Dyna-LfLH: Learning Agile Navigation in Dynamic Environments from Learned Hallucination

Saad Abdul Ghani[1], Zizhao Wang[2], Peter Stone[2,3], and Xuesu Xiao[1]

***Abstract*— This paper introduces Dynamic Learning from Learned Hallucination (Dyna-LfLH), a self-supervised method for training motion planners to navigate environments with dense and dynamic obstacles. Classical planners struggle with dense, unpredictable obstacles due to limited computation, while learning-based planners face challenges in acquiring high-quality demonstrations for imitation learning or dealing with exploration inefficiencies in reinforcement learning. Building on Learning from Hallucination (LfH), which synthesizes training data from past successful navigation experiences in simpler environments, Dyna-LfLH incorporates dynamic obstacles by generating them through a learned latent distribution. This enables efficient and safe motion planner training. We evaluate Dyna-LfLH on a ground robot in both simulated and real environments, achieving up to a 25% improvement in success rate compared to baselines.**

I. Introduction

Dynamic obstacles present a significant challenge for autonomous mobile robots, requiring them to adapt their motion plans in real-time to avoid collisions. Pedestrians crossing streets unexpectedly or other robots performing independent tasks in warehouses exemplify the types of dynamic obstacles that challenge autonomous mobile robots. Such obstacles are often characterized by unpredictable motion patterns, demanding intelligent navigation strategies that can anticipate and respond to rapid changes in the environment.

Navigating around unpredictable dynamic obstacles in a highly dense environment poses significant computational challenges for classical navigation systems, making it inefficient to draw samples or perform optimization iterations in real-time. Recently, machine learning approaches have been used to successfully maneuver around such obstacles in a data-driven manner [1], [2]. However, both Imitation Learning (IL) and Reinforcement Learning (RL) approaches depend on high-quality training data that is difficult and inefficient to acquire in the former and latter cases respectively.

[1]George Mason University {sghani2, xiao}@gmu.edu [2]The University of Texas at Austin zizhao.wang@utexas.edu, pstone@cs.utexas.edu [3]Sony AI

This work has taken place in the RobotiXX Laboratory at George Mason University and the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. RobotiXX research is supported by National Science Foundation (NSF, 2350352), Army Research Office (ARO, W911NF2320004, W911NF2420027, W911NF2520011), Air Force Research Laboratory, US Air Forces Central, Google DeepMind, Clearpath Robotics, Raytheon Technologies, Tangenta, Mason Innovation Exchange, and Walmart. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858), Office of Naval Research (N00014-24-1-2550), ARO (FAIN W911NF-17-2-0181, W911NF-23-2-0004, W911NF-25-1-0065), DARPA (Cooperative Agreement HR00112520004 on Ad Hoc Teamwork), Lockheed Martin, and Good Systems, a research grand challenge at the University of Texas at Austin. The views and conclusions contained in this document are those of the authors alone. Peter Stone serves as the Chief Scientist of Sony AI and receives financial compensation for that role. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

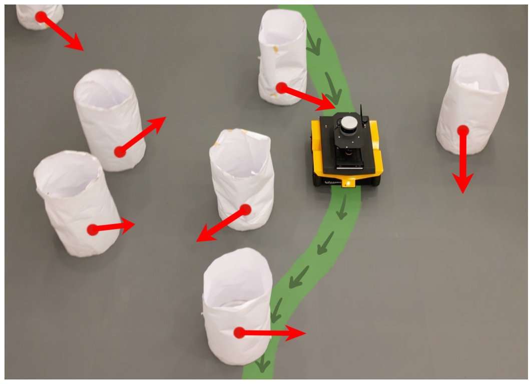

Fig. 1: A mobile robot navigating through a dense and dynamic obstacle field using Dyna-LfLH.

Learning from Hallucination (LfH) [2]–[5] is a paradigm that can safely and efficiently provide a variety of training data for collision avoidance without the need of actually training in challenging obstacle configurations. In LfH, the robot gathers motion plans from past navigation experiences in relatively easy or completely open environments, imagines other more difficult obstacle configurations for which the existing motion plans would also be optimal (i.e., hallucination), and then learns a motion planner based on the hallucinated obstacle configurations and motion plans. This process circumvents the data dependency of IL and RL as one can generate a large amount of data safely and efficiently without the need for an expert supervisor or trial-and-error exploration. However, existing LfH methods are only designed to hallucinate static environments and fail to perform well in dynamic ones.

In this paper, we propose a new Dynamic Learning from Learned Hallucination (Dyna-LfLH) approach (Fig. 1). We design a novel latent distribution that can be learned through Dyna-LfLH in a self-supervised manner and then sampled from to generate a variety of dynamic obstacle configurations. Paired with existing optimal motion plans, these dynamic obstacle configurations are used to learn a motion planner to navigate in environments filled with a large number of dynamic obstacles. Dyna-LfLH is tested on a ground robot both in simulated and physical environments. Superior navigation performance is achieved when compared to LfLH [2], a classical sampling-based motion planner (DWA) [6], a state of the art sampling-based model predictive controller (Log-MPPI) [7], and an IL method [8].

II. Related Work

This section reviews classical motion planning and recent machine learning techniques for mobile robot navigation in dynamic environments. We also introduce the recent LfH paradigm, which our Dyna-LfLH belongs to.

A. Classical Motion Planning

Two popular approaches [9] to solve motion planning in dynamic environments are Artificial Potential Fields (APF) [10] and velocity-based methods [6]. In APF, the environment is modeled as a field of attractive and repulsive forces, guiding the robot through space. The goal has an attractive potential field while obstacles have a negative potential field. The resultant force is calculated to guide the robot towards the goal and away from the obstacles. Velocity-based methods directly work on the robot's and obstacles' kinematics and dynamics. First, the robot's and obstacles' kinematics are taken into account and an initial kinematic trajectory is created to avoid the obstacles. Then, using the robot's dynamics, a motion plan is created to closely follow the initial kinematic trajectory. Dynamic Window Approach (DWA) [6] is a well-known example of a velocity-based method.

Compared to APF and velocity-based motion planners, Dyna-LfLH uses configuration space, which decomposes the environment into free and obstacle spaces. An advantage of our learning approach is that its computation is not dependent on obstacle density and movement during deployment, because Dyna-LfLH simply queries a pre-trained neural network to produce feasible and fast navigation behaviors.

B. Machine Learning for Navigation

Machine learning approaches have been applied to mobile robot navigation in different ways [1], such as either applying learning in conjunction with classical methods [11] or using IL [12] or RL [13] to learn an end-to-end planner [14].

Most learning methods require either high-quality (IL) or extensive (RL) training data. Dyna-LfLH 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.

C. Learning from Hallucination (LfH)

LfH [2]–[5] generates training data by hallucinating obstacle configurations where existing plans are optimal, thus eliminating the need for expert demonstrations or risky exploration. Researchers have designed hallucination techniques to project the *most constrained* [3], a *minimal* [4], or a *learned* [2] obstacle configuration onto the robot perception. Hallucination has also been used to enable multi-robot navigation in narrow hallways [5] and to augment existing global motion plans for which the global path is optimal for [15].

However, all existing LfH approaches assume that the environment is static, or enforce static perception through hallucinating virtual fields [5]. In this work, we generalize the existing LfH formulation into dynamic environments and show our new Dyna-LfLH can hallucinate appropriate dynamic obstacles to safely and efficiently provide training data to learn an agile motion planner to navigate through highly-cluttered, fast-moving, hard-to-predict obstacles.

III. Approach

In this section, we introduce our Dyna-LfLH approach. We first formalize the problem of motion planning in dynamic environments and then reformulate its "inverse" problem, i.e., dynamic obstacle hallucination using the LfH paradigm. Finally, we propose an algorithm to learn a dynamic hallucination function from which we can generate a variety of dynamic obstacle configurations to train a motion planner.

A. Problem Definition

Robot motion planning in static environments is typically formulated in configuration space (C-space), representing the set of all possible robot configurations. In a particular environment, C is partitioned into $C = C_{free} \cup C_{obst}$, where C_{free} denotes collision-free configurations and C_{obst} represents configurations blocked by obstacles or constraints. A motion plan $p \in \mathscr{P}$ consists of actions $p = \{u^t\}_{t=0}^{T-1}$, $u^t \in \mathscr{U}$, where $\mathscr{P}$ is the plan space over discrete time horizon T and $\mathscr{U}$ is its action space. The motion planning problem is then to find $p = f(C_{obst}|c_c, c_g)$ to move the robot from its current configuration c_c, through a sequence of configurations c^t, to its goal configuration c_g, such that $c^t \cap C_{obst} = \emptyset$, $\forall t$ and p is the optimal path from c_c to c_g, according to some metric.

To generalize into dynamic environments, the partition becomes time-dependent, i.e., $C = C^t_{free} \cup C^t_{obst}$, $t \in [1, T]$. The new motion planning problem then becomes $p = f(\{C^t_{obst}\}_{t=1}^T | c_c, c_g)$ such that $c^t \cap C^t_{obst} = \emptyset$, $\forall$ t. For simplicity, we assume the obstacle states C^t_{obst} are known during planning, though this assumption is relaxed in our implementation.

In previous LfH approaches [2]–[4], the "inverse" problem of motion planning, i.e., the hallucination of obstacle configuration space, such that p is optimal, is defined as $\{C^i_{obst}\}_{i=1}^\infty = f^{-1}(p|c_c, c_g)$. Notice that the inverse function is a mapping from a motion plan p to a *set* of obstacle configurations, since multiple obstacle configurations can make p optimal. In most cases, this set is infinitely large.

In Dyna-LfLH, we generalize the previous static hallucination to a dynamic one, i.e., $\{\{C^{t,i}_{obst}\}_{t=1}^T\}_{i=1}^\infty = f^{-1}(p|c_c, c_g)$, that is, generating all possible obstacle configuration *sequences* which make the given motion plan p optimal over the time horizon T. Since it is impossible to produce all (infinite) possible obstacle sequences, we approximate $\{\{C^{t,i}_{obst}\}_{t=1}^T\}_{i=1}^\infty$ using a learned distribution, from which we can numerously sample a large number of obstacle sequences over time horizon T. To be specific, we learn a hallucination function g, which outputs such a distribution:

$$\{C^t_{obst}\}_{t=1}^T \sim g(p|c_c, c_g). \tag{1}$$

B. Approximating $\{C^t_{obst}\}_{t=1}^T$ with Discrete Obstacles

Despite the enormous space of all possible dynamic obstacle sequences $\{C^t_{obst}\}_{t=1}^T$, it is reasonable to assume that they are composed of a few discrete moving obstacles. Therefore, in this paper, we assume $\{C^t_{obst}\}_{t=1}^T$ can be approximated by

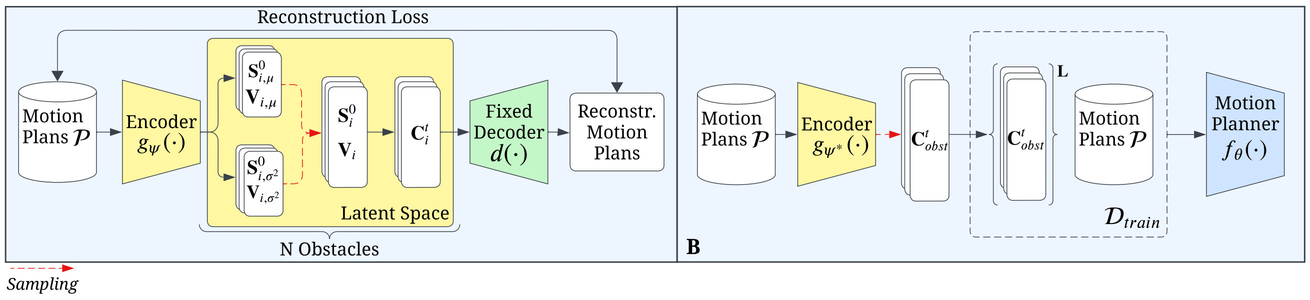
Reconstruction Loss
Motion Plans $\mathcal{P}$
Encoder $g_\psi(\cdot)$
$\mathbf{S}^0_{i,\mu}$ $\mathbf{V}_{i,\mu}$
$\mathbf{S}^0_{i,\sigma^2}$ $\mathbf{V}_{i,\sigma^2}$
$\mathbf{S}^0_i$ $\mathbf{V}_i$
$\mathbf{C}^t_i$
Fixed Decoder $d(\cdot)$
Reconstr. Motion Plans
Latent Space
N Obstacles
Sampling
B
Motion Plans $\mathcal{P}$
Encoder $g_{\psi^*}(\cdot)$
C^t_{obst}
L
C^t_{obst}
Motion Plans $\mathcal{P}$
$\mathcal{D}_{train}$
Motion Planner $f_\theta(\cdot)$

Fig. 2: **A.** The encoder-decoder architecture learns the hallucination function g_ψ (yellow) in a self-supervised manner using past motion plans. The latent space consists of vectors $\mathbf{S}^0$ and $\mathbf{V}$, the N hallucinated obstacles' initial locations and velocities which are sampled from a normal distribution with learned parameters (μ and σ^2). Using $\mathbf{S}^0$ and $\mathbf{V}$, the obstacles C^t are constructed and passed to the fixed, differentiable decoder $d(\cdot)$ (green). $d(\cdot)$ reconstructs a motion plan, $\hat{p}$, that is optimal given C^t. Then, $\hat{p}$ is compared against the original motion plans, p. **B.** Once the hallucination function is trained, we hallucinate and sample S×N dynamic obstacles from g_{ψ^*} that is used to render and create our supervised training set $\mathscr{D}_{train}$. Finally, we train a motion planner $f_\theta(\cdot)$ (dark blue) using a history of L rendered LiDAR scans and our original motion plans, p.

N circular moving obstacles $\{O_i\}_{i=1}^N$ with a fixed radius R at coordinate $\mathbf{C}_i^t = (x_i^t, y_i^t)$ moving in a continuous fashion (obstacles cannot teleport) following first-order dynamics:

$$\mathbf{C}_i^t = \mathbf{S}_i^0 + \mathbf{V}_i \cdot t,\ 1 \le t \le T, \tag{2}$$

where $\mathbf{S}_i^0 = (x_i^0, y_i^0)$ is the starting coordinate of obstacle O_i at $t = 0$, and $\mathbf{V}_i = (v_i^x, v_i^y)$ is its fixed velocity. These assumptions efficiently decompose our original problem of hallucinating the vast space of obstacle configuration sequences, $\{C_{obst}^t\}_{t=1}^T$, into learning a set of structured parameter distributions of (x_i^0, y_i^0), and (v_i^x, v_i^y), sampling from them and forming dynamic obstacles:

$$\{(x_i^0, y_i^0), (v_i^x, v_i^y)\}_{i=1}^N \sim g(p|c_c, c_g). \tag{3}$$

In this work we assume the obstacles only follow first-order dynamics. In future work, it is easy to add complex higher-order dynamics by learning the distributions of acceleration, jerk, snap, and so on. Though for a short time horizon, modeling obstacles to have a constant velocity is sufficient.

C. Learning a Parameterized Hallucination Function

We instantiate the hallucination function g with learnable parameters ψ and learn g_ψ in a self-supervised reconstructive manner using an encoder-decoder structure similar to the static LfLH approach [2]. The encoder $g_\psi(p|c_c, c_g)$ takes the current configuration c_c, goal configuration c_g, and the corresponding plan p as input and produces the probability distributions of x_i^0, y_i^0, v_i^x, and v_i^y for all N dynamic obstacles, as shown in Eqn. (3). We assume x_i^0, y_i^0, v_i^x, and v_i^y are all independent, normally distributed random variables. Then, a potential $\{C_{obst}^t\}_{t=1}^T$ is constructed by applying Eqn. (2) on all N dynamic obstacle parameters sampled from the learned distribution.

The decoder is a 2D classical motion planner without any learnable parameters that is used to generate optimal motion plans $\hat{p}$ from the sampled obstacles. Specifically, $\hat{p} = d(\{C_{obst}^t\}_{t=1}^T \sim g(p|c_c, c_g))$. The goal is to ensure the reconstructed motion plan $\hat{p}$ is the same as the given plan p, indicating p is also optimal for the sampled obstacles $\{C_{obst}^t\}_{t=1}^T$.

Based on a dataset P of past motion plans, either from static/dynamic obstacle environments or completely open spaces, our Dyna-LfLH encoder and decoder find the optimal parameters ψ^* for $g_\psi(\cdot)$ by minimizing a self-supervised loss

$$\psi^* = \underset{\psi}{\operatorname{argmin}} \underset{\substack{p \sim P \\ \hat{p} = d(\{C_{obst}^t\}_{t=1}^T \sim g_\psi(p|c_c, c_g))}}{\mathbb{E}} [\ell(p, \hat{p})], \tag{4}$$

where $\ell(\cdot,\cdot)$ is the reconstruction loss function to encourage the decoder output $d(\{C_{obst}^t\}_{t=1}^T)$ to be similar to the existing motion plan p.

D. Dyna-LfLH

By sampling x_i^0, y_i^0, v_i^x, and v_i^y for all N dynamic obstacles from g_{ψ^*} and constructing $\{C_{obst}^t\}_{t=1}^T$ K times, we can generate a supervised training set for Imitation Learning

$$\mathscr{D}_{train} = \{(\{\{C_{obst}^t\}_{t=1}^T\}^k, p^k, c_c^k, c_g^k)\}_{k=1}^K,$$

with K data points, where p^k is (close to) optimal for $\{\{C_{obst}^t\}_{t=1}^T\}^k$. $\{\{C_{obst}^t\}_{t=1}^T\}^k$ can be transformed into observations in the form of LiDAR scans (with ray tracing) or depth images (with rendering). With the training set $\mathscr{D}_{train}$, a parameterized motion planner $f_\theta(\cdot)$ can be learned by minimizing a supervised learning loss using gradient descent:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underset{\substack{(\{C_{obst}^t\}_{t=1}^T, p, c_c, c_g) \\ \sim \mathscr{D}_{train}}}{\mathbb{E}} [\ell(p, f_\theta(\{C_{obst}^t\}_{t=1}^T | c_c, c_g))]. \tag{5}$$

f_θ will be used to produce motion plans based on the perceived $\{C_{obst}^t\}_{t=1}^T$ during deployment. Notice that during deployment, $\{C_{obst}^t\}_{t=1}^T$ is usually unavailable (unless using explicit future obstacle motion predictors). Therefore, we use the available history $\{C_{obst}^t\}_{t=-L+1}^0$ of length L as inputs to f_θ. Implementation details can be found below.

E. Implementation

As shown in Algorithm 1 line 2, a motion plan $p \in P \subset \mathscr{P}$ consists of a sequence of configurations and linear and angular velocities $\{(x^t, y^t, \text{yaw}^t), (v^t, \omega^t)\}_{t=1}^T$. We instantiate robot configurations in the robot frame so c_c is always $\mathbf{0}$ and therefore ignored. c_g is set to (x^T, y^T, yaw^T), which is included in p and ignored as well. The encoder $g_\psi(\cdot)$ is a network of three one-dimensional convolutional layers followed by fully connected autoregressive layers mapping the input motion plan p to the distribution parameters of the dynamic obstacles' location and velocity (Eqn. 3) in the form of means and log-variances. The decoder d is a re-implementation of Ego-Planner [16] with differential convex optimization layers [17]. The reconstruction loss ℓ in Eqn. 4 is the mean squared error between all positions, linear and angular velocities $\{x^t, y^t, v^t, \omega^t\}_{t=1}^T$ in p and their reconstructed values. To additionally regularize the loss ℓ in Eqn. (4), we impose an obstacle location prior distribution to a normal distribution fitted on all positions $\{(x^t, y^t)\}_{t=1}^T$ in the plan p as an additional loss ℓ_{prior} to prevent obstacles from getting too far away from the plan p. Similarly, an obstacle-obstacle and obstacle-plan collision regularization loss $\ell_{coll} = \sum \max(c - d, 0)^2$ is added, where clearance $c = 0.5$m and d is the distance either between two obstacles or between the obstacle and its closest point on p. We use the same regularization weights as in LfLH [2].

To create $\mathscr{D}_{train}$, $S = 4$ samples of a set of $N = 1$ obstacles are taken from the latent space produced by $g_{\psi^*}(\cdot)$ for every motion plan p (lines 5-14). N dynamic obstacles, $\{\mathbf{C}_i^t\}_{i=1}^N$, $1 \leq t \leq T$, which make motion plan p optimal are constructed using Eqn. (2) and the observations are rendered as 2D LiDAR scans using ray casting given the current robot configuration c^t along the plan p and the corresponding obstacle configuration C_{obst}^t (lines 8-9). $\mathscr{D}_{train}$ is augmented to enhance variability. First, up to five random non-colliding obstacles are added to increase variability in C_{obst}^t. Second, motion plans with velocities over 0.9 m/s are augmented without obstacles to help the model learn fast navigation in open spaces.

In our Dyna-LfLH implementation, $f_{\theta^*}(\cdot)$ (Eqn. (5)) learns to produce one single action u^i given a sequence of $L = 5$ historic LiDAR scans, $\{C_{obst}^t\}_{t=i-L+1}^i$, which comprise a data point in $\mathscr{D}_{train}$ (lines 10-12). Each data point starts at the robot's current configuration $c_c^i = \mathbf{0}$ and the goal c_g^i is a unit vector of a point 2.5 m away on the existing motion plan p.

By taking in a history of L LiDAR scans, the motion planner can implicitly encode and address obstacle dynamics. $f_{\theta^*}(\cdot)$ is modeled as a feed-forward recurrent neural network with two hidden layers, each of size 256 followed by a fully connected layer mapping the hidden layer to linear and angular velocities (line 16). At each time step m during deployment (lines 18-19), $c_c^m = \mathbf{0}$ and c_g^m is instantiated as a unit vector of a point 2.5m away on the global path created using the `move_base` stack.

We use the same Model Predictive Control (MPC) model as LfLH [4] to check for and avoid collisions, with the addition that future LiDAR scans are also simulated based on the two most recent scans. If a collision is imminent, the robot will stop and slowly reverse until no collision is predicted.

IV. Experiments

Dyna-LfLH is implemented on a Clearpath Jackal robot, a four-wheeled, differential-drive, UGV, running the Robot Operating System `move_base` navigation stack. The Jackal has a 720-dimensional, front-facing, 2D LiDAR with a 270° field of view, which is used to instantiate obstacle configuration C_{obst}^t. Dyna-LfLH is used as a local planner. We conduct both simulated and physical experiments to validate our hypothesis that Dyna-LfLH can learn to hallucinate dynamic obstacle configurations where previous motion plans are near-optimal, and agile motion planners can be learned through the learned hallucination.

Algorithm 1 Dyna-LfLH

Input: existing motion plans P, obstacle number N, sampling count S, history sequence length L

```
1:  // Learning Dynamic Hallucination
2:  learn $\psi^*$ for $g_\psi(\cdot)$ with $P$                          ▷ Eqn.(4)
3:  // Dataset Generation
4:  $\mathscr{D}_{train} \leftarrow \emptyset$
5:  for every $p \in P$ do
6:      for $S$ times do
7:          sample $\{(x_i^0, y_i^0), (v_i^x, v_i^y)\}_{i=1}^N$ with $g_{\psi^*}$   ▷ Eqn. 3
8:          create $\{\mathbf{C}_i^t\}_{i=1}^N,\ 1 \leq t \leq T$              ▷ Eqn. 2
9:          render LiDAR scans for $\{C_{obst}^t\}_{t=1}^T$
10:         for every $u^i \in p, L \leq i \leq T-1$ do
11:             $\mathscr{D}_{train} = \mathscr{D}_{train} \cup (\{C_{obst}^t\}_{t=i-L+1}^i, u^i, c_c^i, c_g^i)$
12:         end for
13:     end for
14: end for
15: // Dynamic Learning from Learned Hallucination
16: learn $\theta^*$ for $f_\theta(\cdot)$ with $\mathscr{D}_{train}$               ▷ Eqn. 5
17: // Deployment (each time step $m$)
18: receive $\{C_{obst}^t\}_{t=m-L+1}^m, c_c^m, c_g^m$
19: plan $p = u^m = f_{\theta^*}(\{C_{obst}^t\}_{t=m-L+1}^m \mid c_c^m, c_g^m)$
20: return $p$
```

TABLE I: Key Parameters for DWA and Log-MPPI

DWA	**Linear (x)**	**Angular**
Max Velocity	1.0 m/s	1.57 rad/s
Min Velocity	0.1 m/s	-1.57 rad/s
Acceleration Limit	10.0 m/s^2	20.0 m/s^2
Sampling Resolution	12	40
Simulation Time	2.0 s	
Simulation Granularity	0.02 m	
Log-MPPI	**Linear (x)**	**Angular**
Max Velocity	1.0 m/s	1.5 rad/s
Time Horizon	6.0 s	
Sampling Rate	50 s^{-1}	
Sampled Trajectories	2496	
Exploration Variance	1200.0	
State Dimension	3	
Control Dimension	2	
Cost Function Weights	[2.5, 2.5, 2]	

A. Baselines

Dyna-LfLH is compared with a classical sampling-based motion planner [6], a model predictive path integral (MPPI) controller [7], a state-of-the-art LfH approach [2], and an IL

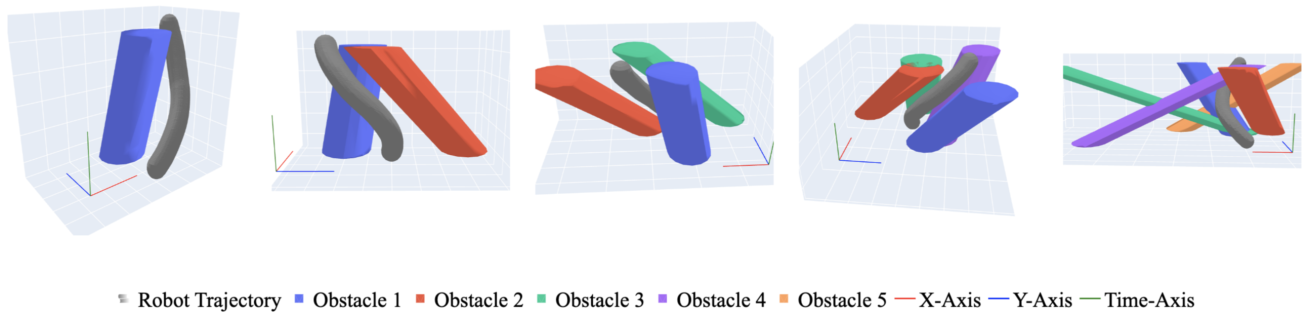

Fig. 3: Example hallucinations of 1, 2, 3, 4, and 5 obstacles respectively. The Z-axis represents time, with the bottom indicating $t = 1$ and the top indicating $t = T$. Robot trajectories are represented by dark gray while obstacles are colored. The robot and the obstacles start at the bottom and move to the top of the graph over time. The steepness of an obstacle is related with its speed. For example, obstacle 1 (blue) is moving slower than obstacle 2 (red) in the second 3D plot. In all cases, the robot trajectories maneuver through the obstacle(s) in a collision-free manner, while the obstacles are generated such that the robot trajectories are near-optimal.

method [8] trained on a large expert dataset [18]. Specifically, the Dynamic Window Approach (DWA) [6] samples actions and evaluates them with a cost function, and Log-MPPI [7] samples from a log-normal mixture distribution and incorporates a 2D costmap. The key parameters for DWA and Log-MPPI are provided in Tab. I. LfLH is trained on 25129 data points (under 10 minutes) of a Jackal robot exploring an open space at 1.0 m/s. After training, it generates 10 static obstacles per data point. Behavior Cloning (BC) [8] is a fully supervised approach using an 8.7-hour expert dataset in dense, dynamic spaces. Dyna-LfLH trains on the same open-space dataset as LfLH (under 10 minutes), generates $N = 1$ dynamic obstacles, and learns a motion planner with different obstacle sequence lengths ($L = 1, 3, 5,$ and 10 previous LiDAR scans). For consistency, all planners have a max linear velocity of 1.0 m/s, the maximum speed in the training data for LfLH and Dyna-LfLH.

B. *Learned Dynamic Hallucination*

In Fig. 3, we present examples of the hallucination results. Five different latent spaces with one to five dynamic obstacles are learned. The corresponding obstacles are sampled from these learned distributions and visualized in a 3D space with the Z-axis representing time. The robot's trajectories successfully navigate through the obstacles without any collision at each time step, while the obstacles are essential to make the robot's maneuvers near-optimal, i.e., the obstacles are the reason why the robot needs to execute such an obstacle avoidance maneuver. Fig. 3 also illustrates that, in most cases, hallucinating just one obstacle is sufficient in explaining the robot's movement, so hallucinating more than one obstacle is unnecessary. Furthermore, the figure shows that the hallucination function learns that both fast and slow-moving obstacles can be avoided using the same motion plan.

C. *Simulation*

We use DynaBARN [19], a simulation testbed for evaluating dynamic obstacle avoidance with a diverse set of 60 environments at different difficulty levels based on the number of obstacles and obstacle motion profiles. For each DynaBARN environment, the robot navigates from one side to the other, and a single collision with any of the obstacles counts as a failure. For every method (DWA, Log-MPPI, LfLH, BC, and Dyna-LfLH), we run 3 navigation trials for each of the 60 environments for a total of 180 trials. In the simulation experiments, the recovery behaviors of all models are turned off to focus only on the main planner's performance. The overall results in terms of success rate in DynaBARN are shown in Fig. 4.

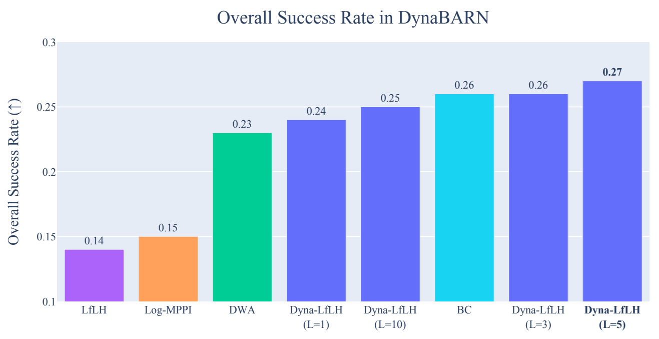

Fig. 4: Simulation Results of 180 trials in DynaBARN. Dyna-LfLH with 5 history scans (L=5) performs the best overall.

The results show that LfLH does not perform well in DynaBARN due to a lack of consideration of obstacle dynamics during hallucination. The state-of-the-art classical planner, Log-MPPI [7], also performs very poorly in this fast-moving and highly cluttered benchmark. DWA performs significantly better than LfLH and Log-MPPI. BC achieves good performance in the simulated DynaBARN after learning on 8.7-hours of expert demonstration data. Dyna-LfLH, trained on 10-minute self-supervised exploration in an open space, achieves a comparable success rate. The Dyna-LfLH with 5 history scans outperforms BC and achieves the best performance across the board.

D. *Physical Experiments*

We also compare the best Dyna-LfLH planner ($L = 5$) with the best planner in each other category, classical (DWA) and IL (BC), in a physical test course, 20 trials each (Fig. 1), with a total of 60 physical trials. We create an enclosed

TABLE II: Physical Experiment Results.

Metrics	DWA	BC	Dyna-LfLH
Success Rate (↑)	0.40	0.15	**0.50**
Avg. TTS (s, ↓)	17.26 ± 8.31	17.67 ± 2.52	**12.60 ± 2.37**

8.2 m × 4.3 m arena with 11 randomly moving obstacles (iRobot Roombas of 0.33 m diameter). The Roombas move with a maximum linear speed of 0.5 m/s with a combination of behaviors such as spiraling outwards, following walls, and bouncing off each other. The success rate and average traversal time on success (TTS) are shown in Tab. II.

Dyna-LfLH achieves the best success rate up to 50%, a 25% improvement over the 2nd best planner, DWA. BC does not work well in the real world and only achieves 15%. We also report the traversal time for each method and Dyna-LfLH achieves the fastest navigation among all successful trials. For safety in the physical experiments, the recovery behaviors of all three methods are turned on.

E. Discussions

Navigating dynamic environments poses a significant challenge for autonomous robots, as evidenced by the low success rates of all evaluated methods (Fig. 4). The simulated DynaBARN environments present particularly difficult scenarios due to the high variability in obstacle motion profiles — individual obstacles frequently change speed and direction, often moving faster than the robot's maximum speed and making sudden, sharp turns. In contrast, the Roombas used in the physical experiments move generally at a constant velocity of 0.5 m/s, which is slower than the robot's maximum speed. This difference in obstacle dynamics makes the physical experiments comparatively easier, contributing to the higher success rates observed in real-world trials.

Nonetheless, our visualization of the learned dynamic obstacles validate our hypothesis that Dyna-LfLH successfully learns to generate dynamic obstacles for which our existing motion plans are optimal. The results from the simulation and physical experiments show that a dynamic motion planner can be learned from such hallucinated data. The learned motion planner also demonstrates generalization to both simulation and the real world.

However, we observe that while theoretically possible to approximate $\{\{C_{obst}^{t,i}\}_{t=1}^{T}\}_{i=1}^{\infty}$ using a learned distribution, in practice, distributions in Eqn. (1) suffer from mode collapse, producing limited samples that fail to approximate the infinitely many obstacle configuration sequences, $\{\{C_{obst}^{t,i}\}_{t=1}^{T}\}_{i=1}^{\infty}$. This likely contributes to the low success rate in Fig. 4. A potential solution is to change the architecture and gradually add variance to the hallucinated obstacles.

V. Conclusions

Dyna-LfLH is a self-supervised method for mobile robot navigation in dynamic environments, capable of navigating among fast-moving, unpredictable obstacles using only past deployment data or data from open spaces. It generates dynamic obstacle configurations to optimize motion plans and provide efficient training data for planners. Both simulated and physical experiments show that a local planner trained with Dyna-LfLH outperforms classical planning methods and supervised approaches that rely on large expert datasets.

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] 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.

[3] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Toward agile maneuvers in highly constrained spaces: Learning from hallucination," *IEEE Robotics and Automation Letters*, pp. 1503–1510, 2021.

[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] 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.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[7] I. S. Mohamed, K. Yin, and L. Liu, "Autonomous navigation of agvs in unknown cluttered environments: Log-mppi control strategy," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 240–10 247, 2022.

[8] A. H. Raj, Z. Hu, H. Karnan, R. Chandra, A. Payandeh, L. Mao, P. Stone, J. Biswas, and X. Xiao, "Rethinking social robot navigation: Leveraging the best of two worlds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[9] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[10] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," *arXiv preprint arXiv:1910.12430*, 2019.

[18] 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.

[19] 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.