2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

ECAHD: Efficient Collision-Aware Hierarchical Diffusion Navigation

Jinu Pahk[1,2], Theo Taeyeong Kim[1,2], Jun ki Lee[1] and Byoung-Tak Zhang[1,2]

***Abstract*— Safe and fast robot navigation is crucial for deploying robots in real-time interactive environments. Recent diffusion-based approaches for path planning and navigation leverage data-driven learning to generalize across diverse tasks and to quickly generate trajectories, but research explicitly incorporating collision awareness remains limited. In this work, we propose Efficient Collision-Aware Hierarchical Diffusion Navigation (ECAHD), a hierarchical diffusion-based framework designed for both safety and computational efficiency. ECAHD generates a sparse trajectory for global path planning and a dense trajectory for local path refinement. The robot follows a rapidly sampled sparse global trajectory, and when a potential collision is detected, a collision-aware guidance diffusion mechanism—which accounts for the robot's shape—adjusts the local trajectory accordingly. Conventional full-sequence diffusion planners suffer from slow sampling speeds and performance degradation when collision-aware guidance is applied across the entire trajectory. ECAHD addresses these issues by significantly reducing the number of waypoints predicted by the global diffusion planner, while delegating robot shape aware collision guidance to the local diffusion planner. This separation not only accelerates planning but also preserves global trajectory quality, as goal-conditioned sampling is no longer disrupted by collision-related constraints. Furthermore, ECAHD allows for increasing the number of global trajectory samples to enhance performance, without incurring substantial computational overhead. In maze2d-large planning tests, ECAHD improved success rates by approximately 1.3% while reducing collision rates by more than 50%, all while cutting inference time by nearly half. In D4RL Navigation benchmark, ECAHD achieved the fastest goal-reaching time and the lowest collision rate among compared methods, demonstrating its effectiveness in efficient and safe robot navigation.**

I. Introduction

With the rapid advancement of generative models, they have been successfully applied to a wide range of fields, including natural language processing [1], computer vision [2], and image generation [3], thanks to their high fidelity and robustness. More recently, research applying generative models to robotics has attracted increasing attention, with approaches such as Decision Transformer [4], which uses an autoregressive model for sequence-based planning and control, Diffuser [6], which performs trajectory planning with diffusion models, and Diffusion Policy [9], which applies diffusion to visuomotor policy learning.

Diffusion models [5], in particular, have demonstrated superior capability in capturing multimodal distributions compared to other generative models, allowing them to generate diverse trajectories for the same planning task, rather than a single deterministic plan [6]. This property has led to studies leveraging diffusion models for global path planning and navigation in maze environments [10], as well as for local path planning that focuses on avoiding obstacles based on cost maps [11].

Compared to classical planners such as A* [12] or RRT [13], these data-driven, learning-based approaches offer faster inference and greater adaptability to various tasks and environmental conditions. Moreover, they can implicitly learn complex planning heuristics directly from data, reducing the need for manual tuning of cost functions or heuristics [6].

However, existing diffusion-based navigation approaches have several limitations. Some methods only generate global trajectories and thus require a separate local planner for fine-grained obstacle avoidance [10], while others focus solely on local collision avoidance, making additional global path planning necessary [11]. Furthermore, unlike traditional planners that can explicitly account for the robot's shape [13], many diffusion-based navigation models rely solely

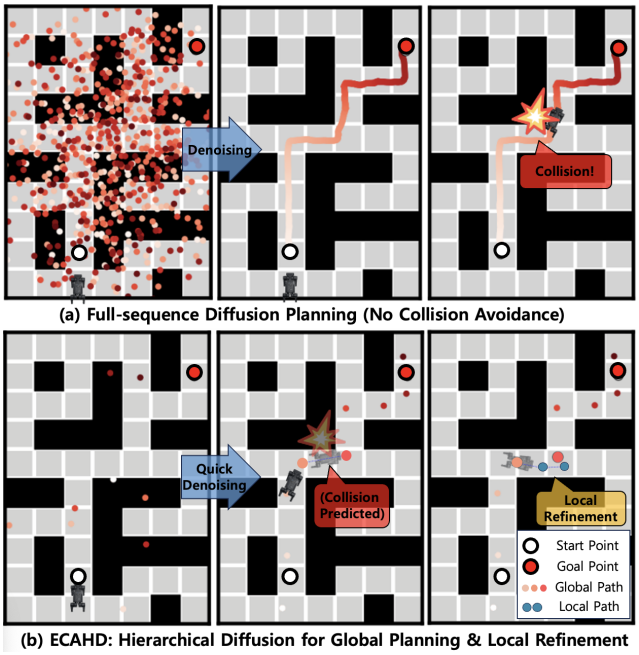

Fig. 1. (a) Planning using Fully-sequence Diffusion (FD) and (b) Hierarchical Diffusion (HD). Without considering collisions during sampling, collisions may occur during navigation. However, the proposed HD with collision guidance predicts collisions during global path planning and then uses a dense diffuser to adjust the local plan.

This work was partly supported by grants from the IITP (RS-2021-II211343-GSAI/10%, RS-2022-II220951-LBA/15%, RS-2022-II220953-PICA/15%), the NRF (RS-2024-00353991-SPARC/15%, RS-2023-00274280-HEI/15%), the KEIT (RS-2025-25453780/15%), and the KIAT (RS-2025-25460896/15%), funded by the Korean government.

[1] Seoul National University, Seoul 08826, Republic of Korea
[2] Tommoro Robotics

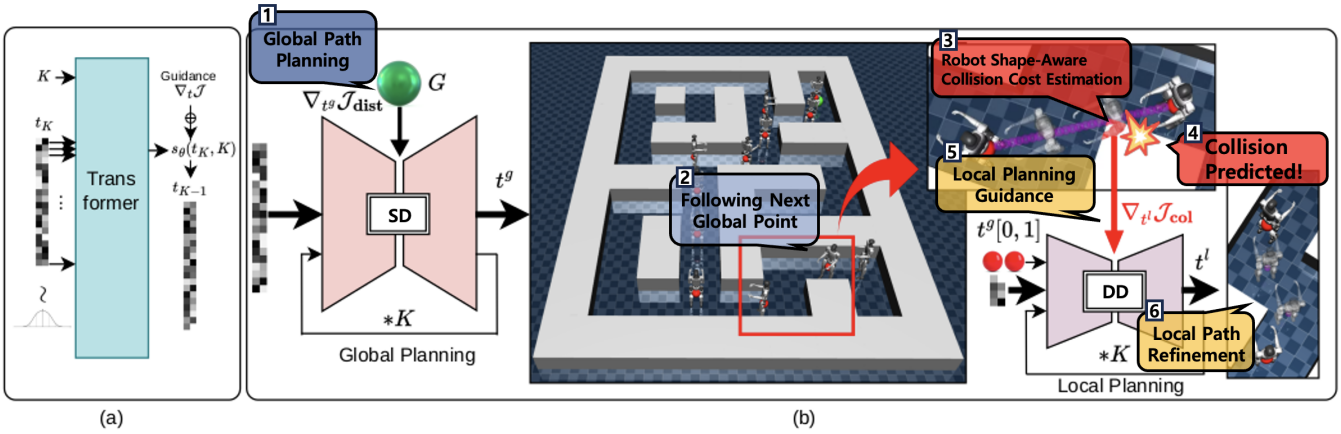

Fig. 2. Overall framework. (a) Transformer predicts the score function from the state at each step. (b) Sparse Diffusion gets goal distance guidance, and Dense Diffusion gets collision surface guidance, generating a local plan aware of robot shape and collisions.

on map or cost map information, overlooking the physical footprint of the robot.

This motivates our research question: How can we design a learning-based navigation framework that achieves both fast, efficient global path planning and precise, robot shape-aware local collision avoidance, without compromising overall planning performance due to collision guidance? To address this, we propose Efficient Collision-Aware Hierarchical Diffusion Navigation (ECAHD), a unified data-driven framework for both global and local planning. Inspired by recent advances in hierarchical diffusion planning [31], ECAHD consists of two complementary diffusion modules: (1) Sparse Diffusion (SD) rapidly generates a coarse global trajectory, enabling fast path initialization even with incomplete maps. (2) Dense Diffusion (DD) refines local segments when potential collisions are detected, using a differentiable collision cost that explicitly considers the robot's shape.

By combining these modules, ECAHD learns and generates global paths significantly faster than FD (Full-sequence Diffusion). Furthermore, by offloading the computational burden of collision avoidance to the DD local planner, ECAHD prevents performance degradation in global planning caused by additional guidance computation, ultimately achieving faster overall navigation and improved collision avoidance.

Our main contributions are:

• Hierarchical Diffusion Planning: A fully data-driven framework that integrates performance-focused global path generation with safety-aware local refinement, enhancing both planning speed and safety.

• Shape-Aware Collision Avoidance: The differentiable collision cost explicitly captures the robot's shape and size from a top-down view, and is seamlessly applied to DD's guidance without requiring extra training.

• Efficient and Robust Navigation: The separation of global and local planning enables fast initialization and targeted refinement. Fast global planning allows multi-sampling for better performance, while local refinement is triggered only when needed based on the collision cost, ensuring efficient real-time navigation.

II. Related Works

A. Robot Path Planning and Collision Avoidance

Robot path planning and collision avoidance are core challenges in autonomous navigation, requiring feasible paths from start to goal while avoiding obstacles. Classical methods like A* [12], Dijkstra, and RRT [13] are widely used, particularly in static environments. A* finds the shortest path via heuristic guidance, while RRT handles high-dimensional spaces by incrementally expanding a search tree.

For collision avoidance, traditional approaches rely on cost maps with inflated obstacle boundaries [24], while reactive methods such as Velocity Obstacles (VO) [25] and Dynamic Window Approach (DWA) [26] compute safe velocities by predicting obstacle motion.

More recently, learning-based methods like Deep Reinforcement Learning (DRL) [27] and Imitation Learning (IL) [28] have shown promise, especially in dynamic environments. DRL learns collision-free policies through trial and error, while IL mimics expert demonstrations for better generalization in complex settings [40].

B. Probabilistic Diffusion Model Path Planning

Probabilistic Diffusion Models (PDMs), originally developed for image generation [3], have also been applied to robot path planning. PDMs excel at modeling multimodal solution spaces, making them effective for planning tasks with diverse feasible trajectories under varying constraints and preferences.

Diffusion models generate paths by reversing a forward process that gradually adds noise to feasible paths. Sampling starts from noise and iteratively refines into a feasible trajectory via the learned reverse process:

a) Unconditional reverse step [7]:

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where

- $\mathbf{x}_t \in \mathbb{R}^{2\times P}$ is the noisy trajectory at diffusion step t, represented as a sequence of P robot states in 2D workspace,
- $\mu_\theta(\mathbf{x}_t, t)$ is the denoiser network (parameterized by θ) that predicts the reverse mean or score-based update at step t,
- σ_t is the variance (noise scale) prescribed by the forward diffusion schedule at step t,
- $\mathbf{z}$ is a Gaussian noise sample drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$,
- and $\mathbf{x}_{t-1}$ is the refined trajectory sample after one reverse step.

b) Guided reverse step:

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_t\,\mathbf{z} + \gamma\,\nabla_{\mathbf{x}_t}\mathcal{J}(\mathbf{x}_t), \tag{2}$$

where γ is the guidance weight. Conventional classifier guidance injects $\nabla_{\mathbf{x}_t}\log p_\phi(\mathbf{y}\,|\,\mathbf{x}_t)$ from a separately trained classifier [8]. In contrast, ECAHD avoids training an extra classifier by directly substituting a differentiable surrogate objective $\mathcal{J}(\mathbf{x}_t)$ that encodes both goal progress and collision cost (Sec. III-B).

Beyond navigation, PDMs have also been applied to robotic manipulation via Imitation Learning (IL), successfully tackling contact-rich tasks like insertion [37], manipulation [39], and peg-in-hole assembly [38], where learning from demonstrations ensures skill transfer and generalization.

In navigation, DiPPeR [10] generates global paths directly from map images without explicit pre-processing or traditional planners, but lacks embodiment-aware local refinement, limiting its performance in narrow passages. NoMaD [30] learns reactive goal-masked diffusion policies from visual inputs, enabling exploration and real-time adaptation in partially observable environments. However, NoMaD does not explicitly separate global and local planning.

Together, these works highlight the flexibility of PDMs for navigation, but neither achieves a fully hierarchical framework integrating fast global planning, shape-aware local refinement, and real-time collision avoidance.

To make the presentation self-contained, we briefly summarize the problem setting adopted in this paper. A robot is required to generate a 2D trajectory $\mathbf{x} = [x_0, \ldots, x_{P-1}]$ from a start S to a goal G on a given map M, while avoiding collisions with its footprint R. Diffusion models address this by iteratively denoising trajectories from $t = K$ down to $t = 1$, with optional guidance terms that encourage goal progress and penalize collisions. This preliminary formulation parallels the style of prior works such as [30], and provides the foundation for the hierarchical method proposed in Sec. III.

III. Methods

A. Notation for Algorithms 1–2.

We denote the planning map by M, start/goal by S, G, robot footprint by R, global and local horizons by P^g and P^l, and the number of denoising steps by K. The global and local diffusion models have parameters θ_g and θ_l, respectively. The cost function $\mathcal{J}$ returns a pair $(\texttt{dist\_cost}, \texttt{collision\_cost})$. When `isSearchEnabled` is activated, we sample I candidate sparse global trajectories $\{t^{g,(i)}\}_{i=1}^{I}$ in parallel and define goal-reaching trajectories as those satisfying $\min_k \|t_k^{g,(i)} - G\| < \epsilon$. Among the goal-reaching candidates, we select the one with the shortest path length $L(t^g) = \sum_k \|t_{k+1}^g - t_k^g\|$; if none reaches the goal, we select the trajectory with minimum $L(t^g)$ (we use $I = 10$ and $\epsilon = 1.0$). Guidance is injected only in local DD steps via $\gamma\nabla\mathcal{J}$ (cf. Eqs. (1)–(2)). We assume a complete static a priori map M (or M_{plan}) is available during planning, and J is computed by querying this map. Therefore, ECAHD in its current form targets static known environments; dynamic or partially observed settings require an external perception/prediction or mapping module.

Algorithm 1 Navigation via Hierarchical Diffusion Planning

Require: Map M_{plan}, Diffusion model θ_g, θ_l, Cost function $\mathcal{J}$, Start position S, Robot shape points R, Goal position G

for $p = 0, 1, 2, ..., P^g - 1$ **do**
  if `isSearchEnabled` **then**
    $\{t_p^g\} \leftarrow \{\}$
    for $i = 0, 1, 2, ..., I - 1$ **parallel do**
      $\{t_p^g\} \leftarrow \{t_p^g\} \cup$ SparseDiffusion(M_{plan}, $P^g - p$, S, G, θ_g, $\mathcal{J}$)
    end for
    Global Trajectory $t_p^g \leftarrow$ BestofN($\{t_p^g\}$)
  else
    Global Trajectory $t_p^g \leftarrow$ SparseDiffusion(M_{plan}, $P^g - p$, S, G, θ_g, $\mathcal{J}$)
  end if
  Local Trajectory $t_p^l \leftarrow$ LocalPlan($t_p^g[0], t_p^g[1]$)
  Local Value $d_p^l, c_p^l \leftarrow \mathcal{J}(t_p^l, t_p^g[1], P^l, R, M_{plan})$
  if $c_p^l > c_{th}$ **then**
    Local Trajectory $t_p^l \leftarrow$ DenseDiffusion(M_{plan}, P^l, $t_p^g[0]$, $t_p^g[1]$, θ_l, $\mathcal{J}$)
  end if
  $S \leftarrow$ MoveAgent(t_p^l)
end for

B. Hierarchical diffusion planning considering robot shape and collision

Unlike conventional diffusion planning or diffusion policy approaches, the hierarchical diffusion planning framework proposed in this work explicitly separates the planning process into two stages: global planning and local refinement. In the global planning stage, a sparse trajectory is generated by performing state inpainting between the start and goal, prioritizing long-horizon feasibility and overall task performance. In contrast, the local refinement stage performs dense trajectory prediction between nearby waypoints along the global path, with a primary focus on collision avoidance.

After the sparse trajectory is predicted through the global SD process, a general navigation plan is already in place, allowing the robot to move toward the goal by following the sequence of points. However, without using a local planner

Algorithm 2 Dense Diffusion and Guidance function

function DENSEDIFFUSION(Map M_{plan}, Local trajectory horizon P^l, Local Start position $t_p^g[0]$, Local goal position $t_p^g[1]$, Dense Diffusion Model θ_l, Cost function $\mathcal{J}$, Robot shape points R, Denoising steps K)
  $t_K^l \sim \mathcal{N}(0, I)$
  for $k = K-1$ **down to** 1 **do**
    $d_k(t_k^l), c_k(t_k^l) \leftarrow \mathcal{J}(t_k^l, t_p^g[1], P^l, R, M_{plan})$
    $s_{k-1} \leftarrow s_{\theta_l}(t_k^l, k) + \omega \nabla_{t_k^l}(d_k(t_k^l) + c_k(t_k^l))$
    $t_{k-1}^l = \text{DDIM}(s_{k-1}, t_k^l, \alpha_k, \alpha_{k-1}, \eta)$
  end for
  return Local plan t_0^l
end function

function COST FUNCTION $\mathcal{J}$(Predicted trajectory t_k, Goal position G, Trajectory horizon P, Robot shape points R, Map M)
  `dist_cost` $\leftarrow$ MSE(t_k, G)
  `collision_cost` $\leftarrow 0$
  for $p = 0, \ldots, P-2$ **do**
    $\psi_{start} \leftarrow$ atan2($t_k[p+1]$ - $t_k[p]$)
    $\psi_{end} \leftarrow$ atan2($t_k[p+2]$ - $t_k[p+1]$)
    $R_{start} \leftarrow$ GetRobotShapePoints($t_k[p]$, ψ_{start}, R)
    $R_{end} \leftarrow$ GetRobotShapePoints($t_k[p+1]$, ψ_{end}, R)
    for each (x, y) in DDA(R_{start}, R_{end}) **do**
      `map_value` $\leftarrow$ BilinearInterpolation(M, x, y)
      `collision_cost` $\leftarrow$ `collision_cost` + `map_value`
    end for
  end for
  return `dist_cost`, `collision_cost` $* \frac{\sigma_{\text{dist}}}{\sigma_{\text{col}}}$
end function

explicitly designed or trained for collision avoidance, simply following the sparse plan linearly can result in collisions with obstacles or walls. In ECAHD, the local plan computed between each pair of adjacent sparse points is evaluated using a cost function $\mathcal{J}$, which assesses both the distance to the local goal d^l and the collision cost c^l. If the collision cost c^l is over a predefined threshold c_{th}, a new local plan is generated using dense diffusion to ensure collision-aware navigation. The complete algorithm for this process is described in Algorithm 1.

The algorithm for the DenseDiffusion function and the cost function $\mathcal{J}$ is outlined in Algorithm 2. During local path sampling, we follow the standard denoising update of diffusion models and apply the DDIM(Denoising Diffusion Implicit Models) scheme for efficiency [8]. At each denoising step, the score function is predicted by a Transformer-based denoiser (see Fig. 2(a)), which serves as the backbone for both the global (θ_g) and local (θ_l) diffusion modules. A shallow CNN (ResNet-18) optionally encodes the map or visual observations to provide contextual features, which are concatenated with trajectory tokens and processed by the Transformer. This design allows the model to jointly reason about long-horizon dynamics and local collision risks. In practice, two separate networks are instantiated: one for Sparse Diffusion (SD) and one for Dense Diffusion (DD), sharing the same backbone structure but trained with different horizon lengths.

For training, we use a trajectory reconstruction (noise-prediction) objective with a K-step forward diffusion. The model is trained to denoise these bundles using mean squared error (MSE) loss between predicted and clean trajectories, with weighting determined by a scheduling matrix. Optimization is performed with the AdamW optimizer (lr $= 10^{-4}$, $\beta = (0.9, 0.999)$, weight decay 10^{-2}), and a linear warm-up is applied in the first $5\,000$ steps. Training proceeds on datasets of trajectory rollouts, where both sparse and dense segments are sampled. Guidance is applied only at sampling time, not during training, making the base model a standard unconditional diffusion planner.

A guidance term $\gamma \nabla_{t_k^l} \mathcal{J}(t_k^l, t_p^g[1], P^l, R, M_{plan})$ is injected to bias the trajectory toward goal proximity while avoiding collisions. This ensures that the model generates a feasible trajectory while simultaneously considering and minimizing gradients related to the distance to the goal and collision risks. Unlike previous work in Diffusion planning, such as Diffuser [6] and Diffusion Policy [9], which rely on training a separate classifier or using classifier-free guidance, this approach addresses a critical issue highlighted in [31]: the difficulty in learning a classifier to predict collision costs due to the sparse and discrete nature of the states, while collision costs are continuous. To overcome this, instead of employing a separate classifier or classifier-free training, we propose a differentiable cost function that directly accounts for collision avoidance during sampling. The differentiability of this approach is demonstrated in Section III.B.

The cost function $\mathcal{J}$ calculates two components: `dist_cost`, which represents the remaining distance to the goal point, and `collision_cost`, which reflects the likelihood of a collision with the environment, considering the robot's shape. Assuming the robot is oriented toward the next point in the plan at each step, a set of points representing the interior of the robot's body, denoted by R, is sampled and placed around each planned point. The predicted continuous trajectory of the robot is then generated by connecting the corresponding internal points between the current and next positions using the Digital Differential Analyzer (DDA) algorithm [32]. DDA is a classic line-drawing and interpolation algorithm from computer graphics that incrementally steps through grid coordinates between two endpoints. In our setting, it efficiently enumerates intermediate points along the swept area of the robot's footprint, ensuring that all potential contact locations are checked. This process is illustrated in Fig. 2 (b), where the local trajectory connecting global plan points (red dots) is depicted as purple points representing the robot's predicted path. To compute the local trajectory cost without losing differentiability, bilinear interpolation is applied to a discrete cost map, generated from the occupancy

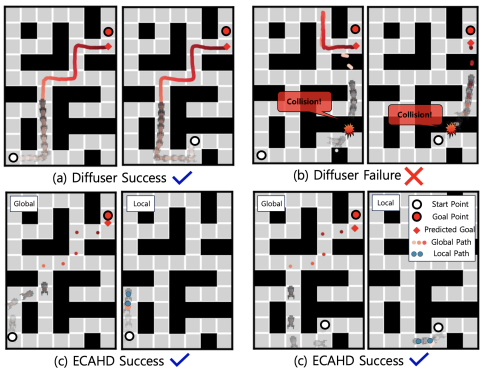

Fig. 3. Example of **planning** with collision guidance. (a) FD without guidance reaches the goal but often collides with walls. (b) FD with guidance struggles, frequently stalling near obstacles. (c,d) Our ECAHD results: two representative examples are shown, both demonstrating that SD provides a feasible global path and DD refines it locally to avoid collisions.

map, to yield the final `collision_cost`.

C. Cost Function Differentiability

The distance cost penalizes the discrepancy between the predicted trajectory t_k and the goal position G using the Mean Squared Error (MSE) function, which is a quadratic function of t_k, and is therefore continuous and differentiable.:

$$\text{dist\_cost}(t_k) = -\frac{1}{N}\sum_{i=1}^{N} \|t_k[i] - G\|^2 \tag{3}$$

To calculate the collision cost, the robot's orientation between adjacent state points $t_k[p+1]$ and $t_k[p]$ must first be computed using the atan2 function. The atan2 function is differentiable with respect to x and y, as shown below.

$$\psi^p_{\text{start}} = \text{atan2}(t_k[p{+}1, y]{-}t_k[p, y], t_k[p{+}1, x]{-}t_k[p, x]) \tag{4}$$

$$\frac{\partial \text{atan2}(y,x)}{\partial x} = -\frac{y}{x^2+y^2}, \frac{\partial \text{atan2}(y,x)}{\partial y} = \frac{x}{x^2+y^2} \tag{5}$$

$$\frac{\partial \psi^p_{\text{start}}}{\partial t_k[p,x]} = -\frac{\Delta y}{\Delta x^2+\Delta y^2}, \frac{\partial \psi^p_{\text{start}}}{\partial t_k[p,y]} = \frac{\Delta x}{\Delta x^2+\Delta y^2} \tag{6}$$

The robot's shape points R, determined by its orientation and position, must be rotated and translated. The rotation function $\mathcal{R}(R, \psi^p_{\text{start}})$ involves trigonometric functions of ψ^p_{start}, and the translation is a simple addition, making both operations differentiable.

$$R^p_{\text{start}} = \mathcal{R}(R, \psi^p_{\text{start}}) + t_k[p] \tag{7}$$

$$\mathcal{R}(R,\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} R_x \\ R_y \end{bmatrix} \tag{8}$$

To compute the collision cost, continuous coordinates along the line between adjacent robot shape points R^p_{start} and R^p_{end} are generated using the Digital Differential Analyzer (DDA). DDA itself is not the source of differentiability; it only produces interpolation points that are differentiable functions of the trajectory endpoints. The increments in x and y between two points are given by:

$$\Delta x = \frac{x_2 - x_1}{N}, \Delta y = \frac{y_2 - y_1}{N} \tag{9}$$

Next, to compute the collision cost at each interpolated point (x_i, y_i), the map values are obtained using bilinear interpolation. The bilinear interpolation is differentiable with respect to the coordinates x and y, as shown below. Given four neighboring grid points $(x_0, y_0), (x_1, y_0), (x_0, y_1), (x_1, y_1)$, bilinear interpolation estimates the value at (x, y) as:

$$\begin{aligned} \text{Interp}(M, x, y) = {} & (1-t_x)(1-t_y)M(x_0, y_0) \\ & + t_x(1-t_y)M(x_1, y_0) \\ & + (1-t_x)t_y M(x_0, y_1) \\ & + t_x t_y M(x_1, y_1) \end{aligned} \tag{10}$$

where t_x and t_y are normalized distances between (x, y) and the neighboring grid points, given by:

$$t_x = \frac{x - x_0}{x_1 - x_0}, \quad t_y = \frac{y - y_0}{y_1 - y_0} \tag{11}$$

The partial derivatives of the interpolation function with respect to x and y are:

$$\begin{aligned} \frac{\partial \text{Interp}(M,x,y)}{\partial x} = \frac{1}{x_1 - x_0}\Big[& -(1-t_y)M(x_0, y_0) \\ & + (1-t_y)M(x_1, y_0) \\ & - t_y M(x_0, y_1) \\ & + t_y M(x_1, y_1)\Big] \end{aligned} \tag{12}$$

$$\begin{aligned} \frac{\partial \text{Interp}(M,x,y)}{\partial y} = \frac{1}{y_1 - y_0}\Big[& -(1-t_x)M(x_0, y_0) \\ & - t_x M(x_1, y_0) \\ & + (1-t_x)M(x_0, y_1) \\ & + t_x M(x_1, y_1)\Big] \end{aligned} \tag{13}$$

IV. Experiments

A. Collision Guidance Impact

We evaluated Hierarchical Diffusion (HD) planning against FD and SD across Circular, Quadruped, and Humanoid robots. **Success Rate** measures if any point in the trajectory reached within 1.0 units of the goal, while **Collision Rate** is the ratio of actual to maximum collision cost (max: 1000 per grid cell). **Inference time** covers total sampling time, with HD including both SD (global) and DD (local) steps. DD was triggered when $c_{th} = 0.1$. Robot radius was 0.4, and each maze2D-large grid cell was 1. Quadruped and Humanoid used Unitree Go1 and G1 top views, respectively. The dataset was built by generating A* trajectories between random point pairs, sampled into 10 sparse waypoints for SD and 2 dense local points between each pair for DD.

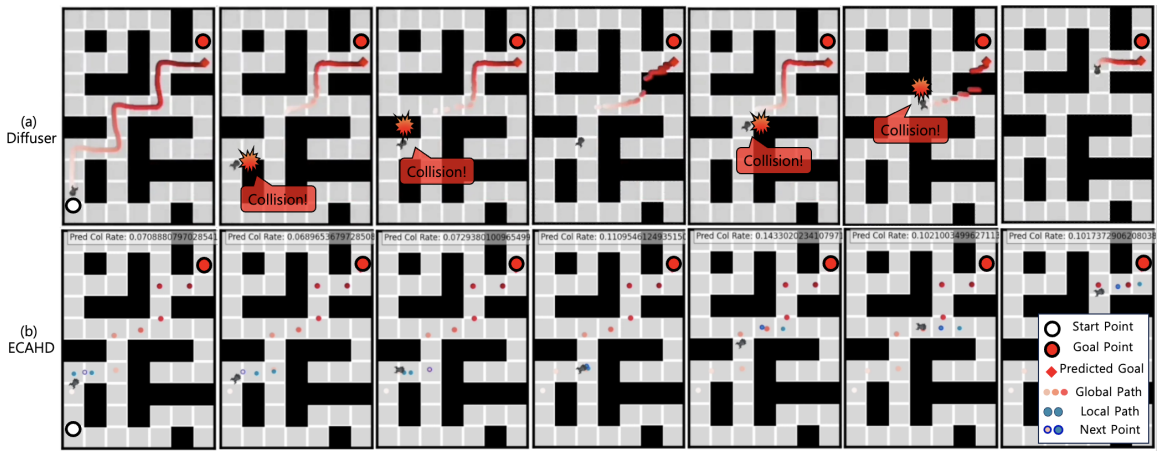

Fig. 4. Example of **Navigation** result. (a) Diffuser shows frequent wall collisions, while (b) ECAHD achieves a cleaner trajectory thanks to the local DD.

TABLE I

THE PLANNING RESULTS OF HD, SD, AND FD. BOLDFACE INDICATES THE BEST RESULT AMONG METHODS WITH COLLISION GUIDANCE APPLIED, AND UNDERLINE INDICATES THE OVERALL BEST RESULT ACROSS ALL METHODS.

		FD [6]		SD		HD (ours)	
Collision Guidance		×	✓	×	✓	×	✓ **(ECAHD)**
Success Rate (%) ↑	Circular	91.41	64.84	90.63	83.59	86.72	**89.06**
	Quadruped	85.94	69.53	89.84	80.47	87.50	**89.06**
	Humanoid	81.25	67.97	85.94	84.38	92.97	**84.43**
	Average	86.20	67.45	88.80	82.81	89.06	**87.52**
Collision Rate (%) ↓	Circular	0.06165	0.05004	0.08646	0.03983	0.04452	**0.03365**
	Quadruped	0.07513	0.04689	0.11323	0.04197	0.05840	**0.03516**
	Humanoid	0.08923	0.04370	0.11358	0.04112	0.05401	**0.03534**
	Average	0.07534	0.04654	0.10442	0.04097	0.05231	**0.03472**
Inference Time (s) ↓		2.762	11.881	1.449	2.443	1.484	**1.527**

TABLE II

ABLATION STUDY ON THE COLLISION GUIDANCE OPTIONS.

	Point	Line	Surface	Dilation	Embodiment	Succ(%)↑	Coll(%)↓
	✓					-	-
[35]	✓			✓		-	-
		✓				89.06	0.4675
		✓		✓		87.50	0.4372
(ours)			✓		✓	94.53	**0.0363**
			✓	✓	✓	**98.44**	0.1320

TABLE III

D4RL NAVIGATION EXPERIMENTS.

	Diffuser [6]	DiPPeR [10]	LDP+A* [11]	**ECAHD** w/o DD	**ECAHD** **w/ DD**
Episode Reward ↑	34.820	**67.672**	55.516	37.969	46.862
First Reach ↓	569.414	676.422	570.281	187.977	**182.797**
Collision Rate (%) ↓	0.1820	0.1551	0.0940	0.1062	**0.0921**
Inference Time (s) ↓	2.762	2.364	12.501	**1.963**	1.994

As shown in Table I, in all cases, adding collision guidance reduces the collision rate. In the case of FD, adding collision guidance significantly also decreases the previously high success rate. This is presumed to be because the gradient of the goal guidance and the collision avoidance guidance were not positively correlated, simultaneously affecting the score function. On the other hand, in HD, where goal distance guidance and collision guidance are separated, adding guidance resulted in the highest success rate among the models with guidance and resulted in the lowest collision rate. This characteristic can be observed in Fig. 3. From the inference time, it can be seen that FD's time increases sharply due to the time needed to compute guidance over the entire sequence. In contrast, SD and HD do not experience significant increases in inference time since they do not need to compute guidance for many densely sampled points. Especially, since SD has a significantly shorter inference time compared to FD, this advantage can be utilized to perform multiple samplings during the robot's navigation, allowing for high-quality global planning, as shown in Algorithm 1.

In all cases, ECAHD achieved the highest success rate and the lowest collision rate, while showing the shortest inference time among methods with collision guidance applied.

B. Ablation Study

Table II shows the results of the ablation study on the method of providing collision avoidance guidance. **Point**, **Line**, **Surface** refer to how values are assigned from the cost map to measure the degree of collision for the robot. Indexing the discrete cost based solely on the coordinates of points is non-differentiable, which means that the trajectory point cost calculation method using occupancy map **dilation** from [35] cannot be applied to this algorithm. Another approach is to provide cost guidance by continuously connecting the points within the trajectories and determining the extent to which this line overlaps with the interpolated cost map. Since this method does not involve direct indexing, it is differentiable, but it has the drawback of not being able to account for the robot's **shape**. In this study, we provide guidance based on the overlap between the surface generated by connecting the robot's trajectory points using DDA and the bilinearly interpolated cost map. This approach considers the robot's shape while achieving the highest success rate and the lowest collision rate. Since this algorithm takes the shape into account, there is no need for cost map dilation that adjusts for the robot's size, and applying this approach even resulted in a lower collision rate. It is worth noting that in our framework, the cost map does not serve as an absolute hard constraint but rather as a guidance signal injected into the diffusion process. Excessive dilation can therefore distort this probabilistic guidance, overly biasing the denoising trajectory away from feasible regions and reducing the effectiveness of collision avoidance. This explains why we observed cases where dilation improved the success rate but simultaneously degraded the collision performance.

C. Simulation Navigation

We conducted a simulation navigation experiment in the Maze2D-Large environment from D4RL (see Table III). Each robot's state observation consisted of 2D position (x, y), heading angle θ, and a cost map encoding nearby obstacles. The action space was defined as linear velocity v and angular velocity ω, controlling the robot's motion at each step. Three metrics were used:

Episode Reward: The average cumulative reward per episode, where the reward increases as the robot approaches the goal. **First Reach**: The average number of steps taken per episode for the robot to first reach the goal, indicating planning efficiency.

Inference Time: The combined average inference time per episode, which includes a single global path planning step performed before navigation and the cumulative local planning steps during navigation.

Diffuser [6] and DiPPeR [10] each generated 100 points per trajectory for global path planning, and interactive navigation was performed without a separate local planner. LDP [11] predicted the global path using A*, and for local planning, it generated 3 to 5 points at each step, similar to ECAHD. Leveraging the computational efficiency of Sparse Diffusion (SD), ECAHD can afford to run 10 parallel global path sampling steps within the same real-time inference budget. Here, "improved speed" refers not just to absolute runtime reduction but to relative efficiency compared to full-sequence diffusion: since SD requires significantly fewer denoising steps, multiple candidate trajectories can be evaluated in parallel without exceeding latency constraints. We then select the trajectory with the highest score based on the diffusion prediction and cost function. In ECAHD w/ DD, the local path planner was only activated when $c_p^l > 0.1$, as specified by the algorithm.

ECAHD achieved a comparable episode reward to other algorithms while reaching the goal the fastest on average and showing the lowest collision rate. Furthermore, despite incorporating multiple model samplings and collision-aware guidance, ECAHD demonstrated the lowest inference time.

D. Runtime Analysis of Guidance and DD Activation

To analyze the computational cost underlying the inference time reported above, the collision guidance in Algorithm 2 has complexity $\mathcal{O}(KP^l|R|N_{\text{interp}})$, where K is the number of denoising steps, P^l the local horizon, $|R|$ the footprint samples, and N_{interp} the interpolation steps. We measure the wall-clock time of a single DD step, including denoising, cost evaluation, gradient computation, and update, on an NVIDIA A100 80GB PCIe GPU with dual Intel Xeon Gold 6338 CPUs; Table IV reports the per-step runtime.

TABLE IV
RUNTIME BREAKDOWN PER DD STEP (S).

Component	w/o ∇J	w/ ∇J
Denoiser	0.00034	0.00038
Cost J (forward)	0.00079	0.00094
Gradient ∇J	–	0.00177
Update	0.00006	0.00006
Total	0.00126	0.00330

With guidance, 53.7% of the DD step time is spent on ∇J. Since DD is applied only to short local segments when collision risk exceeds a threshold, we further examine worst-case behavior by varying the DD activation ratio. For activation ratios of 10%, 50%, and 100%, the average episode inference time was 0.154 s, 0.759 s, and 3.193 s, respectively, showing near-linear scaling and bounded runtime.

V. CONCLUSION

In this paper, we proposed ECAHD, a hierarchical diffusion-based navigation framework that unifies global planning and local collision-aware refinement in a fully data-driven manner. By separating global and local stages, ECAHD resolves the trade-off between goal-reaching performance and collision avoidance faced by prior full-sequence diffusion planners.

With fast goal-conditioned global planning and shape-aware local refinement triggered only when necessary, EC-

AHD achieves lower collision rates while maintaining high success rates and low computational cost.

While this work focuses on static known environments, future work will extend ECAHD to dynamic obstacles, online sensor fusion (e.g., vision or LiDAR), and real-time feedback for physically actuated robots toward embodied diffusion-based navigation in complex real-world settings.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.

[2] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.

[3] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in Proc. 32nd Int. Conf. Mach. Learn. (ICML), 2015, pp. 2256–2265.

[4] L. Chen, K. Lu, M. Zhou, T. LePaige, J. P. Lewis, and H. Jiang, "Decision Transformer: Reinforcement Learning via Sequence Modeling," in Advances in Neural Information Processing Systems (NeurIPS), vol. 34, 2021, pp. 15084–15097.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 6840–6851, 2020.

[6] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis," in Proc. 39th Int. Conf. Mach. Learn. (ICML), vol. 162, 2022, pp. 9902–9915.

[7] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020, pp. 6840–6851.

[8] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in Proc. 9th Int. Conf. Learn. Represent. (ICLR), 2021, pp. 1–17.

[9] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," in Proc. Robotics: Science and Systems (RSS), 2023.

[10] J. Liu, M. Stamatopoulou, and D. Kanoulas, "DiPPeR: Diffusion-based 2D Path Planner applied on Legged Robots," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Yokohama, Japan, 2024, pp. 9264–9270, doi: 10.1109/ICRA57147.2024.10610013.

[11] W. Yu, J. Peng, H. Yang, J. Zhang, Y. Duan, J. Ji, and Y. Zhang, "LDP: A Local Diffusion Planner for Efficient Robot Navigation and Collision Avoidance," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 1–8. Available: https://arxiv.org/abs/2407.01950

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100–107, 1968.

[13] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report, Dept. Comput. Sci., Iowa State Univ., 1998.

[14] A. Diouf, B. Belzile, M. Saad, and D. St-Onge, "Spherical Rolling Robots Design, Modeling, and Control: A Systematic Literature Review," arXiv preprint arXiv:2310.02240, 2023.

[15] Z. Wang, Y. Li, L. Xu, H. Shi, Z. Ma, Z. Chu, C. Li, F. Gao, K. Yang, and K. Wang, "SF-TIM: A Simple Framework for Enhancing Quadrupedal Robot Jumping Agility by Combining Terrain Imagination and Measurement," arXiv preprint arXiv:2408.00486, 2024.

[16] L. Vanroye, A. S. Sathya, J. De Schutter, and W. Decré, "FATROP: A Fast Constrained Optimal Control Problem Solver for Robot Trajectory Optimization and Control," 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, USA, October 2023

[17] T. Mikolajczyk, E. Mikołajewska, H. F. N. Al-Shuka, T. Malinowski, A. Kłodowski, D. Y. Pimenov, T. Paczkowski, F. Hu, K. Giasin, D. Mikołajewski, and M. Macko, "Recent Advances in Bipedal Walking Robots: Review of Gait, Drive, Sensors and Control Systems," Sensors, vol. 22, no. 12, pp. 4440, 2022.

[18] J. Lee, M. Dai, J. Kim, and A. D. Ames, "Safety-critical Locomotion of Biped Robots in Infeasible Paths: Overcoming Obstacles during Navigation toward Destination," 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, Oct. 2023

[19] Y. Tong, H. Liu, and Z. Zhang, "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects," IEEE/CAA Journal of Automatica Sinica, vol. 11, no. 2, pp. 301-328, 2024.

[20] K. Mombaur, A. Truong, and J.-P. Laumond, "Transferring Human Motion Skills to Humanoid Robots," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, May 2024

[21] C. Khazoom, D. G. Diaz, Y. Ding, and S. Kim, "Humanoid Self-Collision Avoidance Using Whole-Body Control with Control Barrier Functions," in Proc. 21st IEEE-RAS International Conference on Humanoid Robots (Humanoids), Ginowan, Japan, Nov. 2022, pp. 558-565.

[22] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in Proc. First Int. Symp. Search Techn. Artif. Intell. Robot., 2008.

[23] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Trans. Robot. Autom., vol. 12, no. 4, pp. 566–580, 1996.

[24] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA: MIT Press, 2005.

[25] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," Int. J. Robot. Res., vol. 17, no. 7, pp. 760–772, 1998.

[26] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robot. Autom. Mag., vol. 4, no. 1, pp. 23–33, 1997.

[27] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2017, pp. 285–292.

[28] J. Ho and S. Ermon, "Generative adversarial imitation learning," in Adv. Neural Inf. Process. Syst., 2016, pp. 4565–4573.

[29] R. Anderson, et al., "Probabilistic diffusion models for generating optimal paths in stochastic environments," IEEE Robot. Autom. Lett., vol. 6, no. 3, pp. 5432–5439, 2021.

[30] A. Sridhar, D. Shah, C. Glossop and S. Levine, "NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 63-70, doi: 10.1109/ICRA57147.2024.10610665.

[31] Chen, C., Deng, F., Kawaguchi, K., Gulcehre, C., and Ahn, S., "Simple Hierarchical Planning with Diffusion," 2024 International Conference on Learning Representations (ICLR), 2024.

[32] Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F., Computer Graphics: Principles and Practice (3rd ed.), Addison-Wesley, 2013.

[33] K. Kiran, et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 4909-4926, June 2021.

[34] B. Chen, D. M. Monso, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann, "Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion," arXiv preprint arXiv:2407.01392, 2024.

[35] M. Missura and M. Bennewitz, "Predictive Collision Detection for the Dynamic Window Approach," 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 8620-8626. doi: 10.1109/ICRA.2019.8794390.

[36] O. Khatib, L. Sentis, J. H. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," Int. J. Humanoid Robotics, vol. 1, no. 1, pp. 29–43, 2004.

[37] S. Beltran-Hernandez, et al., "Adaptive Imitation Learning Framework for Robotic Complex Contact-Rich Insertion Tasks," Frontiers in Robotics and AI, vol. 7, 2020.

[38] D. Abu-Dakka, et al., "Generalizing Peg-in-Hole Tasks by Learning from Demonstrations," IEEE Robotics and Automation Letters, vol. 2, no. 4, pp. 1941-1948, 2017.

[39] L. Tai, et al., "A Survey of Deep Learning-Based Imitation Learning Techniques for Robotic Manipulation," IEEE Access, vol. 9, pp. 123846-123865, 2021.

[40] O. Doukhi and D.-J. Lee, "Deep Reinforcement Learning for End-to-End Local Motion Planning of Autonomous Aerial Robots in Unknown Outdoor Environments: Real-Time Flight Experiments," Sensors, vol. 21, no. 7, p. 2534, 2021. DOI: 10.3390/s21072534.