Learning from Hallucinating Critical Points for Navigation in Dynamic Environments

Saad Abdul Ghani[1], Kameron Lee[2], and Xuesu Xiao[1]

***Abstract*— Generating large and diverse obstacle datasets to learn motion planning in environments with dynamic obstacles is challenging due to the vast space of possible obstacle trajectories. Inspired by hallucination-based data synthesis approaches, we propose Learning from Hallucinating Critical Points (LfH-CP), a self-supervised framework for creating rich dynamic obstacle datasets based on existing optimal motion plans without requiring expensive expert demonstrations or trial-and-error exploration. LfH-CP factorizes hallucination into two stages: first identifying when and where obstacles must appear in order to result in an optimal motion plan, i.e., the critical points, and then procedurally generating diverse trajectories that pass through these points while avoiding collisions. This factorization avoids generative failures such as mode collapse and ensures coverage of diverse dynamic behaviors. We further introduce a diversity metric to quantify dataset richness and show that LfH-CP produces substantially more varied training data than existing baselines. Experiments in simulation demonstrate that planners trained on LfH-CP datasets achieves higher success rates compared to a prior hallucination method.**

I. Introduction

Dynamic obstacles present a fundamental challenge for autonomous mobile robots, as their trajectories reside in a vast and high-dimensional space. Describing obstacle trajectories with velocity, acceleration, and higher order derivatives adds dimensions to the space, making it difficult for planners to anticipate and respond to complex motion patterns in real time. Consequently, intelligent navigation strategies must account for these dynamics and react in real time to avoid collisions effectively.

Learning-based models have recently demonstrated success in navigating such environments by leveraging collected data [1]. Two dominant paradigms—Imitation Learning (IL) and Reinforcement Learning (RL)—provide structured ways to gather and learn from experience. However, both paradigms face critical limitations: IL requires large numbers of expert demonstrations, while RL demands extensive trial-and-error exploration. Moreover, robust planners depend on training datasets that are sufficiently diverse to capture the variability of dynamic obstacles. As a result, learning-based planning has not performed to the same extent as in vision and language domains, where internet-scale labeled datasets are readily available.

The Learning from Hallucination (LfH) paradigm [2] provides a compelling solution to this challenge by synthesizing training data from prior navigation experiences. LfH takes past motion plans from simpler, even fully open environments and generate more constrained and complex scenarios, where those prior plans would be optimal. This process, known as "hallucination", enables safe and inexpensive creation of large datasets without relying on expensive expert demonstrations or extensive trial-and-error exploration. However, prior methods often produce datasets with limited diversity, constraining their applicability and scalability, especially when facing the vast and high-dimensional space of dynamic obstacles [3], [4].

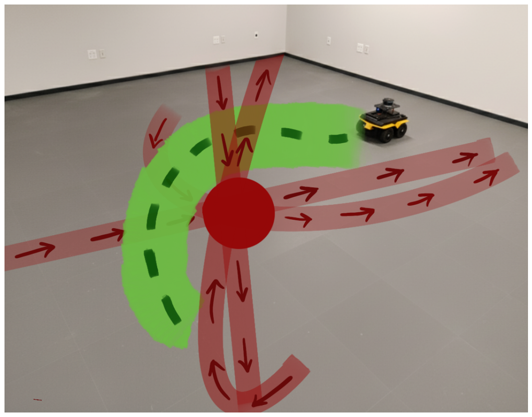


Fig. 1: A mobile robot's optimal path around the hallucinated critical point and its generated paths using LfH-CP.

In this work, we introduce Learning from Hallucinating Critical Points (LfH-CP), a self-supervised framework for generating rich datasets of dynamic obstacle trajectories. LfH-CP factorizes hallucination into two stages: first identifying the *critical points* of obstacle trajectories—specific times and locations obstacles must appear in order to result in an optimal motion plan—and then procedurally generating diverse trajectories with varying orders of complexity that pass through these points while avoiding collisions. These generated obstacle trajectories and original motion plans can then be used to train motion planners in a supervised manner. To quantify the richness of the generated datasets, we introduce a diversity metric and show that LfH-CP produces substantially more varied training data than existing methods. We validate these results in simulated experiments on a ground robot, demonstrating that planners trained on LfH-CP datasets achieve superior navigation performance compared to prior hallucination-based baselines. In summary, this work makes the following contributions:

- Propose LfH-CP, a self-supervised framework for generating large and diverse datasets of dynamic obstacle trajectories.

[1]George Mason University {sghani2, xiao}@gmu.edu
[2]Thomas Jefferson High school intern affiliated with the RobotiXX Lab, George Mason University. 2027klee@tjhsst.edu.

- Introduce a diversity metric to quantify dataset richness.
- Demonstrate effectiveness of LfH-CP through simulated robot experiments.
- Release code on GitHub to facilitate future research[1].

II. Related Work

This section reviews learning based techniques used in autonomous mobile robots for navigating in dynamic environments. It also reviews the recent LfH paradigm.

A. Machine Learning for Dynamic Obstacle Avoidance

Machine learning has been leveraged in mobile robot navigation in various ways [1], such as either integrating learning with classical methods [5]–[9] or using IL [10], [11] or RL [12]–[14] to develop end-to-end planners [15], [16]. Going beyond simple obstacle avoidance [17]–[19], learning is heavily used in social robot navigation where classical methods fail to calculate trajectories for human-populated spaces [11], [20]–[23]. Another use case is off-road navigation, in which learning can help to reason about the unstructured terrain underneath the robot [24]–[29]. Learning approaches can also directly navigate robots with RGB input alone [30]–[35]. Furthermore, advanced methods are emerging that seek richer representations of the environment, such as Inverse Reinforcement Learning (IRL) [36] to infer agent intent and Graph Neural Networks (GNNs) [37] to model the relational dynamics between multiple agents.

However, despite their success, most learning methods require either high-quality (IL) or extensive (RL) training data, such as those derived from human demonstrations or from trial-and-error exploration respectively, both of which are very difficult to acquire among dense and dynamic obstacles. LfH-CP is a self-supervised learning approach that automatically generates diverse training data, addressing the conundrum of needing to know what good navigation behavior is, without prior knowledge of how to achieve it.

B. Learning from Hallucination (LfH)

LfH [2]–[4], [38], [39] has been proposed to alleviate the difficulty of acquiring high-quality or extensive training data using completely safe exploration in open spaces or existing successful navigation experiences. Based on existing motion plans, the idea of hallucination is to generate obstacle configurations in which the existing plans would be optimal. Hallucination allows robots to reflect on past success and produces other training scenarios where such successful navigation can be repeated. Researchers have designed hallucination techniques to project the *most constrained* [2], a *minimal* [4], or a *learned* [3], [39] obstacle configuration onto the robot perception. Hallucination has also been used to enable multi-robot navigation in narrow hallways [38] and to augment existing global motion plans for which the global path is optimal [40].

However, the learned hallucination approaches [3], [39] face a fundamental problem that limits their generalization and usefulness. Dyna-LfLH [39] extends LfLH [3] by incorporating a velocity component in the hallucination model. Though this approach theoretically works well, the learnable components tend to mode collapse in the presence of the fixed, optimal decoder. LfLH, in comparison, can partially overcome mode collapse by hallucinating more obstacles in static environments. With dynamic obstacles, however, adding too many hallucinated obstacles can give the illusion of a safe "tunnel" where the the robot is never at risk of collision.

In this work, we introduce LfH-CP, which hallucinates realistic instantaneous obstacles at critical points and generates dynamic obstacle trajectories from them. This approach enables the safe and efficient generation of diverse, complex training scenarios for learning motion planners to navigate through highly cluttered, fast-moving, and unpredictable obstacles.

III. Approach

In this section, we begin by presenting the motion planning problem in dynamic environments and introduce the notion of *critical configurations*. Using this notion, we reformulate the problem under the Learning from Hallucination (LfH) paradigm, which provides the foundation for our proposed LfH-CP approach, visualized briefly in Fig. 2.

A. The Motion Planning Problem

Motion planning is often framed in the configuration space (C-space), which represents all possible robot configurations in a given environment. It is split into C_{obst}, the set of infeasible configurations blocked by obstacles or restricted by kinodynamic constraints, and $C_{\text{free}} = \text{C-space} \setminus C_{\text{obst}}$, the set of feasible configurations. In dynamic environments, the C-space evolves over time, yielding C^t_{obst} and C^t_{free} where $t \in \{1, \dots, H\}$ over a discrete time horizon H.

A motion plan is a sequence of actions that moves the robot from its current configuration c_c to a goal configuration c_g through intermediate configurations c^t such that $c^t \in C^t_{\text{free}}, \forall t$. The planning problem is then to find such a function that generates such feasible plans. Formally,

$$p = f(\{C^t_{\text{obst}}\}^H_{t=1} \mid c_c, c_g),$$

where $p = \{u^t\}^{H-1}_{t=0}$, $u^t \in \mathscr{U}$, and $\mathscr{U}$ denotes the robot's action space. An optimal planner $f^*(\cdot)$ seeks an optimal plan p^* that minimizes a given cost, such as travel time or path length.

B. The Critical Configurations

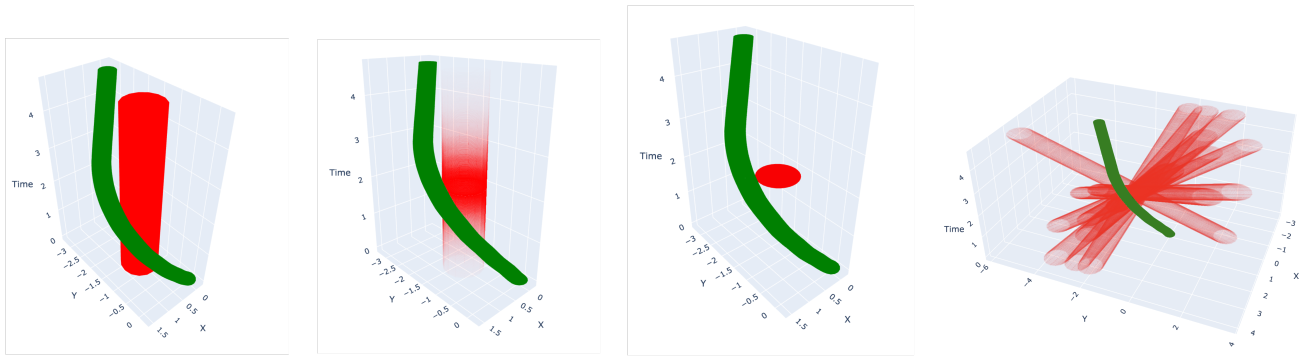
We posit that an optimal plan p^* does not depend on the full time-varying obstacle configurations $\{C^t_{\text{obst}}\}^H_{t=1}$, but rather on a smaller subset of time steps, which we term the *critical configurations*, denoted as $\mathbf{K}$:

$$\mathbf{K} = \{C^t_{\text{obst}} \mid t \in \mathscr{T}\}, \quad \mathscr{T} \subseteq \{1, 2, \dots, H\}, \qquad (1)$$

where $\mathscr{T}$ represents the critical time step(s). For other time step(s), the entire C-space can be free.

The intuition is simple: when navigating around a moving obstacle, the obstacle configurations only strongly influence

[1]https://github.com/Saadmaghani/LfH-CP



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

Fig. 2: The LfH-CP method visualized for motion plan (green) and one obstacle (red). We first learn where the obstacle should be to make our plan optimal using Eqn. 4 (Fig. 2a). Then using Eqn. 6, we learn where the obstacle should be temporally to make the plan optimal (Fig. 2b,2c). Finally using Eqn. 7, numerous obstacle trajectories can be generated from the obstacle's critical point (Fig. 2d).

the avoidance behavior when being close to the robot, usually instantaneously, while it matter less long before or after. Consequently, the plan only needs to consider the obstacle configurations at time steps $\mathscr{T}$, rendering the obstacles *instantaneous* and seemingly *teleport* in and out of existence at their defined locations. Despite this abstraction, the motion plan still remains feasible and optimal.

Formally, the planning problem can be reformulated as

$$p^* = f^*(\mathbf{K} \mid c_c, c_g).$$

This reformulation, however, raises the non-trivial problem of identifying the critical configurations $\mathbf{K}$ from $\{C^t_{\text{obst}}\}^H_{t=1}$. To address this, we leverage the LfH paradigm which enables direct synthesis of the critical configurations $\mathbf{K}$ from past motion plans p.

C. Reformulation in LfH paradigm

The LfH paradigm provides a natural way to tackle the challenge of identifying critical configurations by solving the inverse problem: given a motion plan p, hallucinate all obstacle configurations such that p is optimal. Formally in dynamic environments this is expressed as:

$$\{\{C^{t,i}_{\text{obst}}\}^H_{t=1}\}^\infty_{i=1} = f^{-1}(p \mid c_c, c_g),$$

that is, hallucinating all (possibly infinite) obstacle configuration sequences over time horizon H that make p optimal, given that p moves the robot from c_c to c_g.

In our reformulation, LfH would focus on hallucinating the *critical configurations* directly:

$$\{\mathbf{K}^i\}^\infty_{i=1} = f^{-1}(p \mid c_c, c_g), \tag{2}$$

representing all critical obstacle configurations that determine plan optimality.

Since it is impossible to predict all (infinite) possible configurations, we approximate it using a learned distribution. Concretely, the hallucination function $h(\cdot)$ outputs a distribution over the critical configurations:

$$\mathbf{K} \sim h(p \mid c_c, c_g). \tag{3}$$

This formulation forms the basis of our LfH-CP approach.

D. Hallucinating Critical Points

The hallucination function is parameterized by learnable parameters ψ and we learn $h_\psi(\cdot)$ in an encoder-decoder manner. Similar to prior approaches, we assume C_{obst} can be represented by a set of N circular obstacles $\{O_i\}^N_{i=1}$ with a fixed radii,

$$C_{\text{obst}} \approx \{O_i\}^N_{i=1}.$$

The distribution of obstacles can be represented by a Gaussian over their locations:

$$\{(x_i, y_i)\}^N_{i=1} \sim \mathcal{N}(\mu_i, \Sigma_i).$$

This assumption simplifies the problem of hallucinating obstacle configurations to hallucinating parameter distributions.

$h_\psi(\cdot)$ is learned in two phases. In the first phase, we learn *where* obstacles should be to make p optimal. The encoder takes the motion plan, current robot configuration, and goal configuration and outputs distributions over obstacle locations:

$$\{(\mu_i, \Sigma_i)\}^N_{i=1} = h_\psi(p \mid c_c, c_g).$$

To learn the distributions, the decoder $d(\cdot)$ is a classical motion planner without any learnable parameters that produces an optimal plan p^* given an obstacle configuration:

$$p^* = d(C_{\text{obst}} \sim h_\psi(p \mid c_c, c_g) \mid c_c, c_g).$$

The objective function is to minimize the reconstruction loss between p and p^*:

$$\psi^* = \underset{\psi}{\text{argmin}} \underset{\substack{p\sim P \\ p^*=d(C_{\text{obst}}\sim h_\psi(p|c_c,c_g)|c_c,c_g)}}{\mathbb{E}} \ell(p, p^*). \tag{4}$$

This ensures that obstacles are placed to make p optimal.

In the second phase we learn *when* obstacles should appear by estimating $\mathscr{T}$. In addition to the inputs and outputs in the first phase, $h_\psi(\cdot)$ outputs logits $\alpha_i \in \mathbb{R}^H$ for each obstacle. α_i is passed through the Gumbel-Softmax distribution to produce $\mathbf{k}_i \in [0, 1]^H$, an H-dimensional probability distribution.

Scaling by the maximum value yields a soft, continuous mask $\mathbf{m}_i = \mathbf{k}_i / \max(\mathbf{k_i})$, representing the temporal presence of obstacle i: $\mathbf{m}_i^t = 0$ indicates absence, $\mathbf{m}_i^t = 1$ indicates full presence at time step t.

Finally, to construct $\mathscr{T}$ and $\mathbf{K}$, the argument-maximum is taken over $\mathbf{m}$. Formally,

$$C_{\text{obst}} \approx \{O_i\}_{i=1}^N \sim \{(\mu_i, \Sigma_i, \alpha_i)\}_{i=1}^N = h_\psi(p \mid c_c, c_g),$$
$$\mathbf{K} \approx \{O_i^{t_i^{\text{crit}}}\}_{i=1}^N, \quad t_i^{\text{crit}} = \operatorname{argmax} \mathbf{m}_i. \tag{5}$$

For training, a soft version of the critical configurations, $\mathbf{K}_{\text{soft}}$, is constructed directly from $\mathbf{m}_i$ and passed to the decoder:

$$p^* = d(\mathbf{K}_{\text{soft}} \sim h_\psi(p \mid c_c, c_g) \mid c_c, c_g),$$

allowing the decoder to account for partial obstacle presence. The objective remains the same:

$$\psi^* = \operatorname*{argmin}_\psi \mathop{\mathbb{E}}_{\substack{p \sim P \\ p^* = d(\mathbf{K}_{\text{soft}} \sim h_\psi(p|c_c,c_g)|c_c,c_g)}} \ell(p, p^*). \tag{6}$$

E. Rendering Obstacle Trajectories

The generation function $g(\mathbf{K})$ uses the critical configurations to generate unlimited obstacle trajectories over time horizon H. Specifically,

$$\{C_{obst}^t\}_{t=1}^H \approx \{\{O_i^t\}_{i=1}^N\}_{t=1}^H = g(\mathbf{K} \sim h_{\psi^*}(p \mid c_c, c_g)). \tag{7}$$

$g(\mathbf{K})$ is not tied to a specific parameterization, but it must satisfy two constraints: (i) each obstacle i must be at (x_i, y_i) at $t = \operatorname{argmax} \mathbf{m}_i$, and (ii) generated trajectories must remain collision-free with the plan p. Examples of $g(\cdot)$ are functions that sample smooth trajectories from first, second, or third order dynamics (see Eqn. 9), or piece-wise functions that define obstacle motion over H.

F. Learning from Hallucinating Critical Points

After sampling critical points for all N obstacles from $h_{\psi^*}(\cdot)$ and generating S obstacle trajectories using $g(\cdot)$, we construct a supervised training dataset for IL:

$$\mathscr{D}_{\text{train}} = \{(\{C_{\text{obst}}^{t,j}\}_{t=1}^H, p^j, c_c^j, c_g^j)\}_{j=1}^S.$$

Here, each plan p^j is (roughly) optimal for its corresponding configuration space $\{C_{\text{obst}}^{t,j}\}_{t=1}^H$. The configuration space is transformed into sensor observations (e.g., LiDAR scans via ray tracing), enabling us to train a motion planner $f_\theta(\cdot)$ to reproduce p^j from $\{C_{\text{obst}}^{t,j}\}_{t=1}^H$. Formally,

$$\theta^* = \operatorname*{argmin}_\theta \mathop{\mathbb{E}}_{\substack{(\{C_{\text{obst}}^t\}_{t=1}^H, p, c_c, c_g) \\ \sim \mathscr{D}_{train}}} \left[\ell(p, f_\theta(\{C_{\text{obst}}^t\}_{t=1}^H \mid c_c, c_g))\right]. \tag{8}$$

During deployment, $f_{\theta^*}(\cdot)$ will be used to plan around the sensed dynamic environment.

IV. Implementation

In this section we discuss the particular instantiation of all data and models discussed in Sec. III.

A. Learning h_ψ

We collect a dataset of motion plans P, where each plan is a sequence of $\mathbb{SE}(2)$ robot configurations (x^t, y^t, ψ^t) and linear and angular velocity actions (v^t, ω^t). Each plan is segmented using a sliding window of size 233 ($\approx$5 s of odometry time). Configurations are expressed in the robot frame, so $c_c = \mathbf{0}$ and is omitted. The final configuration of each segment is treated as the goal c_g and is contained in p, so it is not modeled separately.

The hallucination function h_ψ is implemented as a 3-layer convolutional encoder (channels [16,32,64], kernels [5,5,3], stride 2), followed by N autoregressive linear layers that output μ, Σ, and α.

Training proceeds in two phases. In Phase 1 of training, h_ψ is optimized for 1000 epochs to predict μ and Σ, generating $N = 10$ circular obstacles (radius 0.5 m) centered at sampled $\{(x_i, y_i)\}_{i=1}^N$ coordinates. In Phase 2, the network learns α over 1500 epochs. Initially, temporal masks $\mathbf{m}_i = \mathbf{1}$, i.e. obstacle is present at all time steps. Over 1000 epochs, $\mathbf{m}_i$ is annealed to one-hot vectors at $t_i^{\text{crit}} = \operatorname{argmax} \mathbf{m}_i$ via a Gumbel–Softmax with temperature τ decaying from 2048 to 0.1. The final 500 epochs train with one-hot $\mathbf{m}_i$.

The decoder d is a re-implementation of Ego-Planner [41] with convex optimization layers. It uses $\mathbf{m}_i$ to scale obstacle radii and safety clearances over time, so that $\mathbf{m}_i^t \to 0$ nullifies collisions with obstacle i at time t.

The reconstruction loss is mean-squared error between the input plan p and reconstructed plan p^*. Following prior methods, we stabilize training by adding priors and penalties: (i) a Gaussian prior fitted to p is imposed on obstacle centers, biasing obstacles toward the trajectory; and (ii) penalties are applied for obstacle–obstacle and obstacle–plan overlap.

B. Generating $\mathscr{D}_{train}$

For generating $\mathscr{D}_{\text{train}}$, we sample $S_1 = 1$ critical point per obstacle from h_{ψ^*} for each $p^j \in P$, forming $\mathbf{K}^j$. To isolate only the obstacles essential for optimizing the plan, we filter based on reconstruction loss: obstacles are added incrementally in order of contribution, and retained only if they reduce the loss by at least 1%. The process stops when no further improvement is observed or when $N_{\text{max}} = 7$ is reached. Finally, we require at least a 90% overall reduction in reconstruction loss relative to the baseline straight-line plan from start to goal, obtained when obstacles are unoptimized and placed far away. The resulting $\mathbf{K}_{\text{filtered}}^j$ thus contains only the obstacle critical points that contribute most to optimizing p^j.

Obstacle trajectories are instantiated using $g(\mathbf{K}_{\text{filtered}}^j)$, which samples velocities from a uniform distributions between $[1, 2]$ m/s, and applies the first-order equation of motion to produce smooth trajectories. Collision checks ensure that the generated trajectories do not intersect the corresponding plan p^j:

$$g\Big(\{(x_i^{t_i^{\text{crit}}}, y_i^{t_i^{\text{crit}}})\}_{i=1}^{|\mathbf{K}_{\text{filtered}}^j|}\Big) = \Big\{\mathbf{S}_i + \mathbf{V}_i t\Big\}_{i=1}^N,$$
$$t \in [1 - t_i^{\text{crit}}, T - t_i^{\text{crit}}], \tag{9}$$
$$\forall i = \{1, \ldots, |\mathbf{K}_{\text{filtered}}^j|\},$$

where $\mathbf{S}_i = (x_i^{t_i^{\text{crit}}}, y_i^{t_i^{\text{crit}}})$, and $\mathbf{V}_i$ is the (x, y) components of the sampled velocity.

Using Eqn. (9), we sample $S_2 = 50$ trajectories from each $\mathbf{K}^j_{\text{filtered}}$, producing up to $S_2 \times N_{\max}$ obstacle sequences per data point j. These are rendered as 2D LiDAR scans given $c^{t,j}$ along p^j.

To improve robustness, $\mathscr{D}_{\text{train}}$ is further augmented. First, up to 20 random non-colliding obstacle trajectories are added to C_{obst} to introduce noisy, non-optimizing obstacles. Second, obstacle-free motion plans with speed above 0.9 m/s and direction aligned with the goal 90% of the time (cosine similarity ≥ 0.9) are added to encourage fast goal-directed navigation in open spaces.

C. Learning and Deployment of f_{θ^*}

The motion planner f_{θ^*} predicts a sequence of $M_a = 5$ actions, $\{u^i\}_{i=1}^{M_a}$, conditioned on $M_l = 5$ LiDAR scans (comprising $M_l - 1$ historical scans and the current scan) representing obstacle configurations $\{C^t_{\text{obst}}\}_{t=i-M_l+1}^{i+M_a-1}$ and on $M_l = 5$ past actions. Using multiple past scans ($M_l > 1$) enables the planner to capture obstacle dynamics, while predicting multiple future actions ($M_a > 1$) supports longer-horizon reasoning. Each training instance is anchored at the robot's current configuration $c^i_c = \mathbf{0}$, with the goal c^i_g defined as a unit vector pointing to the end of plan p.

The planner $f_{\theta^*}(\cdot)$ is implemented as a causal transformer with two encoder layers, two attention heads, and a 256-dimensional feed-forward network. LiDAR scans are first projected into a 256-dimensional latent space using a 2-layer 1D CNN (kernel size 5). The transformer output is concatenated with the goal vector and processed by a 2-layer MLP head to predict linear and angular velocities.

During deployment, at each time step t', the robot's current configuration is set as $c^{t'}_c = \mathbf{0}$, and the goal $c^{t'}_g$ is a unit vector pointing 2.25 m ahead along the global path given by the `move_base` navigation stack. The planner predicts M_a actions but executes only the first at each step. Following prior work, an MPC safety module monitors collisions: if a collision is imminent, the robot halts first and then reverses to avoid obstacles.

V. Experiments and Evaluation

In this section, we first demonstrate that a single critical point is sufficient to produce an optimal trajectory. We then evaluate the hallucination of critical points and introduce a metric for dataset richness, showing how our method generates diverse obstacle trajectories that maximize this metric. Finally, we present results for learning a motion planner from the hallucinated trajectories and demonstrate that the learned planner outperforms baseline methods in simulated environments.

A. Hallucinating Critical Points

In Fig. 5, we compare the original trajectories with those reconstructed from obstacles at the end of Phase 1 (P1) and from obstacles instantiated solely at their critical points at the end of Phase 2 (P2). Visually, the P2 reconstructed trajectories remain similar to their P1 counterpart. Such similarities validate our hypothesis that obstacles only appearing at critical configurations are sufficient to make a motion plan optimal, instead of requiring the obstacles' constant presence.

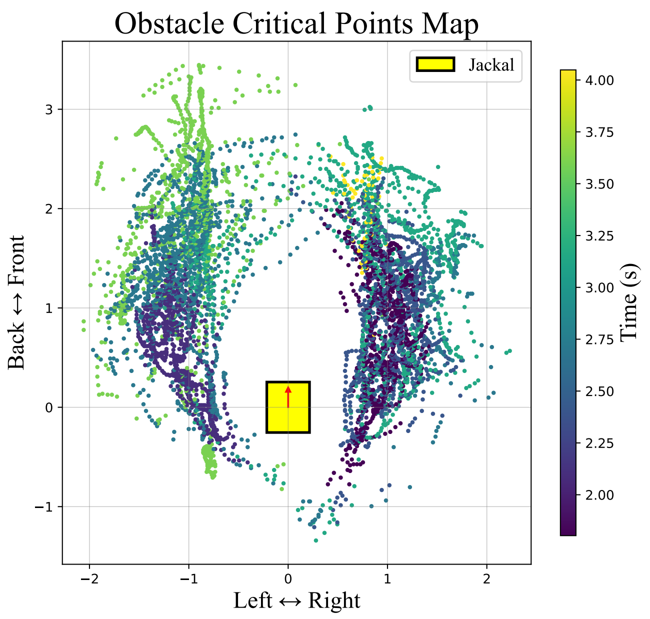


Fig. 3: Obstacle critical points map.

In Fig. 3, we visualize the hallucinated critical points. Each point indicates a location and time such that if an obstacle passes through it, there exists an optimal motion plan moving the robot from its current configuration $(0, 0)$ to a goal configuration behind the obstacle while minimizing travel distance. The map is generated by sampling one critical point per obstacle, filtering out non-essential points, and plotting their positions. When constructing $\mathscr{D}_{\text{train}}$, a moving window along the robot trajectory shifts the hallucinated critical points closer to the robot's current configuration, making the effective map denser in time and space.

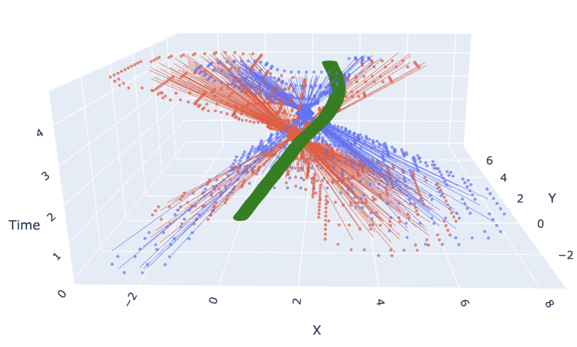
In Fig. 4, we illustrate the diversity of obstacle trajectories that can be generated from a *single* critical point. Despite the variations in their paths, all these obstacles can be avoided using the same underlying motion plan, which remains near-optimal in terms of travel efficiency.

B. Measuring Dataset Coverage

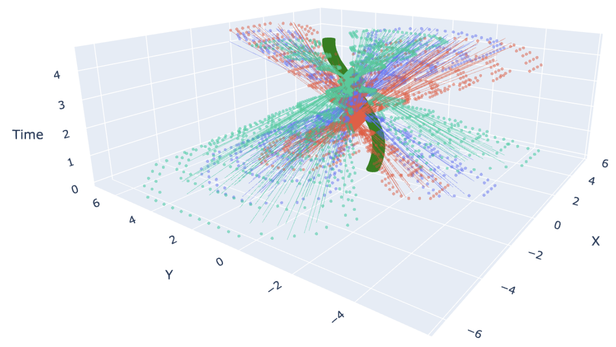
To quantify the richness of the generated datasets, we define a *Dataset Coverage Score (DCS)*, which measures how well the dataset spans the space of relevant obstacle configurations. We consider four key components:

- Distance between the robot and an obstacle, r;
- Angle between the robot and an obstacle, θ;
- Obstacle speed, s; and
- Obstacle heading in the robot frame, ψ.

For each component, we compute marginal coverage by checking whether there exists at least one data point within each bin across the specified bounds and at the specified resolution. The bounds and resolutions used for these calcu-



(a)



(b)

Fig. 4: 100 obstacle trajectories are generated from each critical point. The robot trajectory (green) remains collision-free and near-optimal, regardless of the variations and combinations of obstacle trajectories. That is, the same robot trajectory is optimal and collision-free for any single combination of these 100^N possible obstacle trajectory sets. Fig. 4a show a total of 200 obstacle trajectories generated from $N = 2$ critical points, while Fig. 4b show a total of 300 obstacle trajectories generated from $N = 3$ critical points.

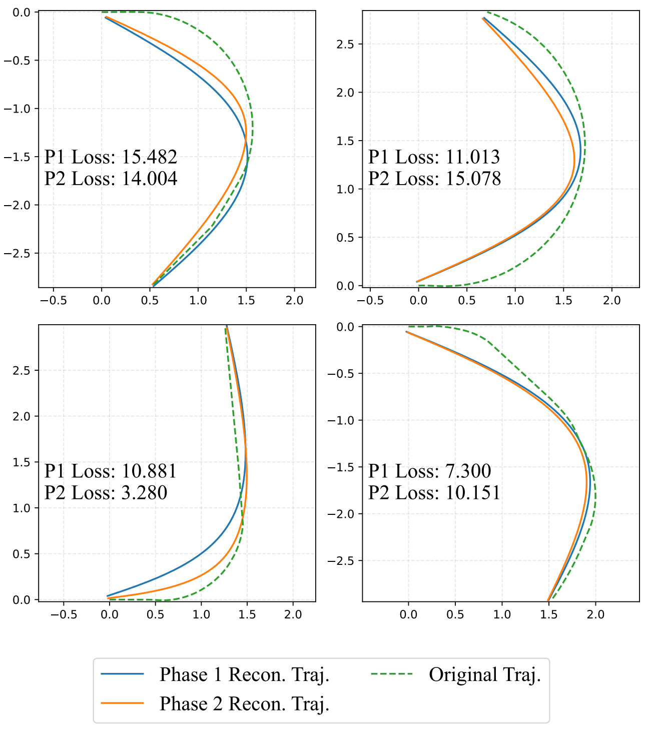


Fig. 5: Reconstructed trajectories from Phase 1 (P1) and Phase 2 (P2) using obstacles at their critical points. Trajectories remain visually close to each other, with only minor variations in reconstruction loss.

lations are summarized in Tab. I.

We define overall dataset coverage as the joint coverage across all four components. Full coverage is achieved when every resolution bin within the specified bounds of each component contains at least one data point. DCS is then given by the ratio of the observed joint coverage to this theoretical upper bound.

The coverage achieved by our method is reported in Tab. II. LfH-CP can achieve almost 100% coverage when considering up to three metrics, while for all four metrics a coverage of 62.21% is reached. In contrast, Dyna-LfLH achieves only 11.05% over the the same range. To further demonstrate robustness, Fig. 6 shows how increasing the number of generated samples improves the coverage score. This indicates that our approach avoids mode collapse and effectively populates diverse obstacle scenarios.

TABLE I: Bounds and Resolution for Dataset Coverage.

Component (unit)	Lower bound	Upper bound	Resolution
r (m)	0.15	2	0.1
θ (deg)	-180	180	5
s (m/s)	1	2	0.1
ψ (deg)	-180	180	5

TABLE II: DCS for all metric combinations for our LfH-CP and comparitive approach.

Metric Combination	Coverage LfH-CP(%)	Coverage Dyna-LfLH (%)
r, θ, s, ψ	62.21	11.05
r, θ, s	100.0	74.61
θ, s, ψ	99.18	49.30
r, s, ψ	100.0	69.6
r, θ, ψ	93.20	40.49
θ, s	100.0	99.13
r, s	100.0	89.47
r, θ	100.0	89.40
r, ψ	100.0	86.52
θ, ψ	100.0	89.06
s, ψ	100.0	91.41
r	100.0	89.47
θ	100.0	100.0
s	100.0	100.0
ψ	100.0	100.0

C. Learning from Hallucinated Critical Points

We evaluate the learned motion planner using DynaBARN [42], a simulation testbed for dynamic obstacle avoidance. DynaBARN supports generation of environments populated with moving obstacles at varying speeds, following

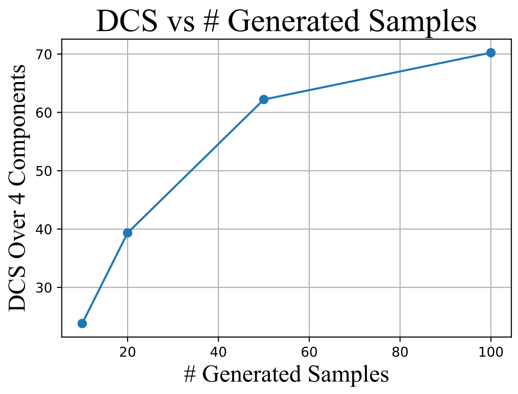


Fig. 6: DCS increases with more samples.

TABLE III: DynaBARN Parameters.

Difficulty	# Worlds	Number of obstacles	Range of speed
Easy	20	[5, 10]	[0.5, 1.0]
Medium	10	[5, 10]	[1.0, 0.5]
Medium	10	[10, 20]	[0.5, 1.0]
Hard	20	[10, 20]	[1.0, 2.0]

trajectories with different motion patterns.

We generate 60 environments with 3 levels of difficulty similar to the original DynaBARN implementation using the parameters in Tab. III. We compare LfH-CP against Dyna-LfLH. Each planner is evaluated over 2 trials in each of the 60 environments, yielding a total of 120 trials. Simulation results are summarized in Fig. 7.

The overall success rates in DynaBARN are low due to the fast-moving and highly cluttered environments. LfH-CP achieves a higher success rate at 30.83%, showing that hallucinated critical points provide strong navigation performance. In contrast, Dyna-LfLH underperforms with success rates of 22.5%, due to mode collapse during hallucination that reduces obstacle diversity and generalization.

Despite achieving high dataset coverage, LfH-CP does not perform particularly well during deployment. One reason is that the plans are optimized for only a few obstacles that are rendered at a time. While random non-colliding obstacles are added for robustness, the planner may still struggle when multiple obstacles appear simultaneously during deployment. A potential solution would be to generate multiple obstacle samples passing through the same critical point within the same demonstration. Finally, the Dataset Coverage Score (DCS) measures coverage for individual obstacles, ensuring at least one sample exists per resolution bin. However, in real scenarios, multiple obstacles often appear together, suggesting that DCS could be extended to account for joint coverage of two or more obstacles simultaneously.

VI. Conclusions

In this paper, we present LfH-CP, a self-supervised framework for creating rich dynamic obstacle datasets based on existing optimal motion plans to create supervised training data. Situated within the LfH paradigm, LfH-CP does not require expensive expert demonstrations for IL or trial-and-error exploration for RL and extends to dynamics obstacles. Facing the vast and high-dimensional space of dynamic obstacles, LfH-CP further tackles the mode collapse problem that degrades previous LfH approaches' performance by hallucinating only the critical points for those obstacles, i.e., where and when these obstacles have to appear to assure an existing motion plan's optimality. Our new obstacle diversity metric, DCS, shows that LfH-CP produces substantially more varied training data than existing baseline and DCS scales with more samples, instead of saturates due to mode collapse. LfH-CP also achieves improved navigation performance in dynamics obstacles compared to state-of-the-art approaches.

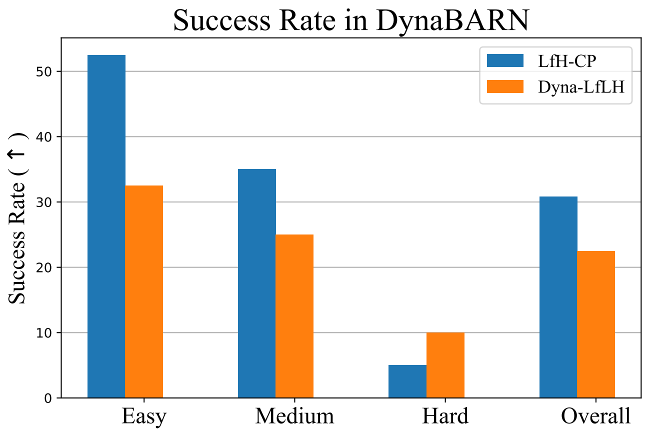


Fig. 7: Simulation Results of 120 trials in DynaBARN.

References

[1] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[2] ——, "Toward agile maneuvers in highly constrained spaces: Learning from hallucination," *IEEE Robotics and Automation Letters*, pp. 1503–1510, 2021.

[3] Z. Wang, X. Xiao, A. J. Nettekoven, K. Umasankar, A. Singh, S. Bommakanti, U. Topcu, and P. Stone, "From agile ground to aerial navigation: Learning from learned hallucination," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 148–153.

[4] X. Xiao, B. Liu, and P. Stone, "Agile robot navigation through hallucinated learning and sober deployment," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[5] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[6] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "Appld: Adaptive planner parameter learning from demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4541–4547, 2020.

[7] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone, "Appli: Adaptive planner parameter learning from interventions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[8] Z. Xu, G. Dhamankar, A. Nair, X. Xiao, G. Warnell, B. Liu, Z. Wang, and P. Stone, "Applr: Adaptive planner parameter learning from reinforcement," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[9] X. Xiao, Z. Wang, Z. Xu, B. Liu, G. Warnell, G. Dhamankar, A. Nair, and P. Stone, "Appl: Adaptive planner parameter learning," *Robotics and Autonomous Systems*, vol. 154, p. 104132, 2022.

[10] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017, pp. 1527–1533.

[11] X. Xiao, T. Zhang, K. M. Choromanski, T.-W. E. Lee, A. Francis, J. Varley, S. Tu, S. Singh, P. Xu, F. Xia, S. M. Persson, L. Takayama, R. Frostig, J. Tan, C. Parada, and V. Sindhwani, "Learning model predictive controllers with real-time attention for real-world navigation," in *Conference on robot learning*. PMLR, 2022.

[12] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[13] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking reinforcement learning techniques for autonomous navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9224–9230.

[14] Z. Xu, X. Xiao, G. Warnell, A. Nair, and P. Stone, "Machine learning methods for local motion planning: A study of end-to-end vs. parameter learning," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2021, pp. 217–222.

[15] J. Zeng, R. Ju, L. Qin, Y. Hu, Q. Yin, and C. Hu, "Navigation in unknown dynamic environments based on deep reinforcement learning," *Sensors*, vol. 19, no. 18, p. 3837, 2019.

[16] B. Wullt, P. Matsson, T. B. Schön, and M. Norrlöf, "Neural motion planning in dynamic environments," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 10 126–10 131, 2023.

[17] X. Xiao, Z. Xu, Z. Wang, Y. Song, G. Warnell, P. Stone, T. Zhang, S. Ravi, G. Wang, H. Karnan *et al.*, "Autonomous ground navigation in highly constrained spaces: Lessons learned from the benchmark autonomous robot navigation challenge at icra 2022 [competitions]," *IEEE Robotics & Automation Magazine*, vol. 29, no. 4, pp. 148–156, 2022.

[18] X. Xiao, Z. Xu, A. Datar, G. Warnell, P. Stone, J. J. Damanik, J. Jung, C. A. Deresa, T. D. Huy, C. Jinyu *et al.*, "Autonomous ground navigation in highly constrained spaces: Lessons learned from the third barn challenge at icra 2024 [competitions]," *IEEE Robotics & Automation Magazine*, vol. 31, no. 3, pp. 197–204, 2024.

[19] B. Liu, X. Xiao, and P. Stone, "A lifelong learning approach to mobile robot navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1090–1096, 2021.

[20] R. Mirsky, X. Xiao, J. Hart, and P. Stone, "Conflict avoidance in social navigation–a survey," *ACM Transactions on Human-Robot Interaction*, 2024.

[21] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone, "Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 807–11 814, 2022.

[22] D. M. Nguyen, M. Nazeri, A. Payandeh, A. Datar, and X. Xiao, "Toward human-like social robot navigation: A large-scale, multi-modal, social human navigation dataset," *arXiv preprint arXiv:2303.14880*, 2023.

[23] A. Francis, C. Pérez-d'Arpino, C. Li, F. Xia, A. Alahi, R. Alami, A. Bera, A. Biswas, J. Biswas, R. Chandra *et al.*, "Principles and guidelines for evaluating social robot navigation algorithms," *ACM Transactions on Human-Robot Interaction*, vol. 14, no. 2, pp. 1–65, 2025.

[24] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[25] H. Karnan, K. S. Sikand, P. Atreya, S. Rabiee, X. Xiao, G. Warnell, P. Stone, and J. Biswas, "Vi-ikd: High-speed accurate off-road navigation using learned visual-inertial inverse kinodynamics," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3294–3301.

[26] A. Datar, C. Pan, M. Nazeri, and X. Xiao, "Toward wheeled mobility on vertically challenging terrain: Platforms, datasets, and algorithms," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[27] A. Datar, C. Pan, M. Nazeri, A. Pokhrel, and X. Xiao, "Terrain-attentive learning for efficient 6-dof kinodynamic modeling on vertically challenging terrain," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5438–5443.

[28] A. Datar, C. Pan, and X. Xiao, "Learning to model and plan for wheeled mobility on vertically challenging terrain," *IEEE Robotics and Automation Letters*, 2024.

[29] A. Pokhrel, A. Datar, M. Nazeri, and X. Xiao, "Cahsor: Competence-aware high-speed off-road ground navigation in se (3)," *arXiv preprint arXiv:2402.07065*, 2024.

[30] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7226–7233.

[31] H. Karnan, G. Warnell, X. Xiao, and P. Stone, "Voila: Visual-observation-only imitation learning for autonomous navigation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2497–2503.

[32] K. S. Sikand, S. Rabiee, A. Uccello, X. Xiao, G. Warnell, and J. Biswas, "Visual representation learning for preference-aware path planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 303–11 309.

[33] S. Ravi, G. Wang, S. Satewar, X. Xiao, G. Warnell, J. Biswas, and P. Stone, "Visually adaptive geometric navigation," in *2023 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2023.

[34] M. H. Nazeri and M. Bohlouli, "Exploring reflective limitation of behavior cloning in autonomous vehicles," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 1252–1257.

[35] M. Nazeri, J. Wang, A. Payandeh, and X. Xiao, "Vanp: Learning where to see for navigation with self-supervised vision-action pre-training," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 2741–2746.

[36] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1557–1577, 2016.

[37] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *201S9 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5899–5905.

[38] J.-S. Park, X. Xiao, G. Warnell, H. Yedidsion, and P. Stone, "Learning perceptual hallucination for multi-robot navigation in narrow hallways," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 033–10 039.

[39] S. A. Ghani, Z. Wang, P. Stone, and X. Xiao, "Dyna-lflh: Learning agile navigation in dynamic environments from learned hallucination," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[40] D. Das, Y. Lu, E. Plaku, and X. Xiao, "Motion memory: Leveraging past experiences to accelerate future motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 16 467–16 474.

[41] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, 2020.

[42] A. Nair, F. Jiang, K. Hou, Z. Xu, S. Li, X. Xiao, and P. Stone, "Dynabarn: Benchmarking metric ground navigation in dynamic environments," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2022, pp. 347–352.