2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)
New York, USA, October 25-27 2021

Machine Learning Methods for Local Motion Planning: A Study of End-to-End vs. Parameter Learning

Zifan Xu[1], Xuesu Xiao[1], Garrett Warnell[1,2], Anirudh Nair[1], and Peter Stone[1,3]

Abstract— **While decades of research efforts have been devoted to developing classical autonomous navigation systems to move robots from one point to another in a collision-free manner, machine learning approaches to navigation have been recently proposed to learn navigation behaviors from data. Two representative paradigms are end-to-end learning (directly from perception to motion) and parameter learning (from perception to parameters used by a classical underlying planner). These two types of methods are believed to have complementary pros and cons: parameter learning is expected to be robust to different scenarios, have provable guarantees, and exhibit explainable behaviors; end-to-end learning does not require extensive engineering and has the potential to outperform approaches that rely on classical systems. However, these beliefs have not been verified through real-world experiments in a comprehensive way. In this paper, we report on an extensive study to compare end-to-end and parameter learning for local motion planners in a large suite of simulated and physical experiments. In particular, we test the performance of end-to-end motion policies, which directly compute raw motor commands, and parameter policies, which compute parameters to be used by classical planners, with different inputs (e.g., raw sensor data, costmaps), and provide an analysis of the results.**

I. INTRODUCTION

For several decades now, autonomous vehicle navigation systems have been a topic of study and development in the the robotics research community. While a large and varied body of literature on the topic exists, the paradigm most commonly employed in deployed autonomous navigation systems is one of hierarchical planning, in which a high-level global planner computes a feasible path from the vehicle's position to a specified destination, and a low-level motion controller [1]–[3] then seeks to move the vehicle along that path while satisfying local constraints such as collision avoidance. To instantiate the levels of this hierarchy, system designers have classically employed hand-crafted and engineered techniques from the planning and optimal control communities [1]–[3]. Increasingly, however, the research community has begun to consider a fundamentally different way to design such systems, i.e., through the use of tools from machine learning.

[1] The Department of Computer Science, The University of Texas at Austin, Austin, TX 78712 {zfxu, ani.nair}@utexas.edu, {xiao, warnellg, pstone}@cs.utexas.edu [2]The Computational and Information Sciences Directorate, Army Research Laboratory [3]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.

Incorporating machine learning into the design process for autonomous vehicle navigation systems is a topic that has recently seen an explosion of popularity within the research community [4]. Broadly speaking, the literature on this topic seeks to leverage experiential data (gathered, e.g., in simulation or the real-world) and, using that data and tools from the machine learning community, automatically compute autonomous navigation systems [5]–[11]. While this general approach can be instantiated in a number of ways, we are interested here in two particular formulations of the problem. The first formulation we consider is that in which machine learning is used in an end-to-end fashion, i.e., what is learned is a single function called a *motion policy* that directly maps sensor measurements to low-level motor commands [5]. In contrast, the second formulation we are concerned with here is that of parameter learning, i.e., what is learned is a higher-level function called a *parameter policy* that maps sensor measurements to hyperparameter values for a classical motion control system [12]–[16].

These two formulations, i.e., end-to-end learning and parameter learning, constitute two fundamentally different approaches to the application of machine learning to autonomous navigation system design. For example, especially in the context of modern neural network function approximators, it is relatively easy to envision a motion policy improving continually with more and more data, whereas, with parameter learning, it is less clear if the classical motion control component imposes limits on this type of improvement. On the other hand, by employing engineered motion controllers, there is perhaps reason to believe that the parameter learning approach may generalize better to new environments, especially with limited access to training data. Additionally, the local controllers used in the parameter learning approach typically exhibit desirable properties such as provable safety guarantees and certain notions of explainability, whereas motion policies typically do not. Given the postulated differences between end-to-end and parameter learning described above, it is natural to ask: are these hypotheses borne out empirically?

In this paper, we seek to provide an answer to the above question by performing a systematic experimental comparison of end-to-end learning and parameter learning for autonomous navigation systems. More specifically, we use reinforcement learning (RL) to train both an end-to-end motion policy and a parameter policy in simulation. We vary the number of training environments, and evaluate the performances on a held-out set of testing environments. We quantify the results using a variety of navigation metrics,

2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) | 978-1-6654-1764-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSRR53300.2021.9597689

and the results allow us to analyze the extent to which each approach generalizes to new environments. Additionally, we deploy the learned systems on a physical robot in the real-world in order to validate the approaches' sim-to-real transferability.

We find that, in general, the end-to-end learning approach does not generalize as well to novel navigation environments as the parameter learning approach, even given a very large set of training environments. However, the end-to-end learning approach achieves the fastest traversal time in the training environments. The parameter learning approach requires much less data and demonstrates performance improvement in several navigation metrics even for the smallest number of training environments, with performance gradually increasing with the size of the training set.

II. BACKGROUND

This section reviews existing work on classical and end-to-end learning-based navigation planners, with a focus on how they are compared with each other.

A. Classical Planners

Mobile robot navigation addresses the problem of moving a robot from one point to another in a collision-free manner. Given a rough global path from a coarse and fast global planner (e.g., Dijkstra's [17], A* [18], D* [19] algorithms), the robotics community has developed many different classical local planners to plan motion commands to follow the global path, to minimize time and energy to get to the goal, and to maintain clearance to obstacles. In general, there exist two schools of classical local planners: sampling-based and optimization-based planners. Sampling-based planners, such as the DWA planner [1], sample feasible commands and finds the optimal one according to a pre-defined cost function; optimization-based planners, such as the E-BAND planner [2], start with an initial local trajectory and runs optimization until the trajectory converges.

Oftentimes, successfully deploying these classical planners requires extensive engineering effort [3]. For example, planner parameters, such as obstacle inflation radius, maximum velocities, and optimization objective weights, need to be manually tuned to fit specific navigation scenarios. To relax this dependency on robotics expertise, automatic parameter tuning methods have been proposed [20]. More recently, Adaptive Planner Parameter Learning (APPL) [12] has gone beyond static planner parameters and extended to dynamic and adaptive parameters for classical local planners, leveraging teleoperated demonstration [13], corrective interventions [14], evaluative feedback [15], and reinforcement learning [16]. Adaptive parameters allow classical planners to achieve optimal performance in different regions of a deployment environment. In this study, for a fair comparison against self-supervised end-to-end planners, we adopt reinforcement learning [16] to learn *parameter policies* that compute adaptive planner parameters to be used by the underlying classical planners at every time step.

B. End-to-End Learning-Based Planners

In spite of the success of classical navigation planners, end-to-end machine learning techniques have been recently proposed to directly produce motion commands from raw sensory input, such as LiDAR scans or depth images. Most such end-to-end *motion policies* combine the classical cascaded global and local planning paradigm into one piece, along with their corresponding components, including costmap, sampling, optimization, etc. As described in Xiao, et al.'s recent survey [4], approaches based on end-to-end learning have achieved navigation capable of obstacle avoidance [5]–[11], terrain negotiation [21], [22], and social navigation [23], [24], while minimizing engineering effort by learning in a data-driven fashion. However, these end-to-end systems are typically thought to be data-hungry and to generalize poorly to new environments. Moreover, unlike classical systems, they lack explainability and provable safety guarantees. To address some of these shortcomings, Pfeiffer, et al. [25] and Xiao, et al. [4] have both suggested not to replace global planners, but only local planners, with end-to-end learning, especially for large and complex navigation scenes. In terms of navigation performance, Xiao, et al. have pointed out that end-to-end learning-based navigation planners are rarely compared with their classical counterparts [4].

Motivated by this lack of performance comparison between parameter learning approaches based on classical planners and end-to-end learning approaches, this paper focuses on local planning and presents a comprehensive study of both paradigms within a plethora of simulated environments in the Benchmark Autonomous Robot Navigation (BARN) dataset [26] and different physical environments.

III. TESTED APPROACHES

This section details the two tested approaches: end-to-end learning and parameter learning. We test the approaches on a specially designed navigation task described in Sec. III-A with the same continuous Reinforcement Learning (RL) algorithm (Sec. III-D). Sec. III-B and III-C detail how we model the navigation task as Partially Observable Markov Decision Processes (POMDPs) represented by tuples $(\mathcal{S}, \mathcal{O}, \mathcal{A}_e, \mathcal{T}_e, \gamma_e, R_e)$ and $(\mathcal{S}, \mathcal{O}, \mathcal{A}_p, \mathcal{T}_p, \gamma_p, R_p)$ for end-to-end learning and parameter learning respectively. The two approaches share the same states $\mathcal{S}$ and observations $\mathcal{O}$, but different continuous actions $\mathcal{A}$, transition dynamics $\mathcal{T}$, reward functions R, and discount factor γ. We will detail them in the corresponding sections. Sec. III-D describes the RL algorithm and the distributed framework that enables the training over multiple environments.

A. Task Definition

We test our approaches using a goal-oriented local *obstacle-avoidance* (OA) navigation task defined in the BARN dataset [26]. Fig. 2 shows three environments in the dataset. The goal of this task is not to test a local planner's ability to follow a long-term global plan, but instead to evaluate

the planner's ability to navigate a robot through obstacle-occupied environments quickly and safely. As shown in Fig. 2, the environments have relatively short navigation scenarios (5m $\times$ 5m), where the robot navigates across obstacle fields with varying densities and structures. Such a navigation scenario can be thought of as a segment of a long-term navigation task, and the learned policy will still have the ability to follow any global plan. Each environment with a pre-defined starting position (x_i, y_i) uniquely defines a single OA task that navigates a robot in the direction of the y-axis (indicated by the blue arrow in Fig. 2). The goal point for the global planner is set at a consistent point beyond a finish line $y_f = 5$m. Given the current location (x_t, y_t), the agent will succeed at the task when it reaches the finish line or $y_t > y_f$.

The objective of OA is to minimize the time cost T of reaching the finish line and the number of collisions C during an episode with a time limit T_{max}. The reward function for OA is:

$$R_t = +20 \cdot R_g + 1 \cdot R_p + 0.1 \cdot R_c, \tag{1}$$

where $R_g = \mathbb{1}(y_t > y_f)$ is the indicator function of reaching the finish line, and $R_p = y_{t+1} - y_t$ is a dense shaping reward of the change of the y coordinate that encourages local progress. The third term R_c is a penalty reward for collisions:

$$R_c = \begin{cases} -1/(d_t + 0.05) & d_t < 0.3 \\ 0 & d_t \geq 0.3 \end{cases}, \tag{2}$$

where d_t is the distance to the closest obstacle at time step t represented by the minimum element in a vector of LiDAR scans, and we use a cutoff distance of 0.3m which is empirically chosen based on the the geometric size of the robot we experiment on. A small number (0.05) is added to the denominator to avoid numerical instability. The coefficients of the three terms are selected so that the policies can learn desirable navigation behaviors for both approaches. Since the parameter learning approach assures safety (collision-free) with the classical local planner, we do not apply the penalty reward term to the parameter learning approach. We will show in Sec. IV that this change will not deteriorate the collision avoidance behavior.

B. Parameter Learning Approach

We first describe the common state and observation spaces between the two approaches. The state, $s = (x, y) \in \mathcal{S}$, represents the actual location of the robot, where (x, y) follows the same Cartesian coordinate axis defined by the obstacle field. An observation, $o = (\bar{o}, p)$, is composed of a perceptual observation $\bar{o}$ and a planned global path p. We compare systems that utilize two different types of perceptual observations: LiDAR scan o_l and costmap o_c. A LiDAR scan observation, $o_l \in \mathcal{O}_l = \{R\}^{N_l}$, is a N_l-length vector from an onboard LiDAR sensor. We instantiate the global path p with a local goal g, which is a point 1m away from the robot on the global path. We append its angle relative to the orientation of the robot, i.e., $\psi = \texttt{arctan2}(g_y, g_x)$. Therefore, the observation with LiDAR scan perception becomes $o = (o_l, \psi) \in \{R\}^{N_l+1}$. The observation with costmap perception, $o = o_c \in \mathcal{O}_c = \{0, 0.5, 1\}^{N_c \times N_c}$, is a $N_c \times N_c$ 2D matrix of uninflated costmap, with 0, 0.5, and 1 denoting the free space, global path, and obstacles, respectively. Fig. 1 shows an example of costmap perception.

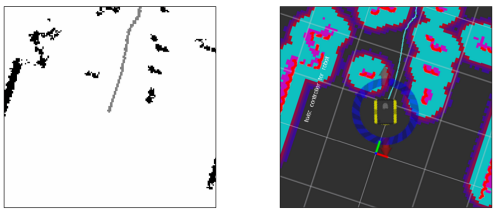

Fig. 1. An example of costmap perception represented as a grayscale image (left) and the corresponding robot visualized in the simulation (right).

The parameter learning approach utilizes a parameter policy trained by RL to dynamically adjust the planner parameters $\theta \in \Theta$ of a classical planner f. The POMDP under this context denotes a *meta-environment* composed of both the underlying navigation world $\mathcal{W}$ (the physical, obstacle-occupied world) and the classical planner f. At each time step t, the agent receives an observation o_t and executes an action $a_t = \theta_t \in \mathcal{A}_p$. Then the environment will transition to the next time step $s_{t+1}, o_{t+1} \sim \mathcal{T}_p(\cdot|s_t, \theta_t)$ by navigating with the classical planner f for a fix period of time, $\Delta t_p = 1$s, under the selected planner parameter set θ_t.

C. End-to-End Learning Approach

The POMDP for the end-to-end learning approach shares the same state and observation space as we described in Sec. III-B. The motion policy learned by this approach controls the robot by selecting a two-dimensional continuous action vector, $a = (v, \omega) \in \mathcal{A}_e$ that encodes the robot's linear and angular velocity. Similar to the parameter learning approach, at each time step t, the agent receives an observation o_t and executes an action a_t. Then the environment will transition to the next time step $s_{t+1}, o_{t+1} \sim \mathcal{T}_e(\cdot|s_t, \theta_t)$ by executing the actual linear and angular velocity for a fix period of time $\Delta t_e = 0.2$s.

D. Learning Algorithm

Both approaches are trained by TD3 [27], an off-policy algorithm based on the common actor-critic framework. To compare between the two approaches, we use the same neural network architectures inherited from APPLR [16]. More specifically, we employ multilayer perceptrons (MLP) with two 512-hidden-unit layers to process the LiDAR scan observation. For the costmap observation, a light-weight convolution neural network (CNN) is employed to learn the embedded features of the image, then followed by the same MLP neural network as the LiDAR scan. We denote the structure of the CNN as sequential layers: **Input** $\rightarrow$ **Conv2D[16, 8, 4]** $\rightarrow$ **Conv2D[32, 4, 2]** $\rightarrow$ **Conv2D[32, 3, 1]** $\rightarrow$ **FC[512]** $\rightarrow$ **FC[512]**, where **FC** is the fully connected layers with the number denoting the number of hidden units, and **Conv2D** is the 2D convolution neural network with the numbers denoting the number of channels, kernel size, and stride respectively.

We study generalizibility of the policies learned over different sizes of training sets. We define a set of navigation environments with different obstacle fields as a training set. To learn from such a training set with multiple environments simultaneously, we implement a distributed training framework (similar to Gorila [28]) with one serial learner and multiple actors running the environments in parallel. Then, one or multiple actors run one environment to generate the experiences in the global replay buffer.

We learn the hyper-parameters with APPLR [16]. For a fair comparison, we use the same hyper-parameters for the end-to-end learning approach except the discount factor: we use a discount factor γ of 0.99 accounting for the shorter time step in the end-to-end learning approach, in contrast to 0.95 for the parameter learning approach.

IV. TEST RESULTS

We compare learning performance of three approaches: parameter policy with LiDAR input (*params_lidar*), parameter policy with costmap input (*params_costmap*), and end-to-end motion policy with LiDAR input (*e2e_lidar*). We do not include end-to-end motion policy with costmap input (*e2e_costmap*) results due to its poor performance, but we discuss possible reasons. In Sec. IV-A, we provide simulation details. In Sec. IV-B, we present test results of the three approaches under three different training environment sizes. In Sec. IV-C, we present test results on a physical robot.

A. Simulation Specifications

We implement all the approaches on a simulated ClearPath Jackal differential-drive ground robot. The specifications of this simulated robot are kept exactly the same as in APPLR [16], including the ROS `move_base` navigation stack [29] and classical local planner DWA [1]. We query the costmap from the navigation stack. The original costmap is first rotated to align with the current robot orientation, and then clipped as a 84 $\times$ 84 matrix centered at the robot, which covers roughly the same range as the clipped LiDAR scan.

The OA tasks are defined in the BARN dataset [26] with 300 simulated navigation environments. The obstacle fields in the environments cover different navigation difficulty levels, ranging from relatively open environments to highly-constrained spaces (three example environments with different difficulties are shown in Fig. 2). Given 300 environments, we randomly select 50 environments as the evaluation set. Then, from the remaining 250 environments, we randomly select 10, 50 and all 250 environments as three different training configurations. We train the policies simultaneously using a distributed training framework on a computing cluster. For each of the training process, we run 250 actors in parallel, each assigned with one CPU core, and equally allocate the actors to the environments, i.e., the 10-environment configuration has 25 actors running per environment.

B. Simulated Tests

The three approaches, *params_lidar*, *params_costmap*, and *e2e_lidar*, are trained in the three training configurations

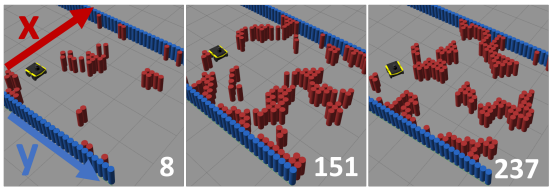

Fig. 2. Indexed Example Navigation Environments in BARN [26]. y-axis is the pre-defined navigation direction. Higher indices (ranging from 1-300) represent more difficult environments.

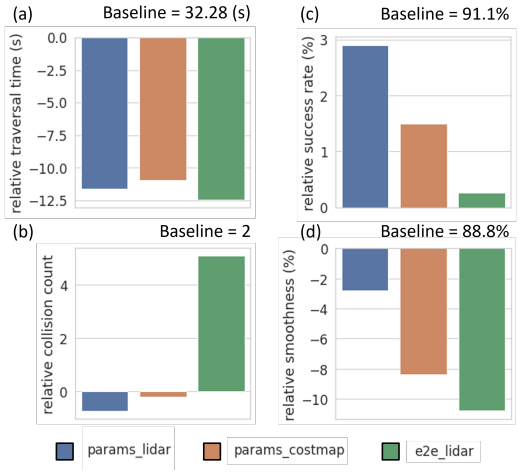

Fig. 3. Test results of the policies on the held-out evaluation environments trained in 250 training environments. The relative performances of the metrics are reported by subtracting the metrics' values of the classical DWA planner (the values are indicated as *Baselines* on the top of each figure). Fig. (a), (b) are the relative traversal time and collision (the lower the better), and Fig. (c), (d) are the relative success rate and smoothness (the higher the better).

with 10, 50 and 250 environments. After training with one training configuration, we evaluate the policy on both the held-out evaluation environments and the original training environments with 20 trials per environment. The following metrics are used to evaluate the navigation performance:

- **Traversal time:** the average total time used to finish the task. Only the successful trials are counted.
- **Success rate:** the percentage of successful trials.
- **Collisions:** the average number of collisions per trial. Note that we use the same cutoff distance (0.3m) as in Eqn. 2 to determine a "soft" collision, which indicates a risky state rather than an actual collision.
- **Smoothness:** the percentage of positive linear velocity commands in all velocity commands.

We begin with the maximum number (250) of training environments. As shown in Fig. 3(a), the three policies achieve similar traversal time, which is about 10s less than the baseline DWA planner. Additionally, as the collision metric shows in Fig. 3(b), in spite of the significant improvements in terms of traversal time, the *params_lidar* and *params_costmap* policies do not sacrifice the collision-free guarantee from classical local planners. The *e2e_lidar* policy, however, undergoes severe degradation in terms of collision avoidance. This degradation is also reflected in the success rate metric (Fig. 3(c)), for which the *e2e_lidar* policy suffers from the worse performance among the three policies. Interestingly, for all three approaches, smoothness

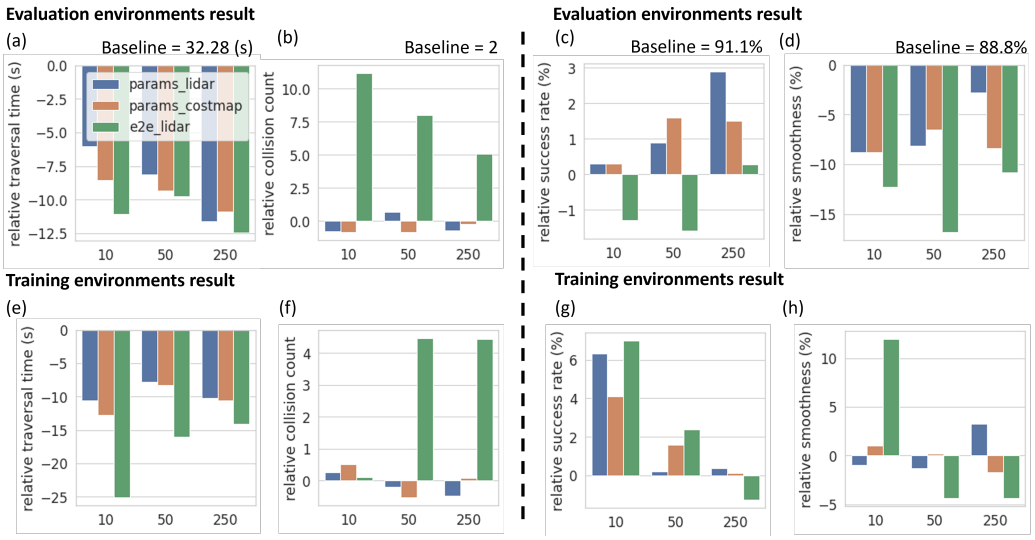

Fig. 4. Test results of the policies on both evaluation environments (top row) and training environments (bottom row) under three training sizes: 10, 50 and 250 environments. The relative performances metrics are reported by subtracting the metrics' values of the classical DWA planner (the values are indicated as *Baselines* at the top of evaluation result figures). The metrics on the left-hand side of the dashed line are the lower the better, while the metrics on the right-hand side are the higher the better.

drops compared to the default DWA, which we hypothesize as an inevitable cost to achieve faster traversal time.

Fig. 4 extends test results to all three training configurations, and the results on both the evaluation environments and training environments are reported. The experiments with 10 training environments is to test possible generalization under a limited number of training environments. After training, in the training environments (bottom row of Fig. 4), *e2e_lidar* achieves the best performance with about 25s reduction of traversal time, almost the same amount of collision, more than 6% increase of success rate, and more than 10% improvement of smoothness compared to the default DWA planner (the first green column in each bottom figure in Fig. 4), which indicates a very good learning results with this training configuration. However, *e2e_lidar* suffers from catastrophic failure in the held-out unseen evaluation environments (the first green column in each top-row figures). Instead, the *params_lidar* and *params_costmap* policies demonstrate considerable improvements in terms of traversal time and success rate and maintains nearly the same performance for other metrics in the training environments. Only a small degradation is shown in the evaluation environments even with only 10 training environments. By analyzing the change of these metrics with increasing number of training environments, we can see a general tendency of improvement in terms of all the metrics in the evaluation environments, which indicates a better generalization with more environments seen during training. We also notice a relatively stable collision count over all the tests for *params_lidar* and *params_costmap*, which shows that tuning the parameters of a classical local planner inherits its collision-free assurance.

To study the effect of two types of perceptual observations, we compare the two variants of parameter learning approach: *params_lidar* and *params_costmap*. As shown in Fig. 4(a) and 4(c), better traversal time and success rate are achieved by *params_costmap* with 10 and 50 training environments.

TABLE I

AVERAGE TRAVERSAL TIME AND SUCCESS RATE

	Easy	Medium	Hard
params_lidar	12.42s (3/3)	24.23s (3/3)	23.40s (3/3)
params_costmap	11.73s (3/3)	14.65s (3/3)	19.04s (3/3)
e2e_lidar	11.25s (3/3)	15.17s (1/3)	NA (0/3)
DWA	26.46s (3/3)	29.80s (3/3)	53.66s (3/3)

This better generalization is possibly because the costmap contains rich historical information and acts as a better representation of the agent's state. This better representation facilitates the generalization to unseen environments. Using *e2e_costmap*, the robot fails to learn a collision-avoidance policy. We hypothesize possible explanations as follows: the image data of the costmap usually takes longer to query and process, which may lead to instability between time steps. Given higher frequency of the end-to-end motion policies (5Hz v.s. 1Hz for parameter policies), the motion policies become more sensitive to this instability, which may account for the failure of *e2e_costmap*.

C. Physical Tests

We deploy the policies, including the default DWA system, in three real-world navigation environments with different difficulty levels. We label them as easy, medium and hard respectively, as shown in Fig. 5. We launch three trials for each policy and measure traversal time for the successful trials. The average traversal time and the success rate (in parentheses) are reported in Table I.

As expected, the end-to-end motion policy shows poor generalization. In the three navigation environments, the end-to-end motion policy only exhibits successful navigation in the easy environment and fails two and all three trials in the medium and hard environment, respectively. Among the successful trials, *e2e_lidar* achieves the best traversal time, which is consistent with the simulation results. Interestingly, the *params_costmap* policy outperforms the *params_lidar* policy in all three environments. This is possibly accounted

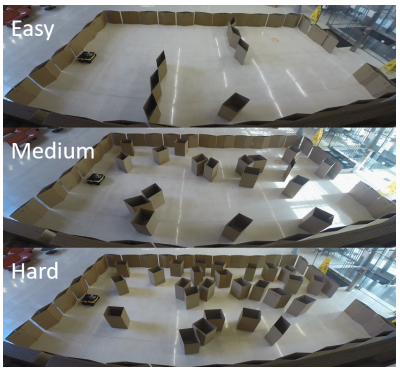

Fig. 5. Real-world navigation environments with three difficulty levels.

for by the increasing raw sensor noise level in the real-world. The costmap, on the other hand, is built upon a history with a longer time horizon, which is more robust to real-world sensor noise. In general, the two parameter policies demonstrate desirable generalization to real-world navigation, while the end-to-end motion policy suffers catastrophic failures similar to the simulation results.

V. CONCLUSIONS

In this paper, we systematically tested the generalizability of an end-to-end learning approach and a parameter learning approach for autonomous navigation systems. Our results indicated that the parameter learning approach has better generalizibility by performing well on the evaluation set with limited access to training data. Additionally, such generalizibility kept improving with increasing size of training data. In contrast, despite very good learning results on the training environments, catastrophic failures, i.e., collisions, were common in the evaluation environments for end-to-end learning policies, even when being trained on a large set of training environments. Real-world physical test results were consistent with simulation results. While these results are based on manually selected hyper-parameters, e.g., coefficients in the reward function and training parameters, how the hyper-parameters influence the generalizibility remains to be investigated in the future.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] A. Francis, A. Faust, H.-T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T.-W. E. Lee, "Long-range indoor navigation with prm-rl," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1115–1134, 2020.

[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] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Toward agile maneuvers in highly constrained spaces: Learning from hallucination," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1503–1510, 2021.

[9] 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.

[10] Z. Wang, X. Xiao, A. J. Nettekoven, K. Umasankar, A. Singh, S. Bommakanti, U. Topcu, and P. Stone, "From agile ground to aerial navigation: Learning from learned hallucination," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[11] H. Karnan, G. Warnell, X. Xiao, and P. Stone, "Voila: Visual-observation-only imitation learning for autonomous navigation," *arXiv preprint arXiv:2105.09371*, 2021.

[12] X. Xiao, Z. Wang, Z. Xu, B. Liu, G. Warnell, G. Dhamankar, A. Nair, and P. Stone, "Appl: Adaptive planner parameter learning," *arXiv preprint arXiv:2105.07620*, 2021.

[13] 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.

[14] 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.

[15] Z. Wang, X. Xiao, G. Warnell, and P. Stone, "APPLE: Adaptive planner parameter learning from evaluative feedback," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 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] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136

[19] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[20] D. Teso-Fz-Betoño, E. Zulueta, U. Fernandez-Gamiz, A. Saenz-Aguirre, and R. Martinez, "Predictive dynamic window approach development with artificial neural fuzzy inference improvement," *Electronics*, vol. 8, no. 9, p. 935, 2019.

[21] 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*, 2019.

[22] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, 2021.

[23] 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.

[24] R. Mirsky, X. Xiao, J. Hart, and P. Stone, "Prevention and resolution of conflicts in social navigation–a survey," *arXiv preprint arXiv:2106.12113*, 2021.

[25] 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.

[26] 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.

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

[28] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. D. Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, and D. Silver, "Massively parallel methods for deep reinforcement learning," 2015.

[29] OSRF, "Ros wiki move_base," http://wiki.ros.org/move_base, 2018.