state (pos & vel) consistency.

Method	Diffuser	Diffuser + KeyCond	EDMP	Structured-Diffuser
Vel MAE	0.2187	0.2064	0.2012	**0.0767**

TABLE II: Maze2D velocity consistency: mean absolute error (MAE) between generated velocities and finite-difference derivatives of positions over 100 trajectories.

quantitative and qualitative results show that Structured-Diffuser reaches dataset-level smoothness earlier and generates smooth trajectories more consistently. This behavior is encouraged by LTV-GP-based covariance, which encodes temporal dependencies across trajectory states.

In Maze2D, the LTV-GP-based covariance explicitly couples position and velocity across the trajectory. We evaluate this by comparing generated velocities with finite-difference derivatives of generated positions (Fig. 8). Structured-Diffuser achieves substantially lower velocity MAE than the baselines (Table II), demonstrating stronger position–velocity consistency.

Training efficiency. As shown in Fig. 5, Structured-Diffuser reaches dataset-level smoothness and high success rates earlier in training than vanilla Diffuser. This suggests that encoding task and motion structure in the prior reduces the amount of training needed to obtain useful trajectories.

Phase-dependent task structure. In many robotics tasks, desired behavior depends not only on the robot state but also on when it occurs (e.g., contact timing). This timing–configuration coupling is difficult to encode with state-wise guidance, especially in partially denoised trajectories where timings may shift under noise. Our framework instead predicts task-phase timings at the upper level and conditions the lower-level prior on them with uncertainty, which is useful for manipulation tasks with critical interaction timing.

D. Hardware Experiment

We deploy the planner on a physical Unitree G1 humanoid equipped with an Inspire hand, with the torso held at a fixed reference by a low-level locomotion controller. ArUco markers provide object and box poses. The planner generates a 60-step trajectory, which is interpolated and executed at 20 Hz without online replanning. We first evaluate randomized-location trials using models trained on the full

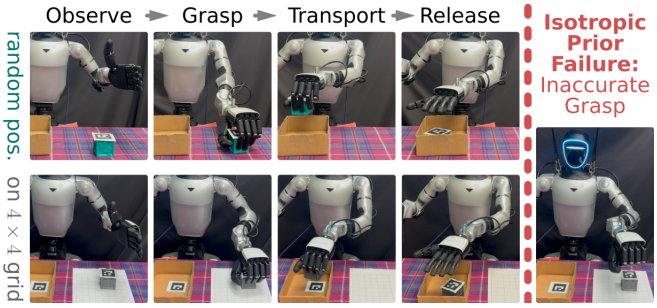

Fig. 9: **Representative G1 hardware rollouts.** The robot localizes the object with the mounted camera and executes pick-and-place into the box. Top: randomly placed object. Bottom: object placed on a fixed 4×4 grid, identical across planners. Right: isotropic-prior failure due to an inaccurate grasp at the same grid location shown in the bottom sequence.

simulation dataset, with object and box positions sampled within $15\,\mathrm{cm} \times 15\,\mathrm{cm}$ and $7\,\mathrm{cm} \times 7\,\mathrm{cm}$ regions, respectively. Representative rollouts are shown in Fig. 9.

For quantitative evaluation, we test 16 designated object locations on a 4×4 grid using diffusion models trained on a separate simulation dataset of 63 trajectories, whose size is deliberately reduced to demonstrate data efficiency. For sim-to-real transfer, we collect 19 grasp configurations (rather than full trajectories) in real hardware, and fine-tune only the key-state predictor on those configurations for less than one minute; the diffusion planners remain fixed. Each planner was evaluated once at each of the 16 grid locations. Structured-Diffuser succeeded in 8/16 trials, compared with 3/16 for Diffuser.

V. Conclusion

In this paper, we propose a hierarchical diffusion planner that embeds task and motion structure directly in the noise model. We validate the approach across three robotic domains, demonstrating improved task success, trainig efficiency and, where evaluated, data efficiency and inter-state consistency. We further demonstrate hardware deployment on a physical G1 humanoid. The lower-level structured diffusion is agnostic to how the key states and timings are obtained; in our experiments, the upper level combines diffusion models, lightweight MLPs, inverse kinematics, and simple task-specific rules. Moreover, our method does not require handcrafted reward terms or auxiliary objectives for guidance, and it does not rely on advanced neural architectures beyond conditioning on upper-level-generated features.

Future work includes disentangling the contributions of the prior mean and covariance, extending the framework to more complex whole-body humanoid manipulation, investigating richer upper-level task specifiers, such as vision-language models, and more systematically studying sensitivity to upper-level module variants and related hyperparameters.

VI. Acknowledgment

ChatGPT and Claude were used in polishing the writing. CODEX and Claude Code were used to polish our code base.

REFERENCES

[1] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1916–1923.

[2] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9902–9915.

[3] 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," *The International Journal of Robotics Research*, vol. 44, no. 10–11, pp. 1684–1704, 2025.

[4] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" in *The Eleventh International Conference on Learning Representations*, 2023.

[5] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M 2 diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[6] K. Mizuta and K. Leung, "CoBL-diffusion: Diffusion-based conditional robot planning in dynamic environments using control barrier and lyapunov functions," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 801–13 808.

[7] C. Pan, Z. Yi, G. Shi, and G. Qu, "Model-based diffusion for trajectory optimization," *Advances in Neural Information Processing Systems*, vol. 37, pp. 57 914–57 943, 2024.

[8] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[9] M. Seo, Y. Cho, Y. Sung, P. Stone, Y. Zhu, and B. Kim, "PRESTO: Fast motion planning using diffusion models based on key-configuration environment representation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 10 861–10 867.

[10] W. Xiao, T.-H. Wang, C. Gan, R. Hasani, M. Lechner, and D. Rus, "SafeDiffuser: Safe planning with diffusion probabilistic models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[11] C.-C. Hsu, B. Wen, J. Xu, Y. Narang, X. Wang, Y. Zhu, J. Biswas, and S. Birchfield, "SPOT: SE(3) pose trajectory diffusion for object-centric manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 4853–4860.

[12] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "SE(3)-DiffusionFields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5923–5930.

[13] J.-B. Bouvier, K. Ryu, K. Nagpal, Q. Liao, K. Sreenath, and N. Mehr, "DDAT: Diffusion policies enforcing dynamically admissible robot trajectories," in *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2025.

[14] J. Liang, J. K. Christopher, S. Koenig, and F. Fioretto, "Simultaneous multi-robot motion planning with projected diffusion models," in *International Conference on Machine Learning*. PMLR, 2025, pp. 37 162–37 180.

[15] Z. Yin, T. Lai, L. Barcelos, J. Jacob, Y. Li, and F. Ramos, "Diverse motion planning with stein diffusion trajectory inference," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 15 610–15 616.

[16] R. Wu, H. Chen, M. Zhang, H. Lu, Y. Li, and Y. Li, "Neural dynamics augmented diffusion policy," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 13 234–13 241.

[17] E. Nachmani, R. S. Roman, and L. Wolf, "Non gaussian denoising diffusion models," *arXiv preprint arXiv:2106.07582*, 2021.

[18] X. Yu, X. Gu, H. Liu, and J. Sun, "Constructing non-isotropic gaussian diffusion model using isotropic gaussian diffusion model for image editing," *Advances in Neural Information Processing Systems*, vol. 36, pp. 76 839–76 851, 2023.

[19] C. Curreli, D. Muhle, A. Saroha, Z. Ye, R. Marin, and D. Cremers, "Nonisotropic Gaussian diffusion for realistic 3D human motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 1871–1882.

[20] N. Jia, T. Zhu, H. Liu, and Z. Zheng, "Structured diffusion models with mixture of Gaussians as prior distribution," *arXiv preprint arXiv:2410.19149*, 2024.

[21] L. Wang, H. Yu, C. Yu, S. Gao, and H. Christensen, "Controllable motion generation via diffusion modal coupling," *arXiv preprint arXiv:2503.02353*, 2025.

[22] D. Blessing, X. Jia, and G. Neumann, "End-to-end learning of Gaussian mixture priors for diffusion sampler," in *The Thirteenth International Conference on Learning Representations*, 2025.

[23] H. Guo, C. Lu, F. Bao, T. Pang, S. Yan, C. Du, and C. Li, "Gaussian mixture solvers for diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 25 598–25 626, 2023.

[24] X. Zhou, X. Chen, and J. Yang, "Diff-Refiner: Enhancing multi-agent trajectory prediction with a plug-and-play diffusion refiner," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 10 779–10 785.

[25] H. Ren, Y. Zeng, Z. Bi, Z. Wan, J. Huang, and H. Cheng, "Prior does matter: Visual navigation via denoising diffusion bridge models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 12 100–12 110.

[26] M. Dai, Q. Zhuang, H.-Y. Xu, Y.-J. Shuai, Y. Wang, Q. Dou, and X.-S. Wei, "Where should action generation begin? A learnable source prior for generative robot policies," *arXiv preprint arXiv:2606.17408*, 2026.

[27] Z. Chang, Y. Chen, B. Park, H. Yu, and Y. Chen, "Function-space diffusion for motion planning," *arXiv preprint arXiv:2607.02977*, 2026.

[28] C. Chen, F. Deng, K. Kawaguchi, C. Gulcehre, and S. Ahn, "Simple hierarchical planning with diffusion," in *The Twelfth International Conference on Learning Representations*, 2024.

[29] M. Sharma, A. Fishman, V. Kumar, C. Paxton, and O. Kroemer, "Cascaded diffusion models for neural motion planning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 361–14 368.

[30] Z. Dong, J. Hao, Y. Yuan, F. Ni, Y. Wang, P. Li, and Y. Zheng, "Diffuserlite: Towards real-time diffusion planning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 122 556–122 583, 2024.

[31] D. Wang, C. Liu, F. Chang, and Y. Xu, "Hierarchical diffusion policy: manipulation trajectory generation via contact guidance," *IEEE Transactions on Robotics*, 2025.

[32] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 081–18 090.

[33] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[34] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna, "EDMP: Ensemble-of-costs-guided diffusion for motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 351–10 358.

APPENDIX

First, we drive the marginal $q(\tau^j \mid \tau^0)$ (6) by induction. Assume (6) holds for $i = j - 1$. Then, $\tau^j \mid \tau^0$ is Gaussian with mean and variance

$$\begin{aligned}
&\mathbb{E}[\tau^j \mid \tau^0] = \sqrt{\alpha_j}\,\mathbb{E}[\tau^{j-1} \mid \tau^0] + (1 - \sqrt{\alpha_j})\xi \\
&= \sqrt{\alpha_j}\left(\sqrt{\bar{\alpha}_{j-1}}\tau^0 + (1 - \sqrt{\bar{\alpha}_{j-1}})\xi\right) + (1 - \sqrt{\alpha_j})\xi \\
&= \sqrt{\bar{\alpha}_j}\,\tau^0 + (1 - \sqrt{\bar{\alpha}_j})\xi, \\
&\mathrm{Var}[\tau^j \mid \tau^0] = \alpha_j(1 - \bar{\alpha}_{j-1})\mathcal{K} + (1 - \alpha_j)\mathcal{K} = (1 - \bar{\alpha}_j)\mathcal{K}.
\end{aligned}$$

Hence the form holds for step j, as claimed.

Second, we derive the posterior (7). Both $q(\tau^i \mid \tau^{i-1})$ and $q(\tau^{i-1} \mid \tau^0)$ are Gaussian distributions with known forms. Using Bayes' rule on $q(\tau^{i-1} \mid \tau^i, \tau^0) \propto q(\tau^i \mid \tau^{i-1})\, q(\tau^{i-1} \mid \tau^0)$, we obtain (7).