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

Hierarchical Motion Planning Adaptation via Guided Diffusion

Woosung Kim and Nicola Bezzo

***Abstract*— Mobile robots deployed for persistent operations in partially known environments need to be able to recover and adapt against unforeseen changes in dynamics, e.g., due to failures, or external disturbances. This paper presents a novel hierarchical framework capable of zero-shot adaptation to environmental and dynamic changes. At the high level, an abstract planner generates a collision-free global path, adapting to degraded mobility by inflating a dynamic safety buffer around obstacles to ensure the route remains navigable. At the low level, a concrete planner employs a conditional Denoising Diffusion Probabilistic Model (DDPM) to refine the abstract path into a smooth, executable trajectory. The key to our approach is conditioning the diffusion model's generation process on the robot's online-estimated dynamic limits. Our framework's effectiveness and robustness are validated in both complex simulations and real-world hardware experiments, demonstrating its ability to ensure mission success under unstructured and unexpected fault situations.**

I. Introduction

Despite a growing interest in robotic technologies, their use to assist, augment, or substitute human capabilities still poses several challenges. In fact, robotic applications are typically built with specific systems and environments in mind and exhaustive pre-training to account for all possible situations that can arise in the real world is unfeasible, computationally prohibitive, and may still result in incomplete knowledge since unforeseen situations may still arise. The goal of this paper is to *quickly and autonomously adapt motion planning methods for a robot which may undergo changes in dynamics and operate between different environments* hence empowering the robot with persistent and resilient autonomy capabilities.

Existing control and machine learning approaches that deal with the problem of adaptation at run-time [1], [2], [3] rely on highly precise representations of a system and this is typically the default assumption in the field. However, it is challenging and often unfeasible to create a perfect model replica of a system, especially when the system may change in unpredictable ways over time due to intermittent or slowly progressing failures, platform aging, or changing environmental conditions. A representative example is the NASA Mars rovers Spirit and Opportunity, which experienced hardware degradation during long-duration deployment, including mobility-related failures that required adapted operational strategies [4]. Even domain randomization techniques [5], [3] are not sufficient to enable such adaptation since they rely on strong assumptions on the system model structure and have the problem that the solution is not general enough to be repurposed on different environment settings.

Humans, on the other hand, don't use complex or precise models to adapt their motion in front of new environment conditions or changes in dynamics [6]. We typically create a rough plan and leverage and fine-tune our motor skills as we follow such plan, adapting both our plan and actions as needed; for example we may decide to re-route based on the conditions detected in the current path or change our motion by slowing down, jumping, sliding, running, while adapting only specific subtasks without the need to recompute an entire trajectory.

Woosung Kim and Nicola Bezzo are with the Autonomous Mobile Robots Lab (AMR Lab) and the Departments of Electrical and Computer Engineering and Systems Engineering, University of Virginia, Charlottesville, VA 22903, USA {mzg8qg, nbezzo}@virginia.edu

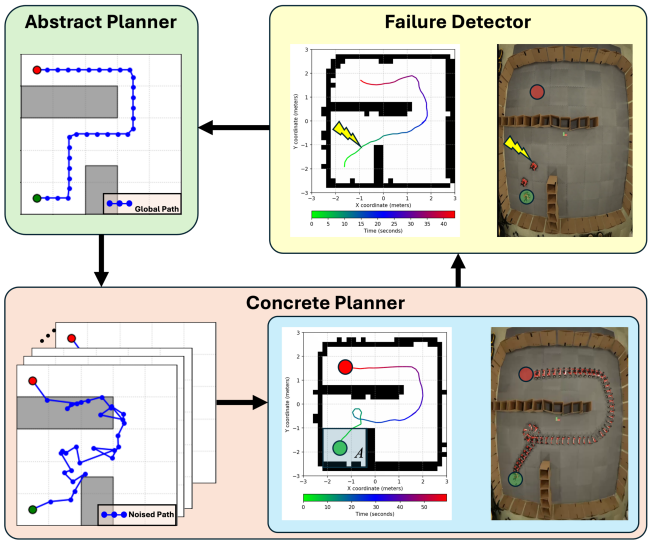


Fig. 1. Overview of the proposed hierarchical planner. An abstract Planner generates a coarse global path on a map. A Failure Detector tracks fault situations and updates plant dynamic limit. A concrete planner conditions on the updated limits to produce a smooth, executable trajectory.

Following these considerations, and differently from current state of the art that focuses on highly precise representations of a system, in this paper we enable fast online system adaptation through a hierarchical approach that adapts a *low resolution/high abstraction planner* coupled with a *high resolution/low abstraction control level and real world* representation.

Fig. 1 illustrates the proposed hierarchy which has two core responsibilities: a high-level abstract planner ensures safety by generating a coarse, collision-free global path, while a low-level concrete planner ensures feasibility by refining the coarse path into a smooth, executable trajectory. This low-level planner is powered by a conditional Denoising Diffusion Probabilistic Model (DDPM) [7]. The advantage of our approach lies in its ability to perform zero-shot adaptation at runtime, addressing both newly detected obstacles and ensuring trajectory feasibility under a dynamic shift. Instead of complex online retraining or system identification, our system adapts by learning the new dynamic constraints online and modifying the conditions supplied to the generative model.

Our contribution is twofold: **i**) We propose a hierarchical planning framework that enables online, zero-shot adaptation to unexpected changes in the robot's dynamic constraints. At the high level, a global planner maintains a collision-free

route; at the low level, a conditional diffusion model adjusts each local segment into a dynamically feasible path under changing dynamic capabilities. This is achieved by conditioning a diffusion model, without fine-tuning or retraining. **ii**) We integrate this with reactive obstacle avoidance and abstract-path adherence by guided denoising with a local Artificial Potential Field (APF) composed of the abstract path and adaptive Euclidean Signed Distance Fields (ESDF) [8] inflation. This yields trajectories that remain within the robot's updated feasibility set, avoid newly detected obstacles, and stay aligned with the abstract path.

We demonstrate the effectiveness of our approach through simulations and experiments in realistic and complex environments, showing that despite environmental and dynamic changes, the robot is able to safely reach its goal.

II. Related Work

Robot motion planning has been explored through various algorithmic paradigms. Our work draws upon principles from traditional, optimization-based, and modern learning-based approaches.

Traditional path planners like A* are foundational for finding globally optimal, collision-free geometric paths but do not account for robot dynamics, producing waypoints that are often unsmooth and infeasible without significant post-processing [9]. To bridge this gap, optimization-based methods such as Model Predictive Path Integral (MPPI) control generate short-horizon, dynamically feasible trajectories by sampling and weighting control sequences [10]. MPPI is effective for reactive local navigation and can be combined with a higher-level planner that provides intermediate waypoints or global guidance. However, since the optimization itself is still performed over a limited horizon, its performance in complex environments remains strongly dependent on the quality of that guidance, and purely local formulations can still be susceptible to local minima.

Reinforcement Learning (RL) offers a model-free alternative for developing complex control policies [11]. However, RL agents typically require extensive training in simulation and generate actions sequentially, which can lead to compounding errors. More importantly, policies learned via RL are optimized for a specific set of dynamics and lack the flexibility to adapt to sudden hardware failures without costly retraining.

Generative approaches, particularly Denoising Diffusion Probabilistic Models (DDPMs), have recently emerged as a compelling solution for motion planning [12], [13]. By generating entire trajectories in a single inference pass, they inherently avoid compounding errors. Their key advantage lies in their conditionality; the generation process can be guided by various factors, including environmental maps and, as we explore, the robot's own dynamic constraints, enabling powerful zero-shot adaptation.

Beyond initial path generation, a critical area of research addresses how planners can adapt to sudden hardware failures or changes in dynamics at runtime. One strategy involves geometrically transforming the control space; for instance, by using Schwarz-Christoffel Mapping (SCM) to create a map that translates control inputs from an expert system to a learner with degraded capabilities, compensating for model discrepancies [14]. Another promising direction is meta-learning, which can train a model to quickly adapt to a new fault with only a few data points gathered at runtime, enabling proactive and safe replanning [15]. While powerful, these methods either focus on mapping control inputs rather than generating novel trajectories or still require a brief fine-tuning phase upon encountering a new fault.

Our work builds on the motivation of these adaptive frameworks but proposes a distinct, generative solution. Instead of geometrically mapping control inputs or fine-tuning a predictive model, our diffusion-based planner achieves zero-shot adaptation. By conditioning the trajectory generation process on the robot's newly learned dynamic limits, it instantly produces a dynamically feasible and safe path without an intermediate adaptation step, offering a complementary approach to the challenge of runtime adaptation. To the best of our knowledge, this is the first work to leverage diffusion models for robot adaptation against concurrent changes in dynamics, the environment, and unexpected fault situations.

III. Problem Formulation

This work addresses the challenge of creating a motion planner that can reactively handle both external environmental changes and internal dynamic faults. We consider a robot with a planar pose $\boldsymbol{x} = (x, y, \psi) \in \mathrm{SE}(2)$ operating in a workspace $\mathcal{W} \subset \mathbb{R}^2$.

In practice, many common ground-robot faults (e.g., tire deflation, unilateral motor derating/overheating, brake drag, tread damage) manifest primarily as loss of *turning* authority rather than complete loss of mobility [16]. These effects are typically asymmetric, affecting left vs right turning differently, so the effective yaw-rate ($\omega(t)$) limit become unequal and time-varying. By contrast, moderate reductions in forward speed often preserve reachability (the robot can follow a valid path but arrives later), whereas sudden degradation of left/right turning capacity can invalidate the curvature required by the current plan. We therefore model dynamic faults as asymmetric, time-varying yaw-rate limits, represented as dynamic constraints, $\mathcal{D}_t = \{\omega_L(t), \omega_R(t)\}$, representing the maximum achievable left and right angular velocities, which can change unpredictably over time. This choice directly motivates a planner that remains feasible under the constraint $\omega(t) \in [-\omega_R(t), \omega_L(t)]$.

Given these time-varying constraints, we seek to find a planning policy Π that generates a trajectory $\boldsymbol{\tau}(t) : [0, T] \to \mathrm{SE}(2)$ to steer the robot from a starting position $\boldsymbol{p}_\mathrm{s}$ to a goal position $\boldsymbol{p}_\mathrm{g}$. The workspace also contains a set of obstacles, $\mathcal{O}_t$, that are discovered and mapped online. A valid trajectory must be simultaneously safe (i.e., avoid collisions) with respect to the environment and feasible under the robot's current dynamics ensuring liveness (i.e., eventually reaching the desired goal). Hence, we want to satisfy the following safety conditions for all $t \in [0, T]$:

$$\mathcal{B}(\boldsymbol{\tau}(t)) \cap \mathcal{O}_t = \emptyset \quad \wedge \quad \omega(t) \in [-\omega_R(t), \omega_L(t)], \tag{1}$$

where $\mathcal{B}(\boldsymbol{\tau}(t))$ is the robot's footprint. We also enforce liveness via a constraint:

$$\|(x(T), y(T)) - \boldsymbol{p}_\mathrm{g}\|_2 \leq \varepsilon. \tag{2}$$

where $||\cdot||_2$ is the L2 norm.

The core challenge is that both the environmental constraints ($\mathcal{O}_t$) and the dynamic constraints ($\mathcal{D}_t$) can change abruptly during a mission, invalidating the current plan. This

leads to our central research question: *How can a robot autonomously respond to a significant change in its dynamics and, in response, immediately generate a new trajectory $\boldsymbol{\tau}'$ that satisfies* (1) *and* (2) *under its newly constrained dynamics?* Our approach leverages a conditional diffusion model to solve this unified problem without requiring any online retraining.

IV. Approach

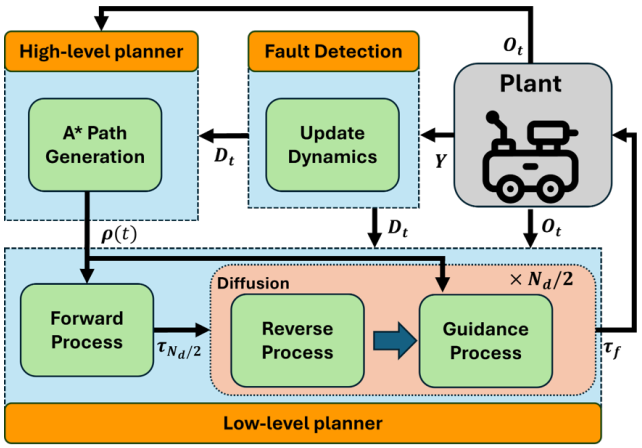


Fig. 2. Overview of the proposed hierarchical planner. A high-level A* computes a global path; a fault-detection updates dynamic limits when deviations are detected; and a low-level diffusion module, guided by the local map, refines the path into a smooth, feasible, obstacle-avoiding trajectory.

We propose a hierarchical planning framework designed to be reactive to both external environmental changes and internal dynamic faults. As illustrated in Fig. 2, the architecture comprises two layers: a high-level planner that uses onboard sensor observations Y to produce a coarse global path, and a low-level diffusion-based trajectory generator that refines this path under the current dynamic limits $\mathcal{D}_t$ into a smooth, dynamically feasible, obstacle-avoiding trajectory that remains aligned with the abstract path. When sustained tracking deviation is detected, the fault-detection module updates $\mathcal{D}_t$ and triggers a safe replan, yielding a new feasible trajectory. We use a diffusion model that generates a feasible concrete trajectory for output by iteratively denoising a noisy sample over N_{d} steps. The proposed diffusion generative model has two main benefits: i) it is highly receptive to diverse conditioning, allowing us to steer trajectory generation in real time using not only the goal position (g_x, g_y) but also time-varying dynamic limits $\mathcal{D}_t$ [17]; and ii) by producing entire trajectory sequences jointly, its multi-step generation is well-suited for long horizons, reducing cumulative error relative to single-step predictive models where errors compound.

The core of our framework is the ability to adapt its generation process to the robot's changing dynamics, ensuring robust navigation in unstructured environments, even when experiencing kinematic faults. The following subsections detail each component of this framework.

A. High-level abstract planning.

The high-level planner is responsible for generating a global, collision-free path $\boldsymbol{\rho}(t) = \{\boldsymbol{p}(t), \ldots, \boldsymbol{p}_g\}$ from the robot's current position $\boldsymbol{p}(t) = (x(t), y(t))$ to a goal $\boldsymbol{p}_g = (x_g, y_g)$ in a partially observed environment. While any high level planning method can be leveraged, in this work, we employ the A* algorithm for its computational efficiency. Using onboard sensors and a SLAM algorithm, the robot continuously updates its map of the environment $\mathcal{O}(t)$. The framework constantly validates $\boldsymbol{\rho}$ against this updated map. If $\boldsymbol{\rho}$ becomes obstructed by newly detected obstacles, the high-level planner re-generates a new collision-free global path.

The geometric plan output of the A* is leveraged by the following low-level diffusion-based planner to guide the creation of dynamically feasible trajectories.

B. Low-level concrete planning.

To convert the global path $\boldsymbol{\rho}(t)$ into an executable motion plan, a local window $L(t) \subset \mathcal{W}$ is considered, centered around the current robot state $\boldsymbol{x}(t)$. This window limits planning to a tractable horizon while still allowing the robot to consider nearby obstacles during planning, as illustrated in Fig. 3. If the global goal $\boldsymbol{p}_g$ lies outside the window, the local goal $\boldsymbol{p}_g^L(t)$ is selected as the furthest point along $\boldsymbol{\rho}$ that remains within $L(t)$:

$$\boldsymbol{p}_g^L(t) = \boldsymbol{p}_k \quad \text{s.t.} \quad k = \max\{i \mid \boldsymbol{p}_i \in L(t) \wedge \boldsymbol{p}_i \in \boldsymbol{\rho}(t)\}. \tag{3}$$

We employ a conditional diffusion model [18] to generate a trajectory, $\boldsymbol{\tau}^*(h;t) = \boldsymbol{p}_h,\ h = 1, \ldots, H$, consisting of a fixed-length sequence of H waypoints that respect the input limits $\mathcal{D}_t$ while avoiding obstacles $\mathcal{O}_t$. At each diffusion step

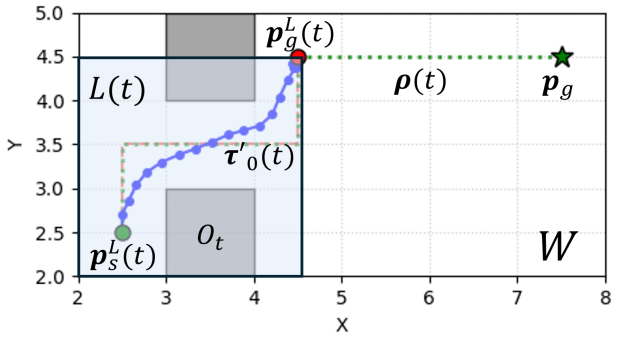


Fig. 3. Setup of the low-level diffusion planner with local window

n, the trajectory is represented by a diffusion embedding $\boldsymbol{\tau}_n(h;t)$:

$$\boldsymbol{\tau}_n(h;t) = \boldsymbol{z}_n(h), \quad \boldsymbol{z}_n(h) = [\Delta \boldsymbol{p}_{n,h}, \boldsymbol{c}_{n,h}], \tag{4}$$

where $\boldsymbol{z}_n(\cdot) \in \mathbb{R}^d$, $\Delta \boldsymbol{p}_{n,h} = \boldsymbol{p}_{n,h} - \boldsymbol{p}_{n,h-1}$ (with $\Delta \boldsymbol{p}_{n,0} = \boldsymbol{0}$), and $\boldsymbol{c}_n(h)$ is a condition vector for waypoint h that encodes relevant context, including local start position $\boldsymbol{p}_s^L$, local goal position $\boldsymbol{p}_g^L(t)$, the robot's heading $\psi(t)$, and current dynamic limits $\omega_L(t), \omega_R(t)$:

$$\boldsymbol{c}_{n,h} = [\boldsymbol{p}_{n,h}, \boldsymbol{p}_s^L, \boldsymbol{p}_g^L, \boldsymbol{e}(\omega_L), \boldsymbol{e}(\omega_R), \boldsymbol{e}(\psi)], \tag{5}$$

where $\boldsymbol{e}(x) = [cos(x), sin(x)]$ is a unit orientation vector, and the time dependence is dropped for brevity.

The diffusion process begins by converting the abstract global path $\boldsymbol{\rho}$ into an initial trajectory $\boldsymbol{\tau}_0$ over H waypoints. During the forward process, Gaussian noise is incrementally added to $\boldsymbol{\tau}_0$ producing $\boldsymbol{\tau}_n$ for $n = 1, \ldots, N_d/2$. Using only half the steps ensures that $\boldsymbol{\tau}_{N_d/2}$ still retains the global geometry of $\boldsymbol{\tau}_0$, which constraints the generative diversity of the model so that trajectories remain aligned with the path while reducing the number of denoising steps required.

For N_d/k when k is smaller, the trajectories were found to have insufficient flexibility to adjust, whereas for large k, the trajectories were indistinguishable from pure noise taking a longer time to compute the output. We therefore selected $k = 2$ to balance flexibility and efficiency.

This noisy trajectory is then passed to the **Reverse Process**, which iteratively denoises it, yielding $\boldsymbol{\tau}'_n$ at each step $n = N_d/2, \ldots, 0$. To enforce collision avoidance and alignment with the global path $\boldsymbol{\rho}$, **Guidance** is applied at each reverse step. The final trajectory $\boldsymbol{\tau}^*(t) = \boldsymbol{\tau}'_0$ respects dynamic limits $\mathcal{D}_t$ and moves from $\boldsymbol{p}_s$ to $\boldsymbol{p}_g$. The remainder of this section goes into detail about each of the three key diffusion stages.

1) Forward process. Starting from $\boldsymbol{\tau}_0$, a fixed Markov noising kernel q is leveraged to progressively add Gaussian noise over $N_d/2$ steps, following a variance schedule $\{\beta_n\}_{n=1}^{N_\mathrm{d}/2}$. The forward process begins by encoding $\boldsymbol{\rho}$ into the initial trajectory $\boldsymbol{\tau}_0$ for forward diffusion. At each step $n = 1, \ldots, N_d/2$, the noisy trajectory $\boldsymbol{\tau}_n$ is generated as:

$$q(\boldsymbol{\tau}_n|\boldsymbol{\tau}_{n-1}) = \mathcal{N}(\boldsymbol{\tau}_n; \sqrt{1-\beta_n}\boldsymbol{\tau}_{n-1}, \beta_n \boldsymbol{I}), \tag{6}$$

where $\boldsymbol{I} \in \mathbb{R}^{H\cdot d \times H\cdot d}$ is the identity matrix. Exploiting a property of the kernel q, $\boldsymbol{\tau}_n$ can be directly sampled from $\boldsymbol{\tau}_0$ without iterating through intermediate steps:

$$q(\boldsymbol{\tau}_n|\boldsymbol{\tau}_0) = \mathcal{N}(\boldsymbol{\tau}_n; \sqrt{\bar{\alpha}_n}\boldsymbol{\tau}_0, (1-\bar{\alpha}_n)\boldsymbol{I}), \quad \bar{\alpha}_n = \prod_{i=1}^{n}(1-\beta_i). \tag{7}$$

Using this direct sampling property, noise is added to $\boldsymbol{\tau}_0$ up to the noised trajectory $\boldsymbol{\tau}_{N_\mathrm{d}/2}$ before running the reverse process.

2) Reverse process. Given the noised trajectory $\boldsymbol{\tau}_{N_d/2}$, the reverse process uses a learned denoising model $\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_n, n, \boldsymbol{c}_n)$ to iteratively refine it into a dynamically feasible and smooth trajectory $\boldsymbol{\tau}'_0$. The denoising model is implemented as a Temporal U-net with attention [13], designed for 1-D sequential data such as trajectories. To enforce task-specific constraints, including start and goal positions, the network is conditioned using $\boldsymbol{c}_{n,h}$ from (5). At each residual block i of the U-net, the full conditioning vector $\boldsymbol{c}_n$ is fed into two Multi Layer Perceptrons (MLPs) that generate the FiLM parameters $\boldsymbol{\gamma}_i(\boldsymbol{c}_n)$ and $\boldsymbol{\beta}_i(\boldsymbol{c}_n)$. Together, these parameters apply an affine transformation to the intermediate feature maps $\boldsymbol{F}_i$ as:

$$\mathrm{FiLM}(\boldsymbol{F}_i|\boldsymbol{c}_n) = \boldsymbol{\gamma}_i(\boldsymbol{c}_n) \odot \boldsymbol{F}_i + \boldsymbol{\beta}_i(\boldsymbol{c}_n), \tag{8}$$

where $\odot$ denotes element-wise multiplication.

The denoising neural network, $\boldsymbol{\epsilon}_\theta$, is trained for the reverse process by predicting the added noise $\epsilon \sim \mathcal{N}(0, I)$, conditioned on the noisy trajectory $\boldsymbol{\tau}_n$, and context $\boldsymbol{c}_n$ by minimizing the loss $\mathcal{L}$:

$$\mathcal{L}(\theta) = \mathbb{E}_{n,\boldsymbol{\tau}_0,\boldsymbol{\epsilon},\boldsymbol{c}_n}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_n}\boldsymbol{\tau}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, n, \boldsymbol{c}_n)||^2\right]. \tag{9}$$

Once trained, $\boldsymbol{\epsilon}_\theta$ is used to parameterize the reverse process $p_\theta(\boldsymbol{\tau}_{n-1}|\boldsymbol{\tau}_n)$, where p_θ is the learned probability distribution for a denoising step, and θ is the learned network parameter. Starting from $\boldsymbol{\tau}_{N_\mathrm{d}/2}$, the model iteratively samples from p_θ for $n = N_\mathrm{d}/2, \ldots, 0$, progressively denoising the trajectory. At each step n, $\boldsymbol{\tau}'_n$ is progressively denoised, however, the reverse process does not account for obstacles $\mathcal{O}_t$. To ensure safety, each $\boldsymbol{\tau}'_n$ is processed through the guidance step presented next. After completing all $N_d/2$ reverse and guidance steps, a safe trajectory $\boldsymbol{\tau}^*(t)$ is obtained that respects the start and goal states as well as dynamics $\mathcal{D}_t$.

3) Guidance Process. In order to deal with collision avoidance with obstacles $\mathcal{O}(t) \in L(t)$, a guidance mechanism is incorporated at each denoising step. This guidance actively adjusts each waypoint to avoid obstacles and remain aligned with the global path $\boldsymbol{\rho}$, using APFs within the local planning window $L(t)$ [19].

The repulsive field generates a gradient based on the distance to the nearest obstacle, while the attractive field produces a gradient based on the distance to the global path, both evaluated at each waypoint $\boldsymbol{p}'_h \in \boldsymbol{\tau}'_n(h),\ h = 1, \ldots, H$.

The repulsive force $\boldsymbol{F}_\mathrm{rep}$ pushes the waypoint away from obstacles.

$$\boldsymbol{F}_\mathrm{rep}(\boldsymbol{p}'_h) = \eta_\mathrm{rep}\left(\frac{d_\mathrm{s} - ||\boldsymbol{v_r}(\boldsymbol{p}'_h)||}{d_\mathrm{s}}\right)^2 \frac{\boldsymbol{v_r}(\boldsymbol{p}'_h)}{||\boldsymbol{v_r}(\boldsymbol{p}'_h)||}, \tag{10}$$

where η_rep is the guidance strength parameter for repulsion, $\boldsymbol{p}_\mathrm{obs}(\boldsymbol{p}'_h)$ is the closest point on an obstacle to the waypoint. The force is directed along the repulsive vector $\boldsymbol{v_r}(\boldsymbol{p}'_h) = \boldsymbol{p}'_h - \boldsymbol{p}_\mathrm{obs}(\boldsymbol{p}'_h)$. $\boldsymbol{F}_\mathrm{rep}$ is only applied when the waypoint $\boldsymbol{p}'_h$ is within a predefined safety distance d_s to an obstacle, and its magnitude increases quadratically as it gets closer, providing a strong but smooth push.

The attractive force $\boldsymbol{F}_\mathrm{att}$ keeps $\boldsymbol{\tau}'_n$ aligned with the global path $\boldsymbol{\rho}$:

$$\boldsymbol{F}_\mathrm{att}(\boldsymbol{p}'_h) = \eta_\mathrm{att}\frac{\boldsymbol{p}^0_n - \boldsymbol{p}'_h}{||\boldsymbol{p}^0_n - \boldsymbol{p}'_h||}, \tag{11}$$

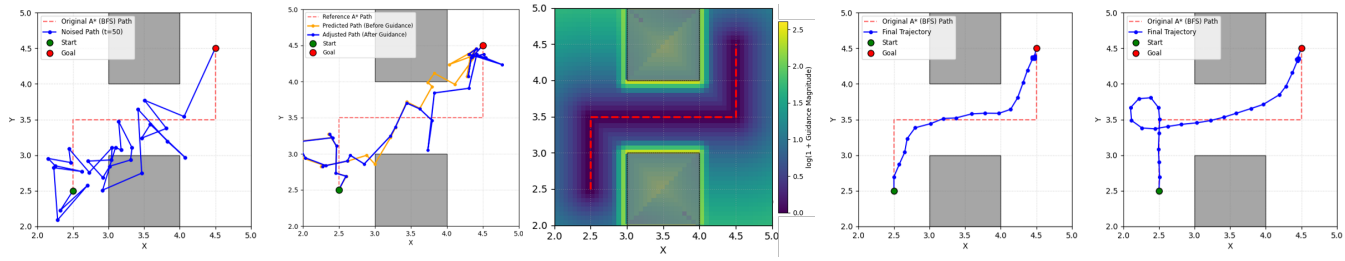
where η_att is the coefficient of attractive guidance and $\boldsymbol{p}^0_h \in \boldsymbol{\tau}_0(h)$ is the hth waypoint in the initial trajectory obtained from ρ. This force is directed along the vector from the waypoint $\boldsymbol{x}_h$ to its corresponding target waypoint on the global reference path.

The total guidance correction for a waypoint is the sum of these forces:

$$\boldsymbol{g}(\boldsymbol{p}'_h) = \boldsymbol{F}_\mathrm{rep}(\boldsymbol{p}'_h) + \boldsymbol{F}_\mathrm{att}(\boldsymbol{p}'_h) \tag{12}$$

This guidance is integrated into the reverse diffusion process at each timestep n [20]. The guidance process is as follows:

1) Noise Prediction: The network $\boldsymbol{\epsilon}_\theta$ predicts the noise from the current noisy trajectory of full states $\boldsymbol{\tau}_n$: $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_n, n, \boldsymbol{c}_n)$.
2) $\boldsymbol{\tau}_0$ Estimation: An estimate of the final output trajectory at diffusion step n, $\hat{\boldsymbol{\tau}}_n$, is computed using the predicted noise $\hat{\boldsymbol{\epsilon}}$:
$$\hat{\boldsymbol{\tau}}_n = \frac{1}{\sqrt{\bar{\alpha}_n}}(\boldsymbol{\tau}_n - \sqrt{1-\bar{\alpha}_n}\hat{\boldsymbol{\epsilon}}). \tag{13}$$
3) Guidance Correction Application: Guidance is applied to estimated waypoint $\hat{\boldsymbol{p}}_h$ from $\hat{\boldsymbol{\tau}}_n$:
$$\hat{\boldsymbol{p}}^*_h = \hat{\boldsymbol{p}}_h + \boldsymbol{g}(\hat{\boldsymbol{p}}_h). \tag{14}$$
The guided trajectory, $\hat{\boldsymbol{\tau}}^*_n$, is then constructed with updated $\hat{\boldsymbol{p}}^*_h$.
4) Next Step Sampling: The guided trajectory $\hat{\boldsymbol{\tau}}^*_n$ is used to sample $\boldsymbol{\tau}'_{n-1}$ by computing the linear–Gaussian



(a) (b) (c) (d) (e)

Fig. 4. Low-level planning adaptation sequence: (a) forward process from A* path to noised path, (b) intermediate reverse process step with (c) guidance heatmap used in the guidance process, (d) final output for nominal case, and (e) final output for fault case with $\omega_R(t) = 0$.

posterior distribution $q(\boldsymbol{\tau}'_{n-1}|\boldsymbol{\tau}'_n, \hat{\boldsymbol{\tau}}^*_n)$, leveraging the forward kernel (6). This ensures the guidance correction influences the remainder of the denoising process.

This method allows us to inject environmental constraints $\mathcal{O}(t)$ into the generative process. The final executable trajectory is formed by the sequence of corrected spatial coordinates $\boldsymbol{\tau}^*(h)$, producing a path by iteratively pushing waypoints away from nearby obstacles and pulling them back toward the global path ensuring feasibility, safety and alignment with the higher-level abstract trajectory.

As shown in Fig. 4, the low-level planner generates a noisy trajectory $(\boldsymbol{\tau}_{N_\mathrm{d}/2})$ from $\boldsymbol{\rho}(t)$ using the forward process (Fig. 4(a)). Considering the robot's dynamics, the planner runs the reverse process iteratively with guidance (Figs. 4(b)(c)) to generate an executable path $(\boldsymbol{\tau}'_0)$ when the robot is in nominal state (Fig. 4(d)) or if the robot's dynamics are impaired. For instance, when its turning ability fails ($\omega_L(t)$=0), it generates a feasible alternative using only the remaining available maneuvers (Fig. 4(e)).

C. Training.

The performance of our conditional diffusion model is fundamentally dependent on the quality and diversity of its training data. We generate a comprehensive dataset of expert trajectories using a Hybrid A* planner, which extends the traditional A* search to a continuous state space of (x, y, ψ) to produce smooth paths that respect non-holonomic constraints [9]. Using this planner, we generate trajectories with a horizon length H in environments with randomly placed obstacles and a wide variety of dynamic turning limits (ω_L, ω_R) to ensure the dataset is rich with challenging scenarios. We train the Temporal U-Net $\boldsymbol{\epsilon}_\theta$ on this expert dataset using Adam (batch size 128) with a diffusion step of $N_\mathrm{d} = 100$ steps.

D. Receding Horizon Predictive Replanning

To achieve smooth, continuous navigation, our framework employs a predictive receding horizon strategy. A new trajectory is generated after the robot has followed the current plan for a certain number of receding horizon steps, $h_r \leq H$. Instead of executing a full local trajectory, the system proactively generates the next trajectory $\boldsymbol{\tau}(t')$. This process hinges on a designated receding horizon waypoint, (x_{h_r}, y_{h_r}), chosen partway through the current trajectory. This waypoint serves as the starting point for $\boldsymbol{\tau}(t')$, which is generated in a background thread before the robot reaches it.

A critical component for ensuring a seamless, kinematically continuous transition is the calculation of the initial heading for the new plan. This heading is derived from the direction of motion encoded in the current plan at the handover point, rather than from the robot's instantaneous pose. The heading can be calculated using the waypoint $\boldsymbol{x}_{h_r} = [x_{h_r}, y_{h_r}]^T$ and its successor $\boldsymbol{x}_{h_r+1}$. Once the robot reaches $\boldsymbol{x}_{h_r}$, the system discards the remainder of $\boldsymbol{\tau}(t)$ and seamlessly begins executing the pre-computed $\boldsymbol{\tau}(t')$. This rolling-horizon cycle of predictive replanning enables the robot to move continuously and reactively toward its goal.

E. Fault detection & safety buffer update.

As mentioned, the hierarchical planner can generate feasible and safe trajectories under changing dynamic situations on the fly. However, to rapidly adapt to a dynamic fault, the framework must first detect the anomaly and estimate the robot's new dynamic limits. This section details our three-stage process for this autonomous adaptation: 1) detecting a fault by monitoring trajectory tracking performance, 2) estimating the robot's dynamic limits and updating a corresponding safety buffer, and 3) triggering a safe replanning sequence.

1) Fault Detection via Cross-Track Error Monitoring. In order to detect faults, the robot continuously monitors its adherence to the planned trajectory at a uniform sampling period Δt by evaluating the cross-track error $d_\perp(t_k)$, defined as the normal distance from the robot's position $\boldsymbol{p}(t_k)$ to the nearest waypoint of the planned trajectory $\boldsymbol{\tau}^*$ [21]. This metric captures deviations from the intended trajectory that indicate a potential change in the robot's effective dynamics $\mathcal{D}_t$.

To filter out transient spikes, a sliding-window mean over the last N_{err} steps is computed, and a fault is triggered if this mean exceeds a threshold δ_f:

$$\frac{1}{N_\mathrm{err}} \sum_{k=1}^{N_{err}} d_\perp(t_k) > \delta_\mathrm{f} \tag{15}$$

2) Dynamic Re-estimation and Adaptive Safety Buffer. Once a dynamic fault is detected, the robot updates its angular velocity limits $\mathcal{D}'_t$ based on recent motion. Let the recent pose history be $\boldsymbol{P} = \{\boldsymbol{x}(t_k)\}_{k=0}^{N_{err}}$. The instantaneous angular velocity $\omega(k)$ for each step is approximated by the finite difference of the heading:

$$\hat{\omega}(t_k) = \frac{\psi(t_k) - \psi(t_{k-1})}{t_k - t_{k-1}}, \tag{16}$$

for $k = 1, \ldots, N_{err}$. The updated limits $\mathcal{D}'_t = (\omega'_L, \omega'_R)$ are then computed empirically from the observed angular velocities:

$$\begin{aligned} \omega'_L &= \max\{\hat{\omega}(t_k) \mid \hat{\omega}(t_k) > 0\} \\ \omega'_R &= |\min\{\hat{\omega}(t_k) \mid \hat{\omega}(t_k) < 0\}|. \end{aligned} \tag{17}$$

A severe degradation in turning ability, particularly on one side, requires wider maneuvers to change direction. For instance, a robot that can no longer turn left must execute a wide right-hand loop to achieve a leftward orientation, which demands more space. To account for this, we introduce an adaptive safety buffer, d_{s}, which is used to inflate obstacle boundaries in the map via an ESDF. This inflation effectively blocks narrow passages and forces the high-level planner to find routes through more open areas, accommodating the robot's impaired mobility.

In the nominal limit, the robot's ability to reorient locally with bounded curvature allows for a simple, fixed safety buffer α accounting for the robot's footprint $\mathcal{B}(\boldsymbol{\tau}(t))$. On the other hand, in the critically impaired limit, one turning direction is effectively unavailable; any changes in heading must be achieved by a same-direction loop using the degraded control authority.

$$d_{\mathrm{s}} = \alpha + \begin{cases} 0 & \text{if } \min(\omega_L, \omega_R) > \omega_{\mathrm{c}} \\ \frac{2v_t}{\max(\omega_L, \omega_R)} & \text{if } \min(\omega_L, \omega_R) \leq \omega_{\mathrm{c}}, \end{cases} \tag{18}$$

where ω_{c} is the critical turning-limit threshold, the minimum yaw rate required to execute planned curvatures. With the updated d_{safe} and $\mathcal{D}'_t$, the framework initiates a replan with newly calculated safety margins.

V. Simulations

Simulations were conducted to validate the proposed framework's efficiency and its robustness to sudden faults. Our approach was also compared against several baselines in complex maze environments.

A. Simulation Setup

Simulations were performed in the ROS-based MVSim [23] virtual simulator, using three maze environments from the D4RL benchmark [22]: U-Maze, Medium Maze, and Large Maze (depicted in Fig. 5), where designated start (green) and goal (red) positions define the navigation tasks. The simulated robot uses a unicycle model and is equipped with a 2D LiDAR sensor with 5m range. All simulations were executed on a laptop with an NVIDIA RTX 4090 GPU and an Intel i9-14900HX CPU.

B. Baseline Planners for Comparison

Our approach was compared against three baselines:

1) **MPPI with Global Planner (MPPI-Global):** A hierarchical planner using the same A* global planner with ESDF-based guidance as our approach for global planning, together with a Model Predictive Path Integral (MPPI) [24] controller for local, reactive trajectory optimization [10], where the controller is provided with the estimated dynamic limits D_t during simulation.
2) **Reactive MPPI (MPPI-Local):** A purely reactive MPPI controller that optimizes trajectories directly toward the final goal without a global path.
3) **PPO with Global Planner (PPO-Global):** A controller trained using Proximal Policy Optimization (PPO) [25] to follow the same pre-computed A* global path with ESDF-based guidance used by our approach. This baseline is trained under predefined nominal or fault conditions to represent a near-optimal policy for reaching the goal under those specific conditions.

C. Evaluation Scenarios and Metrics

All planners were evaluated across nine dynamic turn-rate profiles (1 nominal + 8 faulted) in three maze environments (U-Maze, Medium, Large). A trial is considered successful if the robot reaches the goal without collisions. For Medium and Large mazes, we assess three distinct goal positions each; for every *<environment, profile, goal>* configuration, we run three trials recording *success rate* (%) and *average and minimum completion times* (s) over the trials that reach the goal.

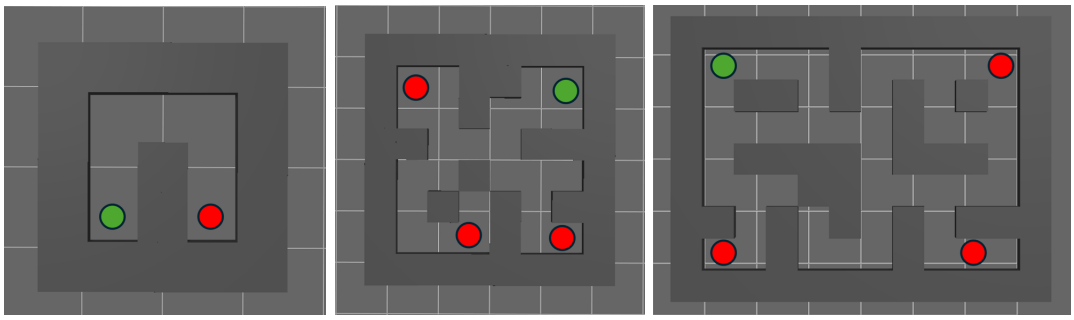
To rigorously probe robustness, we varied the robot's maximum left/right turn-rate limits to produce the following nine cases with faults never experienced before and not known prior to running the trials:

- **Nominal:** $\mathcal{D}_t = \{50,\ 50\}$ deg/s.
- **Partial fault (one side degraded):** $\mathcal{D}_t \in \{\{50,\ 20\},\ \{20,\ 50\},\ \{30,\ 20\},\ \{20,\ 30\}\}$ deg/s.
- **Complete fault (one side disabled):** $\mathcal{D}_t \in \{\{50,\ 0\},\ \{0,\ 50\},\ \{25,\ 0\},\ \{0,\ 25\}\}$ deg/s.

The complete results across all 9 cases $\times$ 3 environments $\times$ 3 goals for Medium/Large mazes with 3 trials each are summarized in Table I. The final trajectories for four representative cases are presented in Fig. 6 to highlight how our proposed hierarchical diffusion planner is able to adapt robot trajectory differently under different fault conditions. For example, under partial turning authority (Figs. 6(b)(d)) the robot performs less adjustments (i.e., less loops) in comparison to the complete loss of one side turning capabilities presented in Figs. 6(a)(c). Fig. 6(d) with $\mathcal{D}_t = [20, 30]$deg/s highlights also how the proposed generative approach is able to adjust trajectories to perform less rapid turns with wider loops when needed to respect the specific dynamic constraints. Note that the environment is unknown a priori as demonstrated by the path in Figs. 6(a)(b) in which the robot ends up first in the bottom right corner of the maze before finding its way to the goal.

Table I shows that across the full evaluation grid, our approach achieves 100% success in all environments and fault settings, while MPPI based methods degrade substantially under partial/complete turning-loss conditions. MPPI based methods show a lower average time compared to ours; this is because their successful trials cluster around shorter goal settings, while they fail longer, harder ones. MPPI-Local frequently gets trapped in local minima in larger mazes; MPPI-Global remains strong in nominal settings but fails more often as turning constraints tighten. PPO-Global achieves near-optimal times because it follows a pre-computed global path under the specific fault settings for which it was trained, but it does not generalize to diverse environments or unseen faults.

Crucially, our approach's success stems from its zero-shot robustness. The diffusion-based low-level planner requires approximately 3 s to generate a trajectory at inference time on the evaluation platform. It utilizes a single conditional diffusion model across all environments and fault cases without any retraining or fine-tuning. This inherent adaptability allows our model to succeed by actively exploring the maze and adjusting planning policy considering dynamically feasible maneuvers, such as reorientation loops, to overcome tight physical limits where baselines often fail. This capacity for generalization underscores not only the model's



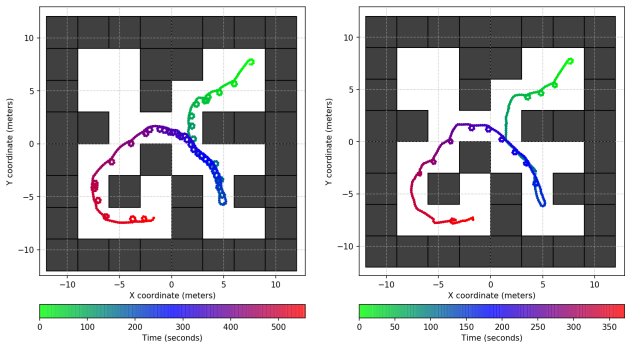
(a) (b) (c)

Fig. 5. The three simulation environments used for evaluation from the D4RL benchmark [22]: (a) U-Maze, (b) Medium Maze, and (c) Large Maze. The green and red circles indicate the start and goal positions for each trial.

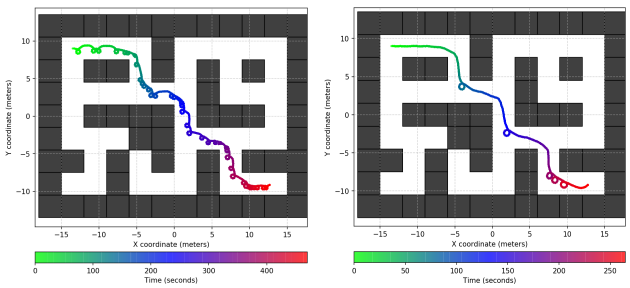
TABLE I

COMPREHENSIVE NAVIGATION PERFORMANCE ACROSS NOMINAL AND FAULT CONDITIONS

Environment	Planner	Nominal			Partial loss			Complete loss		
		S (%)	Avg T (s)	Best T (s)	S (%)	Avg T (s)	Best T (s)	S (%)	Avg T (s)	Best T (s)
U-Maze	Our Approach	100	152.54	149.20	100	175.39	149.36	100	258.39	219.30
	MPPI-Global	100	156.69	150.22	58.33	279.78	148.63	8.33	168.10	168.10
	MPPI-Local	33.33	101.20	101.20	16.67	107.00	104.70	8.33	107.52	107.52
	PPO-Global	100	75.00	75.00	100	73.25	72.00	100	98.25	83.00
Medium Maze	Our Approach	100	248.51	178.22	100	302.46	173.67	100	430.19	223.32
	MPPI-Global	66.67	267.24	172.64	47.22	292.53	171.05	0.56	378.45	269.99
	MPPI-Local	11.11	121.33	121.33	13.89	132.12	110.23	0	-	-
	PPO-Global	100	112.00	72.00	100	125.90	87.00	100	133.56	101.00
Large Maze	Our Approach	100	248.57	162.39	100	259.90	174.22	100	458.40	235.45
	MPPI-Global	77.78	270.01	167.21	72.22	267.34	165.20	0.56	338.91	263.35
	MPPI-Local	11.11	129.54	129.54	0	-	-	0	-	-
	PPO-Global	100	133.33	94.00	100	153.7	103.00	100	200.44	124.00



(a) $\mathcal{D}_t = \{50, 0\}$deg/s (b) $\mathcal{D}_t = \{50, 20\}$deg/s



(c) $\mathcal{D}_t = \{0, 50\}$deg/s (d) $\mathcal{D}_t = \{20, 30\}$deg/s

Fig. 6. Simulation results illustrating trajectories under complete and partial loss of turning capabilities. Our method successfully generates adaptive maneuvers, respecting dynamic constraints $\mathcal{D}_t$ not known a priori.

robustness but also its practical applicability for diverse and unpredictable situations.

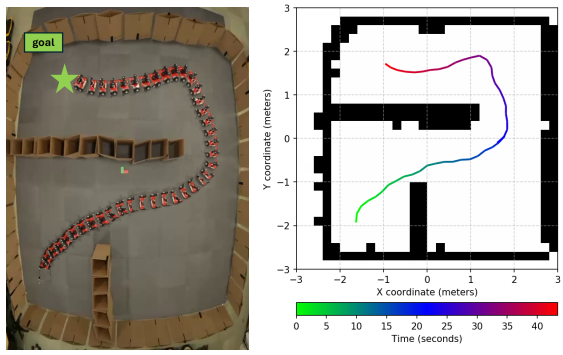
VI. PHYSICAL EXPERIMENTS

We validated the framework on a Husarion ROSbot2 in a maze environment (Fig. 7). A blue zone defined nominal operation; leaving this zone triggered loss of left or right turning capability. The robot mapped the environment online using onboard LiDAR and Gmapping SLAM [26]. Under nominal conditions, the planner successfully reached the goal (Fig. 7(a), 7(b)).

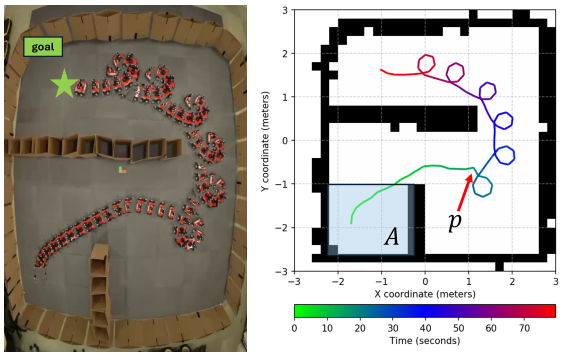
As illustrated in Figs. 7(c),(d), and Fig. 1, leaving the blue nominal zone triggers the fault. Under nominal operation, the dynamic constraints were $\mathcal{D}_t = \{\omega_L(t), \omega_R(t)\} = \{50, 50\}\,\mathrm{deg/s}$; when the fault was triggered, they switched to $\{50, 0\}$ or $\{0, 50\}\,\mathrm{deg/s}$ depending on whether right (Fig. 1) or left (Figs. 7(c),(d)) turning was disabled. In real hardware, when the robot deviated from the currently generated path beyond a small threshold, the planner adjusted its policy by conditioning on the estimated $\mathcal{D}_t$ and replanned accordingly. The point p marks the instant at which the policy adjustment occurs. Our approach successfully detected the fault and quickly adapted its planning policy, generating looping maneuvers to complete the task. These results corroborate our simulation findings, demonstrating the practical applicability and robustness of our proposed hierarchical planner.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we presented a novel hierarchical planning framework that leverages a conditional diffusion model for zero-shot adaptation to unexpected changes in a robot's dynamics. The framework's ability to instantly generate



(a) $\mathcal{D}_t = \{50, 50\}$deg/s (b) $\mathcal{D}_t = \{50, 50\}$deg/s



(c) $\mathcal{D}_t = \{0, 50\}$deg/s (d) $\mathcal{D}_t = \{0, 50\}$deg/s

Fig. 7. Real-robot experiment results in a maze environment: (a) Sequence of snapshots for Nominal conditions and (b) the associated final trajectory; (c) sequence of snapshots for a loss of left turn capabilities and (d) the associated final trajectory. The blue zone A marks the area of no fault. p indicates where the planner detects and adapts to the fault, performing several loops to complete the task.

safe and feasible trajectories in response to severe hardware faults, without requiring any retraining, was validated through complex simulations and on a physical robotic platform where it succeeded as baselines failed. Future work will extend the framework to affordance-aware adaptation, multi-robot coordination, and broader validation across diverse robotic platforms and sim-to-real settings.

VIII. Acknowledgment

This work is based on research sponsored by DARPA YFA award #D25AC00371.

References

[1] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," in *Conference on Robot Learning*. PMLR, 2023, pp. 859–870.

[2] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2019.

[3] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient domain randomization with bayesian optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 911–918, 2021.

[4] J. A. Townsend, P. Bellutta, M. Keuneke, M. Seibert, A. Stroupe, J. Wright, E. Ferguson, D. Forgette, J. Herman, H. Justice *et al.*, "Mars exploration rovers 2004-2013: Evolving operational tactics driven by aging robotic systems," in *SpaceOps 2014 Conference*, 2014, p. 1884.

[5] R. Polvara, M. Patacchiola, M. Hanheide, and G. Neumann, "Sim-to-real quadrotor landing via sequential deep q-networks and domain randomization," *Robotics*, vol. 9, no. 1, p. 8, 2020.

[6] T. Felin and M. Holweg, "Theory is all you need: Ai, human cognition, and causal reasoning," *Strategy Science*, 2024.

[7] H. Sasaki, C. G. Willcocks, and T. P. Breckon, "Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models," *arXiv preprint arXiv:2104.05358*, 2021.

[8] H. Wang, Y. Lin, W. Zhang, W. Ye, M. Zhang, and X. Dong, "Safe autonomous exploration and adaptive path planning strategy using signed distance field," *IEEE Access*, vol. 11, pp. 144 663–144 675, 2023.

[9] S. Sedighi, D.-V. Nguyen, and K.-D. Kuhnert, "Guided hybrid a-star path planning algorithm for valet parking applications," in *2019 5th international conference on control, automation and robotics (ICCAR)*. IEEE, 2019, pp. 570–575.

[10] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[11] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 3796–3810, 2021.

[12] J. Carvalho, A. T. Le, P. Kicki, D. Koert, and J. Peters, "Motion planning diffusion: Learning and adapting robot motion planning with diffusion models," *IEEE Transactions on Robotics*, 2025.

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

[14] S. Gao and N. Bezzo, "A schwarz-christoffel mapping-based framework for sim-to-real transfer in autonomous robot operations," *Journal of Intelligent & Robotic Systems*, vol. 111, no. 2, p. 57, 2025.

[15] E. Yel, S. Gao, and N. Bezzo, "Meta-learning-based proactive online planning for uavs under degraded conditions," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 320–10 327, 2022.

[16] D. Zhuo-hua, C. Zi-xing, and Y. Jin-xia, "Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3428–3433.

[17] G. Zhou, S. Swaminathan, R. V. Raju, J. S. Guntupalli, W. Lehrach, J. Ortiz, A. Dedieu, M. Lázaro-Gredilla, and K. Murphy, "Diffusion model predictive control," *arXiv preprint arXiv:2410.05364*, 2024.

[18] 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*, p. 02783649241273668, 2023.

[19] W. Teshome, K. Behzad, O. Camps, M. Everett, M. Siami, and M. Sznaier, "Real-time adaptive motion planning via point cloud-guided, energy-based diffusion and potential fields," *IEEE Robotics and Automation Letters*, 2025.

[20] K. M. Lee, S. Ye, Q. Xiao, Z. Wu, Z. Zaidi, D. B. D'Ambrosio, P. R. Sanketi, and M. Gombolay, "Learning diverse robot striking motions with diffusion models and kinematically constrained gradient guidance," *arXiv preprint arXiv:2409.15528*, 2024.

[21] N. Mohammad and N. Bezzo, "Soft actor-critic-based control barrier adaptation for robust autonomous navigation in unknown environments," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 2361–2367.

[22] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[23] J.-L. Blanco-Claraco, B. Tymchenko, F. J. Mañas-Alvarez, F. Cañadas-Aránega, Á. López-Gázquez, and J. C. Moreno, "Multivehicle simulator (mvsim): Lightweight dynamics simulator for multiagents and mobile robotics research," *SoftwareX*, vol. 23, p. 101443, 2023.

[24] J. Higgins, N. Mohammad, and N. Bezzo, "A model predictive path integral method for fast, proactive, and uncertainty-aware uav planning in cluttered environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 830–837.

[25] W. Chen and L. Li, "A study of ppo algorithms combining curiosity and imitation learning in maze environments," in *2024 2nd International Conference on Artificial Intelligence and Automation Control (AIAC)*, 2024, pp. 411–416.

[26] B. Balasuriya, B. Chathuranga, B. Jayasundara, N. Napagoda, S. Kumarawadu, D. Chandima, and A. Jayasekara, "Outdoor robot navigation using gmapping based slam algorithm," in *2016 moratuwa engineering research conference (mercon)*. IEEE, 2016, pp. 403–408.