# APPLE: Adaptive Planner Parameter Learning From Evaluative Feedback

Zizhao Wang, Xuesu Xiao, Garrett Warnell, and Peter Stone

***Abstract*—Classical autonomous navigation systems can control robots in a collision-free manner, oftentimes with verifiable safety and explainability. When facing new environments, however, fine-tuning of the system parameters by an expert is typically required before the system can navigate as expected. To alleviate this requirement, the recently-proposed Adaptive Planner Parameter Learning paradigm allows robots to *learn* how to dynamically adjust planner parameters using a teleoperated demonstration or corrective interventions from non-expert users. However, these interaction modalities require users to take full control of the moving robot, which requires the users to be familiar with robot teleoperation. As an alternative, we introduce APPLE, Adaptive Planner Parameter Learning from *Evaluative Feedback* (real-time, scalar-valued assessments of behavior), which represents a less-demanding modality of interaction. Simulated and physical experiments show APPLE can achieve better performance compared to the planner with static default parameters and even yield improvement over learned parameters from richer interaction modalities.**

***Index Terms*—Autonomous vehicle navigation, learning from demonstration, motion and path planning.**

## I. INTRODUCTION

MOBILE robot navigation is a well-studied problem in the robotics community. Many classical approaches have been developed over the last several decades and several of them have been robustly deployed on physical robot platforms moving in the real world [1], [2], with verifiable guarantees of safety and explainability.

However, prior to deployment in a new environment, these approaches typically require parameter re-tuning in order to achieve robust navigation performance. For example, in cluttered environments, a low velocity and high sampling rate are necessary in order for the system to be able to generate safe and smooth motions, whereas in relatively open spaces, a large maximum velocity and relatively low sampling rate are needed in order to achieve optimal navigation performance. This parameter re-tuning requires robotics knowledge from experts who are familiar with the inner workings of the underlying navigation system, and may not be intuitive for non-expert users [3]. Furthermore, using a single set of parameters assumes the same set will work well on average in different regions of a complex environment, which is often not the case.

To address these problems, Adaptive Planner Parameter Learning (APPL) is a recently-proposed paradigm that opens up the possibility of dynamically adjusting parameters to adapt to different regions, and enables non-expert users to fine-tune navigation systems through modalities such as teleoperated demonstration [4] or corrective interventions [5]. These interaction modalities require non-expert users to take full control of the moving robot during the entire navigation task, or, at least, when the robot suffers from poor performance. However, non-expert users who are inexperienced at controlling the robot may be unwilling or unable to take such responsibility due to perceived risk of human error and causing collisions.

Fortunately, even non-expert users who are not willing to take control of the robot are typically still able to observe the robot navigating and provide real-time positive or negative assessments of the observed navigation behavior through *evaluative feedback*. For example, even non-expert users can know to provide negative feedback when a robot is performing poorly, e.g., getting stuck in highly constrained spaces [6], [7] or driving unnecessarily slowly in open spaces [8]. This more-accessible modality provides an interaction channel for a larger community of non-expert users with mobile robots (Fig. 1).

In this work, we introduce a machine learning method that can leverage evaluative feedback in the context of autonomous navigation called *Adaptive Planner Parameter Learning from Evaluative Feedback* (APPLE). Based on a parameter library [4], [5] or a parameter policy [9], APPLE learns how to choose

Manuscript received February 24, 2021; accepted July 12, 2021. Date of publication July 30, 2021; date of current version August 20, 2021. This letter was recommended for publication by Associate Editor L. Tapia and Editor S. J. Guy upon evaluation of the reviewers' comments. This work has taken place in the Learning Agents Research Group (LARG) at the University of Texas at Austin. LARG research was supported in part by grants from the National Science Foundation under Grants CPS-1739964, IIS-1724157, and NRI-1925082, in part by the Office of Naval Research under Grant N00014-18-2243, in part by the Future of Life Institute under Grant RFP2-000, in part by Army Research Office under Grant W911NF-19-2-0333, in part by DARPA, in part by Lockheed Martin, in part by General Motors, and in part by Bosch. The views and conclusions contained in this document are those of the authors alone. The work of P. 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 and objectivity in research. *(Corresponding author: Zizhao Wang.)*

Zizhao Wang is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas 78712 USA (e-mail: zizhao.wang@utexas.edu).

Xuesu Xiao is with the Department of Computer Science, University of Texas at Austin, Austin, Texas 78712 USA (e-mail: xiao@cs.utexas.edu).

Garrett Warnell is with the Department of Computer Science, University of Texas at Austin, Austin, Texas 78712 USA, and also with the Computational and Information Sciences Directorate, Army Research Laboratory, Austin, Texas 78712 USA (e-mail: garrett.a.warnell.civ@mail.mil).

Peter Stone is with the Department of Computer Science, University of Texas at Austin, Austin, Texas 78712 USA, and also with Sony AI (e-mail: pstone@cs.utexas.edu).

Digital Object Identifier 10.1109/LRA.2021.3100940

2377-3766 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.

Fig. 1. For non-expert users who are unable or unwilling to take control of the robot, evaluative feedback, e.g. *good job* (green thumbs up) or *bad job* (red thumbs down), is a more accessible human interaction modality, but still valuable for improving navigation systems during deployment.

appropriate navigation planner parameters at each time step in order to adapt to different parts of the deployment environment. Specifically, APPLE treats the scalar human feedback as the value for the state-action pair in the Reinforcement Learning framework during training, in which the action is the parameter set to be used by the underlying navigation system. During deployment, APPLE selects the parameters to maximize the expected human feedback value. We implement APPLE both in the Benchmarking Autonomous Robot Navigation (BARN) [10] environments and also in real-world, highly constrained obstacle courses. In both training and unseen environments, APPLE is able to outperform the planner with default parameters, and even to improve over APPL variants learned from richer interaction modalities, such as teleoperated interventions. Our experimental results indicate that evaluative feedback is a particularly valuable form of human interaction modality for improving navigation systems during deployment.

## II. RELATED WORK

In this section, we review existing work on machine learning for mobile robot navigation, adaptive planner parameters, and learning from human evaluative feedback.

### A. Learning for Navigation

While autonomous navigation has been studied by the robotics community for decades, machine learning approaches have recently been extensively applied to this problem as well. Xiao, *et al.* [11] presented a survey on using machine learning for motion control in mobile robot navigation: while the majority of learning approaches tackle navigation in an end-to-end manner [12], [13], it was found that approaches using learning in conjunction with other classical navigation components were more likely to have achieved better navigation performance. These methods included those that learned sub-goals [14], local planners [6]–[8], [15], [16], or planner parameters [4], [5], [9], [17]–[19]. Learning methods have also enabled navigation capabilities that complement those provided in the classical navigation literature, including terrain-aware [20]–[22] and social [23]–[25] navigation.

APPLE leverages the aforementioned hybrid learning and classical architecture, where the learning component only learns to select appropriate set of planner parameters, and interacts with the underlying classical navigation system.

### B. Adaptive Parameters for Classical Navigation

Considering classical navigation systems' verifiable safety, explainability, and stable generalization to new environments, and the difficulty in fine-tuning those systems, learning adaptive planner parameters is an emerging paradigm of combining learning and planning. Examples include finding trajectory optimization coefficients using Artificial Neural Fuzzy Inference Improvement [17], optimizing two different sets of parameters for straight-line and U-turn scenarios with genetic algorithms [19], or designing novel systems that can leverage gradient descent to match expert demonstrations [18]. Recently, the APPL paradigm [4], [5], [9] has been proposed, which further allows parameters to be appropriately adjusted during deployment "on-the-fly", in order to adapt to different regions of a complex environment. APPL also learns from non-expert users using teleoperated demonstration [4], corrective interventions [5], or trial-and-error in simulation [9].

APPLE utilizes an accessible but sparse modality of human interaction in evaluative feedback, which is suitable for non-expert users who are not able to take control of the robot. It is also suitable for scenarios where extensive trial-and-error is not feasible and a handcrafted reward function is not available, e.g., using Reinforcement Learning (RL) [9]. Similar, or even better, navigation performance, compared to that learned from richer interaction modalities, can be achieved using APPLE.

### C. Learning From Human Feedback

The method we propose in this letter uses evaluative feedback from a human to drive a machine learning process that seeks to increase the performance of an autonomous navigation system. Because evaluative feedback is a relatively easy signal for humans to provide, several methods have been proposed to allow machines to learn from such signals over the past several decades. Broadly speaking, most of these methods can be understood as trying to interpret the feedback signal in the context of the classical RL framework. For example, the COACH framework [26] interprets evaluative feedback as the policy-dependent *advantage*, i.e., it is assumed that the feedback indicates how much better or worse the agent's current behavior is compared to what the human currently expects the agent to do. The TAMER framework [27], on the other hand, can be thought of as interpreting evaluative feedback to be the *value*, or expected payoff, of the current behavior if the agent were to act in the future in the way the human desires. Yet other approaches interpret evaluative feedback directly as reward or some related statistic [28]–[30].

APPLE adopts a similar learning from feedback paradigm, but instead of taking actions as raw motor commands, APPLE's action space is the parameters used by the underlying navigation system. During training, APPLE learns the value of state-action pairs based on a scalar human feedback. During deployment,

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.

APPLE selects the parameters to maximize the expected human feedback.

## III. APPROACH

In this section, we introduce APPLE, which has two novel features: (1) compared to learning from demonstration or interventions, which require human driving expertise in order to take full control of the moving robot, APPLE requires instead just evaluative feedback that can be provided even by non-expert users; (2) in contrast to previous work [4], [5] that selects the planner parameter set based on how similar the deployment environment is to the demonstrated environment, APPLE is based on the expected evaluative feedback, i.e., the actual navigation performance. APPLE's performance-based parameter policy has the potential to outperform previous approaches that are based on similarity.

### A. Problem Definition

We denote a classical parameterized navigation system as $G : \mathcal{X} \times \Theta \to \mathcal{A}$, where $\mathcal{X}$ is the state space of the robot (e.g., goal, sensor observations), $\Theta$ is the parameter space for $G$ (e.g., max speed, sampling rate, inflation radius), and $\mathcal{A}$ is the action space (e.g., linear and angular velocities). During deployment, the navigation system repeatedly estimates state $x$ and takes action $a$ calculated as $a = G(x; \theta)$. Typically, a default parameter set $\bar{\theta}$ is tuned by a human designer trying to achieve good performance in most environments. However, being good at everything often means being great at nothing: $\bar{\theta}$ usually exhibits suboptimal performance in some situations and may even fail (is unable to find feasible motions, or crashes into obstacles) in particularly challenging ones [4].

To mitigate this problem, APPLE learns a parameter policy from human evaluative feedback with the goal of selecting the appropriate parameter set $\theta$ (from either a discrete parameter set library or from a continuous full parameter space) for the current deployment environment. In detail, a human can supervise the navigation system's performance at state $x$ by observing its action $a$ and giving corresponding evaluative feedback $e$. Here, the evaluative feedback can be either discrete (e.g., "good/bad job") or continuous (e.g., a score ranging in $[0, 1]$). During feedback collection, APPLE finds (1) a parameterized predictor $F_\phi : \mathcal{X} \times \Theta \to \mathcal{E}$ that predicts human evaluative feedback for each state-parameter pair $(x, \theta)$, and (2) a parameterized parameter policy $\pi_\psi : \mathcal{X} \to \Theta$ that determines the appropriate planner parameter set for the current state.

Based on whether APPLE chooses the parameter set from a library or the parameter space, we introduce the discrete and continuous parameter policies in the following two sections, respectively.

### B. Discrete Parameter Policy

In some situations, the user may already have $K$ candidate parameter sets (e.g., the default set or sets tuned for special environments like narrow corridors, open spaces, etc.) which together make up a parameter library $\mathcal{L} = \{\theta^i\}_{i=1}^K$ (superscript $i$ denotes the index in the library). In this case, APPLE uses the provided evaluative feedback $e$ in order to learn a policy that selects the most appropriate of these parameters given the state observation $x$.

To do so, we parameterize the feedback predictor $F_\phi$ in a way similar to the value network in DQN [31], where the input is the observation $x$ and the output is $K$ predicted feedback values $\{\hat{e}^i\}_{i=1}^K$, one for each parameter set $\theta^i$ in the library $\mathcal{L}$, as a prediction of the evaluative feedback a human user would give if the planner were using the respective parameter set at state $x$. We form a dataset for supervised learning, $\mathcal{D} := \{x_j, \theta_j, e_j\}_{j=1}^N$ ($\theta_j \in \mathcal{L}$, subscript $j$ denotes the time step) using the evaluative feedback collected so far, and $F_\phi$ is learned via supervised learning to minimize the difference between predicted feedback and the label,

$$\phi^* = \arg\min_\phi \mathop{\mathbb{E}}_{(x_j, \theta_j, e_j) \sim \mathcal{D}} \ell(F_\phi(x_j, \theta_j), e_j) \tag{1}$$

where $\ell(\cdot, \cdot)$ is the categorical cross entropy loss if the feedback $e_j$ is discrete (e.g., "good job/bad job" or an integer score between 1 to 5), or mean squared error given continuous feedback.

To achieve the best possible performance, the parameter policy $\pi(\cdot|x)$ chooses the parameter set that maximizes the expected human feedback (the discrete parameter policy doesn't require any additional parameters beyond $\phi$ for $F$, so the $\psi$ is omitted here for simplicity). More specifically,

$$\pi(\cdot|x) = \arg\max_{\theta \in \mathcal{L}} F_{\phi^*}(x, \theta). \tag{2}$$

Compared to RL, especially DQN, discrete APPLE has a similar architecture and training objective. However, an important difference is that while RL optimizes future (discounted) cumulative reward, APPLE greedily maximizes the current feedback. The reason is that we assume, while supervising the robot's actions, the human will not only consider the current results but also future consequences and give the feedback accordingly. This assumption is consistent with past systems such as TAMER [27]. Under this interpretation of feedback, APPLE can also be though of as trying to maximize some notion of future performance.

### C. Continuous Parameter Policy

If a discrete parameter library is not available or desired, APPLE can also be used over continuous parameter spaces (e.g., deciding the max speed from $[0.1, 2]$ m/s). In this scenario, APPLE can still learn from either discrete or continuous feedback. However, learning from discrete feedback of finer resolutions or even continuous feedback should lead to better performance.

In this setting, we parameterize the parameter policy $\pi_\psi$ and the feedback predictor $F_\phi$ in the actor-critic style. With the collected evaluative feedback $\mathcal{D} := \{x_j, \theta_j, e_j\}_{j=1}^N$, the training objective of $F_\phi$ is still to minimize the difference between predicted and collected feedback, as specified by Eqn. (1). For the parameter policy $\pi_\psi$, beyond choosing the action that maximizes expected feedback, its training objective is augmented by maximizing the entropy of policy $\mathcal{H}(\pi_\psi(\cdot|x))$ at state $x$. Using the same entropy regularization as Soft Actor Critic (SAC) [32],

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PARAMETER LIBRARY AND RANGE: *MAX_VEL_X* (V), *MAX_VEL_THETA* (W), *VX_SAMPLES* (S), *VTHETA_SAMPLES* (T), *OCCDIST_SCALE* (O), *PDIST_SCALE* (P), *GDIST_SCALE* (G), *INFLATION_RADIUS* (I)

| | v | w | s | t | o | p | g | i |
|---|---|---|---|---|---|---|---|---|
| $\theta_1$ | 0.50 | 1.57 | 6 | 20 | 0.10 | 0.75 | 1.00 | 0.30 |
| $\theta_2$ | 0.26 | 2.00 | 13 | 44 | 0.57 | 0.76 | 0.94 | 0.02 |
| $\theta_3$ | 0.22 | 0.87 | 13 | 31 | 0.30 | 0.36 | 0.71 | 0.30 |
| $\theta_4$ | 1.91 | 1.70 | 10 | 47 | 0.08 | 0.71 | 0.35 | 0.23 |
| $\theta_5$ | 0.72 | 0.73 | 19 | 59 | 0.62 | 1.00 | 0.32 | 0.24 |
| $\theta_6$ | 0.37 | 1.33 | 9 | 6 | 0.95 | 0.83 | 0.93 | 0.01 |
| $\theta_7$ | 0.31 | 1.05 | 17 | 20 | 0.45 | 0.61 | 0.22 | 0.23 |
| min | 0.2 | 0.31 | 4 | 8 | 0.10 | 0.10 | 0.01 | 0.10 |
| max | 2.0 | 3.14 | 20 | 40 | 1.50 | 2.00 | 1.00 | 0.60 |

$\pi_\psi$ favors more stochastic policies, leading to better exploration during training:

$$\psi^* = \arg\min_{\psi} \mathop{\mathbb{E}}_{\substack{x_j \in \mathcal{D} \\ \tilde{\theta}_j \sim \pi_\psi(\cdot|x_j)}} \left[ -F_\phi(x_j, \tilde{\theta}_j) + \alpha \log \pi_\psi(\tilde{\theta}_j | x_j) \right], \tag{3}$$

where $\alpha$ is the temperature controlling the importance of the entropy bonus and is automatically tuned as in SAC [32].

### D. Deployment

During deployment, we measure the state $x_t$, use the parameter policy to obtain a set of parameters $\theta_t \sim \pi_{\psi^*}(\cdot|x_t)$ at each time step, and apply that parameter set to the navigation planner $G$.

## IV. EXPERIMENTS

In our experiments, we aim to show that APPLE can improve navigation performance by learning from evaluative feedback, in contrast to a teleoperated demonstration or a few corrective interventions, both of which require the non-expert user to take control of the moving robot. We also show APPLE's generalizability to unseen environments. We implement APPLE on a ClearPath Jackal ground robot in BARN [10] with 300 navigation environments randomly generated using Cellular Automata, and in two physical obstacle courses.

### A. Implementation

The Jackal is a differential-drive robot equipped with a Velodyne LiDAR that we use to obtain a 720-dimensional planar laser scan with a 270° field of view, denoted as $l_t$. The robot uses the Robot Operating System `move_base` navigation stack with Dijkstra's global planner and the default DWA local planner [2]. From the global planner, we query the relative local goal direction $g_t$ (in angle) as the averaged tangential direction of the first 0.5m global path. The state space of the robot is the combination of the laser scan and local goal $x_t = (l_t, g_t)$. The parameter space consists of the 8 parameters of the DWA local planner as described in Tab. I, and the action space is the linear and angular velocity of the robot, $a_t = (v_t, \omega_t)$.

For discrete APPLE, we construct the parameter library shown in Tab. I with the default DWA parameter set $\theta_1$ and parameter sets $\theta_{2\sim7}$ learned in the APPLI work [5]. Here, we use parameter sets learned in previous work, and we have found that this is important-without a reasonably good set to start with, APPLE is typically unable to learn, e.g., smaller *inflation_radius* and larger *vtheta_samples* in $\theta_2$ achieve good navigation in tight spaces, while larger *max_vel_x* and *pdist_scale* in $\theta_4$ perform well in open ones. Without prelearned parameter sets, one can obtain a library via coarse tuning to create various driving modes (e.g. increase *max_vel_x* to create an aggressive mode). For continuous APPLE, the parameter ranges for the parameter policy $\pi_\psi$ to select from are listed in the same table.

Implementation-wise, for discrete APPLE, $F_\phi(x, \theta)$ is a fully-connected neural network with 2 hidden layers of 128 neurons, taking the 721 dimensional $x_t$ as input and outputting the 7 predicted feedback signals $\hat{e}_{t,1\sim7}$ for $\theta_{1\sim7}$ respectively. Parameter policy $\pi_\psi$ uses $\epsilon$-greedy exploration with $\epsilon$ decreasing from 0.3 to 0.02 during the first half of the training. For continuous APPLE, $F_\phi(x, \theta)$ shares the same architecture as discrete APPLE, except for different input (concatenation of $x_t$ and $\theta_t$) and output (scalar $\hat{e}_t$). The parameter policy $\pi_\psi$ also uses the same architecture, mapping $x_t$ to $\theta_t$.

To evaluate the performance of APPLE, we use APPLI with the same parameter library in Tab. I upper part, APPLR with the same parameter ranges in Tab. I lower part, and the Default DWA planner as three baselines. Since APPLD does not generalize well without a confidence-based context predictor [5] and APPLI can therefore outperform APPLD, we do not include APPLD as one of the baselines. Despite using the same library, APPLI chooses the parameter set based on the similarity between the current observation and the demonstrated environments, while discrete APPLE uses the expected feedback.

### B. Simulated Experiments

We begin by testing APPLE on the BARN dataset, with simulated evaluative feedback generated by an oracle (proxy human). Note that although the proxy human for the simulated experiments appears to be similar to the APPLR work [9], the simulated experiments aim to validate different APPLE setups with easily accessible feedback before physical experiments. The intended use case for APPLE is still during *physical* deployments with *real* humans. The benchmark dataset consists of 300 simulated navigation environments ranging from easy ones with a lot of open spaces to challenging ones where the robot needs to get through dense obstacles. Navigation trials in three example environments with low, medium, and high difficulty levels are shown in Fig. 2. We randomly select 250 environments for training APPLE and hold the remained 50 environments as the test set.

For the simulated feedback, we use the projection of the robot's linear velocity along the local goal direction, i.e., $e_t = v_t \cdot \cos(g_t)$, and it greedily encourages the robot to move along the global path as fast as possible. Then we discretize it to different number of levels ($\infty$ levels mean using continuous feedback) to study the effect of feedback resolutions. Notice,

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.

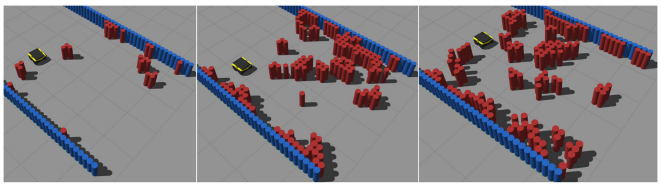

Fig. 2. Simulated Environments in BARN Dataset with Low, Medium, and High Difficulty Levels.

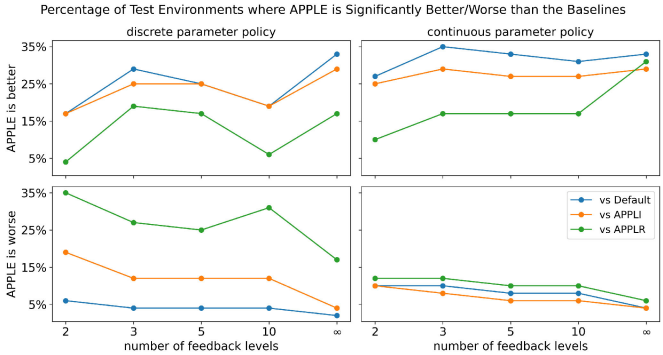


Fig. 3. APPLE Learning from Different Feedback Resolutions.

this simulated evaluative feedback is suboptimal as it doesn't consider future states, and we expect actual human evaluative feedback would be more accurate. For example, when the robot is leaving an open space and is about to enter a narrow exit, a human would expect the robot to slow down to get through the exit smoothly, but the simulated oracle still encourages the robot to drive fast. The oracle provides its evaluative feedback at 1 Hz, while APPLE, APPLI and APPLR dynamically adjust the parameter set for the DWA planner at the same frequency.

After training in 250 environments with a total of 2.5 M feedback signals collected, we evaluate APPLE with discrete and continuous parameter policies (denoted as APPLE (disc.) and APPLE (cont.)), as well as three baselines, on the 50 test environments by measuring the traversal time for 20 runs per environment. The proxy human aims at improving navigation efficiency and thus reducing traversal time. We then conduct t-tests to compute the percentage of environments in which APPLE with different feedback resolutions is significantly better/worse ($p < 0.05$) than baselines, as shown in Fig. 3.

Despite learning from suboptimal evaluative feedback, APPLE (disc.) and APPLE (cont.) still outperform the Default DWA and APPLI at all feedback resolutions. These results demonstrate the advantage of APPLE over APPLI, which selects the parameter set with a performance-based predictor (Eqns. 2 and 3) rather than a similarity-based predictor. Among all feedback resolutions, there is significant improvement of 3 feedback levels over 2, for both discrete and continuous policies. Further increasing resolutions doesn't improve performance much (the slight decreases are likely due to stochasticity in learning), except for using continuous feedback (i.e., $\infty$ levels) for continuous policy. These results suggest that as few as three discrete feedback levels are needed to improve navigation performance. Most of the cases where APPLE achieves worse performance are due to a corner case for the global planner where it keeps switching between two global paths and thus confuses the local planner. Surprisingly,

TABLE II
TRAVERSAL TIME IN TRAINING AND UNSEEN ENVIRONMENT

| | Default | APPLI | APPLE (disc.) |
|---|---|---|---|
| **Training** | 143.1±20.0s | 79.8±8.1s | 75.2±4.1s |
| **Unseen** | 150.5±24.0s | 86.4±1.1s | 83.9±4.6s |

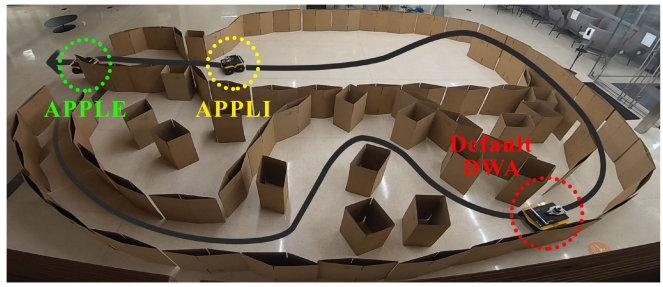


Fig. 4. APPLE Running in an Unseen Physical Environment.

despite that in theory APPLR is expected to perform the best, APPLE (disc.) performs only slightly worse than APPLR, and APPLE (cont.) outperforms APPLR in 29% of test environments with continuous feedback. The worse performance of APPLR may come from challenging optimization of cumulative rewards or reward design. Lastly, comparing the left and right parts of Fig. 3, APPLE (cont.) is comparable with APPLE (disc.) with low feedback resolutions and performs much better with continuous feedback, because of its larger model capacity in the parameter space.

### C. Physical Experiments

We also apply APPLE on a physical Jackal robot. In a highly constrained obstacle course (Fig. 1), we first apply APPLI which provides 5 sets of parameters (1 default $\theta_1$ and 4 learned $\theta_{2\sim5}$). Then to train a discrete APPLE policy which uses the same parameter library, one of the authors follows the robot autonomously navigating and uses an Xbox joystick to give binary evaluative feedback (instead of 3 feedback levels for easier collection) at 2 Hz. The author aims at teaching APPLE to reduce traversal time. To reduce the burden of giving a large amount of feedback, the user is only requested to give negative feedback by pressing a button on the joystick when he thinks the robot's navigation performance is bad, while for other instances, positive feedback is automatically given. In other words, we interpret the absence of human feedback to be the same as if the human had provided positive feedback. While this interpretation is not standard in the literature (and even undesirable at times [34]), we found that it yielded good results for the application studied here. Because APPLR [9] requires an infeasible amount of trial and error in the real world, it is not included as a baseline in the physical experiments.

The entire APPLE training session lasts roughly 30 minutes, in which the robot navigates 10 trials in the environment shown in Fig. 1. APPLE learns in an online fashion with 30% probability of random exploration. After the training, the learned APPLE model is deployed in the same training environment. We compare APPLE to APPLI with the same sets of parameters and the confidence

Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.

measure of context prediction [5], and the DWA planner with static default parameters. Each experiment is repeated five times. The results are shown in Tab II. APPLE achieves the fastest average traversal time with the smallest variance in the training environment.

To test APPLE's generalizability, we also test APPLE in an unseen environment (Fig. 4) and show the results in Tab. II. In the unseen environment, APPLE has slightly increased variance, but still has the fastest average traversal time compared to the other two baselines.

## V. CONCLUSIONS

In this work, we introduce APPLE, *Adaptive Planner Parameter Learning from Evaluative Feedback*. In contrast to most existing end-to-end machine learning for navigation approaches, APPLE utilizes existing classical navigation systems and inherits all their benefits, such as safety and explainability. Furthermore, instead of requiring a full expert demonstration or a few corrective interventions that need the user to take full control of the robot, APPLE just needs evaluative feedback as simple as "good job" or "bad job" that can be easily collected from non-expert users. Moreover, comparing with APPLI which selects the parameter set based on the similarity with demonstrated environments, APPLE achieves better generalization by selecting the parameter set with a performance-based criterion, i.e., the expected evaluative feedback. We show APPLE's performance improvement with simulated and real human feedback, as well as its generalizability in both 50 unseen simulated environments and an unseen physical environment. In this letter, we use relatively dense feedback signals from the human user in the physical experiments (and different resolutions of simulated feedback signals in the simulated experiments) to reduce the amount of time needed to train a good APPLE policy. These dense feedback signals may not always be practical, for example, the user may not always be paying attention. Therefore an important direction for future investigation is to study how little feedback is needed to yield good performance. Another important direction is to evaluate APPLE's generality with human subjects with different expertise levels and feedback criteria using an extensive user study.

## REFERENCES


[1] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1993, pp. 802–807.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[3] K. Zheng, "ROS navigation tuning guide," in *Proc. Robot Operating Syst.*, 2017, pp. 197–226.

[4] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "APPLD: Adaptive planner parameter learning from demonstration," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4541–4547, Jul. 2020.

[5] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone, "APPLI: Adaptive planner parameter learning from interventions," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020.

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

[7] B. Liu, X. Xiao, and P. Stone, "A lifelong learning approach to mobile robot navigation," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1090–1096, Apr. 2021.

[8] X. Xiao, B. Liu, and P. Stone, "Agile robot navigation through hallucinated learning and sober deployment," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021.

[9] Z. Xu *et al.*, "APPLR: Adaptive planner parameter learning from reinforcement," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021.

[10] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking metric ground navigation," in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, 2020, pp. 116–121.

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

[12] M. Bojarski *et al.* "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[13] 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 *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1527–1533.

[14] G. J. Stein, C. Bradley, and N. Roy, "Learning over subgoals for efficient navigation of structured, unknown environments," in *Proc. Conf. Robot Learn.*, 2018, pp. 213–222.

[15] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation," in *Proc. Conf. Robot Learn.*, 2017, pp. 185–194.

[16] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 2007–2014, Apr. 2019.

[17] 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," *Electron.*, vol. 8, no. 9, pp. 935–953, 2019.

[18] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable gaussian process motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 10 598–10 604.

[19] A. Binch, G. P. Das, J. P. Fentanes, and M. Hanheide, "Context dependant iterative parameter optimisation for robust robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3937–3943.

[20] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment, "Robot navigation from human demonstration: Learning control behaviors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1150–1157.

[21] S. Siva, M. Wigness, J. Rogers, and H. Zhang, "Robot adaptation to unstructured terrains by joint representation and apprenticeship learning," in *Robotics: Sci. Syst.*, 2019.

[22] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," vol. 6, no. 2, pp. 1312–1319, 2021.

[23] J. Hart *et al.*, "Using human-inspired signals to disambiguate navigational intentions," in *Proc. Int. Conf. Social Robot.*. Springer, 2020, pp. 320–331.

[24] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, "Crowdsteer: Realtime smooth and collision-free robot navigation in dense crowd scenarios trained using high-fidelity simulation," in *IJCAI*, 2020, pp. 4221–4228.

[25] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3052–3059.

[26] J. MacGlashan *et al.*, "Interactive learning from policy-dependent human feedback," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2285–2294.

[27] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proc. 5th Int. Conf. Knowl. Capture*, 2009, pp. 9–16.

[28] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone, "A social reinforcement learning agent," in *Proc. Int. Conf. Auton. Agents*, 2001, pp. 377–384.

[29] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *Proc. AAAI Conf. Artif. Intell.*, vol. 6, 2006, pp. 1000–1005.

[30] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *Proc. IEEE Int. Conf. Rehabil. Robot.*, 2011, pp. 1–7.

[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*., 2018, pp. 1861–1870.

[33] T. K. Faulkner, E. S. Short, and A. L. Thomaz, "Policy shaping with supervisory attention driven exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 842–847.


Authorized licensed use limited to: University of Texas at Austin. Downloaded on September 23,2021 at 00:19:17 UTC from IEEE Xplore. Restrictions apply.