In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation* (ICRA 2021), Xi'an, China, June 2021

# Agile Robot Navigation through Hallucinated Learning and Sober Deployment

Xuesu Xiao[1], Bo Liu[1], and Peter Stone[1,2]

***Abstract*— Learning from Hallucination (LfH) is a recent machine learning paradigm for autonomous navigation, which uses training data collected in completely safe environments and adds numerous imaginary obstacles to make the environment densely constrained, to learn navigation planners that produce feasible navigation even in highly constrained (more dangerous) spaces. However, LfH requires hallucinating the robot perception during deployment to match with the hallucinated training data, which creates a need for sometimes-infeasible prior knowledge and tends to generate very conservative planning. In this work, we propose a new LfH paradigm that does not require runtime hallucination—a feature we call "sober deployment"—and can therefore adapt to more realistic navigation scenarios. This novel Hallucinated Learning and Sober Deployment (HLSD) paradigm is tested in a benchmark testbed of 300 simulated navigation environments with a wide range of difficulty levels, and in the real-world. In most cases, HLSD outperforms both the original LfH method and a classical navigation planner.**

## I. INTRODUCTION

Machine learning techniques have been recently applied to mobile robot navigation to develop robots that are capable of moving from one point to another within obstacle-occupied environments in a collision-free manner [1]–[7]. Besides classical planning methods [8], [9], machine learning approaches can produce effective planners from data instead of hand-crafted rules and heuristics.

Among the thrust of learning to navigate, Imitation Learning (IL) [1] and Reinforcement Learning (RL) [3] are the two main streams. While the former requires expert demonstration, the latter learns from trial-and-error. Their initial successes indicate a promising potential future for these data-driven approaches, which do not require sophisticated engineering and in-situ adjustment [5], [10]. However, most learning approaches require a large amount of training data in order to produce good navigation behaviors, especially in challenging unseen environments.

Learning from Hallucination (LfH) [6] is a recently proposed paradigm to address the difficulty of obtaining high-quality training data. Using LfH, the robot collects training data in an obstacle-free, and thus completely safe, environment with a random exploration policy. During training, the most constrained surrounding obstacle configuration is synthetically projected onto the robot's perception, which allows the effective action taken by the robot in the open space to be the only feasible, and therefore optimal, action. A control policy is learned with training data as if the robot had been moving in those constrained spaces. Thanks to the inherent safety of navigating in a completely open training environment, the robot can autonomously generate a large amount of training data with no human supervision or any costly failure during trial-and-error learning.

However, one major drawback of LfH [6] is that the perception also needs to be hallucinated during deployment, with the help of a fine-resolution reference path (prior knowledge that is sometimes infeasible to obtain), and it requires other modules to address out-of-distribution scenarios. This runtime hallucination adds extra computation to the perception and becomes ineffective when only sparse future waypoints are available. Furthermore, hallucinating to be always in the most constrained environment during deployment causes the planner to become unnecessarily conservative and fail to adapt to some realistic situations, e.g., an actual open space.

The Hallucinated Learning and Sober Deployment (HLSD) approach proposed in this work eliminates the necessity of hallucination during deployment and allows the robot to perceive its actual surroundings. This "sober deployment" relaxes the requirement for a high-resolution reference path and enables the robot to adapt to the real deployment environment. Through a novel hallucination strategy during training, the robot is able to learn from many obstacle configurations as augmentations sampled in addition to the *minimal unreachable set*, which is the *smallest* set of obstacles required to cause the actions performed in the obstacle-free training environment to be optimal, given a specific goal. Our simulated experiment on 300 benchmark testbeds [11] and our real-world experiment using a physical robot show that after seeing an extensive amount of carefully designed hallucinated training data, the robot is able to efficiently produce agile maneuvers without runtime hallucination. Superior navigation performance is achieved compared to the original LfH approach with extra access to a high-resolution global path and runtime hallucination, and also to a classical navigation planner.

## II. RELATED WORK

This section presents related work in mobile robot navigation, using classical motion planning and recent machine learning techniques.

### A. Classical Motion Planning

In terms of classical motion planning, mobile robot navigation is the problem of moving a robot from one point

[1]Xuesu Xiao, Bo Liu, and Peter Stone are with Department of Computer Science, University of Texas at Austin, Austin, TX 78712 {xiao, bliu, pstone}@cs.utexas.edu. [2]Peter Stone is with Sony AI. This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARO (W911NF-19-2-0333), 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.

to another within an obstacle-occupied space in a collision-free manner. Such planning usually happens in the robot Configuration Space (C-Space) [12] and produces (asymptotically) optimal motion plans based on a predefined metric, such as maximum clearance, shortest path, or a combination thereof [13]. These metrics are optimized in terms of C-space representation (e.g. cellular decomposition [13]), global planning (e.g. Dijkstra's), and local planning. For example, the optimization-based Elastic Bands (E-Band) planner [8] uses repulsive forces generated from obstacles to deform and optimize an initial trajectory, primarily to achieve maximum clearance. The Dynamic Window Approach (DWA) [9] samples feasible actions and scores them based on a weighted score of distance to obstacles, closeness to a global path, and progress toward the goal. The new hallucination strategy proposed in this paper assumes the planner learned from hallucinated training data primarily seeks the shortest path. We leave hallucination strategies that optimize for maximum clearance for future work.

### B. Machine Learning for Navigation

Despite decades of effort to develop classical autonomous navigation systems [8], [9], machine learning has recently shown promise for creating competitive end-to-end planners with obstacle avoidance [1], enabling terrain-based navigation [4], [14], allowing robots to move around humans [2], and tuning parameters for classical navigation systems [5], [15], [16]. All these learning methods require either extensive or high-quality training data, such as that derived from trial-and-error exploration or from human demonstrations [17].

To address the difficulty in obtaining extensive or high-quality training data, self-supervised LfH [6] collects training data in an obstacle-free environment with complete safety, and then learns a local navigation policy through synthetically projecting the most constrained C-space ($C^*_{obst}$) onto the robot perception. The most constrained C-space corresponds to the obstacle configuration with as many obstacles as possible such that the executed motion is the *only* feasible and therefore *optimal* motion plan: if any more obstacles were to be added, then the motion plan would no longer be feasible. However, LfH also requires hallucination of $C^*_{obst}$ during deployment. This hallucination "on-the-fly" requirement assumes prior knowledge such as a relatively high-resolution global path is available. Furthermore, this previous work [6] only works well when the linear velocity in the training set is mostly constant, due to the fact that the robot always hallucinates navigating in the most constrained scenarios and the trained local planner is therefore relatively conservative. Varying speeds in the same most constrained space will lead to ambiguity, as shown in our experiments (Section IV). In this work, we eliminate the necessity of hallucination "on-the-fly", and therefore the requirement of an available high-resolution global path. During sober deployment, the local planner only needs a single local goal point, along with the real, rather than hallucinated, perception. In addition, the novel hallucination technique also teaches the robot to vary its speed in response to the real environment, instead of the hallucinated most constrained spaces.

## III. APPROACH

In this section, we present our Hallucinated Learning and Sober Deployment (HLSD) technique for mobile robot navigation. In Sec. III-A, we start with a simplified example, which assumes the robot to be a point mass that follows a relatively simple path, to introduce the idea of a minimal unreachable set, $C^{min}_{obst}$, for a given optimal plan. We provide necessary conditions for being a minimal unreachable set, as the basis for many unreachable sets for the optimal plan. In Sec. III-B, we show how we use one special case, $\overline{C^{min}_{obst}}$, to represent all other minimal unreachable sets. We also present how we adapt the point mass example to deal with real-world robots. In Sec. III-C, we introduce a sampling method to generate augmentations to the representative minimal unreachable set ($\overline{C^{min}_{obst}}$) to generalize for sober deployment.

### A. Point Robot Example

We first present a simplified example, which assumes a point mass robot going through three non-colinear configurations. In this case, the robot's workspace is exactly the C-space. We use the same notation used by Xiao *et al*. to formalize LfH [6]: 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 $p \in \mathscr{P}$ is a sequence of low-level 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 "dual" problem of finding $f(\cdot)$, LfH [6] includes a method to find the (unique) *most constrained* unreachable set corresponding to $p$, $C^*_{obst}$. In this work, however, we introduce the definition of a (not unique) *minimal* unreachable set, $C^{min}_{obst}$, corresponding to $p$:

***Definition** 3.1:* $\mathscr{C}^{min}_{obst} \doteq \{C^{min}_{obst} \mid \forall c \in C^{min}_{obst}, f(C^{min}_{obst} \setminus \{c\} \mid c_c, c_g) \neq f(C^{min}_{obst} \mid c_c, c_g)\}$

In other words, every member, $C^{min}_{obst} \in \mathscr{C}^{min}_{obst}$, is a minimal set of obstacles that lead to $p$ being an optimal plan.[1] Any path that arises from an optimal plan can be approximated by connected line segments. The simplest case of a robot path following an optimal plan $p$ to move from $c_c$ to $c_g$ (except a straight line) is composed of configurations on two straight line segments, $c_c - c_m$ and $c_m - c_g$. The intermediate turning point is defined as $c_m$. Since $c_m$ is one configuration on the robot's path, we say $p$ *goes through* $c_m$ (Fig. 1). In the following, given a point-mass robot moving from $c_c$ to $c_g$ according to some optimal plan $p$ computed by $f(\cdot)$, we show two necessary conditions ($\Longrightarrow$) for an unreachable set ($C_{obst}$) to be a minimal unreachable set ($C^{min}_{obst}$), to aid in identifying the representative unreachable set ($\overline{C^{min}_{obst}}$) used in Sec. III-B.

[1] $C^{min}_{obst}$ is minimal in the sense that no subset of it also leads to the same optimal plan, i.e. nothing can be removed such that $p$ remains optimal.

***Proposition** 3.1:* If $c \in C_{obst}^{min}$, then the optimal plan for the unreachable set $C_{obst}^{min} \setminus \{c\}$, $\hat{p} = f(C_{obst}^{min} \setminus \{c\} \mid c_c, c_g)$, must go through $c$.

*Proof:* Assume otherwise. Since $\hat{p} = f(C_{obst}^{min} \setminus \{c\} \mid c_c, c_g) \neq f(C_{obst}^{min} \mid c_c, c_g) = p$ (***Definition** 3.1*), for $C_{obst}^{min} \setminus \{c\}$, the path arising from $\hat{p}$ is shorter than the one arising from $p$. Since we assume $\hat{p}$ does not go through $c$, adding $c$ back to $C_{obst}^{min} \setminus \{c\}$ does not affect the feasibility of $\hat{p}$ for $C_{obst}^{min}$ and does not change the path length. The path arising from $\hat{p}$ is still shorter than the one arises from $p$ in $C_{obst}^{min}$, thus contradicting the optimality of $p$ for $C_{obst}^{min}$. ■

***Proposition** 3.2:* ($\Longrightarrow$ 1) $\forall C_{obst} \in \mathscr{C}_{obst}^{min}, c_m \in C_{obst}$.[2]

*Proof:* Consider any circle $\mathscr{B}(c_m, \epsilon)$ that centers at $c_m$ with radius $\epsilon$. $\mathscr{B}$ intersects $c_c - c_m$ at $c_1$ and $c_m - c_g$ at $c_2$ (Fig. 1 a). Assume there exists no configuration $c \in \mathscr{B}$ such that $\exists C_{obst} \in \mathscr{C}_{obst}^{min}$ and $c \in C_{obst}$. Consider the path $c_c - c_1 - c_2 - c_g$: since $c_c - c_m - c_g$ is feasible, then $c_c - c_1$ and $c_2 - c_g$ are feasible. Moreover, by assumption $c_1 - c_2$ is also feasible. But $c_c - c_1 - c_2 - c_g$ is shorter than $c_c - c_m - c_g$ since $c_1 - c_2$ is shorter than $c_1 - c_m - c_2$ due to triangle inequality. This contradicts the optimality of $c_c - c_m - c_g$. Therefore, $\exists c \in \mathscr{B}$ that belongs to some $C_{obst} \in \mathscr{C}_{obst}^{min}$. Since this is true for $\lim_{\epsilon \to 0} \mathscr{B}(c_m, \epsilon)$, and $C_{obst}$ is a closed set [13], the limit point $\mathscr{B}(c_m, 0) = c_m \in C_{obst}$. ■

We name the union of all robot configurations in the triangle defined by $c_c$, $c_m$, and $c_g$ as $G_1$. On the other side of line segment $c_c - c_g$, we name the union of all configurations in the half ellipse, whose focal points locate at $c_c$ and $c_g$, and whose major axis length is $|c_c c_m| + |c_m c_g|$, as $G_2$. We define $G = G_1 \cup G_2$ (the grey area in Fig. 1).

***Proposition** 3.3:* ($\Longrightarrow$ 2) $\forall C_{obst} \in \mathscr{C}_{obst}^{min}$, $\forall c \in G$, $\{c_p \mid c_p \text{ on line segments } c_c - c - c_g\} \cap C_{obst} \neq \emptyset$

*Proof:* Assume $\exists c \in G$, $\{c_p \mid c_p \text{ on line segments } c_c - c - c_g\} \cap C_{obst} = \emptyset$. Then $c_c - c - c_g$ is a feasible path. The length of $c_c - c - c_g$ is the sum of distances from $c$ (inside ellipse) to the two ellipse focal points $c_c$ and $c_g$, which is, per definition of an ellipse, shorter than its major axis length $|c_c c_m| + |c_m c_g|$. This contradicts the optimality of $p$. ■

Based on the two necessary conditions of $C_{obst} \in \mathscr{C}_{obst}^{min}$ (***Proposition** 3.2* and *3.3*), one class of minimal unreachable sets could be simply constructed by connecting $c_m$ with some point $c_e$ on the left boundary of the ellipse with a straight line $c_m - c_e$ ($C_{obst}^{min_1}$ in Fig. 1 a), or two line segments $c_m - c_g(c_c)$ and $c_g(c_c) - c_e$ ($C_{obst}^{min_2}$ in Fig. 1 b), if not all configurations on $c_m - c_e$ are in $G$. In particular, for efficiency, we simply represent all minimal unreachable sets with a special case, $C_{obst}^{\overline{min}}$ (Fig. 1 d), which is all configurations along the straight line $c_m$ and $c_m'$ (the reflective symmetry point of $c_m$ with respect to $c_c - c_g$). Empirical evidence of this approximation's sufficiency for the purpose of learning will be provided in Sec. IV. Here, we further provide one more observation to help develop intuition regarding how to identify a $C_{obst}^{min}$.

***Proposition** 3.4:* $\forall C_{obst}^{min}$, $\forall c \in C_{obst}^{min}$, $c \in G$.

[2] $C$ is a topological space, $C_{obst}$ is a closed set, and $C_{free} = C \setminus C_{obst}$ is an open set. $c_m$ is a boundary point of both $C_{obst}$ and $C_{free}$. The robot can come arbitrarily close to the obstacles while remaining in $C_{free}$ [13].

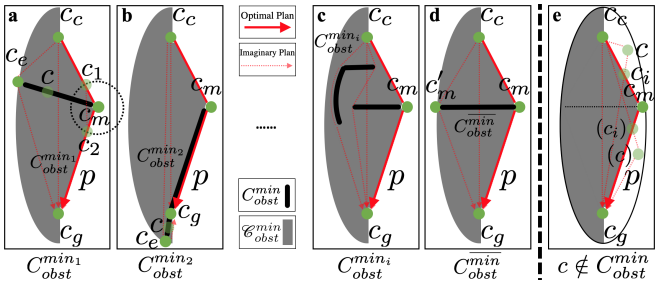


Fig. 1: $C_{obst}^{min_1}$, $C_{obst}^{min_2}$, ... , $C_{obst}^{min_i}$, $C_{obst}^{\overline{min}} \in \mathscr{C}_{obst}^{min}$.

*Proof:* Assume $c \in C_{obst}^{min}$ and $c \notin G$, only two possibilities exist:

(1) $c$ is outside the entire (left and right half) ellipse whose focal points locate at $c_c$ and $c_g$, and whose major axis length is $|c_c c_m| + |c_m c_g|$: based on ***Proposition** 3.1*, the optimal plan $\hat{p} = f(C_{obst}^{min} \setminus \{c\} \mid c_c, c_g)$ must go through $c$. The shortest possible path, which goes through $c$, is $c_c - c - c_g$, if it exists. However, based on the definition of ellipse, for any $c$ outside the ellipse, $|c_c c| + |c c_g| > |c_c c_m| + |c_m c_g|$. This contradicts the optimality of $\hat{p}$.

(2) $c$ is inside the entire ellipse, but outside $G$: $c$ must be in the right half of the ellipse but outside $G_1$ (Fig. 1 e). Again, $\hat{p}$ must go through $c$. The shortest possible path which goes through $c$ is $c_c - c - c_g$, if it exists, which must intersect either $c_c - c_m$ or $c_m - c_g$ at some point $c_i$. Due to substructure optimality (i.e. a sub-path of a shortest path is still a shortest path), the shortest path between any points on $c_c - c_m - c_g$ must be its sub-path. If $c_i$ is on $c_c - c_m$, then the shortest path from $c_i$ to $c_g$ must be $c_i - c_m - c_g$. $c_c - c - c_i - c_m - c_g$ is longer than $c_c - c_m - c_g$, and therefore not optimal. If $c_i$ is on $c_m - c_g$, the shortest path between $c_c$ and $c_i$ must be $c_c - c_m - c_i$. $c_c - c_m - c_i - c - c_g$ is longer than $c_c - c_m - c_g$, and therefore not optimal. In both cases, the shortest possible path going through $c$ is longer than $c_c - c_m - c_g$. This contradicts the optimality of $\hat{p}$.

Therefore, $\forall C_{obst}^{min}$, $\forall c \in C_{obst}^{min}$, $c \in G$. ■

### *B. Realistic Nonholonomic Robot*

Based on the propositions discussed in Sec. III-A, we present the hallucination technique for a realistic robot. In realistic scenarios, we approximate all $C_{obst}^{min}$ in $\mathscr{C}_{obst}^{min}$ with $C_{obst}^{\overline{min}}$. We hypothesize that all $C_{obst}^{min}$ are sufficiently similar that using just one leads to learning that is just as good as if we used them all, especially when (1) $c_c$, $c_m$, and $c_g$ extracted from realistic trajectories are close to each other, and (2) $C_{obst}$ is instantiated in terms of discrete LiDAR beams. Empirical evidence of this sufficiency will be provided in Sec. IV. However, a realistic robot cannot be modeled as a simple point mass because: (1) the size is not negligible and we need a path for the *center* of the robot that causes no part of the robot to collide with an obstacle; (2) nonholonomic robots cannot change motion direction instantly, so we need to generalize to a *continuously* turning path from the *piece-wise* point mass example. In Sec. III-B, we aim to adapt the point mass example (Sec. III-A) to real-world robots, and therefore need to address these differences.

To address (1), we define one point to the left and another

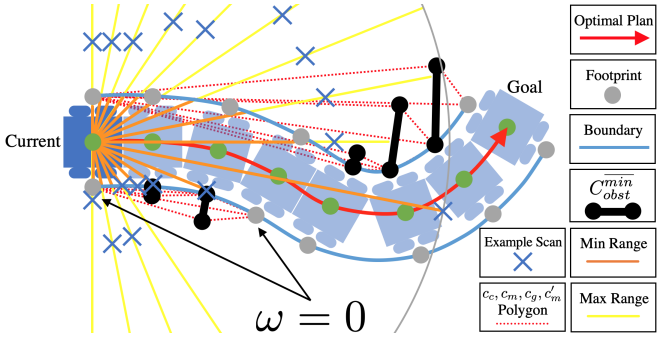


Fig. 2: Applying the point robot example to realistic robot case: $C_{obst}^{min}$ is instantiated as LiDAR beams whose maximum range is determined by ray casting.

to the right of the centroid of the robot, offset by the robot width, as footprint points, as shown in Fig. 2. The instantaneous linear velocity along the line between these two footprint points is zero for nonholonomic vehicles. The polygon defined by a sequence of footprint points must belong to free space. The actual path executed by the optimal plan $p$ follows the middle of this area. To address (2), we define $c_\omega$ as configurations between the current and goal configurations with a non-zero angular velocity ($\omega \neq 0$). Given the current configuration, each $c_\omega$, and each $c_\omega$'s next configuration, their left or right footprint points are treated as $c_c$, $c_m$, and $c_g$ in the *point robot* case: based on the sign of $\omega$, the robot turns left or right, and the left or right footprint points are chosen. For each triple of *point robot* $c_c$, $c_m$, and $c_g$, for efficiency, we approximate all different $C_{obst}^{min}$ with $\overline{C_{obst}^{min}}$ (Fig. 1 d). For a realistic optimal plan $p$ with *actual* $c_c$ (current) and $c_g$ (goal) and multiple *point robot* $c_c - c_m - c_g$ triples in between, we define the union of all $\overline{C_{obst}^{min}}$ as $\overline{\mathscr{C}_{obst}^{min}}$.

In particular, we define an "opposite" function of $f(\cdot)$: $\overline{\mathscr{C}_{obst}^{min}} = o(p \mid c_c, c_g)$ as the *minimal* hallucination function.[3] This function finds $\overline{\mathscr{C}_{obst}^{min}}$ based on the fact that $p$ is an optimal plan. As visualized in Fig. 2, $C_{obst}$ is instantiated in terms of discrete LiDAR range readings. For each LiDAR beam, we define a minimum and a maximum range, which limit the possible range readings of this particular beam based on the optimal plan $p$. The minimum range of all beams is determined by the left and right boundary, as configurations within the boundary must be in $C_{free}$. We project all $\overline{C_{obst}^{min}}$ onto the corresponding LiDAR beams and for the beams directly intersect any $\overline{C_{obst}^{min}}$, the maximum range is set as the distance from the robot to the intersection point. For other beams, the maximum range is simply the LiDAR's physical limit, or manually pre-processed to a certain threshold.

### C. Sampling between Min and Max Range

Given the representative minimal unreachable set, we want to find all possible unreachable sets, or their sensor readings, that could lead to $p$ being the optimal plan. Based on the LiDAR scan with a minimum and maximum range for each beam (end of Sec. III-B), a sampling strategy is devised to create many obstacle sets, $C_{obst}$, in which $p$ is optimal.

Our sampling strategy aims at creating different range readings that (1) resemble real-world obstacles, and (2) respect uncertainty/safety. For (1), most real-world obstacles have a certain footprint, and their surface contribute to continuity among neighboring beams. Starting from the first beam, a random range is sampled between min and max with a uniform distribution. Moving on to the neighboring beam, with a probability $\alpha$, we increase, or decrease, the previous range by a small random amount, and assign the value to the current beam. This practice is to simulate the continuity in neighboring beams. We make sure this value is within the min and max range of that beam. With probability $1-2\alpha$, we start from scratch and randomly sample between this beam's min and max values. This practice is to simulate the scenarios where the next beam misses the current obstacle, and reaches another one or does not reach any obstacle at all. For (2), we add an offset value as a function of the optimal plan $p$ to the ranges. For example, given a faster speed of $p$, a larger positive offset is added to the range reading (obstacles are farther away), because faster motion is correlated with more uncertainty or less safety (details can be found in Sec. IV-A). One example scan sample is shown in Fig. 2 as blue $\times$'s.

The entire HLSD pipeline is described in Alg. 1. The inputs to the algorithm are a random exploration policy $\pi_{rand}$ in open space; the minimal hallucination function $o(\cdot)$; a *sampling count* of hallucinated $C_{obst}$ to be generated per data point; the *offset*$(\cdot)$ function; the probability $\alpha$; and a parameterized planner $f_\theta(\cdot)$. For every data point $(p, c_c, c_g)$ in $\mathscr{D}_{raw}$ collected using $\pi_{rand}$ in open space (line 2), we hallucinate $\overline{\mathscr{C}_{obst}^{min}}$ (line 5) and generate the min and max values for every LiDAR beam (line 6). Lines 8–15 correspond to the sampling technique to generate random laser scans. We instantiate $C_{obst}$ as LiDAR readings $L$ and add it to $\mathscr{D}_{train}$ (line 16). This process is repeated *sampling count* times for every data in $\mathscr{D}_{raw}$. Finally, we train $f_\theta(\cdot)$ with supervised learning (line 19). This hallucinated learning enables sober deployment with perception of the real configuration $C_{obst}^{real}$ without runtime hallucination (Lines 21–22).

## IV. EXPERIMENTS

Simulated and physical experiments are conducted to validate our hypothesis that HLSD can achieve better performance (faster, smoother, safer) than a classical method and LfH with runtime hallucination. In our experiments, we use a Clearpath Jackal robot, a four-wheeled, differential-drive, nonholonomic, Unmanned Ground Vehicle (UGV), running the Robot Operating System (ROS) `move_base` navigation stack. Its DWA local planner is replaced with HLSD. The robot navigates without a map. The global planner (Dijkstra's algorithm) assumes unknown regions are free and replans when obstacles are perceived. The local environment is known to the local planner.

### A. Implementation

In order to instantiate Alg. 1, with $\boldsymbol{o(\cdot)}$ described in detail in Sec. III-B, one still needs to define $\boldsymbol{f_\theta(\cdot)}$, $\boldsymbol{\pi_{rand}}$, ***sampling***

[3]Note the inverse function $f^{-1}(\cdot)$ does not exist. Technically, $c_g$ can be uniquely determined by $p$ and $c_c$, but we include it as an input to $o(\cdot)$ for notational symmetry with $f(\cdot)$.

***count***, $\boldsymbol{\alpha}$, and ***offset*$(\cdot)$**:

$\boldsymbol{f_\theta(\cdot)}$: For $p = \{u_i\}_{i=1}^{t} = f_\theta(C_{obst}^{real} \mid c_c, c_g)$, instead of requiring a high-resolution global path from the global planner (Dijkstra's) to construct $C_{obst}^{*}$ in LfH [6], we only query a single local goal $c_g$ on the global path 1m away from the robot at each time step, and $c_c$ is the origin in the robot frame (orientation is ignored for simplicity). The UGV is equipped with a 720-dimensional 2D LiDAR with a 270° field of view, and we clip the maximum range to 1m to reduce the input space ($C_{obst}^{real}$). The planning horizon $t$ of $p$ is set to 1, i.e. only a single action $p = \{u_1\} = \{(v_1, \omega_1)\}$ (linear and angular velocity) is produced. We use the same three-layer neural network, with 256 hidden neurons and ReLU activation for each layer as in LfH [6].

$\boldsymbol{\pi_{rand}}$: $\pi_{rand}$ randomly picks a target $(\hat{v}, \hat{\omega})$ pair and commands the robot to reach that speed with constant increments/decrements considering acceleration limit. After reaching $(\hat{v}, \hat{\omega})$, $\pi_{rand}$ keeps that command with some probability (0.9) or otherwise generates a new target command. We limit the output $v \in [0, 1.0]$m/s and $\omega \in [-1.57, 1.57]$rad/s. During training in a simulated open space, control inputs ($v$ and $\omega$) and robot configurations ($x$, $y$, and $\psi$) are recorded. Unlike the dataset collected by LfH [6], which mostly contains $v = 0.4$m/s, our dataset contains a variety of $v$ values. 12585 data points are collected in a 505s real-time simulation, including a variety of motions in an open space.

***sampling count***, $\boldsymbol{\alpha}$, and ***offset*$(\cdot)$**: We set *sampling count* to 10 and $\alpha$ to 0.48. The *offset*$(\cdot)$ function linearly maps current $v$ in the range [0.3, 1.0]m/s to an offset value between [0, 1]m. Considering the fact that a real robot also needs to turn even in open space due to nonholonomic constraints, as opposed to an ideal point mass robot which does not, we also hallucinate $C_{obst} = \emptyset$ for configurations where $v > 0.8$m/s. Considering highly constrained spaces, we additionally hallucinate the most constrained $C_{obst}^{*}$ for configurations where $v < 0.3$m/s. So $|\mathscr{D}_{train}| = 12 * |\mathscr{D}_{raw}|$. Training takes less than three minutes on a NVIDIA GeForce GTX 1650 laptop GPU.

After computing $p$ with Alg. 1, we use the same Model Predictive Control model as in LfH [6] to check for collisions. When a collision is detected, the robot enters a two-phase recovery behavior: it first queries the global path immediately in front of the robot, and rotates to align the robot heading with the tangential direction of the global path. If this recovery behavior is still not safe as determined by the collision checking, the robot backs up at $v = -0.2$m/s. Since LfH learns only from the *most constrained* C-space, it requires runtime hallucination to match with training data, and a Turn in Place module to drive the robot out of out-of-distribution scenarios. Neither of those components are required by HLSD. Because our dataset contains varying $v \in [0, 1.0]$m/s, the LfH speed modulation module to adapt mostly constant $v$ to real environments is not necessary either. The robot is able to react to the real obstacle configuration by using $f_\theta(\cdot)$ alone because $\mathscr{D}_{train}$ covers a wide range of distributions.

**Algorithm 1** Hallucinated Learning and Sober Deployment

**Input:** $\pi_{rand}$, $o(\cdot)$, *sampling count*, *offset*$(\cdot)$, $\alpha$, $f_\theta(\cdot)$

// **Hallucinated Learning**
collect motion plans $(p, c_c, c_g)$ from $\pi_{rand}$ in free space and form raw data set $\mathscr{D}_{raw}$
$\mathscr{D}_{train} \leftarrow \emptyset$
**for** every $(p, c_c, c_g)$ in $\mathscr{D}_{raw}$ **do**
  hallucinate $\mathscr{C}_{obst}^{\overline{min}} = o(p \mid c_c, c_g)$
  generate LiDAR range $L_{min}$ and $L_{max}$ with $\mathscr{C}_{obst}^{\overline{min}}$
  **for** iter = 1 : *sampling count* **do**
    $L \leftarrow \emptyset$
    $l^1 \sim [l_{min}^1, l_{max}^1]$, $l^1 \leftarrow l^1 +$ *offset*$(p)$
    add $l^1$ to $L$, $l^{last} \leftarrow l^1$
    **for** i = 2 : $|L|$ **do**
      increase, decrease $l^{last}$ by a randomly selected small amount, or $l^i \sim [l_{min}^i, l_{max}^i]$, $l^i \leftarrow l^i +$ *offset*$(p)$, with probability $\alpha$, $\alpha$, and $1 - 2\alpha$, respectively, and assign to $l^i$
      make sure $l^i \in [l_{min}^i, l_{max}^i]$
      add $l_i$ to $L$, $l^{last} \leftarrow l^i$
    **end for**
    $C_{obst} \leftarrow L$, $\mathscr{D}_{train} = \mathscr{D}_{train} \cup (C_{obst}, p, c_c, c_g)$
  **end for**
**end for**
train $f_\theta(\cdot)$ with $\mathscr{D}_{train}$ by minimizing the error $\mathbb{E}_{(C_{obst}, p, c_c, c_g) \sim \mathscr{D}_{train}} \left[ \ell(p, f_\theta(C_{obst} \mid c_c, c_g)) \right]$

// **Sober Deployment** (each time step)
receive $C_{obst}^{real}, c_c, c_g$
plan $p = \{u_i\}_{i=1}^{t} = f_\theta(C_{obst}^{real} \mid c_c, c_g)$
**return** $p$

### *B. Simulated Experiments*

We first use the Benchmark Autonomous Robot Navigation (BARN) dataset with 300 navigation environments and quantified difficulty levels [11] (example environments shown in Fig. 3) to compare (1) DWA, (2) original LfH trained on the mostly constant 0.4m/s dataset with hallucinated $C_{obst}^{*}$ (LfH 0.4), (3) LfH trained on our varying speed 1.0m/s dataset with $C_{obst}^{*}$ (LfH 1.0), (4) HLSD trained on the 0.4m/s dataset with augmentations to $C_{obst}^{\overline{min}}$ (HLSD 0.4), and (5) HLSD trained on the 1.0m/s dataset with $C_{obst}^{\overline{min}}$ (HLSD 1.0). The baselines are chosen for the following reasons: DWA is a widely used local planner and its implementation is available through the ROS `move_base` navigation stack. We only consider similar self-supervised approaches and exclude methods that require expert supervision [1], [18], because HLSD does not require any expert motion trajectories. Although DWA's default max linear velocity is 0.5m/s, for a fair comparison, we set DWA's max linear velocity to the same as HLSD's (1.0m/s). We find that by also doubling DWA's default sampling rate for linear and angular velocity (12 and 40), the robot's performance is roughly the same as when using the default parameters (but at double the speed). Simulated results are shown in Fig. 4.

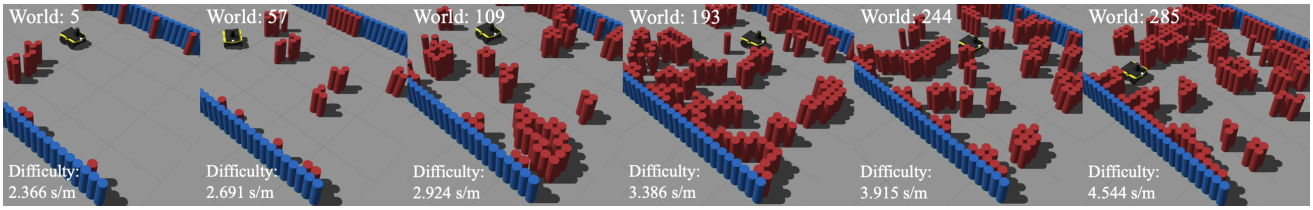


Fig. 3: Simulated Environments of Different Difficulties in the BARN Dataset [11]

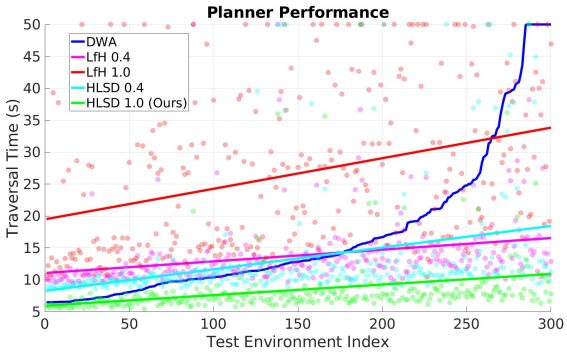


Fig. 4: Simulation Results

In each of the 300 navigation environments generated by Cellular Automaton, the robot navigates between a specified start and goal location without a map. We record the traversal time for each trial, with a maximum of 50s. Three trials are conducted for each planner in each environment, resulting in a total of 4500 trials. The difficulty of the 300 environments are ordered from left to right based on the DWA performance (blue line). The performances of other planners are plotted as dots, for which a line is fit using linear regression. LfH 1.0 (red) fails many trials. The reason is that learning from a dataset with varying speed and always hallucinating the most constrained spaces causes ambiguity: the same most constrained C-space can map to different plans, which confuses the learner. LfH 0.4 (magenta) and HLSD 0.4 (cyan), both trained on the mostly constant 0.4m/s dataset without ambiguity, achieve similar performance and are less sensitive to navigational difficulty, especially LfH 0.4, but LfH 0.4 requires runtime hallucination and other components. Note that DWA has max speed of 1.0m/s, while LfH 0.4 and HLSD 0.4 have up to 0.6m/s max speed, modulated from mostly 0.4m/s learned from the dataset. DWA has an advantage in easy environments (left) due to its fast speed, but in difficult ones (right), LfH 0.4 and HLSD 0.4 are more stable. Also having 1.0m/s max speed, HLSD 1.0 considers the varying linear velocity in the 1.0m/s dataset with the *offset*$(\cdot)$ function (line 9 in Alg. 1), and achieves the best result, outperforming all alternatives across the entire range of difficulties. The means and standard deviations of all five planners calculated from all trials are shown in Tab. I. LfH 1.0 has the largest average time and variance, because the learner is confused by the varying training labels in the same most constrained spaces. DWA has large time and also high variance due to its sampling nature. Again, HLSD 0.4 performs similarly as LfH 0.4 does. HLSD 1.0 still outperforms all other planners.

### C. Physical Experiments

We also deploy the same set of local planners in a physical test course, five trials each (Fig. 5 top). The results are shown in Tab. I. The complicated obstacle course causes DWA to execute many recovery behaviors, and the robot takes a long average time with large variance to finish the traversal. LfH 0.4 and HLSD 0.4 are both able to successfully navigate the robot through, with similar traversal times. But note that LfH 0.4 requires a fine-resolution global path for hallucination during deployment and the Turn in Place module, while HLSD does not and only reacts to the real obstacles without any other extra components. In this physical obstacle course, LfH 1.0 fails every trial due to multiple collisions. Our HLSD 1.0 achieves the best performance both in terms of average time and standard deviation. HLSD deployment in a natural cluttered environment is shown in Fig. 5 bottom.

TABLE I: Simulated and Physical Traversal Time in Seconds

| | DWA | LfH 0.4 | LfH 1.0 | HLSD 0.4 | **HLSD 1.0** |
|---|---|---|---|---|---|
| Sim. | 17.0±12.6 | 13.5±6.4 | 26.7±15.0 | 13.4±9.8 | **8.5±6.3** |
| Phy. | 78.8±10.0 | 67.0±4.4 | $\infty$ | 66.3±0.7 | **45.4±0.4** |

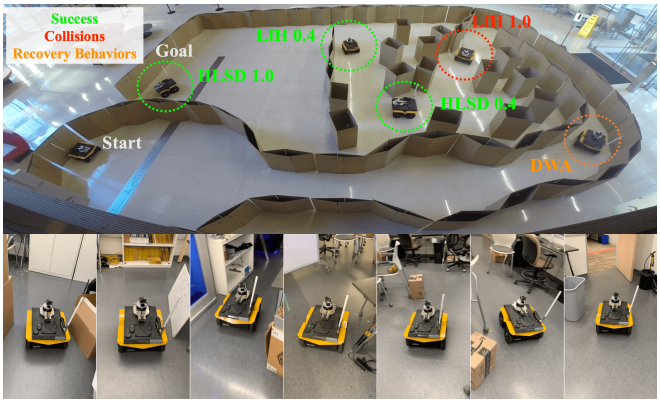


Fig. 5: Physical Experiments (`https://www.youtube.com/watch?v=LZcBN9zgtXg&t=11s`)

## V. CONCLUSIONS

We introduce HLSD, a self-supervised machine learning technique for mobile robot navigation with safety guarantee during training. Similar to LfH [6], the robot safely learns in a completely obstacle-free environment and its perception is hallucinated with obstacle configurations where the actions taken in the free space remain optimal. Instead of overfitting to the most constrained spaces during training and requiring runtime hallucination and other modules to adapt to actual environments, the new HLSD pipeline allows the robot to navigate with the learned planner alone in response to realistic perception. By leveraging a large body of carefully designed hallucinations used for training, the learned planner does not need to deal with many out-of-distribution scenarios and has its own capability to address uncertainty/safety in the real-world during deployment.

## REFERENCES


[1] 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.

[2] 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.

[3] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[4] S. Siva, M. Wigness, J. Rogers, and H. Zhang, "Robot adaptation to unstructured terrains by joint representation and apprenticeship learning," in *Robotics: Science and Systems (RSS)*, 2019.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] X. Xiao, J. Dufek, T. Woodbury, and R. Murphy, "Uav assisted usv visual navigation for marine mass casualty incident response," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6105–6110.

[11] 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.

[12] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

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

[14] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *arXiv preprint arXiv:2102.12667*, 2021.

[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. 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.

[17] 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.

[18] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese *et al.*, "Deep local trajectory replanning and control for robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5815–5822.