In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation* (ICRA 2023), London, England, May 2023

Benchmarking Reinforcement Learning Techniques for Autonomous Navigation

Zifan Xu[1], Bo Liu[1], Xuesu Xiao[2,3], Anirudh Nair[1], and Peter Stone[1,4]

***Abstract*— Deep reinforcement learning (RL) has brought many successes for autonomous robot navigation. However, there still exists important limitations that prevent *real-world* use of RL-based navigation systems. For example, most learning approaches lack safety guarantees; and learned navigation systems may not generalize well to unseen environments. Despite a variety of recent learning techniques to tackle these challenges in general, a lack of an open-source benchmark and reproducible learning methods specifically for autonomous navigation makes it difficult for roboticists to choose what learning methods to use for their mobile robots and for learning researchers to identify current shortcomings of general learning methods for autonomous navigation. In this paper, we identify four major desiderata of applying deep RL approaches for autonomous navigation: (D1) *reasoning under uncertainty*, (D2) *safety*, (D3) *learning from limited trial-and-error data*, and (D4) *generalization to diverse and novel environments*. Then, we explore four major classes of learning techniques with the purpose of achieving one or more of the four desiderata: *memory-based neural network architectures* (D1), *safe RL* (D2), *model-based RL* (D2, D3), and *domain randomization* (D4). By deploying these learning techniques in a new open-source large-scale navigation benchmark and real-world environments, we perform a comprehensive study aimed at establishing to what extent can these techniques achieve these desiderata for RL-based navigation systems.**

I. Introduction

Autonomous robot navigation, i.e., moving a robot from one point to another without colliding with any obstacle, has been studied by the robotics community for decades. Classical navigation systems [1], [2] can successfully solve such navigation problem in many real-world scenarios, e.g., handling noisy, partially observable sensory input but still providing verifiable collision-free safety guarantees. However, these systems require extensive engineering effort, and can still be brittle in challenging scenarios, e.g., in highly constrained environments. This is reflected by a recent competition (The BARN Challenge [3]) held in ICRA 2022, which suggests that even experienced roboticists tend to underestimate how difficult navigation scenarios are for real robots. Recently, data-driven approaches have also been used to tackle the navigation problem [4] thanks to advances in the machine learning community. In particular, Reinforcement Learning (RL), i.e., learning from self-supervised trial-and-error data, has achieved tremendous progress on multiple fronts, including safety [5]–[7], generalizability [8]–[11], sample efficiency [12], [13], and addressing temporal data [14]–[16]. For the problem of navigation, learned navigation systems from RL [17] have the potential to relieve roboticists from extensive engineering efforts [18]–[22] spent on developing and fine-tuning classical systems. Moreover, a simple case study conducted in five randomly generated obstacle courses where classical navigation systems often fail shows that a RL-based navigation has the potential to achieve superior behaviors in terms of successful collision avoidance and goal reaching (Fig. 1 left).

Despite such promising advantages, learning-based navigation systems are far from finding their way into real-world robotics use cases, which currently still rely heavily on their classical counterparts. Such reluctance in adopting learning-based systems in the real world stems from a series of fundamental limitations of learning methods, e.g., lack of safety, explainability, and generalizability. To make things even worse, a lack of well-established comparison metrics and reproducible learning methods further obfuscates the effects of different learning approaches on navigation across both the robotics and learning communities, making it difficult to assess the state of the art and therefore to adopt learned navigation systems in the real world.

To facilitate research in developing RL-based navigation systems with the goal of deploying them in real-world scenarios, we introduce a new open-source large-scale navigation benchmark with a variety of challenging, highly constrained obstacle courses to evaluate different learning approaches, along with the implementation of several state-of-the-art RL algorithms. The obstacle courses resemble highly-constraint real-world navigation environments (Fig. 1 right), and present major challenges to existing classical navigation systems, while RL-based navigation systems have the potential to perform well in them (Fig. 1 left).

We identify four major desiderata that ought to be fulfilled by any learning-based system that is to be deployed: (D1) *reasoning under uncertainty of partially observed sensory inputs*, (D2) *safety*, (D3) *learning from limited trial-and-error data*, and (D4) *generalization to diverse and novel environments*. By deploying four major classes of learning techniques: *memory-based neural network architectures*, *safe RL*, *model-based RL*, and *domain randomization*, we perform extensive experiments and empirically compare a large range of RL-based methods based on the degree to which they achieve each of these desiderata. Moreover, by deploying six selected navigation systems in three qualitatively different real-world navigation environments, we investigate to what degree the conclusions drawn from the benchmark can be

[1]Department of Computer Science, University of Texas at Austin [2]Department of Computer Science, George Mason University [3]Everyday Robots [4]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.

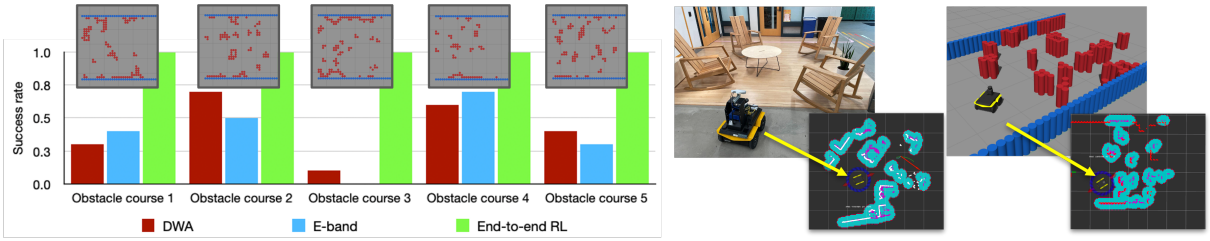

Fig. 1: **Left:** Success rate of two classical navigation systems, DWA [2] (red) and E-band [1] (blue), and vanilla end-to-end RL-based (green) navigation systems (individually trained) in five randomly generated difficult obstacle courses. The insets at the top show top-down views of the five obstacle courses. **Right:** Navigation environments in the real world (left) and the proposed benchmark (right) are similar to the robot perception system (e.g., white/red laser scans and cyan/purple costmaps).

applied to the real world. Supplementary videos and material for this work are available on the project webpage.[1]

II. Desiderata for Learning-based Navigation

In this section, we introduce four desiderata for learning-based autonomous navigation systems and briefly discuss the learning techniques as their corresponding solutions.

(D1) reasoning under uncertainty of partially observed sensory inputs. Autonomous navigation without explicit mapping and localization is usually formalized as a Partially Observable Markov Decision Process (POMDP), where the agent produces the motion of the robot only based on limited sensory inputs that are usually not sufficient to recover the full state of the navigation environment. Most RL approaches solve POMDPs by maintaining a history of past observations and actions [14], [15]. Then, neural network architectures like Recurrent Neural Networks (RNNs) that process sequential data are employed to encode history and address partial observability. In this study, we investigate various design choices of history-dependent architectures.

(D2) safety. Even though in some cases deep RL methods achieve comparable performance to classical navigation, they still suffer from poor explainability and do not guarantee collision-free navigation. The lack of safety guarantee is a major challenge preventing RL-based navigation from being used in the real world. Prior works have addressed this challenge by formalizing the navigation as a multi-objective problem that treats collision avoidance as a separate objective from reaching the goal and solving it with Lagrangian or Lyapunov-based methods [5]. For simplicity, we only explore Lagrangian method and investigate whether explicitly treat safety as a separate objective leads to safer and smoother learned navigation behavior.

(D3) learning from limited trial-and-error data. Although deep RL approaches can alleviate roboticists from extensive engineering effort, a large amount of data is still required to train a typical deep RL agent. However, autonomous navigation data is usually expensive to collect in the real world. Therefore, data collection is usually conducted in simulation, e.g., in the Robot Operating System (ROS) Gazebo simulator, which provides an easy interface with real-world robots. However, simulating a full navigation stack from perception to actuation is more computationally expensive compared to other RL domains, e.g., MuJuCo or Atari games [23], [24], which presents a high requirement for sample efficiency. Most prior works have used off-policy RL algorithms to improve sample efficiency with experience replay [25], [26]. In addition, model-based RL methods can explicitly improve sample efficiency, and are widely used in robot control problems. In this study, we compare two common classes of model-based RL method [12], [13] combined with an off-policy RL algorithm, and empirically study to what extent model-based approaches improve sample efficiency when provided with different amounts of data.

(D4) generalization to diverse and novel environments. The ultimate goal of deep RL approaches for autonomous navigation is to learn a generalizable policy for all kinds of navigation environments in the real world. A simple strategy is to train the agent in as many diverse navigation environments as possible or domain randomization, but it is unclear what is the necessary amount of training environments to efficiently achieve good generalization. Utilizing the large-scale navigation benchmark proposed in this paper, we empirically study the dependence of generalization on the number of training environments.

III. Navigation Benchmark

This section details the proposed navigation benchmark for RL-based navigation systems, which aims to provide a unified and comprehensive testbed for future autonomous navigation research. First, Sec. III-A discusses the difference between the proposed benchmark and existing navigation benchmarks. In Sec. III-B and III-C, the navigation task is formally defined and formulated as a POMDP. More detailed background of MDP and POMDP can be found on the project webpage. Finally, Sec. III-D introduces simulated and real-world environments that benchmark different aspects of navigation performance.

A. Existing Navigation Benchmarks

Our proposed benchmark differs from existing benchmarks in three aspects: (1) **high-fidelity physics**: the navigation tasks are simulated by Gazebo [27], which is based on realistic physical dynamics and therefore tests motion planners that directly produce low-level motion commands, i.e., linear and angular velocies, in contrast to high-level instructions such as turn left, turn right, move forward [28], [29]. In other words,

[1]https://cs.gmu.edu/ xiao/Research/RLNavBenchmark/

we focus on "how to navigate" (motion planning), instead of "where to navigate" (path planning); (2) **ROS integration**: our benchmark is based on ROS [30], which allows seamless transfer of a navigation method developed and benchmarked in simulation directly onto a physical robot with little (if any) effort; and (3) **collision-free navigation**: the benchmark includes both static and dynamic environments, and requires *collision-free* navigation, whereas other benchmarks assume that either collisions are possible [29] or collision-avoidance will be addressed by other low-level controllers out of the scope of the benchmark [28]. A special case is the photo-realistic interactive Gibson benchmark by Xia et. al. [31], which intentionally allows physical interaction with objects (e.g., pushing) and therefore pose no challenges to the collision-avoidance system.

B. Navigation Problem Definition

Definition 1 (Robot Navigation Problem)**.** *Situated within a navigation environment e which includes information of all the obstacle locations at any time t, a start location (x_i, y_i), a start orientation θ_i, and a goal location (x_g, y_g), the navigation problem $\mathcal{T}_e$ is to maximize the probability p of a mobile robot reaching the goal location from the start location and orientation under a constraint on the number of collisions with any obstacle $C < 1$ and a time limit $t < T_{max}$.*

A navigation problem can be formally defined as above. Given the current location (x_t, y_t), the robot is considered to have reached the goal location if and only if its distance to the goal location is smaller than a threshold, $d_t < d_s$, where d_t is the Euclidean distance between (x_t, y_t) and (x_g, y_g), and d_s is a constant threshold.

C. POMDP Formulation

A navigation task $\mathcal{T}_e$ can be formulated as a POMDP conditioned on a navigation environment e, which can be represented by a 7-tuple $(S_e, A_e, O_e, T_e, \gamma_e, R_e, Z_e)$. In this POMDP, the state $s_t \in S_e$ is a 5-tuple $(x_t, y_t, \theta_t, c_t, e)$ with x_t, y_t, θ_t the two-dimensional coordinates and the orientation of the robot at time step t, c_t a binary indicator of whether a collision has occurred since the last time step $t-1$, and e the navigation environment. The action $a_t = (v_t, \omega_t) \in A_e$ is a two-dimensional continuous vector that encodes the robot's linear and angular velocity. The observation $o_t = (\chi_t, \bar{x_t}, \bar{y_t}) \in O_e$ is a 3-tuple composed of the sensory input χ_t from LiDAR scans and the relative goal position $(\bar{x_t}, \bar{y_t})$ in the robot frame. The observation model $Z : S \to O$ maps the state to the observation. The reward function for this POMDP is defined as follows:

$$R_e(s_t, a_t) = +b_f \cdot \mathbb{1}(d_t < d_s) + b_p \cdot (d_{t-1} - d_t) - b_c \cdot c_t, \quad (1)$$

where $\mathbb{1}(d_t < d_s)$ is the indicator function of reaching the goal location, d_t is the Euclidean distance to the goal location, and b_f, b_p, b_c are the coefficient constants. In this reward function, the first term is the true reward function that assigns a positive constant b_f for the success of an agent, which matches with the objective of the navigation task in Definition 1. The second and third terms are auxiliary rewards that facilitate the training by encouraging local progress and penalizing collisions.

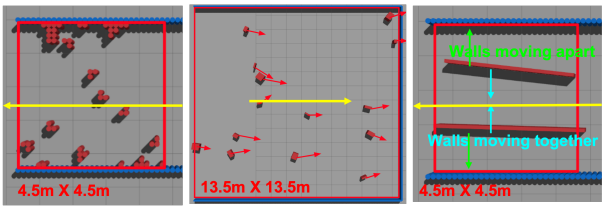

Fig. 2: Three types of navigation environments: `static` (left), `dynamic box` (middle), and `dynamic-wall` (right). The red squares mark the obstacle fields, and the yellow arrows mark the direction of navigation. In `dynamic-wall`, the green (blue) arrows indicate the case when the two walls are moving apart (together). In `dynamic box`, the red arrows indicate the velocities of obstacles.

We perform a grid search over different values of the coefficients in this reward function, and the result shows that the auxiliary reward term $(d_{t-1} - d_t)$ is necessary for successful training, and a much smaller coefficient b_p relative to b_f can lead to a better asymptotic performance. The agent can learn without the penalty reward for collision ($b_c = 0$), but a moderate value of b_c can improve the asymptotic performance and speed up training. For all the experiments in this paper, we fix the coefficients as $b_f = 20$, $b_p = 1$ and $b_c = 4$.

In our experiments, the RL algorithm solves a multi-task RL problem where the tasks are randomly sampled from a task distribution $\mathcal{T}_e \sim p(\mathcal{T}_e)$. Here the task distribution $p(\mathcal{T}_e) := U(\{e_i\}_{i=1}^N)$ is a uniform distribution on a set of N navigation environments $\{e_i\}_{i=1}^N$. The overall objective of this multi-task RL problem is to find an optimal policy $\pi^* = \max_\pi \mathbb{E}_{\mathcal{T}_e \sim p(\mathcal{T}_e), \tau_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R_e(s_t, a_t) \right]$.

D. Navigation Environments

The navigation is performed by a ClearPath Jackal differential-drive ground robot in simulated by the Gazebo simulator. More details of the robot and simulation can be found on the project webpage. Each environment in this benchmark will have a navigation system navigating the robot through a 10m navigation path that passes through a highly constrained obstacle course. Walls are placed at three edges of a square so that passing through the obstacle field is the only path to the goal location (see Fig. 2). The benchmark includes 300 `static` environments, 100 `dynamic-box` environments, and 100 `dynamic-wall` environments. The `static` environments contains a diverse set of obstacle course covering a large range of difficulty levels from easy to hard. A `dynamic-box` environment has small boxes with random shapes and velocities to test the system's immediate reactions to small moving obstacles. A `dynamic-wall` has two walls moving oppositely that requires the system to make a longer-term decision of whether to pass or wait. The detailed procedures of generating these environments can

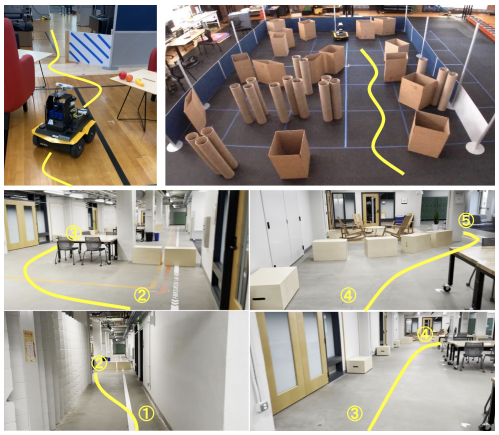

Fig. 3: Real-world benchmark-like (top-right), in-door highly-constrained (top-left), and large-scale (bottom) environments. The yellow curves mark the paths of navigation.

be found on the project webpage. We randomly select 50 environments from each type as the test sets, which are denoted as `static-test`, `dynamic-box-test`, and `dynamic-wall-test`. The remaining environments are denoted as `static-train`, `dynamic-box-train`, and `dynamic-wall-train` respectively. To study the effect of randomization, `static-train` is further separated as `static-train-5`, `static-train-10`, `static-train-50`, `static-train-100`, and `static-train-250` by randomly sampling 5, 10, 50, 100, and all 250 environments from `static-train`.

To test the sim-to-real transferability of the policies learning with different techniques, the navigation systems are deployed in three qualitatively different static navigation environments including a benchmark-like environment (Fig. 3 left), an indoor highly-constrained environment (Fig. 3 right), and a large-scale environment of 30 meters in length. We denote them as `real-world-1`, `real-world-2`, and `real-world-3` respectively.

IV. Experiments

In this section, we present experimental results of each studied technique to achieve the proposed desiderata in Sec. II. We implement distributed training pipelines (similar to [21]) of different RL algorithms including TD3 [32], SAC [33], and DDPG [34]. They perform similarly in the study of different neural network architectures. For simplicity, all the experiments mentioned in this section use TD3 combined with the corresponding techniques, and all the data points are averaged over three independent runs.

A. Memory-based Neural Network Architectures (D1)

To benchmark the performance of different neural network (NN) architectures, deep RL policies represented by architectures of Multilayer Perceptron (MLP), One-dimensional Convolutional Neural Network (CNN), Gated Recurrent Units (GRU), and Transformer with history length of 4 and 8 are trained in `static-train-50`, and the two types of dynamic environments `dynamic-box-train` and `dynamic-wall-train` from Sec. III-D. After training, the policies are tested in their corresponding test sets. In addition, MLP with history length of one is added as a memory-less baseline. Table I shows the success rates of policies with different architectures and history lengths evaluated in `static-test` (left), `dynamic-wall-test` (middle) and `dynamic-box-test` respectively.

Memory-based NNs only marginally improve navigation performance in static environments. In Table I, the policy represented by Transformer with a history length of 4 shows the best success rate of 68%, with a slightly worse success rate of 65% achieved by the baseline MLP. Additionally, a monotonic decrease in success rate with increasing history length is observed in each tested NN architecture. For example, a 32% drop in the success rate of Transformer is shown by increasing the history length from 4 to 8. One possible explanation is that, if only few past observations are useful to make the decision, including more history will make it more difficult to learn a generalized policy in this very diverse training set.

Memory is essential when possible catastrophic failures will happen by making the wrong long-term decisions. Memory usually matters for dynamic environments when a single time frame is not sufficient to estimate the motion of obstacles. Surprisingly, in `dynamic-box` where the dynamic obstacles are completely random, the memory-based NN architectures do not outperform the memory-less baseline. On the other hand, in `dynamic-wall` with a manually designed dynamic challenge, the best success rate of 82% is observed in GRU with a history length of 4, which improves about 15% over the non-memory baseline. During our deployment of the policies, we observe that, in `dynamic-box` even though the memory-less agent does not estimate the motion and adjust its plan in advance, it tends to perform safely and avoids the obstacles when they get close enough. This simple strategy works surprisingly well and achieves similar success rate as the memory-based policies. However, this strategy does not work in the manually designed dynamic challenges like `dynamic-wall` where the agent has to estimate the motion of the obstacles to pass safely.

B. Safe RL (D2)

To investigate to what extent safe RL methods can help to improve safety, a TD3 agent with the Lagrangian-based safe RL method is trained in `static-train-50`, and then tested in `static-test`. The policy is represented by a MLP with its input containing only one history length. Table II shows the success rate, average survival time, and average traversal time of the safe RL agent trained with Lagrangian method and a baseline MLP agent tested in `static-test`. We define survival time as the time cost of an unsuccessful episode (collision or exceeding a time limit of 80s). Traversal time, instead, is the time cost of a successful episode. With the same level of success rate, a longer survival time means that the agent tends to, at least, avoid collisions if it cannot succeed. To compare the safe RL method with classical

Success rate (%) (↑)	static env.			dynamic-box env.			dynamic-wall env.		
	$H=1$	$H=4$	$H=8$	$H=1$	$H=4$	$H=8$	$H=1$	$H=4$	$H=8$
MLP	65 ± 4	57 ± 7	42 ± 2	50 ± 5	35 ± 2	46 ± 3	67 ± 7	72 ± 1	69 ± 4
GRU	-	51 ± 2	43 ± 4	-	48 ± 4	45 ± 1	-	$\mathbf{82 \pm 4}$	78 ± 5
CNN	-	55 ± 4	45 ± 5	-	42 ± 5	40 ± 1	-	63 ± 3	43 ± 3
Transformer	-	$\mathbf{68 \pm 2}$	46 ± 3	-	$\mathbf{52 \pm 1}$	44 ± 4	-	33 ± 28	15 ± 13

TABLE I: (D1) Success rate (%) (↑) of policies trained with different neural network architectures and history lengths. H is the history length of the memory. Bold font indicates the best success rate for each type of environment.

Methods	Baseline (model-free)	Lagrangian method	MPC (model-based)	DWA	TEB
Success rate (%) (↑)	65 ± 4	74 ± 2	70 ± 3	**82**	70
Survival time (s) (↑)	8.0 ± 1.5	16.2 ± 2.5	55.7 ± 4.9	**62.7**	26.9
Traversal time (s) (↓)	$\mathbf{7.5 \pm 0.3}$	8.6 ± 0.2	24.7 ± 2.0	35.6	26.9

TABLE II: (D2) Success rate (↑), survival time (↑), and traversal time (↓) of policies trained with Lagrangian method, MPC with probabilistic transition model, and DWA. The bold font indicates the best number achieved for each type of metric.

navigation systems which are believed to have better safety, we also add evaluation metrics from a classical navigation stack with the Dynamic Window Approach (DWA) [2] local planner.

Lagrangian method reduces the gap between training and test environments. When deployed in the training environments, both the baseline MLP and the safe RL method achieves about 80% success rate. However, in the test environments, the Lagrangian method has a better success rate of 74% compare to 65% by the baseline MLP. We hypothesize that the safety constraint applied by the safe RL methods forms a way of regularization, and therefore, improves the generalization to unseen environments.

Lagrangian method increases the average survival time in failed episodes. As expected, the Lagrangian method increases the average survival time by $8.2s$ compared to the baseline MLP at a cost of $1.1s$ longer average traversal time. However, such improved safety are still worse than the classical navigation systems given the best survival time of $88.6s$ achieved by DWA.

C. Model-based RL (D2 and D3)

To explore how the model-based approaches help with the autonomous navigation tasks, we implement Dyna-style, MPC, and MBPO, and evaluate the methods in static environments. The transition models are either represented by a deterministic NN or a probabilistic NN that predicts the mean and variance of the next state. During the training in `static-train-50`, the policies are saved when 100k, 500k and 2000k transition samples are collected, then tested in `static-test`. The success rates of these policies are reported in Table IV.

Model-based methods do not improve sample efficiency. As shown in the second and third columns in Table IV, better success rates of 13% and 58% are achieved by the baseline MLP method provided by limited 100k and 500k transition samples respectively. In addition, Higher success rates at 500k transition samples are observed in probabilistic models compared to their deterministic counterparts, which indicates a more efficient learning with probabilistic transition models. Notice that MBPO exploits more heavily on the model compared to the Dyna-style method, which leads to much worse asymptotic performance (about 20% success rate in the end).

Model-based methods with probabilistic dynamic models improve the asymptotic performance. In the last column of Table IV, both Dyna-style and MPC with probabilistic dynamic models achieve slightly better success rates of 70% compared to 65% in the baseline MLP method when sufficient transition samples of 2000k are given to the learning agent.

The MPC policy performs conservatively when deployed in unseen test environments and shows a better safety performance. The safety performances of MPC policies with probabilistic dynamic models are also tested (see Table II). We observe that the agents with MPC policies navigate very conservatively with an average traversal time of $24.7s$, which is about two times more than the MLP baseline. In the meantime, MPC policies achieve improved safety with the best survival time of $55.7s$ among the RL-based methods.

D. Domain Randomization (D4)

To explore how model generalization depends on the degree of randomness in the training environments, baseline MLP policies with one history length are trained in the environment sets with 5, 10, 50, 100, and 250 training environments. The trained policies are tested in the same `static-test`. To investigate the performance gap between training and test, the policies trained with 50, 100, and 250 environments are also tested on `static-train-50`, which is part of their training sets. Fig 4 shows the success rate of policies trained with different number of training environments.

The generalization to unseen environments improves with increasing number of training environments. As shown in Fig. 4, the performances on the unseen test environments monotonously increase from 43% to 74% with the number of training environments increasing from 5 to 250. Moreover, the gaps between training and test environments gradually shrink by adding more training environments provided by that the polices are robust enough to maintain similar performances of about 80% on the training environ-

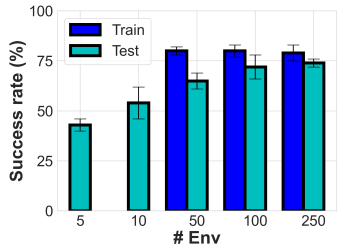

Fig. 4: (D4) Success rate (%) of policies trained with different number of training environments.

	H	# envs	real-world-1 traversal time (↓)	real-world-2 (# successful trials (↑) / total # trials)	real-world-3
MLP	1	50	6.9 (1/3)	10.6 (1/3)	N (0/3)
MLP	1	250	4.6 ± 0.8 (3/3)	$\mathbf{6.6 \pm 0.6}$ **(3/3)**	$\mathbf{22.6 \pm 0.5}$ **(3/3)**
Transformer	4	50	6.1 ± 0.4 (3/3)	6.1 ± 0.1 (2/3)	20.5 ± 2 (2/3)
Lagrangian	1	50	$\mathbf{4.4 \pm 0.6}$ **(3/3)**	7.1 ± 0.1 (2/3)	26.2 (1/3)
MPC	1	50	13.2 ± 0.7 (3/3)	24.8 ± 3.7 (3/3)	N (0/3)
DWA	-	-	16.2 ± 0.7 (3/3)	35.2 ± 8.2 (2/3)	66.9 ± 0.6 (3/3)

TABLE III: Physical experiments. The table shows the traversal time (s) (↓) and the number of successful trials (↑) of 5 RL-based navigation systems and a classical navigation system (DWA) evaluated in three real-world environments. The bold font indicates the best traversal time when all three trials are successful.

ments.

Transition samples	100k	500k	2000k
MLP	$\mathbf{13 \pm 7}$	$\mathbf{58 \pm 2}$	65 ± 4
Dyna-style deterministic	8 ± 2	30 ± 10	66 ± 5
MPC deterministic	0 ± 0	21 ± 10	62 ± 3
Dyna-style probabilistic	0 ± 0	48 ± 4	$\mathbf{70 \pm 1}$
MPC probabilistic	0 ± 0	45 ± 4	$\mathbf{70 \pm 3}$
MBPO	0 ± 0	0 ± 0	21.9 ± 3

TABLE IV: (D3) Success rate (%) (↑) of policies trained with different model-based methods and different number of transition samples. The bold font indicates the best success rate for each number of transition samples.

E. Physical experiments

To study the consistency of the above observations in simulation and the real world, we deploy one baseline MLP policy, one best policy for each studied desideratum, and one classical navigation system (DWA [2]) in the three real-world environments introduced in Sec. III-D. Each deployment is repeated three times, and the average traversal time and the number of successful trials are reported in Table. III. Even though the best memory-based policy, transformer architecture with 4 history length, was only marginally better than the baseline MLP in simulation, in the real world it can navigate very smoothly and fails only once in `real-world-2` and `real-world-3`, while baseline MLP fails most of the trials in all the environments including the benchmark-like environment. One possible reason for this is that simulations are typically more predictable than the real world. Therefore, it is particularly important to use historical data in the real world to estimate the environment and current states of the robot. Similarly, MLP policy trained with 250 environments can successfully navigate in all the environments without any failures, while baseline MLP trained with 50 environments fails most of the trials. Safe RL improves the chances of success in all the environments and can navigate more safely by performing backups and small adjustments of robots' poses. Similar to the simulation, MPC navigates very conservatively and succeeds in all the trials in `real-world-1` and `real-world-2`, but has much more difficulty generalizing to large-scale `real-world-3`.

V. Conclusion

In this section, we discuss the conclusions we draw from these benchmark experiments. We organize these conclusions by the desiderata as follows:

(D1) reasoning under uncertainty of partially observed sensory inputs does not obviously benefit from adding memory in simulated static environments and very random dynamic (`dynamic-box`) environments, but much more significant improvements were observed in the *real world* and in more *challenging dynamic environments* (`dynamic-wall`).

(D2) safety is improved by both safe RL and model-based MPC methods. However, classical navigation systems still achieve the best safety performance at a cost of very long traversal time. Whether RL-based navigation systems can achieve similar safety guarantees as classical navigation systems and whether safety can be improved without significantly sacrificing the traversal time are still open questions.

(D3) the ability to learn from limited trial-and-error data is not improved by the evaluated model-based methods. Currently, we observe that model-based RL methods indeed improve sample-efficiency, but only when the number of imaginary rollouts from the learned model is large (e.g. $\geq 2000k$) and when they are sampled with randomness. We therefore hypothesize that the improvement comes from the robustness brought by learning on more data sampled from the learned model. Hence, this result motivates not only more accurate model learning for reducing the number of imaginary rollouts, but also theoretical understanding of how the model helps improve the robustness or even safety of navigation.

(D4) the generalization to diverse and novel environments is improved by increasing the randomness of training environments. However, a noticeable gap of about 5% between training and test environments is not eliminated by further increasing the number of training environments to 250. This reflects the limitation of simple *domain randomization* to increase the generalization, which is, however, widely used by the community.

In summary, although the proposed benchmark is not intended to represent every real-world navigation scenario, it serves as a simple yet comprehensive testbed for RL-based navigation methods. We observed that for every desideratum, no method can achieve 100% success rate on all *training* environments. Even though we ensured that we have made sure that every environment is indeed individually solvable. This alone indicates that there exists an optimization and generalization challenge when we have a large number of training environments as in our proposed benchmark.

REFERENCES

[1] 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.

[2] 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.

[3] 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 barn challenge at icra 2022," *arXiv preprint arXiv:2208.10473*, 2022.

[4] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, pp. 1–29, 2022.

[5] Y. Chow, O. Nachum, A. Faust, M. Ghavamzadeh, and E. A. Duéñez-Guzmán, "Lyapunov-based safe policy optimization for continuous control," *CoRR*, vol. abs/1901.10031, 2019. [Online]. Available: http://arxiv.org/abs/1901.10031

[6] G. Thomas, Y. Luo, and T. Ma, "Safe reinforcement learning by imagining the near future," 2022.

[7] E. Rodríguez-Seda, D. Stipanovic, and M. Spong, "Lyapunov-based cooperative avoidance control for multiple lagrangian systems with bounded sensing uncertainties," in *2011 50th IEEE Conference on Decision and Control and European Control Conference, CDC-ECC 2011*, ser. Proceedings of the IEEE Conference on Decision and Control, Dec. 2011, pp. 4207–4213, 2011 50th IEEE Conference on Decision and Control and European Control Conference, CDC-ECC 2011 ; Conference date: 12-12-2011 Through 15-12-2011.

[8] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *ICML*, 2019.

[9] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," *arXiv preprint arXiv:1912.01588*, 2019.

[10] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," *arXiv: Learning*, 2018.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[12] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *SIGART Bull.*, vol. 2, no. 4, p. 160–163, jul 1991. [Online]. Available: https://doi.org/10.1145/122344.122377

[13] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[14] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *AAAI Fall Symposia*, 2015.

[15] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Solving deep memory pomdps with recurrent policy gradients," in *ICANN*, 2007.

[16] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[24] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, jun 2013.

[25] 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, pp. 2007–2014, 2019.

[26] A. Wahid, A. Toshev, M. Fiser, and T.-W. E. Lee, "Long range neural navigation policies for the real world," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 82–89, 2019.

[27] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[28] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[29] L. Harries, S. Lee, J. Rzepecki, K. Hofmann, and S. Devlin, "Mazeexplorer: A customisable 3d benchmark for assessing generalisation in reinforcement learning," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–4.

[30] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[31] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, 2020.

[32] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.