2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636402

From Agile Ground to Aerial Navigation: Learning from Learned Hallucination

Zizhao Wang[1], Xuesu Xiao[2], Alexander J Nettekoven[3], Kadhiravan Umasankar[4], Anika Singh[1], Sriram Bommakanti[4], Ufuk Topcu[4], and Peter Stone[2,5]

***Abstract*— This paper presents a self-supervised Learning from Learned Hallucination (LfLH) method to learn fast and reactive motion planners for ground and aerial robots to navigate through highly constrained environments. The recent Learning from Hallucination (LfH) paradigm for autonomous navigation executes motion plans by random exploration in completely safe obstacle-free spaces, uses hand-crafted hallucination techniques to add imaginary obstacles to the robot's perception, and then learns motion planners to navigate in realistic, highly-constrained, dangerous spaces. However, current hand-crafted hallucination techniques need to be tailored for specific robot types (e.g., a differential drive ground vehicle), and use approximations heavily dependent on certain assumptions (e.g., a short planning horizon). In this work, instead of manually designing hallucination functions, LfLH *learns* to hallucinate obstacle configurations, where the motion plans from random exploration in open space are optimal, in a self-supervised manner. LfLH is robust to different robot types and does not make assumptions about the planning horizon. Evaluated in both simulated and physical environments with a ground and an aerial robot, LfLH outperforms or performs comparably to previous hallucination approaches, along with sampling- and optimization-based classical methods.**

I. INTRODUCTION

Although classical navigation systems can safely and reliably move mobile robots from one point to another within obstacle-occupied environments, recent machine learning techniques have demonstrated improvement over their classical counterparts [1], e.g., by learning local planners [2], [3], learning world representation [4], [5], or learning planner parameters [6]–[8]. However, these learning approaches heavily depend on access to high quality training data.

Learning from Hallucination (LfH) [9], [10] is a recently proposed paradigm to address the difficulty of 1) obtaining high-quality training data for traditional Imitation Learning (IL) from expert demonstrations [11] and 2) Reinforcement Learning (RL) from trial-and-error [12]. During LfH training, the robot executes a variety of random motion plans in a completely safe open space, imagines obstacle configurations for which the motion plans are optimal (called *hallucination*), and learns an end-to-end local planner as a mapping from the hallucinated obstacle configurations to the optimal motion plans in open space. The inherent safety of navigating in a completely open training environment allows generation of a large amount of training data with no expert supervision or costly failures during trial-and-error learning. During deployment, learned local planners react to real obstacles within constant time (i.e., querying a pre-trained neural network), which is not dependent on how densely packed the surrounding obstacles are.

However, existing LfH methods require manually designed hallucination functions to generate the most constrained [9] or a minimal [10] obstacle set. While the former requires access to a fine-resolution global path and runtime hallucination during deployment, the latter assumes a short planning horizon to assure that one representative minimal unreachable set can approximately represent all possible minimal unreachable sets. Furthermore, these limitations prevent the manually-designed hallucination functions from extending from simple differential-drive ground vehicles to different robot types, e.g., aerial robots.

The Learning from Learned Hallucination (LfLH) method introduced in this work removes the necessity of manually designing hallucination functions specific to certain robot types and assumptions. In a self-supervised manner, LfLH automatically learns distributions of obstacles which make randomly explored motion plans in open space optimal, samples obstacle configurations from learned obstacle distributions, and finally learns a local planner that maps hallucinated obstacles to optimal motion plans. During deployment, the robot precepts real obstacles and uses the learned local planner to generate motion plans. LfLH is tested on a ground and an aerial robot, both in simulated benchmark testbeds [13] and physical environments. Superior navigation performance is achieved compared to existing LfH approaches [9], [10] and classical sampling-based [14] and optimization-based [15] planners.

II. RELATED WORK

This section reviews classical motion planning and recent machine learning techniques for mobile robot navigation.

A. *Classical Motion Planning*

Classical motion planning techniques for mobile robot navigation mostly work in the robot Configuration Space (C-Space) [16] and mainly comprise two categories: sampling-

[1]Department of Electrical and Computer Engineering {zizhao.wang, anikasingh}@utexas.edu, [2]Computer Science {xiao, pstone}@cs.utexas.edu, [3]Mechanical Engineering nettekoven@utexas.edu, [4]Aerospace Engineering and Engineering Mechanics {kadhirus99, sriram.bommakanti, utopcu}@utexas.edu, University of Texas at Austin, Austin, Texas 78712. [5]Sony AI.

This work has taken place in the Learning Agents Research Group (LARG) and Autonomous System Group at UT Austin, supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243, N00014-20-1-2115), FLI (RFP2-000), ARO (W911NF-19-2-0333), ARL (W911NF2020132), NASA (80NSSC19K0209), DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. 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.

based and optimization-based. Sampling-based motion planners [14] generate sample motion plans and select the best sample based on a certain metric, such as maximum clearance, shortest path, or a combination thereof [16]. Optimization-based planners [15] start with an initial motion plan, then use optimization techniques to iteratively refine the initial plan to avoid obstacles while observing kinodynamic constraints. One common shortcoming of both categories is that when dealing with more constrained obstacle spaces, classical motion planners require increased computation: sampling-based planners require more samples to find a collision-free motion plan to go through all obstacles, while optimization-based planners require more optimization iterations until a feasible plan can satisfy both collision and kinodynamic constraints.

Compared with classical motion planning algorithms, one advantage of the proposed LfLH approach is that its computation is not dependent on obstacle density during deployment, because 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 the classical navigation pipeline in different ways [1], such as constructing a world representation [4], [5], fine-tuning planner parameters [6]–[8], improving navigation performance with experience [2], or enabling social [17] and terrain-aware navigation [3]. Most learning methods require either extensive (RL) or high-quality (IL) training data, such as that derived from trial-and-error exploration or from human demonstrations, respectively.

LfH [9], [10] has been recently proposed to alleviate the difficulty of acquiring extensive or high-quality training data: from random exploration in a completely safe open space with complete safety, motion planners can be learned by synthetically projecting the *most constrained* [9] or augmented *minimal* [10] C-space onto the robot perception. Through carefully designed hallucination functions, these methods have shown fast and agile maneuvers on ground robots compared to classical motion planning and traditional learning approaches. However, the design of specific hallucination functions does not easily extend to other robot types (e.g., aerial robots [9]) and relies on specific assumptions (e.g., a short motion plan/planning horizon to make approximated hallucination valid [10]).

LfLH removes the requirement for a carefully designed hallucination function tailored to a specific robot with strict assumptions, and instead *learns* hallucinated obstacle distributions which assure the motion plans executed in open space are optimal in a self-supervised manner. Sample obstacle configurations can be drawn from the learned obstacle distributions as training data to learn a motion planner.

III. APPROACH

In this section, we present our Learning from Learned Hallucination (LfLH) approach. We first formulate the problem using the LfH framework, then present the proposed approach to *learn* (instead of manually design) a hallucination function, from which a motion planner is finally learned, as shown in Fig. 1.

A. Problem Definition

We adopt the same notation used by Xiao *et al.* to formalize LfH [9] and Hallucinated Learning and Sober Deployment (HLSD) [10]: given a robot's C-space partitioned by unreachable (obstacle) and reachable (free) configurations, $C = C_{obst} \cup C_{free}$, the classical motion planning problem is to find a function $f(\cdot)$ that can be used to produce optimal plans $p = f(C_{obst} \mid c_c, c_g)$ that result in the robot moving from the robot's current configuration c_c to a specified goal configuration c_g without intersecting (the interior of) C_{obst}. Here, a plan $p \in \mathscr{P}$ comprises a sequence of actions $\{u_i\}_{i=1}^{t}$ ($u_i \in \mathscr{U}$, $\mathscr{P}$ and $\mathscr{U}$ are the robot's plan and action space, respectively). Considering the inverse problem of finding $f(\cdot)$, LfH [9] and HLSD [10] use hallucination functions denoted as $g(p \mid c_c, c_g)$, to generate the (unique) *most constrained* and a (not unique) *minimal* obstacle set, respectively, to make a motion plan p generated by a random policy π_{rand} in open space optimal. To instantiate these hallucination functions, hand-crafted rules are designed for specific robot types (e.g. differential drive robots) and do not easily extend to others. LfH [9] further requires a fine-resolution global path and a runtime hallucination function $h(\cdot)$ to augment the real obstacle perceptions to the most constrained cases during deployment. HLSD [10] uses one representative minimal unreachable set to approximately represent all of them. This approximation is accurate only for short planning horizons or motion plans, either requiring frequent replanning or limiting navigation speed at runtime.

LfLH aims to learn a parameterized hallucination function $g_\psi(\cdot)$, which outputs probability distributions of obstacles, in a self-supervised manner, without the need to manually design hallucination functions for each robot type and thus avoiding the subsequent problems described above. Then it samples many times from this learned distribution, $C_{obst} \sim g_\psi(\cdot)$, to generate many obstacle configurations, in which the free-exploration motion plans in open space are close to optimal.

B. Learning Hallucination

We adopt an encoder-decoder architecture to learn the hallucination function $g_\psi(p \mid c_c, c_g)$ parameterized by ψ. Taking the current configuration c_c, goal configuration c_g, and the corresponding plan p as input, the encoder $g_\psi(\cdot)$ generates probability distributions of obstacles. We assume obstacles are ellipses (or ellipsoids), and thus the obstacle distributions are modeled as normal distributions of obstacle locations and sizes in the C-Space. To shape the obstacle distributions such that the given plan p is optimal, LfLH uses a classical motion planner which does not have learnable parameters as the decoder $d(C_{obst} \sim g_\psi(p \mid c_c, c_g))$. The decoder $d(\cdot)$ samples from the obstacle distributions and then computes the optimal motion plan for the sampled obstacles using its built-in algorithms. If the reconstructed motion plan is the same as the given plan p, it indicates p is also optimal for the sampled obstacles. In this work, we

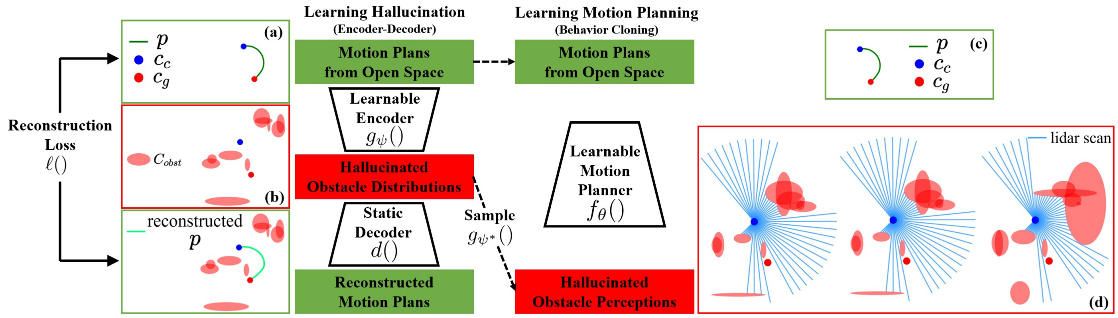

Fig. 1: The Encoder-Decoder architecture learns hallucination function g_ψ from motion plans in open space (a) to obstacle distributions (b). Sampling from hallucinated obstacle distributions and rendering corresponding observations (d), motion planner f_θ is learned with Behavior Cloning using motion plans collected in the open space as ground truth (c). During deployment, f_θ takes in real obstacle perceptions and generates corresponding motion plans.

choose $d(\cdot)$ to be a differentiable optimization-based motion planner for easy training. To use non-differentiable motion planners, one can use approximate gradients algorithms, like REINFORCE [18]. Overall, our encoder-decoder architecture uses gradient descent to find the optimal parameters ψ^* for $g_\psi(\cdot)$ by minimizing a self-supervised loss using a dataset P of motion plans collected using random exploration in an open space:

$$\psi^* = \underset{\psi}{\operatorname{argmin}} \underset{\substack{p \sim P \\ C_{obst} \sim g_\psi(p|c_c,c_g)}}{\mathbb{E}} \big[\ell(p, d(C_{obst})) + \ell_r(C_{obst}, p)\big]. \tag{1}$$

The choice of the reconstruction loss function $\ell(\cdot,\cdot)$ depends on the representation of a motion plan p (e.g., a sequence of raw motor commands or a B-spline trajectory). In addition to the main reconstruction loss $\ell(\cdot,\cdot)$, a regulization loss ℓ_r is employed to stablize training,

$$\ell_r(C_{obst}, p) = \lambda_1 \ell_{prior}(C_{obst}) + \lambda_2 \ell_{coll}(C_{obst}, p), \tag{2}$$

where ℓ_{prior} prevents g_ψ from overfitting by encouraging a larger probability that C_{obst} is sampled from a prior distribution (i.e., minimizing the discrepancy between the output of g_ψ and the prior), ℓ_{coll} is the penalty for collision between obstacles C_{obst} and plan p as well as between each pair of obstacles in C_{obst}, and λ_1, λ_2 are corresponding regularization weights. Implementation details of ℓ, ℓ_{prior}, and ℓ_{coll} can be found in Sec. IV.

After learning $g_{\psi^*}(\cdot)$ in a self-supervised manner, we can sample from the learned obstacle distributions $g_{\psi^*}(p \mid c_c, c_g)$ many times to generate many obstacle configurations C_{obst} where the motion plan p is close to optimal. We then form a training set $\mathscr{D}_{train}$ with individual data points (C_{obst}, p, c_c, c_g). The specific instantiation of C_{obst} depends on the perception modality of the mobile robot. For example, for ground robots with 2D LiDAR, we use 2D ray casting from the onboard sensor to the sampled obstacles to determine range readings for each laser beam; for aerial robots with 3D depth cameras, we use 3D rendering to determine each camera pixel's depth value to the sampled obstacles.

C. Learning from Learned Hallucination

With the training set $\mathscr{D}_{train}$ constructed using the learned hallucination function $g_{\psi^*}(\cdot)$, we learn a parameterized motion planner $f_\theta(\cdot)$ from $\mathscr{D}_{train}$ by minimizing a supervised learning loss using gradient descent:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underset{(C_{obst}, p, c_c, c_g) \sim \mathscr{D}_{train}}{\mathbb{E}} \big[\ell(p, f_\theta(C_{obst} \mid c_c, c_g))\big]. \tag{3}$$

The loss function $\ell(\cdot,\cdot)$ in Eqns. 1 and 3 could take the same form. But in practice, the encoder $g_\psi(\cdot)$'s input plan (Eqn. 1) and the motion planner $f_\theta(\cdot)$'s output plan (Eqn. 3) can take different forms to faciliate learning. Details can be found in Sec. IV. During deployment, the learned f_{θ^*} uses real obstacle perceptions rather than hallucinated ones to generate effective motion plans.

The entire learning pipeline is shown in Alg. 1, including data collection, learning hallucination, learning from learned hallucination, and deployment.

IV. EXPERIMENTS

LfLH is implemented on a ground and an aerial robot to validate our hypothesis that LfLH can automatically learn obstacle configurations where motion plans executed in open

Algorithm 1 Learning from Learned Hallucination

Input: π_{rand}, $g_\psi(\cdot)$, *sampling count*, $f_\theta(\cdot)$

```
// Data Collection
collect motion plans (p, c_c, c_g) in free space with π_rand
    to form motion plan dataset P
// Learning Hallucination
learn ψ* using Eqn. 1 for g_ψ(·) with P
D_train ← ∅
for every (p, c_c, c_g) in P do
    for sampling count times do
        sample C_obst ~ g_ψ*(p | c_c, c_g)
        D_train = D_train ∪ (C_obst, p, c_c, c_g)
    end for
end for
// Learning from Learned Hallucination
learn θ* using Eqn. 3 for f_θ(·) with D_train
```

```
// Deployment (each time step)
receive C_obst, c_c, c_g
plan p = {u_i}_{i=1}^t = f_θ*(C_obst | c_c, c_g)
return p
```

space are near-optimal, and agile motion planners can be learned through the learned halluciantion.

A. Ground Robot

We first implement LfLH on a ground robot and compare LfLH's performance with a classical sampling-based motion planner [14] and state-of-the-art learning approaches from hallucination, including LfH [9] and HLSD [10].

1) Implementation: We use a Clearpath Jackal robot, a four-wheeled, differential-drive, Unmanned Ground Vehicle (UGV), running the Robot Operating System `move_base` navigation stack. The Jackal has a 720-dimensional 2D LiDAR with a 270° field of view, which is used to instantiate obstacle configuration C_{obst} using ray casting. Its DWA local planner is replaced with LfLH.

For data collection (line 2 in Alg. 1), we collect three separate datasets: a dataset with mostly constant 0.4m/s linear velocity (v) and varying angular velocity (ω); a varying v and ω dataset with 1.0m/s max v; and a varying v and ω dataset with 2.0m/s max v. To be specific, π_{rand} samples a target $(\hat{v}, \hat{\omega})$ pair at random according to the max limit and commands the UGV to reach that speed with constant increments/decrements considering its acceleration limit. The most constrained hallucination in LfH [9] only works with the 0.4m/s dataset, while being confused by the ambiguity of varying v in 1.0m/s and 2.0m/s datasets in the most constrained spaces. The minimal hallucination in HLSD [10] works with both the 0.4m/s and 1.0m/s datasets, but does not learn well from the 2.0m/s dataset because the approximation of the representative minimal unreachable set used in HLSD fails to represent all minimal sets in the long trajectory produced by the faster speed. Only LfLH works well with all datasets.

For the UGV, a single odometry point o_i consists of position x_i, orientation ϕ_i, linear velocity v_i, and angular velocity ω_i. We use $N = 125$ odometry points to compose a motion plan p that takes 2.5s and train the hallucination function $g_{\psi^*}(\cdot)$ with Eqn. 1 (line 4 in Alg. 1). The encoder $g_{\psi}(\cdot)$ is modeled as a network with three one-dimensional convolutional layers to extract temporal features and a fully-connected layer to map temporal features to ten obstacles' location and size distributions, in the form of means and variances. The decoder d is Ego-Planner [15], an optimization-based planning algorithm which we reimplement with differential convex optimization layers [19] to enable differentiation through d itself. The reconstruction loss ℓ in Eqn. 1 is the mean squared error of all positions and linear velocities $\{x_i, v_i\}_{i=1}^N$ in p and their reconstructed values. For regularization, the obstacle location prior distribution is a normal distribution fitted on all positions $\{x_i\}_{i=1}^N$ in the plan p, to prevent obstacles from getting too far away from the plan p. Meanwhile, the obstacle size prior is a normal distribution with an empirically chosen mean of 0.3m and variance of 0.0025m^2. The obstacle-obstacle/obstacle-plan collision regularization loss is $\ell_{coll} = \sum \max(c-d, 0)^2$ where clearance $c =$0.5m and d is the distance either between two obstacles or between the obstacle and its closet point on the plan p. Regularization weights are tuned with grid search as $\lambda_1 = 0.3$ and $\lambda_2 = 2.0$.

After training, we construct the training set $\mathscr{D}_{train}$ by sampling ten obstacle configurations from $g_{\psi^*}(\cdot)$ for each collected plan. Beyond the ten hallucinated obstacles, to increase the variance of training samples, we randomly sample five additional obstacles whose distances to p are proportional to the velocity to account for motion uncertainty at fast speeds. To make sure the training samples are valid, we filter out invalid samples where obstacles are too close to the path p (clearance $c < 0.5$m). Finally, the observations, i.e., 2D Lidar scans, are rendered using ray casting given the UGV's configuration c_c and obstacle configurations C_{obst}. The motion planner f_{θ^*} is trained to produce only the first action (v_1, ω_1) in the entire motion plan p for simplicity (line 13 in Alg. 1), and it is represented as a fully-connected network with 2 hidden layers of 256 units. During deployment, the UGV reasons in the robot frame, so c_c is the origin and c_g is a point 1.5m away from the robot on the global path (line 15 in Alg. 1). We use the same Model Predictive Control (MPC) model as HLSD [10] to check for and avoid collisions.

2) Simulated Experiments: We first use the BARN dataset [13] with 300 navigation environments (example environments shown in Fig. 2) randomly generated by Cellular Automata to compare the different motion planners. As a classical sampling-based planner, DWA's [14] max linear velocity is increased from the default 0.5m/s to 2.0m/s for a fair comparison with other planners. We find that by also quadrupling DWA's default sampling rate for linear and angular velocity (to 24 and 80), the UGV's performance is roughly the same as when using the default parameters (but at quadruple the speed). We train nine different planners with LfH [9], HLSD [10], and LfLH on all three datasets. For all the planners, we run three navigation trials in each of the 300 navigation environments in BARN between a specified start and goal location without a map and record the traversal time (with 50s maximum).

We list the average traversal time with the standard deviation in Tab. I. The ∞ sign indicates the motion planner learned using the corresponding method and dataset fails to navigate to the goal in most of the trials (getting stuck or colliding). For the other entries, a low traversal time means efficient and agile navigation performance. LfLH is the only learning method that works well with all three datasets, and achieves the best navigation performance. For clarity, we only plot the test results in each environment (averaged over three trials) using each variant with its fastest working dataset, i.e., LfH 0.4, HLSD 1.0, and LfLH 2.0, along with DWA 2.0 in Fig. 3 (scattered dots). We also fit a line to show the trend of each method. In Fig. 3 we order the navigation environments from left to right with increasing average traversal time achieved by DWA 2.0 (red). Note that the y axis uses a log scale to better visualize the differences in the small traversal time range. LfLH 2.0 (green) achieves similar results in easy environments (left) and significantly outperforms DWA 2.0 in difficult ones (right). Although LfH

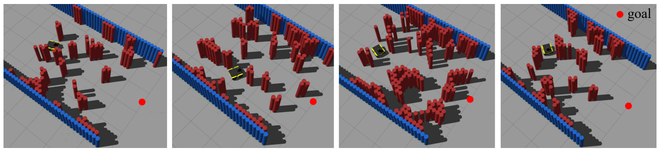

Fig. 2: Four example BARN environments

TABLE I: UGV Simulated Average Traversal Time

DWA 2.0	Dataset	LfH	HLSD	**LfLH**
22.1±11.4s	0.4m/s	13.8±5.3s	13.2±7.9s	13.4±6.4s
	1.0m/s	∞	8.5±5.2s	8.3±3.8s
	2.0m/s	∞	∞	**8.1**±5.4s

0.4 (orange) and HLSD 1.0 (yellow) have a disadvantage due to slow max speed in easy environments, they exhibit good agility in more constrained difficult ones, especially HLSD 1.0. LfLH 2.0 is the only planner that achieves good performance in both easy and difficult environments, shown by the flat green line. Note that the maximal velocity of the UGV is 2.0m/s and the highly constrained BARN environments require slow speeds for agile maneuvering in most places.

3) Physical Experiments: We also deploy the four planners in a physical test course, for five trials each (Fig. 4). The results are shown in Tab. II. The sampling-based DWA planner [14] fails to sample feasible motions in many constrained spaces, and has to execute many recovery behaviors before re-sampling. Therefore DWA takes a long average time with large variance to traverse the course. LfH 0.4 requires a fine-resolution global path [9] and drives smoothly but slowly everywhere along the course. Being too conservative in wide open spaces causes LfH 0.4 to achieve similar results to DWA 2.0, which gets stuck in many places but makes up the time by accelerating in open spaces. HLSD 1.0 also navigates smoothly, but much faster than LfH 0.4, and outperforms DWA 2.0. Our LfLH 2.0 is the only planner that can learn from the fast 2.0m/s dataset, achieving the best performance among all the variants.

B. Aerial Robot

We also evaluate LfLH on an Unmanned Aerial Vehicle (UAV, a quadrotor) where hallucination is more challenging

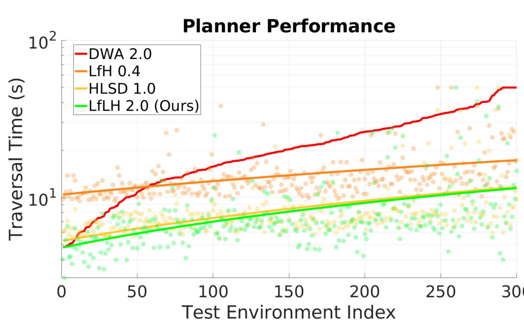

Fig. 3: Simulation results in BARN

TABLE II: UGV Physical Average Traversal Time

DWA 2.0	LfH 0.4	HLSD 1.0	**LfLH 2.0**
73.6±3.8s	78.4±1.8s	50.6±0.8s	**41.1**±0.9s

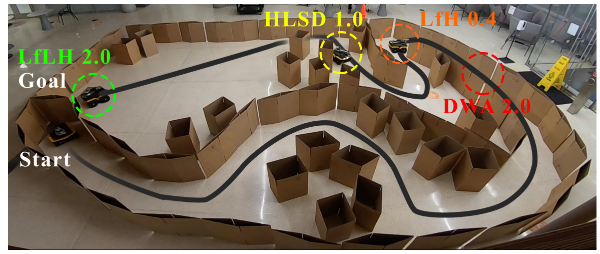

Fig. 4: UGV Physical Experiments: Progress of HLSD 1.0 (yellow), LfH 0.4 (orange), and DWA 2.0 (red) in the obstacle course, when LfLH 2.0 (green) reaches the goal.

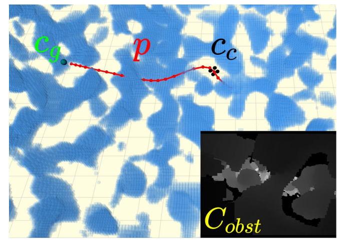

Fig. 5: **Left**: In Ego-Planner's simulator, the obstacles are shown as blue clouds, the goal is marked green (c_g), and the produced motion plan p is marked red. The bottom right is the depth image as an instantiation of C_{obst}.

due to higher dimensionality and more agility. We compare LfLH with a state-of-the-art optimization-based UAV trajectory planning algorithm, Ego-Planner [15].

1) Implementation: We apply LfLH to a simulated UAV in Ego-Planner's simulator and a physical PX4 Vision UAV platform shown in Fig. 5. Both the simulated and physical UAVs use depth input, acquired by rendering in simulation and a Structure Core camera, respectively, to instantiate obstacle configuration C_{obst}.

Due to unavailability of a motion capture system, we collect a dataset of 20-min flight in simulation (line 2 in Alg. 1). We represent the random exploration policy π_{rand} to collect p in an open space using Ego-Planner with 2.0m/s max v in a highly constrained environment. We only record motion plans p but do not record any perception input (as if the UAV were flying in an open space),[1] which is later synthesized by LfLH. Note that previous learning approaches from hallucination cannot solve this 3D aerial navigation task: LfH [9] is not applicable because no global path is available; The approximated minimal unreachable set in HLSD [10] cannot effectively represent 3D obstacles given the long flight plan.

For our LfLH implementation in 3D, the odometry point contains the same information $(x_i, \phi_i, v_i, \omega_i)$ as the UGV (but in 3D), and each plan p consists of $M = 500$ odometry points that takes 2.5s. The encoder g_ψ, decoder d, reconstruction loss ℓ, and regularization ℓ_r are in the same format as for the UGV, except that $g_\psi(\cdot)$ outputs distributions of 15 obstacles, the obstacle size prior has a different mean of 0.6m and variance of $0.2m^2$ in the prior regularization loss ℓ_{prior}, the clearance increases to $c = 0.6$m in the collision regularization loss $\ell_{collsion}$, and $\lambda_1 = 0.5, \lambda_2 = 5.0$ for regularization weights.

[1]We leave a true random exploration policy in an open space as future work, such as random teleoperation or a random policy as in the UGV case.

TABLE III: UAV Simulated Evaluation Results

Metrics	Ego-Planner	LfLH
Survival Time (s)	101.99±62.83	**192.87**±161.37
Survival Distance (m)	174.15±106.74	**213.07**±172.98
SPL	**0.74**	0.56

After training $g_\psi(\cdot)$ (line 4 in Alg. 1), the training set $\mathscr{D}_{train}$ is constructed with the collected plans p and their corresponding observed C_{obst}, i.e., depth images rendered using ray casting given the 15 sampled hallucinated obstacles and five additional obstacles sampled in the same way as the UGV in Sec. IV-A.1. The motion planer f_θ is modeled as four convolutional layers and three fully-connected layers used to extract features from the depth image, goal configuration, and current velocities (h, c_g, v, ω). Then f_θ is trained to produce positions and linear velocities $\{\hat{x}_i, \hat{v}_i\}_{i=1}^M$ in the entire motion plan p. A similar MPC method to HLSD [10] used for ground navigation is also applied to check for collisions by the UAV.

2) Simulated Environments: We first evaluate LfH and Ego-Planner in simulation with a randomly generated forest shown in Fig. 5 left. Each method is tested for 10 trials, and in each trial the UAV keeps navigating to randomly generated goals until the UAV collides with obstacles. Meanwhile, we record the total traversal time and distance, as well as the individual traversal time t_i and distance p_i to each goal i, from which we measure Success weighted by Path Length [20], SPL $= \frac{1}{K}\sum_{i=1}^{K} S_i \frac{l_i}{p_i}$, where S_i is the binary indicator of success for reaching the ith goal, and l_i is the Euclidean distance from the ith start to the ith goal, which is always smaller than p_i.

We list the average and standard deviation (if applicable) of these metrics in Tab. III. Compared with Ego-Planner, LfLH survives (keeps navigating in a collision-free manner) longer both in terms of traversal time and distance, but it has lower SPL. In other words, LfLH trades off aggressive motions for safety. With safer motion plans, it can react faster to unexpected obstacles in the highly constrained environments (e.g., obstacles that are occluded until the robot gets pretty close). To the best of our knowledge, LfLH is the first learning-based planner for aerial robots that can navigate in such highly constrained spaces.

V. CONCLUSIONS

LfLH is a self-supervised machine learning technique for mobile robot navigation that only requires data in open space. In addition to self-supervised learning of a motion planner, LfLH also *learns* self-supervised to generate hallucinated obstacle configurations, from which the motion planner is learned, instead of requring hand-crafted hallucination functions. In contrast to the manually designed ones [9], [10], LfLH is robust to different robot types and can be applied to agile ground and aerial robots navigating at faster speeds. Although in our experiments the ground robot learns from a real random exploration policy, one future research direction is to investigate truly random exploration for aerial vehicles, instead of using trajectories collected from an existing motion planner. In addition to a physical demonstration, other interesting directions include extending LfLH to dynamic obstacles and designing better exploration strategies to cover necessary navigation skills for all possible obstacle configurations.

REFERENCES

[1] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion control for mobile robot navigation using machine learning: a survey," *arXiv preprint arXiv:2011.13112*, 2020.

[2] 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.

[3] 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.

[4] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment, "Robot navigation from human demonstration: Learning control behaviors," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1150–1157.

[5] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," 2017.

[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, 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.

[10] 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.

[11] 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.

[12] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[13] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking metric ground navigation," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020, pp. 116–121.

[14] 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.

[15] 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.

[16] S. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[17] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

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

[20] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.