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 presents a self-supervised learning method to safely learn a motion planner for ground robots to navigate environments with dense and dynamic obstacles. When facing highly-cluttered, fast-moving, hard-to-predict obstacles, classical motion planners may not be able to keep up with limited onboard computation. For learning-based planners, high-quality demonstrations are difficult to acquire for imitation learning while reinforcement learning becomes inefficient due to the high probability of collision during exploration. To safely and efficiently provide training data, the Learning from Hallucination (LfH) approaches synthesize difficult navigation environments based on past successful navigation experiences in relatively easy or completely open ones, but unfortunately cannot address dynamic obstacles. In our new Dynamic Learning from Learned Hallucination (Dyna-LfLH), we design and learn a novel latent distribution and sample dynamic obstacles from it, so the generated training data can be used to learn a motion planner to navigate in dynamic environments. Dyna-LfLH is evaluated on a ground robot in both simulated and physical environments and achieves up to 25% better success rate compared to baselines.**

I. Introduction

Autonomous mobile robots are becoming an ever-increasing sight in today's society. For example, food delivery robots operate in over 30 locations across the US and Europe in various environments such as universities and city centers [1]. Cargo-carrying robots [2], capable of carrying 40lbs of luggage and following their owners, are now commercially available for purchase. With this trend of mobile robots coexisting with humans, it is essential for those robots to plan their motions to safely and efficiently navigate environments occupied by a large number of dynamic obstacles.

However, the uncertainty of obstacle dynamics, compounded by a large number of obstacles, makes it computationally inefficient for classical navigation systems to draw samples or perform optimization iterations to navigate around such obstacles. Recently, machine learning approaches have been used to successfully maneuver around such obstacles in a data-driven manner [3], [4]. 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.

Learning from Hallucination (LfH) [4]–[7] 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.

[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

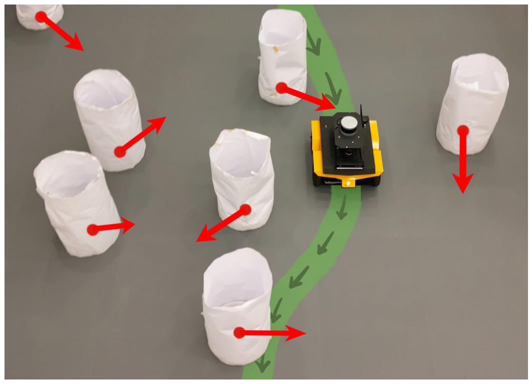
Fig. 1: A mobile robot navigating through a dynamic and dense obstacle field using Dyna-LfLH.

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 time-dependent dynamic obstacle configurations paired with existing optimal motion plans. Augmented with time, 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 [4], a classical sampling-based motion planner [8], and an IL method [9].

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

Though there exist many approaches to tackling motion planning in dynamic environments, two of the most popular ones [10] are Artificial Potential Fields (APF) [11] and velocity-based methods [8]. 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) [8] is a well-known example of a velocity-based method.

Compared with APF and velocity-based classical motion planning algorithms, our proposed Dyna-LfLH approach uses configuration space, which is mainly used within static environments and not in dynamic ones. An advantage of our machine learning approach is that it provides the opportunity of data-driven mobility that can improve with increasing navigation experiences. Furthermore, when used to train a neural planner, 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 [3], such as either applying learning in conjunction with classical methods [12]–[18] or using IL [19], [20] or RL [21]–[23] to learn an end-to-end planner [24], [25]. Going beyond simple obstacle avoidance [26], [27], learning is heavily used in social robot navigation where classical methods fail to calculate trajectories for human-populated spaces [20], [28]–[30]. Another use case is off-road navigation, in which learning can help to reason about the unstructured terrain underneath the robot [31]–[34]. Learning approaches can also directly navigate robots with RGB input alone [35]–[39].

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. Dyna-LfLH is a self-supervised learning approach that can automatically synthesize a variety of training data to address the conundrum of needing to know how a good navigation behavior looks like in order to learn but without knowing how to do it in the first place.

C. Learning from Hallucination (LfH)

LfH [4]–[7] 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* [5], a *minimal* [6], or a *learned* [4] obstacle configuration onto the robot perception. Perceptual hallucination has also been utilized to enable multi-robot navigation in narrow hallways [7].

However, all existing LfH approaches rely on the assumption that the environment is static. 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

The traditional formulation of the robot motion planning problem in environments with static obstacles is based on the idea of configuration space (C-space). The C-space represents the set of all possible robot configurations. In a particular environment, C can be decomposed to $C = C_{free} \cup C_{obst}$, where C_{free} is the set of configurations the robot is free to move in and C_{obst} is the set of unreachable configurations due to obstacles, non-holonomic constraints, etc. A motion plan $p \in \mathscr{P}$ comprises a sequence of actions $p = \{u^t | 0 \leq t \leq T-1\}, u^t \in \mathscr{U}$, where $\mathscr{P}$ is the robot's plan space over time horizon T and $\mathscr{U}$ is it's action space. The motion planning problem is then defined as finding a function $f(\cdot)$ that generates optimal plans $p = f(C_{obst}|c_c, c_g)$ 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 shortest path from c_c to c_g.

To generalize such a static case into a dynamic one, we change the partition $C = C_{free} \cup C_{obst}$ to be time-dependent, i.e., $C = C^t_{free} \cup C^t_{obst}, t \in \mathbb{Z}^+$. Although t can take any possible integers, for our fixed-horizon motion planning problem, we only consider $1 \leq t \leq T$. The new motion planning problem then becomes $p = \{u^t\}_{t=0}^{T-1} = f(\{C^t_{obst}\}_{t=1}^{T}|c_c, c_g)$ such that the robot moves from c_c through c^t towards c_g without intersecting with C^t_{obst} at every t, i.e., $c^t \cap C^t_{obst} = \emptyset, \forall\, 1 \leq t \leq T$. For the convenience of derivation, we assume $\{C^t_{obst}\}_{t=1}^{T}$ is known a priori to produce $p = \{u^t\}_{t=0}^{T-1}$, while we will relax this assumption in our implementation.

In previous LfH approaches [4]–[6], the "inverse" problem of motion planning, i.e., the hallucination of obstacles, 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 static obstacle configurations (not a

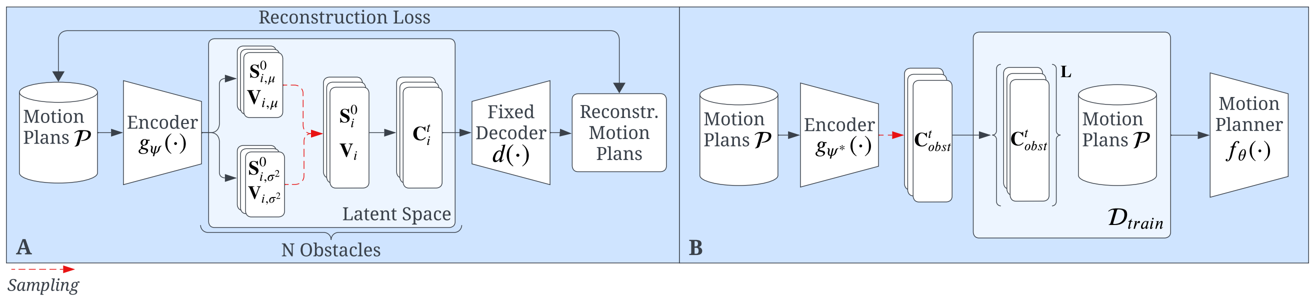

Fig. 2: **A.** The encoder-decoder architecture learns the hallucination function g_ψ in a self-supervised manner using past motion plans. The latent space consists of vectors $\mathbf{S}^0$ and $\mathbf{V}$, the N hallucinated obstacles' initial locations and velocities which are sampled from a normal 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 $d(\cdot)$. $d(\cdot)$ 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$\times$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 $f_\theta(\cdot)$ using a history of L rendered LiDAR scans and our motion plans.

single obstacle configuration), 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 dynamic, time-dependent obstacles, i.e., $\{\{C^{t,i}_{obst}\}^T_{t=1}\}^\infty_{i=1} = f^{-1}(p|c_c, c_g)$, that is, generating all possible dynamic obstacle sequences which make the given motion plan p optimal over the time horizon T. Note that it is impossible to produce all (infinite) possible obstacle sequences. So we approximate $\{\{C^{t,i}_{obst}\}^T_{t=1}\}^\infty_{i=1}$ 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:

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

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

Despite the enormous space of all possible dynamic obstacle sequences $\{C^t_{obst}\}^T_{t=1}$ (Eqn. (1)), it is reasonable to assume that they are composed of a few discrete moving obstacles. Therefore, in this paper, we assume $\{C^t_{obst}\}^T_{t=1}$ can be approximated by N circular moving obstacles $\{O_i\}^N_{i=1}$ with a fixed radius R at coordinate $\mathbf{C}^t_i = (x^t_i, y^t_i)$ moving in a continuous fashion (obstacles cannot teleport) following first-order dynamics:

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

where $\mathbf{S}^0_i = (x^0_i, y^0_i)$ is the starting coordinate of obstacle O_i at $t = 0$, and $\mathbf{V}_i = (v^x_i, v^y_i)$ is its fixed velocity. Those assumptions efficiently decompose our original problem of hallucinating the variety of obstacle sequences $\{C^t_{obst}\}^T_{t=1}$ into learning a set of structured parameter distributions of (x^0_i, y^0_i), and (v^x_i, v^y_i) and sampling from them to comprise the temporal obstacles:

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

Notice that 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.

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 [4]. 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^0_i, y^0_i, v^x_i, and v^y_i for all N dynamic obstacles, as shown in Eqn. (3). We assume x^0_i, y^0_i, v^x_i, and v^y_i all follow a normal distribution. Then, a potential $\{C^t_{obst}\}^T_{t=1}$ is constructed by applying Eqn. (2) on all N dynamic obstacle parameters sampled from the learned distribution. The decoder is a 3D classical motion planner without any learnable parameters that is used to generate optimal motion plans $\hat{p}$ from the sampled obstacles. Specifically, $\hat{p} = d(\{C^t_{obst}\}^T_{t=1} \sim g(p|c_c, c_g))$. 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 $\{C^t_{obst}\}^T_{t=1}$.

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

$$\psi^* = \underset{\psi}{\mathrm{argmin}} \underset{\substack{p \sim P \\ \{C^t_{obst}\}^T_{t=1} \sim g_\psi(p|c_c, c_g)}}{\mathbb{E}} \ell(p, \hat{p}), \tag{4}$$

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

D. *Dyna-LfLH*

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

$$\mathscr{D}_{train} = \{(\{\{C^t_{obst}\}^T_{t=1}\}^k, p^k, c^k_c, c^k_g)\}^K_{k=1},$$

with K data points, where p^k is (close to) optimal for $\{\{C^t_{obst}\}^T_{t=1}\}^k$. $\{\{C^t_{obst}\}^T_{t=1}\}^k$ 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:

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

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

E. Implementation

As shown in Algorithm 1 line 2, in this work, a motion plan $p \in P \subset \mathscr{P}$ consists of a sequence of configurations and linear and angular velocities $\{(x^t, y^t, \text{yaw}^t), (v^t, \omega^t)\}^T_{t=1}$. 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 [40] with differential convex optimization layers [4], [41]. The reconstruction loss ℓ in Eqn. 4 is the mean squared error between all positions, linear and angular velocities $\{x^t, y^t, v^t, \omega^t\}^T_{t=1}$ 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 $\{(x^t, y^t)\}^T_{t=1}$ 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/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 [4].

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 4-13). N dynamic obstacles $\{\mathbf{C}^t_i\}^N_{i=1}, 1 \le t \le T$ 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^t_{obst} (lines 7-8). To increase variation, $\mathscr{D}_{train}$ is augmented in the following ways. First, up to five random non-colliding obstacles are additionally sampled from a uniform distribution within a 3m radius with linear velocity uniformly distributed between $[-\sqrt{2}, +\sqrt{2}]$m/s. To account for motion uncertainty at fast speeds, the obstacles in front of the robot are at distances proportional to the robot's velocity at every point in time on motion plan p. Second, motion plans with a velocity greater than 0.9m/s are augmented with no obstacles so the model also learns to quickly navigate 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, $\{C^t_{obst}\}^i_{t=i-L+1}$, which comprise a data point in $\mathscr{D}_{train}$ (lines 9-11). Each data point starts at the robot's current configuration $c^i_c = \mathbf{0}$ and the goal c^i_g is a unit vector of a point 2.5m 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 15). At each time step m during deployment (lines 17-18), $c^m_c = \mathbf{0}$ and c^m_g 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 [6] to check for and avoid collisions, with the addition that future LiDAR scans are also simulated based on the two most recent scans. The same recovery behaviors are employed when a collision is detected.

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 $\mathscr{P}$                         ▷ Eqn.(4)
 3: $\mathscr{D}_{train} \leftarrow \emptyset$
 4: for every $p \in \mathscr{P}$ do
 5:     for $S$ times do
 6:         sample $\{(x^0_i, y^0_i), (v^x_i, v^y_i)\}^N_{i=1}$ with $g_{\psi^*}$    ▷ Eqn. 3
 7:         create $\{\mathbf{C}^t_i\}^N_{i=1}, 1 \le t \le T$                    ▷ Eqn. 2
 8:         render LiDAR scans for $\{C^t_{obst}\}^T_{t=1}$
 9:         for every $u^i \in p, L \le i \le T-1$ do
10:             $\mathscr{D}_{train} = \mathscr{D}_{train} \cup (\{C^t_{obst}\}^i_{t=i-L+1}, u^i, c^i_c, c^i_g)$
11:         end for
12:     end for
13: end for
14: // Dynamic Learning from Learned Hallucination
15: learn $\theta^*$ for $f_\theta(\cdot)$ with $\mathscr{D}_{train}$              ▷ Eqn. 5

16: // Deployment (each time step $m$)
17: receive $\{C^t_{obst}\}^m_{t=m-L+1}, c^m_c, c^m_g$
18: plan $p = u^m = f_{\theta^*}(\{C^t_{obst}\}^m_{t=m-L+1} \mid c^m_c, c^m_g)$
19: return $p$
```

IV. Experiments

Dyna-LfLH is implemented on a Clearpath Jackal robot, a four-wheeled, differential-drive, unmanned ground vehicle, 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^t_{obst}. 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

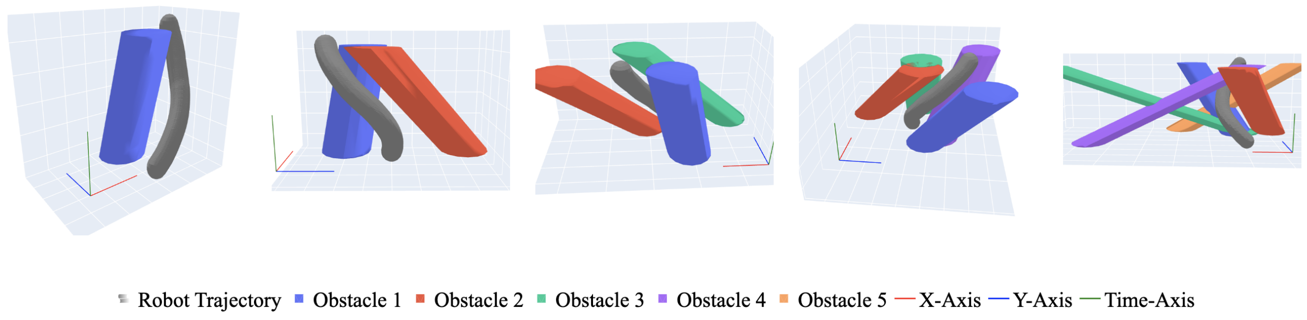

Fig. 3: Example hallucinations of 1, 2, 3, 4, and 5 obstacles respectively. The Z-axis represents the time dimension. Robot trajectories are represented by dark gray while obstacles are colored. The robot and the obstacles start at the bottom and move to the top of the graph over time. In all cases, 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.

previous motion plans are near-optimal, and agile motion planners can be learned through the learned hallucination.

A. Baselines

Dyna-LfLH is compared with a classical sampling-based motion planner [8], a state-of-the-art LfH approach [4], and an IL method [9] trained on a large-scale expert dataset [29]. To be specific, the classical Dynamic Window Approach (DWA) planner [8] samples possible actions and evaluates them based on a pre-defined cost function; LfLH is trained on a dataset of 25129 data points (less than ten minutes) of a Jackal robot randomly exploring in an open space at a maximum speed of 1.0m/s; Behavior Cloning (BC) [9] is a fully supervised approach with a large-scale (8.7 hours) dataset collected from expert human demonstrations in dense and dynamic spaces. Dyna-LfLH uses the same open-space exploration dataset as LfLH (less than ten minutes) for training and we test different obstacle sequence lengths ($L = 1, 3, 5, 10$ previous LiDAR scans). For a fair comparison, the max linear velocity of all the planners are set to 1.0m/s, the maximum speed in the training data, with the classical DWA planner [8] also doubling its sampling rate.

B. Learned Dynamic Hallucination

In Fig. 3, we show a few examples of the hallucination results. Five different latent spaces with one to five dynamic obstacles are learned. Corresponding obstacles are sampled from the learned obstacle distributions and visualized in a 3D space with the Z-axis representing time. The robot trajectories are able to successfully maneuver through the obstacles without any collision for every time step, while the obstacles are also necessary to make the robot maneuvers near-optimal, i.e., the obstacles are the reason why the robot needs to execute such an obstacle avoidance maneuver.

C. Simulation

We use DynaBARN [42], 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 gentle collision with any of the obstacles counts as a failure. For every method (DWA,

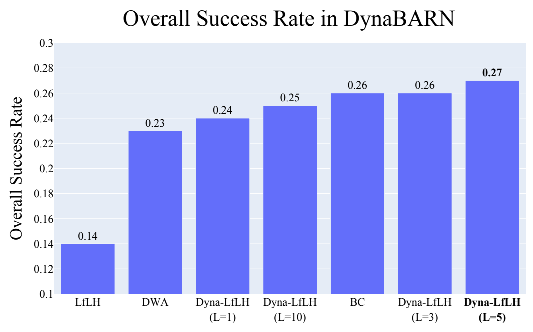

Fig. 4: Simulation Results of 180 trials in DynaBARN.

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. The results show that LfLH does not perform well in DynaBARN due to a lack of consideration of obstacle dynamics during hallucination. DWA performs significantly better than LfLH. BC achieves good performance in the simulated DynaBARN after learning on 8.7-hours of expert demonstration data. However, BC takes almost twice the time on average to reach the goal compared to Dyna-LfLH. Dyna-LfLH, trained on 10-minute self-supervised exploration in an open space, achieves a comparable success rate and faster navigation than BC. 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 8.2×4.3m arena with 11 randomly moving obstacles (iRobot Roombas of a 0.33m diameter). The Roombas move with a maximum linear speed of 0.5m/s with a combination of behaviors such as spiraling outwards, following walls, and bouncing off each other. The results are shown in Tab. I.

Dyna-LfLH achieves the best success rate up to 50%, a 25% improvement over the 2nd best planner, DWA. BC

TABLE I: Physical Experiment Results.

Metrics	DWA	BC	Dyna-LfLH
Success Rate	0.40	0.15	**0.50**
Avg. Traversal Time on Success	17.26±8.31	17.67±2.52	**12.60±2.37**

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 most efficient navigation among all successful trials. For safety in the physical experiments, the recovery behaviors of all three methods are turned on. It should be noted that all three methods heavily rely on their recovery behaviors due to the inherent difficulty of physical experiments.

E. Discussions

Comparing the success rates of different history scans used by Dyna-LfLH in Fig. 4, it can be empirically seen that neither too many ($L = 10$) nor too few ($L = 1$) history scans should be accounted for when predicting the current action. Increasing the number of history scans can be understood as increasing the model complexity which may result in superior performance until the model starts to overfit.

Our visualization of the learned dynamic obstacles and results from the simulation and physical experiments validate our hypothesis that Dyna-LfLH is able to successfully learn to generate dynamic obstacles in which our existing motion plans are optimal. And that a dynamic motion planner can be learned from such hallucinated data. The learned motion planner also shows generalization to both simulation and the real world. However, considering the difficult problem of moving through highly-cluttered, fast-moving, hard-to-predict obstacles, the overall success rate of all the methods, including the best Dyna-LfLH, is still unsatisfactory. How to efficiently hallucinate a diverse set of obstacles, e.g., different sizes and shapes, and how to address out-of-distribution scenarios still remain to be investigated.

V. Conclusions

Dyna-LfLH is a self-supervised machine learning method for mobile robot navigation in dynamic environments. The method can safely learn to navigate among highly-cluttered, fast-moving, hard-to-predict obstacles by only requiring data from existing past deployments or safely collected in open spaces. Dyna-LfLH is capable of generating dynamic obstacle configurations to make existing motion plans optimal and thus safely and efficiently provide training data for a motion planner to learn. As shown in both simulated and physical experiments, a local planner learned from Dyna-LfLH can achieve superior performance compared to existing classical planning methods and supervised learning approaches that require large-scale, high-quality, expert demonstration data.

References

[1] Starship, "Starship technologies," www.starship.xyz, accessed: 4th March 2024.

[2] P. Group, "Gita robots - piaggio fast forward," piaggiofastforward.com/, accessed: 4th March 2024.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[14] 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.

[15] 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.

[16] Z. Wang, X. Xiao, G. Warnell, and P. Stone, "Apple: Adaptive planner parameter learning from evaluative feedback," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7744–7749, 2021.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[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.