

Real-Time Adaptive Motion Planning via Point Cloud-Guided, Energy-Based Diffusion and Potential Fields

Wondmgezahu Teshome¹, Kian Behzad¹, Octavia Camps¹, Michael Everett¹, Milad Siami¹, and Mario Sznai¹

Abstract—Motivated by the problem of pursuit-evasion, we present a motion planning framework that combines energy-based diffusion models with artificial potential fields for robust real time trajectory generation in complex environments. Our approach processes obstacle information directly from point clouds, enabling efficient planning without requiring complete geometric representations. The framework employs classifier-free guidance training and integrates local potential fields during sampling to enhance obstacle avoidance. In dynamic scenarios, the system generates initial trajectories using the diffusion model and continuously refines them through potential field-based adaptation, demonstrating effective performance in pursuit-evasion scenarios with partial pursuer observability.

Index Terms—AI-Based Methods, Motion and Path Planning, Planning under Uncertainty.

I. INTRODUCTION

THIS paper is motivated by the problem of using robots to guide crowds to safety in scenarios involving rapidly evolving threats, such as an active shooter or a forest fire. This problem can be abstracted to real-time motion planning in partially known scenarios characterized by the presence of both static obstacles and dynamic, adversarial agents.

Among the many types of motion planning algorithms, sampling-based methods [1]–[3] have proven effective for high-dimensional problems but often generate jerky paths requiring post-processing and suffer from computational inefficiency in sampling [4]. Meanwhile, optimization-based approaches [5]–[8] use gradient or second-order information to efficiently find smooth trajectories. However, these methods are limited to providing locally optimal solutions [9].

Recent learning-based approaches address some of these limitations by using data-driven models for improved planning efficiency and adaptability in high-dimensional spaces and complex environments [10]. Among these, diffusion models have emerged as particularly promising for trajectory generation, offering powerful distribution modeling capabilities [11]. While diffusion planners excel at capturing multimodal behaviors and handling environmental constraints, they face computational challenges during the iterative sampling process, especially in real-time applications. In contrast, classical methods, such as artificial potential fields (APF) [12], excel at local

Manuscript received March 11, 2025; Revised May 27, 2025; Accepted June 25, 2025. This paper was recommended for publication by Editor Bera upon evaluation of the reviewers comments. This work was supported by NSF grants CNS-2038493 and CMMI-2208182, AFOSR grant FA9550-19-1-0005, ONR grant N00014-21-1-2431, and the Sentry DHS Center of Excellence under Award 22STESE00001-03-03.

¹The authors are with ECE Department, Northeastern University, Boston, MA 02115, USA {teshome.w, behzad.k, m.everett}@northeastern.edu, {camps, siami, msznaier}@coe.neu.edu

©2026 IEEE

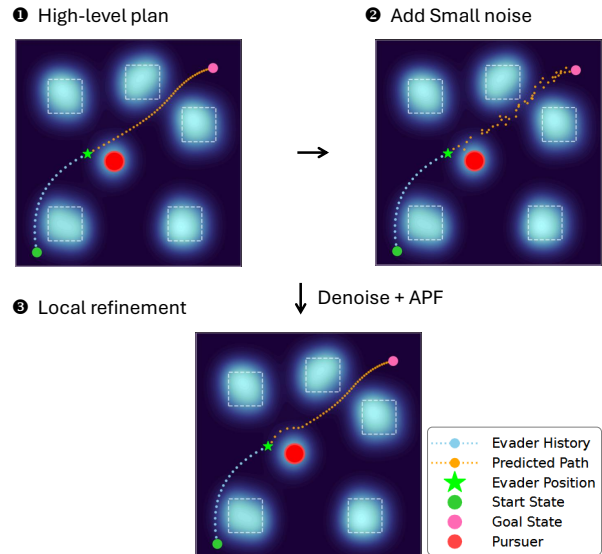


Fig. 1. **Overview of our approach:** (1) Initial energy-based diffusion trajectory planning, where light-blue regions represent high-energy potential fields (obstacles), lightblue dots trace the evader’s (green star) state history, red sphere indicates a dynamic pursuer, and dark purple areas show safe, low-energy navigation spaces. (2) When the pursuer approaches the evader, local path exploration is performed via small noise perturbations. (3) Final trajectory refinement using denoising and APF, demonstrating adaptive path planning that avoids obstacles.

obstacle avoidance but lack the global planning capabilities of learning-based approaches. In this work, we propose a hybrid framework that integrates energy-based diffusion models with APFs, leveraging the global planning capabilities of learned models while maintaining the reactive benefits of classical methods. Our approach, outlined in Fig. 1, processes obstacle information directly from point clouds, bypassing reliance on geometric maps. This addresses a practical constraint in robotic systems, where the map may be unknown but a depth camera or LiDAR sensor could provide point cloud measurements. To handle dynamic environments, we propose a hierarchical framework that integrates high level planning with real-time denoising and APF-based refinement (Figure 2). This real-time denoising step amounts to running a few steps in a diffusion, allowing for balancing real-time computational cost versus performance. As shown in the paper, this approach proves to be particularly effective in pursuit-evasion scenarios. Specifically, our contributions include:

- A real-time motion planning algorithm for environments with both static obstacles and adversarial dynamic agent, which extends energy-based diffusion models to leverage artificial potential fields and a point cloud encoder, and
- Demonstrations of the proposed algorithms in previously

unseen scenarios, compositional scenarios, and a hardware implementation of pursuit-evasion.

II. RELATED WORK

Recent learning-based approaches have shown promise in addressing the limitations of classical methods. For example, MPNet [13] pioneered the use of neural networks for motion planning, while [14] improved obstacle representation using PointNet [15]. Diffusion models enable trajectory generation through iterative denoising. Diffuser [11] introduced trajectory generation conditioned on rewards and constraints. Motion Planning Diffusion (MPD) [16] extended this by learning diffusion models as priors for trajectory distributions. Most closely related to our work is Potential-Based Diffusion Planning [17], which learns separate potential functions that can be composed at test time. While [17] requires explicit knowledge of obstacle locations, our method differs by processing point cloud representations and integrating artificial potential fields during sampling.

In dynamic environments, methods like RRTX [18] enable rapid replanning, while SIPP [19] improves efficiency by grouping safe configurations over time. Classical approaches such as artificial potential fields (APF) have been enhanced for smoother navigation in unstructured settings [20] and adapted for pursuit-evasion tasks [21]. Deep reinforcement learning has also been employed for dynamic obstacle avoidance [22], [23], with architectures like a two-stream Q-network for spatial-temporal planning [24], and cooperative frameworks for multi-robot coordination [25].

In pursuit-evasion scenarios, [26] developed evasion strategies using Deep Q-Networks, while [27] proposed multi-agent DRL systems with adversarial-evolutionary training. Additionally, [28] approached the pursuit-evasion problem from the pursuer perspective, developing a multi-UAV pursuit framework with target prediction capabilities to enhance coordination in obstacle-rich environments.

Despite these advances, DRL-based approaches generally suffer from limitations including poor generalization to novel scenarios and susceptibility to overfitting [29].

Our approach addresses these limitations by integrating learned global planning with reactive local planning through a hybrid diffusion-APF framework. This combines point cloud-based obstacle encoding with energy-based diffusion models, enabling both compositionality between different static obstacle sets and enhanced real-time performance for dynamic scenarios through reactive potential fields.

III. PRELIMINARIES

A. Problem Formulation

In this paper, we address the pursuit-evasion motion planning problem in an environment where an evader must navigate to a goal while avoiding both static obstacles and a dynamic pursuer. Let \mathcal{C} be the configuration space of the evader, with obstacle space \mathcal{C}_{obs} and free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$. The environment contains static obstacles $\mathcal{O}_{static} = \{O_1, O_2, \dots, O_n\}$ represented as point clouds, and a pursuer \mathcal{P} with state $p(t)$ at time t , modeled as a sphere with radius

r_p . Given an initial state $s_s \in \mathcal{C}_{free}$, a goal state $s_g \in \mathcal{C}_{free}$, and the environment representation, our objective is to find a trajectory $\tau : [0, T] \rightarrow \mathcal{C}_{free}$ that satisfies the following constraints:

- Boundary conditions: $\tau(0) = s_s, \tau(T) = s_g$
- Remains collision-free with static obstacles: $\tau(t) \in \mathcal{C}_{free} \forall t \in [0, T]$, and
- Maintains safe distance from the pursuer: $\|\tau(t) - p(t)\| > r_{safe} \forall t \in [0, T]$.

The pursuer is only partially observable, detected when within radius r_{detect} of the evader. The planner must adapt in real-time to both static obstacles and the pursuer.

B. Diffusion Models

Diffusion models learn to generate data with the same distribution as the training data [30], [31]. To this effect, during the training phase, noise is added to the data through a forward diffusion process that transforms trajectories τ_0 with an unknown distribution $q_o(\tau_0)$ into Gaussian distributed ones via

$$q(\tau_t | \tau_{t-1}) = \mathcal{N}(\tau_t; \sqrt{1 - \beta_t} \tau_{t-1}, \beta_t \mathbf{I})$$

where β_t represents the noise level at each step $t \in \{1, \dots, N\}$ and N denotes the number of diffusion steps. The Gaussian nature of this process allows for obtaining τ_N in a single step through

$$q(\tau_N | \tau_0) = \mathcal{N}(\tau_N; \sqrt{\bar{\alpha}_N} \tau_0, (1 - \bar{\alpha}_N) \mathbf{I})$$

with $\alpha_N = 1 - \beta_N$ and $\bar{\alpha}_N = \prod_{s=1}^N \alpha_s$. This is followed by a reverse diffusion process that learns to recover the data by approximating

$$p_\theta(\tau_{t-1} | \tau_t) = \mathcal{N}(\tau_{t-1}; \mu_\theta(\tau_t, t), \sigma_t^2 \mathbf{I})$$

where μ_θ is the learnable mean function and $\sigma_t^2 \mathbf{I}$ is the covariance with variance parameter $\sigma_t^2 = \beta_t(1 - \bar{\alpha}_{t-1}) / (1 - \bar{\alpha}_t)$. This can be achieved by training a neural network ϵ_θ with parameters θ to predict the noise component by minimizing

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau_0, t, \epsilon \sim \mathcal{N}(0, \mathbf{I})} \|\epsilon - \epsilon_\theta(\tau_t, t)\|^2$$

where $\tau_t = \sqrt{\bar{\alpha}_t} \tau_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$. For conditional diffusion models, additional inputs \mathbf{c} are included, modifying the noise prediction network to $\epsilon_\theta(\tau_t, t, \mathbf{c})$. Once the model is trained, data with a distribution $q_o(\tau)$ can be generated by applying the learned denoising process to random noise with distribution $\mathcal{N}(\tau_N; \sqrt{\bar{\alpha}_N} \tau_0, (1 - \bar{\alpha}_N) \mathbf{I})$.

C. Energy-Based Diffusion Models

Recent work has shown that diffusion models can benefit from energy-based parameterization as an alternative to the traditional score-based approach [32]. Score-based models [33] directly estimate the score function $-\nabla_\tau \log p_\theta(\tau_t)$. Energy-based diffusion models take a different approach by first defining a parameterized energy function $f_\theta(\tau, t) = -\|s_\theta(\tau, t)\|^2$, where $s_\theta(\tau, t)$ represents a neural network with vector output [32]. The corresponding score is given by:

$$\epsilon_\theta(\tau, t) = -\nabla_\tau f_\theta(\tau, t) = \nabla_\tau \|s_\theta(\tau, t)\|^2$$

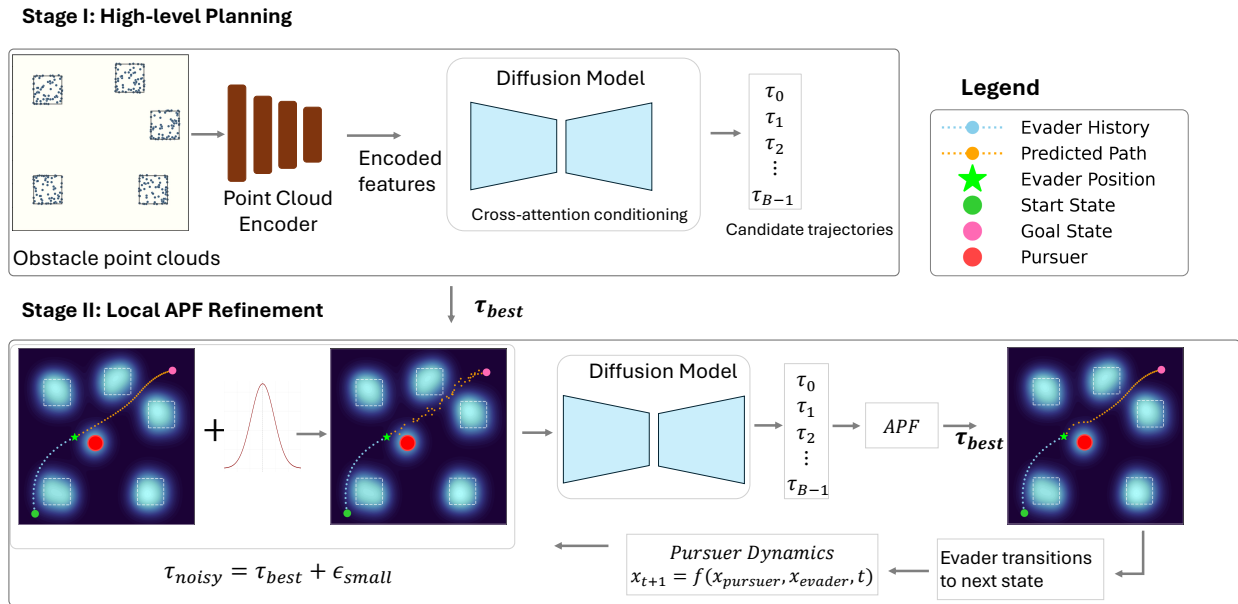


Fig. 2. **Trajectory planning pipeline for a Pursuit-Evasion case:** In the high-level planner, obstacle point clouds are processed by a pre-trained encoder to obtain a low-dimensional encoding of the point cloud features. The diffusion model then generates multiple candidate trajectories conditioned on start/goal states and these obstacle features. The best trajectory is passed to the low-level planner, which performs refinement when a pursuer approaches the evader. This refinement combines denoising with APF while conditioning on the evader’s state history. The refined trajectory is then used as input for the next planning iteration, creating an adaptive system that continuously responds to the pursuer’s movements while navigating through the environment.

While this approach requires computing an additional backward pass gradient, it provides significant benefits by enabling the composition of multiple diffusion models through their unnormalized log-probabilities [32]. For example, [17] used this compositional property to combine multiple motion constraints in trajectory planning.

D. Artificial Potential Fields

Artificial potential field methods [12] provide a motion planning framework by modeling the robot’s configuration space as a virtual force field. In this framework, the total potential field $U(\tau) = U_{att}(\tau) + U_{rep}(\tau)$ combines attractive forces toward goals and repulsive forces away from obstacles. The robot’s trajectory τ typically evolves according to $\Delta\tau = -\nabla U(\tau)$, guiding the robot toward regions of lower potential energy. While traditional APF approaches are known to suffer from local minima issues [34], [35], our integration with learning-based trajectory planning offers key advantages. Specifically, this approach allows the learned model to first generate trajectory proposals, after which the APF guides these initial trajectories away from obstacles with minimal interference to the overall trajectory distribution. In our implementation, we employ a simple yet effective exponential repulsive potential field adapted from [36] around obstacles to ensure collision avoidance during trajectory generation, as detailed in Section IV.

IV. PROPOSED FRAMEWORK FOR ADAPTIVE MOTION PLANNING

We propose a hierarchical trajectory planning framework that combines energy-based diffusion models with artificial

potential fields for both static and dynamic environments. Our method integrates high-level trajectory generation with continuous refinement to handle pursuit-evasion scenarios.

A. Training Conditional Energy-Based Diffusion Model

The overall framework is shown in Fig. 2. At the high-level planning stage, a conditional energy-based diffusion model learns trajectory distributions conditioned on obstacle point clouds. Each obstacle in the scene is represented by a set of points (2D or 3D) sampled from its surface and interior. The point clouds are processed using an encoder that extracts point features, which are then processed through set transformer blocks [37] to produce a multi-scale scene representation. The scene encoding conditions the diffused trajectory features through cross-attention layers at multiple resolutions in the temporal U-Net [11], allowing the model to incorporate obstacle information. The energy-based training objective follows [17], [32]:

$$L_{MSE} = \|\epsilon - \nabla_{\tau} E_{\theta}(\tau_t(\tau_0, \epsilon), t, z)\|^2 \quad (1)$$

where E_{θ} is the energy function parameterized by a temporal U-Net architecture with cross-attention mechanisms that allow diffusion features to attend to obstacle point cloud embeddings at multiple scales and z are the obstacle features. In this paper we adopt classifier-free guidance (CFG) training [38], where we randomly drop obstacle points. This enables the model to jointly learn the distribution of the trajectories, conditioned on the obstacle locations, and the unconditional motion prior.

Algorithm 1 Energy-Based Diffusion Planning with APF for Static Obstacles

```

1: Input: initial start state  $s_s$ , goal state  $s_g$ , pretrained diffusion
   energy function  $E_\theta$  with integrated encoder, diffusion steps  $N$ ,
   static obstacle point cloud  $\mathcal{O}_{\text{static}}$ , guidance scale  $w$ , batch size
    $B$ , APF parameters  $\gamma$ ,  $d_{\text{thresh}}$ ,  $N_{\text{apf}}$ 
2:  $z_{\text{latent}} \leftarrow f_{\text{encoder}}(\mathcal{O}_{\text{static}})$ 
3:  $\tau_N \sim \mathcal{N}(0, \mathbf{I})$ 
4: for  $t = N, \dots, 1$  do
5:    $e_{\text{cond}} \leftarrow \nabla_{\tau} E_\theta(\tau_t, t, z_{\text{latent}})$ 
6:    $e_{\text{uncond}} \leftarrow \nabla_{\tau} E_\theta(\tau_t, t, \mathbf{0})$ 
7:    $e_{\text{comb}} \leftarrow (1 + w)e_{\text{cond}} - we_{\text{uncond}}$ 
8:    $\mu_t \leftarrow \frac{1}{\sqrt{\alpha_t}} \left( \tau_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} e_{\text{comb}} \right)$ 
9:   if  $t < N_{\text{apf}}$  then  $\triangleright$  Apply APF in final denoising stages
10:     $\mathbf{O}_{\text{nearest}} \leftarrow \text{NP}(\mu_t, \mathcal{O}_{\text{static}})$   $\triangleright$  Find nearest obstacle points
    using KD-Tree
11:     $\mathbf{d} \leftarrow \mu_t - \mathbf{O}_{\text{nearest}}$   $\triangleright$  Vector to nearest obstacles
12:     $d \leftarrow \|\mathbf{d}\|$   $\triangleright$  Scalar distances
13:     $\mathbf{F} \leftarrow \gamma \exp(-d/d_{\text{thresh}}) \cdot \mathbf{d}/d$   $\triangleright$  Repulsive force
14:     $\mu_t \leftarrow \mu_t + \mathbf{F}$   $\triangleright$  guide trajectories towards regions of
    lower potential energy
15:  end if
16:   $\tau_{t-1} \leftarrow \mu_t + \sigma_t \xi$ ,  $\xi \sim \mathcal{N}(0, \mathbf{I})$ 
17:   $\tau_{t-1}[0] \leftarrow s_s, \tau_{t-1}[H-1] \leftarrow s_g$   $\triangleright$  Apply hard conditioning
18: end for
19: return  $\tau_0$ 
    
```

B. Trajectory Generation with Integrated Potentials

For scenarios with static obstacles, we propose the trajectory generation approach detailed in Algorithm 1. This approach combines learned energy potentials with APF during the sampling process. It consists of two key components: (1) a reverse diffusion process that generates trajectories using classifier-free guidance, and (2) an integration of APF during the denoising process. Given a static obstacle point cloud $\mathcal{O}_{\text{static}}$, we first get its latent representation through an obstacle encoder that is trained jointly with the energy function E_θ . During the reverse diffusion process, we iteratively denoise the trajectory while incorporating both learned and external potentials. At each step t , we generate trajectories conditioned on the start state, goal state and obstacle features through the following denoising process [31]:

$$\tau_{t-1} = \mu_t + \sigma_t \xi, \quad \xi \sim \mathcal{N}(0, \mathbf{I}) \quad (2)$$

where τ_{t-1} represents the trajectory at timestep $t-1$ and ξ is random Gaussian noise added during the reverse process if $t > 1$, otherwise it is 0. Following [31], the denoised prediction μ_t is computed using:

$$\mu_t = \frac{1}{\sqrt{\alpha_t}} \left(\tau_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} e_{\text{comb}} \right) \quad (3)$$

where e_{comb} combines the conditional and unconditional energy gradients as [38]:

$$e_{\text{comb}} = (1 + w)e_{\text{cond}} - we_{\text{uncond}} \quad (4)$$

where w is the guidance scale that controls the conditioning strength, e_{cond} is the conditional energy gradient, and e_{uncond} is the unconditional energy gradient. The external APF is

integrated with the learned potential during the final denoising stages ($t < N_{\text{apf}}$). The APF generates repulsive forces, which guide the trajectory towards regions of low potential, defined as:

$$\mathbf{F}(\mathbf{x}, \mathcal{O}) = \gamma \exp\left(-\frac{d}{d_{\text{thresh}}}\right) \frac{\mathbf{d}}{d} \quad (5)$$

where $\mathbf{d} = \mathbf{x} - \text{NP}(\mathbf{x}, \mathcal{O})$ and $d = \|\mathbf{d}\|$. Here, $\text{NP}(\mathbf{x}, \mathcal{O})$ finds the nearest obstacle point, with parameters γ and d_{thresh} controlling avoidance strength and influence range, respectively. This potential field generates a repulsive force that increases exponentially as the distance to the nearest obstacle decreases. The complete trajectory update incorporating both the denoised prediction and APF becomes:

$$\tau_{t-1} = \mu_t + \mathbf{F}(\mu_t, \mathcal{O}) + \sigma_t \xi \quad (6)$$

Compositional Sampling: For handling multiple sets of obstacle configurations, we leverage the compositional property of energy-based diffusion models [17], [32]. When sampling trajectories that must respect multiple obstacle configurations, we modify the energy gradient combination in Algorithm 1 as:

$$e_{\text{comb}} = e_{\text{uncond}} + \sum_{i=1}^{N_{\text{obs}}} w_i (e_{\text{cond}}^i - e_{\text{uncond}}) \quad (7)$$

where e_{cond}^i represents the conditional energy gradient for the i -th set of obstacles, and w_i is the corresponding guidance scale. This approach enables handling previously unseen scenarios, as long as they can be decomposed into ones where models are available, by simply composing the component's energy gradients during the sampling process.

C. Adaptive Trajectory Refinement for Dynamic Environments including Adversarial Agent

In principle, this paper's motivating problem of pursuit-evasion (i.e., reaching a goal while avoiding a dynamic pursuer) can be solved through a model predictive approach: At each time instant, exploit the compositional property of Algorithm 1 to generate a trajectory conditioned on the present and predicted future positions of the pursuer, given by

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \Delta t \cdot v \cdot \mathbf{d}_{\text{pursuit}} \quad (8)$$

Here $\mathbf{d}_{\text{pursuit}} = k_p \cdot \frac{\mathbf{s}_t - \mathbf{p}_t}{\|\mathbf{s}_t - \mathbf{p}_t\|}$, where \mathbf{p}_t and \mathbf{s}_t are pursuer and evader positions respectively, v is the pursuer velocity, and k_p controls the pursuit strength. However, this approach is relatively slow and thus may not be able to handle rapidly changing scenarios. To address this challenge, we propose the approach outlined in Algorithm 2. Dynamic adaptation is handled through continuous refinement, where the evader only considers pursuer avoidance within a specified detection radius r_{detect} . As the iteration progresses, the evader's history (executed) states are stored and used as conditioning states for subsequent trajectory sampling. Intuitively, this approach amounts to running just a few denoising steps, starting from the current trajectory, as opposed to a complete reverse diffusion. To maintain dynamically feasible trajectories, we implement velocity-constrained smoothing at transition points between executed and planned states. This smoothing process,

Algorithm 2 Dynamic Trajectory Refinement with Pursuit-Evasion

```

1: Input: initial trajectory  $\tau_{\text{high}}$  from Algorithm 1, goal state  $s_g$ ,
   dynamic simulation steps  $N_{\text{dyn}}$ , static obstacles  $\mathcal{O}_{\text{static}}$ , batch size
    $B$ , APF parameters  $\gamma$ ,  $d_{\text{thresh}}$ , detection range  $r_{\text{detect}}$ 
2: Initialize pursuer state  $\mathbf{p}_0$ , executed history  $\mathcal{H} \leftarrow \tau_{\text{high}}[0]$ 
3: for  $t = 0, \dots, N_{\text{dyn}} - 1$  do
4:   Update pursuer:  $\mathbf{p}_t \leftarrow f_{\text{dynamic}}(t, \tau_{\text{high}}[0])$ 
5:    $\tau_M \leftarrow \tau_{\text{high}} + \sigma\xi$ ,  $\xi \sim \mathcal{N}(0, I)$   $\triangleright$  Add small noise
6:    $\tau_M \leftarrow \text{repeat}(\tau_M, B)$   $\triangleright$  Create batch of candidates
7:   for  $j = M, \dots, 1$  do  $\triangleright$  Refinement steps
8:     Refine  $\tau_j$  using steps 5-8 from Algorithm 1
9:     if  $j < M_{\text{threshold}}$  then  $\triangleright$  Apply APF in final steps
10:       $d_{\text{pursuer}} \leftarrow \|\tau_j[t] - \mathbf{p}_t\|$ 
11:      if  $d_{\text{pursuer}} < r_{\text{detect}}$  then
12:        Apply APF for both  $\mathcal{O}_{\text{static}}$  and  $\mathbf{p}_t$ 
13:      else
14:        Apply APF for  $\mathcal{O}_{\text{static}}$  only
15:      end if
16:    end if
17:  end for
18:   $\tau_{\text{smooth}} \leftarrow \text{Algorithm 3}(\tau_0[t], \tau_0[t+n], \Delta t, n, v_{\text{max}})$   $\triangleright$  Apply
  smooth transitions
19:   $\tau_{\text{best}} \leftarrow \text{select\_best}(\tau_{\text{smooth}})$   $\triangleright$  Based on cost metric
20:  Update history:  $\mathcal{H} \leftarrow \mathcal{H} \cup \{\tau_{\text{best}}[t+1]\}$   $\triangleright$  Update executed
  state
21:   $\tau_{\text{best}}[0 : |\mathcal{H}|] \leftarrow \mathcal{H}$ ,  $\tau_{\text{best}}[H-1] \leftarrow s_g$   $\triangleright$  Apply history and
  goal conditioning
22:   $\tau_{\text{high}} \leftarrow \tau_{\text{best}}$ 
23:  if  $\|\tau_{\text{high}}[t] - s_g\| < \varepsilon$  then
24:    break
25:  end if
26: end for
27: return  $\tau_{\text{high}}$ 
    
```

detailed in Algorithm 3, is applied to a custom window size of the trajectory segments. Our experiments use a small window size (2 or 3). After applying this smoothing, the best trajectory is selected, and the history state is updated. This process continues until the evader reaches within ε distance of the goal state, i.e., $\|\tau_{\text{high}}[t] - s_g\| < \varepsilon$.

Trajectory Selection: From the batch of generated trajectories, we first identify collision-free candidates through obstacle checking. Among these feasible trajectories, we select the trajectory with the lowest cost using a weighted cost metric: $\text{cost}(\tau) = w_l \cdot L(\tau) + w_s \cdot S(\tau)$ where $L(\tau)$ denotes path length, $S(\tau)$ measures trajectory smoothness, and w_l, w_s are their respective weights.

Algorithm 3 Smoothing with Velocity Constraints

```

1: Input: States  $s_1, s_2$ , timestep  $\Delta t$ , steps  $n$ , max velocity  $v_{\text{max}}$ 
2:  $\Delta \mathbf{w} \leftarrow s_2^{\text{pos}} - s_1^{\text{pos}}$   $\triangleright$  Position difference
3:  $\mathbf{v} \leftarrow \Delta \mathbf{w} / \|\Delta \mathbf{w}\|$   $\triangleright$  Unit direction
4:  $\mathbf{v}_{\text{tar}} \leftarrow \Delta \mathbf{w} / (n \cdot \Delta t)$   $\triangleright$  Target velocity
5:  $\mathbf{v}_{\text{base}} \leftarrow \begin{cases} \mathbf{v} \cdot v_{\text{max}} & \text{if } \|\mathbf{v}_{\text{tar}}\| > v_{\text{max}} \\ \mathbf{v}_{\text{tar}} & \text{otherwise} \end{cases}$   $\triangleright$  Clamped velocity
6:  $\mathbf{w}_t \leftarrow s_1^{\text{pos}} + t \cdot \mathbf{v}_{\text{base}}$  for  $t \in [\Delta t, 2\Delta t, \dots, n\Delta t]$ 
7:  $\tau_{\text{smooth}} \leftarrow [\mathbf{w}_t, \mathbf{v}_{\text{base}}]_{t=1}^n$   $\triangleright$  Concatenate positions & velocities
8: return  $\tau_{\text{smooth}}$ 
    
```

V. EXPERIMENTS

Our experimental results demonstrate that our novel integration of energy-based diffusion models with APFs achieves better performance in trajectory generation for both static and dynamic environments. We evaluate our approach in static obstacle avoidance and pursuit-evasion scenarios, evaluating different aspects of generated trajectories for each case.

A. Experimental Setup

Dataset: We evaluate our method in maze environments with 6 and 10 obstacles. Following [16], demonstrations are generated using RRT-connect [39], B-spline interpolation, and GPMP optimization [7]. The training set includes 2,000 obstacle configurations, each with 20 start-goal pairs and 25 diverse trajectories. Trajectories are of size $48 \times d$, where $d = 4$ (2D) or $d = 6$ (3D), with 48 timesteps. For 3D, training uses 10-obstacle environments; testing is on composed setups. Evaluation uses 100 unseen configurations with 20 start-goal pairs each. In pursuit-evasion, a single unseen environment with a dynamic pursuer is used across 5 runs, each with 100 start-goal pairs. The evader always moves faster than the pursuer.

Evaluation Metrics: In static obstacle environments, performance is evaluated using five metrics: *Success Rate* (percentage of runs with at least one collision-free trajectory), *Collision Intensity* (proportion of colliding waypoints), *Waypoint Variance* (path diversity), *Path Length* (PL; trajectory efficiency), and *Computation Time* (average sampling time). For pursuit-evasion, we assess the evader’s Goal Success Rate, Collision Rate (including obstacles and captures), and Path Length. An overall *Score* combines goal achievement and collision avoidance. Metrics marked (\uparrow) are better when higher, and (\downarrow) when lower.

Baselines: We compare our method to several planners. For static obstacles, we benchmark against RRTC-GPMP, BIT* [40], and MPD [16], and include our base model without APF to assess its effect. In the pursuit-evasion task, we compare our Diff-APF planner to the SAC reinforcement learning baseline [41], along with two ablations: Diff-base and Diff-SAPF (APF for static obstacles only). Results highlight the benefits of integrating artificial potential fields for improved obstacle avoidance.

B. Implementation Details

We build on the U-Net architecture from [16], [17], using residual temporal blocks. For sampling, we use DDIM [42] with 5 diffusion steps and classifier-free guidance sampling with a guidance scale of 2.0.

Obstacle data is processed with a set-based encoder for point clouds of size $N \times 64 \times D$ (64 points per obstacle, $D=2, 3$). The encoder combines PointNet-style feature extraction [15] with set transformer blocks [37], producing a 320-dimensional latent vector used for conditioning via cross-attention. The encoder is trained end-to-end with the diffusion model. Training uses 100 diffusion steps, batch size 128, learning rate $1e-4$, on an NVIDIA RTX 2080 Ti. Conditional dropout ($p=0.2$) is applied during training to improve generalization.

TABLE I

COMPARISON OF DIFFERENT MOTION PLANNING METHODS FOR STATIC OBSTACLE ENVIRONMENTS. BOLD VALUES INDICATE TOP TWO RESULTS FOR EACH METRIC.

Dataset	Method	Success	Collision	Waypoint	Computation	Path
		Rate(%) \uparrow	Intensity(%) \downarrow	Variance \uparrow	Time (s) \downarrow	Length \downarrow
Maze2D-6obs	RRTC-GPMP	100.00 \pm 0.00	0.00 \pm 0.00	1.89 \pm 0.27	3.4 \pm 0.02	1.78 \pm 0.07
	BIT*	99.55 \pm 1.43	0.52 \pm 0.13	0.28 \pm 0.05	0.32 \pm 0.01	1.46 \pm 0.06
	MPD	78.8 \pm 9.06	7.94 \pm 2.06	0.26 \pm 0.08	2.11 \pm 0.03	1.58 \pm 0.07
	Diff-base(ours)	99.80 \pm 1.21	1.94 \pm 1.17	1.56 \pm 0.31	0.29 \pm 0.01	1.84 \pm 0.08
	Diff-APF(ours)	100.00 \pm 0.00	0.35 \pm 0.19	1.75 \pm 0.29	0.47 \pm 0.03	1.90 \pm 0.09
Maze2D-10obs	RRTC-GPMP	100.00 \pm 0.00	0.00 \pm 0.00	2.40 \pm 0.37	3.43 \pm 0.03	1.87 \pm 0.08
	BIT*	99.80 \pm 1.21	0.79 \pm 0.16	0.50 \pm 0.09	0.32 \pm 0.01	1.55 \pm 0.08
	MPD	59.90 \pm 13.35	11.57 \pm 2.25	0.31 \pm 0.19	2.11 \pm 0.02	1.58 \pm 0.09
	Diff-base(ours)	92.80 \pm 5.84	7.00 \pm 1.63	1.02 \pm 0.30	0.3 \pm 0.01	1.78 \pm 0.10
	Diff-APF(ours)	99.85 \pm 0.85	2.90 \pm 1.07	1.94 \pm 0.60	0.6 \pm 0.05	2.10 \pm 0.22

C. Results and Discussion

We evaluate our method in both static and dynamic obstacle scenarios. As shown in Table I, in static environments, our APF-enhanced model achieves superior performance across multiple metrics, with a 100% success rate and only 0.35% collision intensity in 6-obstacle settings—matching RRTC-GPMP in quality but with much faster sampling (0.47s vs 3.4s). It also shows high trajectory diversity (waypoint variance: 1.75). While BIT* yields shorter paths, our method offers better success rates.

The advantages of our approach become more pronounced in complex environments with 10-obstacles. our APF model maintains a 99.85% success rate, outperforming MPD (59.90%) and closely matching RRTC-GPMP and BIT*. APF integration reduces collisions from 7.00% to 2.90% and increases waypoint variance from 1.02 to 1.94 over the base model.

For pursuit-evasion, Diff-APF achieves the highest score (84.2%, Table II), outperforming SAC while generating shorter, safer paths. Qualitative results (Fig. 3) show our method adapting better to unseen obstacles, generating smooth, collision-free trajectories.

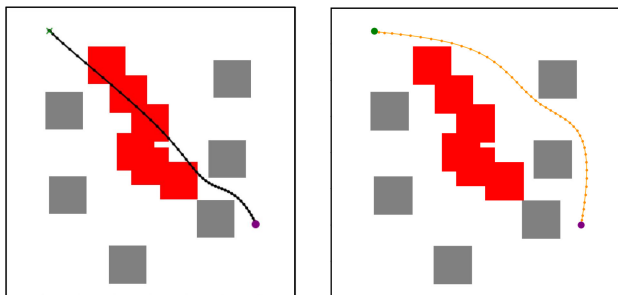


Fig. 3. Trajectory generation comparison on Maze2D in the presence of unseen objects. *left*: MDP fails to avoid the obstacles. *right*: Our method successfully avoids them.

In 3D (Fig. 4), our approach successfully navigates a complex arrangement of 20 static obstacles created by composing two separate obstacle configurations, demonstrating generalization. In pursuit-evasion (Fig. 5), our method avoids both static and dynamic threats, unlike SAC, which struggles in unseen configurations.

Ablation studies (Fig. 6) show APF-enhanced models maintain >99.8% success in complex settings. Collision intensity stabilizes after 20 DDIM steps, suggesting that increasing sampling steps further mainly impacts computational cost. Sampling time scales linearly with the number of steps, increasing approximately 9x from 5 to 50 steps.

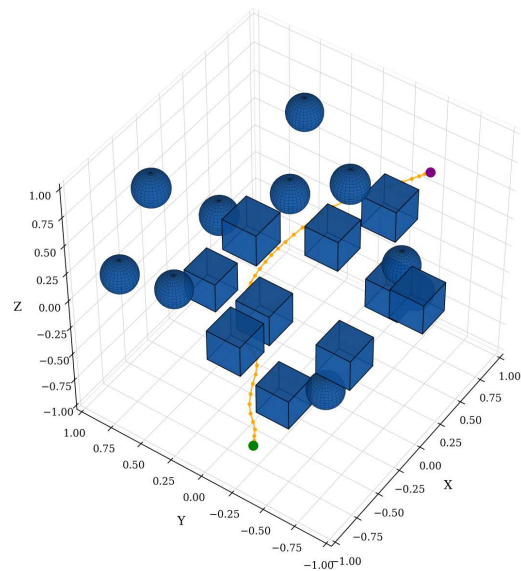


Fig. 4. Trajectory generation on Maze3D environment with box and sphere obstacles.

TABLE II
QUANTITATIVE RESULTS ON PURSUIT-EVASION TASK
COMPARING DIFFERENT METHODS.

Method	Score(%) \uparrow	PL \downarrow
SAC	70.0	3.20
Diff-base	30.6	1.54
Diff-SAPF	57.6	1.65
Diff-APF	84.2	1.69

D. Physical Demonstrations: Pursuer-Evader Case

To validate our theory in an actual setup, we conducted experiments using sensor-rich RC QCars. The QCar1 by

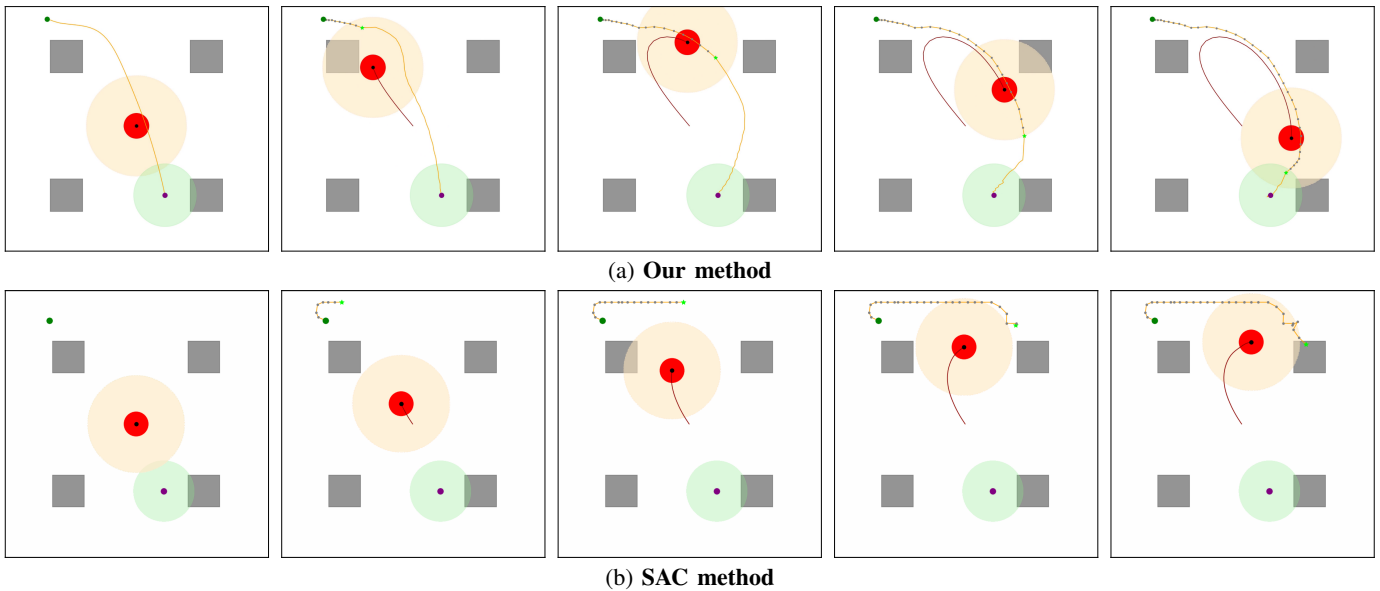


Fig. 5. Pursuit-evasion simulation: Our method (top) successfully adapts to unseen obstacle configurations while the DRL-based SAC method (bottom) struggles. Orange lines show evader path, red circles represent pursuers with detection zones (light yellow circular regions), green areas indicate safe zones, and green/purple dots mark start/goal states.

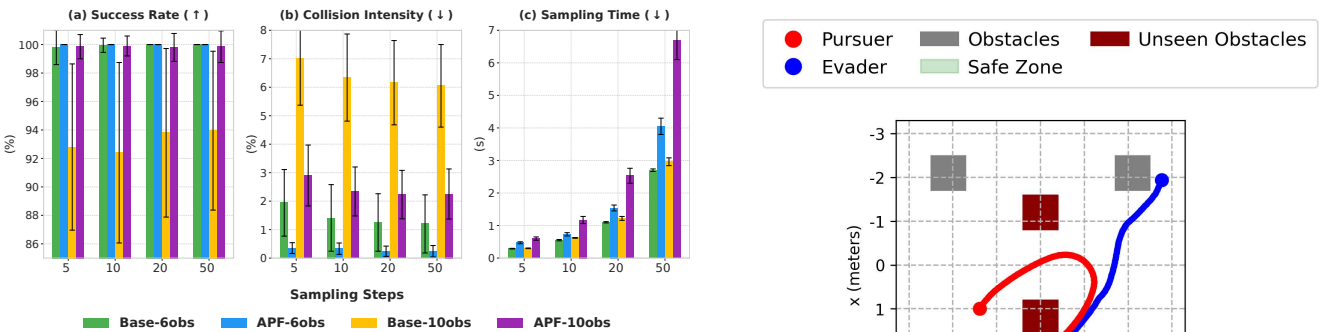


Fig. 6. Ablation on DDIM Sampling Steps: Comparison between Base and APF methods. (a) trajectory planning success rate, (b) collision intensity, and (c) computational sampling time in Maze2D environments with 6 and 10 obstacles. Error bars indicate standard deviation.

Quanser is a 1/10-scale autonomous vehicle platform designed for robotics and AI research. The platform includes key onboard sensors such as an IMU, RGB-D camera, and LiDAR, and is equipped with motion capture markers for integration with external tracking systems, making it ideal for experiments in localization, control, and navigation. Real-time performance evaluation using NVIDIA GeForce RTX 3080 Ti connected to the QCar1 shows our method achieves 0.15 seconds per iteration compared to MPD (0.73 seconds) and BIT* (0.3 seconds). SAC provides direct action selection (0.0004 seconds per action) rather than trajectory generation. Figure 7 shows a pursuit-evasion experiment in a 6x6 m² environment with 6 total obstacles: 4 known obstacles and 2 additional unseen obstacles. The top visualization displays the executed trajectories of both the evader and pursuer, while the bottom shows the corresponding physical lab setup. Additional experiments and video demonstrations are available at <https://github.com/wondmgezahu/RAMP>.

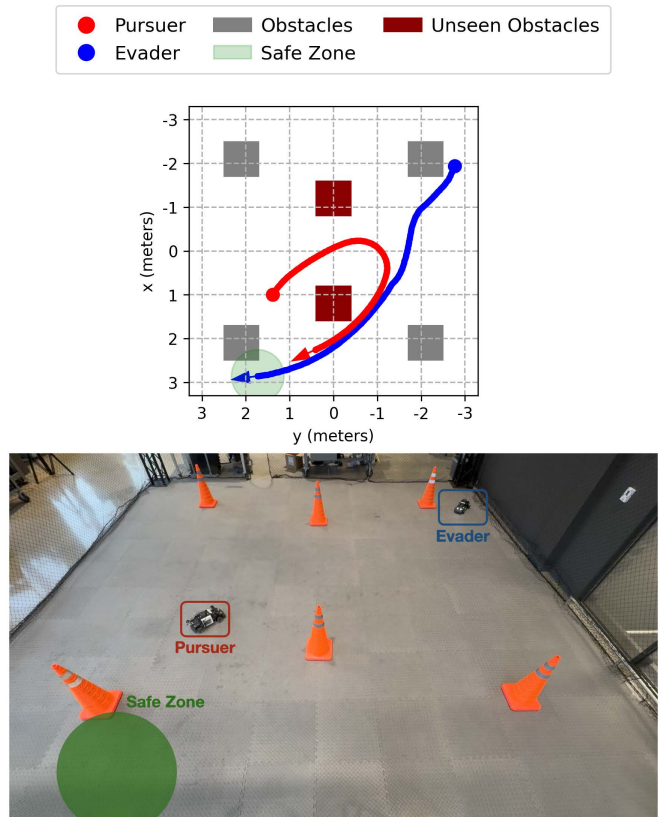


Fig. 7. Pursuit-evasion experiment showing pursuer and evader trajectory visualization (top) and corresponding lab setup (bottom).

VI. CONCLUSION

We presented a novel approach combining energy-based diffusion models with artificial potential fields for motion planning. By integrating local potential fields with learned diffusion models, our method effectively handles both static and

dynamic obstacles while maintaining computational efficiency. The approach demonstrates superior performance in static environments with up to 100% success rate and significantly improves pursuit-evasion scenarios, achieving 84% success rate. These results suggest that combining learned models with classical potential fields can enhance the robustness and adaptability of motion planning systems. Building on these promising results, future work will continue to improve the scalability of the pursuit-evasion scenario as the number of obstacles increases. Furthermore, we plan to investigate strategies for learning the APF parameters to handle increasingly complex environments.

REFERENCES

- [1] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.
- [2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [4] L. Petrović, "Motion planning in high-dimensional spaces," *arXiv preprint arXiv:1806.07457*, 2018.
- [5] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *Int. J. of robotics research*, vol. 32, no. 9–10, pp. 1164–1193, 2013.
- [6] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [7] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *Int. J. of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.
- [8] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 ICRA*, 2011, pp. 4569–4574.
- [9] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.
- [10] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021.
- [11] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.
- [12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.
- [13] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *ICRA*, 2019, pp. 2118–2124.
- [14] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," in *CoRL*, 2021, pp. 355–364.
- [15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE CVPR*, 2017, pp. 652–660.
- [16] J. Carvalho, A. T. Le, M. Baiert, 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*. IEEE, 2023, pp. 1916–1923.
- [17] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du, "Potential based diffusion motion planning," *arXiv preprint arXiv:2407.06169*, 2024.
- [18] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.
- [19] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *ICRA*, 2011, pp. 5628–5635.
- [20] L. Zhai, C. Liu, X. Zhang, and C. Wang, "Local trajectory planning for obstacle avoidance of unmanned tracked vehicles based on artificial potential field method," *IEEE Access*, vol. 12, pp. 19 665–19 681, 2024.
- [21] H. Tang, W. Zhang, M. Sun, B. Lin, and Z. Hu, "A pe game with one superior hunter and multi-pursuer against an evader," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 5124–5130.
- [22] F. Hart and O. Okhrin, "Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk," *Neurocomputing*, vol. 568, p. 127097, 2024.
- [23] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *IROS*, 2021, pp. 6456–6463.
- [24] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Transactions on Games*, vol. 10, no. 4, pp. 400–412, 2018.
- [25] R. Han, S. Chen, and Q. Hao, "Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning," in *ICRA*, 2020, pp. 448–454.
- [26] J. Zhu, W. Zou, and Z. Zhu, "Learning evasion strategy in pursuit-evasion by deep q-network," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 67–72.
- [27] X. Qu, W. Gan, D. Song, and L. Zhou, "Pursuit-evasion game strategy of usv based on deep reinforcement learning in complex multi-obstacle environment," *Ocean Engineering*, vol. 273, p. 114016, 2023.
- [28] R. Zhang, Q. Zong, X. Zhang, L. Dou, and B. Tian, "Game of drones: Multi-uav pursuit-evasion game with online motion planning by deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7900–7909, 2022.
- [29] R. B. Grando, R. Steinmetz, V. A. Kich, A. H. Kolling, P. M. Furik, J. C. de Jesus, B. V. Guterres, D. T. Gamarra, R. S. Guerra, and L. Drews, "Improving generalization in aerial and terrestrial mobile robots control through delayed policy learning," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 1028–1033.
- [30] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *ICML*, 2015, pp. 2256–2265.
- [31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [32] Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl, "Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc," in *ICML*, 2023, pp. 8489–8510.
- [33] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.
- [34] X. Yun and K.-C. Tan, "A wall-following method for escaping local minima in potential field based motion planning," in *ICAR'97*, 1997, pp. 421–426.
- [35] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Sixth international conference on intelligent systems design and applications*, vol. 2, 2006, pp. 622–627.
- [36] V. Gazi, B. Fidan, Y. S. Hanay, and İ. Köksal, "Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 2, pp. 149–168, 2007.
- [37] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *ICML*, 2019, pp. 3744–3753.
- [38] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.
- [39] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000, pp. 995–1001.
- [40] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *Int. J. of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.
- [41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*, 2018, pp. 1861–1870.
- [42] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.