APPLD: Adaptive Planner Parameter Learning From Demonstration

Xuesu Xiao, Bo Liu, Garrett Warnell, Jonathan Fink, and Peter Stone

***Abstract*—Existing autonomous robot navigation systems allow robots to move from one point to another in a collision-free manner. However, when facing new environments, these systems generally require re-tuning by expert roboticists with a good understanding of the inner workings of the navigation system. In contrast, even users who are unversed in the details of robot navigation algorithms can generate desirable navigation behavior in new environments via teleoperation. In this letter, we introduce APPLD, *Adaptive Planner Parameter Learning from Demonstration*, that allows existing navigation systems to be successfully applied to new complex environments, given only a human-teleoperated demonstration of desirable navigation. APPLD is verified on two robots running different navigation systems in different environments. Experimental results show that APPLD can outperform navigation systems with the default and expert-tuned parameters, and even the human demonstrator themselves.**

***Index Terms*—Learning from demonstration, autonomous vehicle navigation, imitation learning.**

I. Introduction

DESIGNING autonomous robot navigation systems has been a topic of interest to the research community for decades. Indeed, several widely-used systems have been developed and deployed that allow a robot to move from one point to another [1], [2], often with verifiable guarantees that the robot will not collide with obstacles while moving.

However, while current navigation systems indeed allow robots to autonomously navigate in known environments, they often still require a great deal of tuning before they can be successfully deployed in new environments. Adjusting the high-level parameters, or hyper-parameters, of the navigation systems can produce completely different navigation behaviors. For example, wide open spaces and densely populated areas may require completely different sets of parameters such as inflation radius, sampling rate, planner optimization coefficients, etc. Re-tuning these parameters requires an expert who has a good understanding of the inner workings of the navigation system. Even Zheng's widely-used full-stack navigation tuning guide [3] asserts that fine-tuning such systems is not as simple as it looks for users who are "sophomoric" about the concepts and reasoning of the system. Moreover, tuning 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.

In contrast, it is relatively easy for humans—even those with little to no knowledge of navigation systems—to generate desirable navigation behavior in new environments via teleoperation, e.g., by using a steering wheel or joystick. It is also intuitive for them to adapt their specific navigation strategy to different environmental characteristics, e.g., going fast in straight lines while slowing down for turns.

In this letter, we investigate methods for achieving autonomous robot navigation that are adaptive to complex environments *without* the need for a human with expert-level knowledge in robotics. In particular, we hypothesize that *existing* autonomous navigation systems can be successfully applied to complex environments given (1) access to a human teleoperated demonstration of competent navigation, and (2) an appropriate high-level control strategy that dynamically adjusts the existing system's parameters.

To this end, we introduce a novel technique called *Adaptive Planner Parameter Learning from Demonstration* (APPLD) and hypothesize that it can outperform default or even expert-tuned navigation systems on multiple robots across a range of environments. Specifically, we evaluate it on two different robots, each in a different environment, and each using a different underlying navigation system. Provided with as little as a single teleoperated demonstration of the robot navigating competently in its environment, APPLD segments the demonstration into contexts based on sensor data and demonstrator behavior and uses machine learning both to find appropriate system parameters for each context and to recognize particular contexts from sensor data alone (Fig. 1). During deployment, APPLD provides a simple control scheme for autonomously recognizing context and dynamically switching the underlying navigation system's parameters accordingly. Experimental results confirm our hypothesis: APPLD can outperform the underlying system using default parameters and parameters tuned by human experts, and even the performance of the demonstrator.

Manuscript received February 24, 2020; accepted June 3, 2020. Date of publication June 15, 2020; date of current version June 25, 2020. This letter was recommended for publication by Associate Editor Claes Christian Smith and Editor Dan Popa upon evaluation of the reviewers' comments. *(Xuesu Xiao and Bo Liu contributed equally to this work.) (Corresponding author: Xuesu Xiao.)*

Xuesu Xiao, Bo Liu, and Peter Stone are with the Department of Computer Science, University of Texas at Austin, Austin, TX 78712 USA (e-mail: xiao@cs.utexas.edu; bliu@cs.utexas.edu; pstone@cs.utexas.edu).

Garrett Warnell and Jonathan Fink are with the Computational and Information Sciences Directorate, Army Research Laboratory, Adelphi, MD 20783 USA (e-mail: garrett.a.warnell.civ@mail.mil; jonathan.r.fink3.civ@mail.mil).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3002217

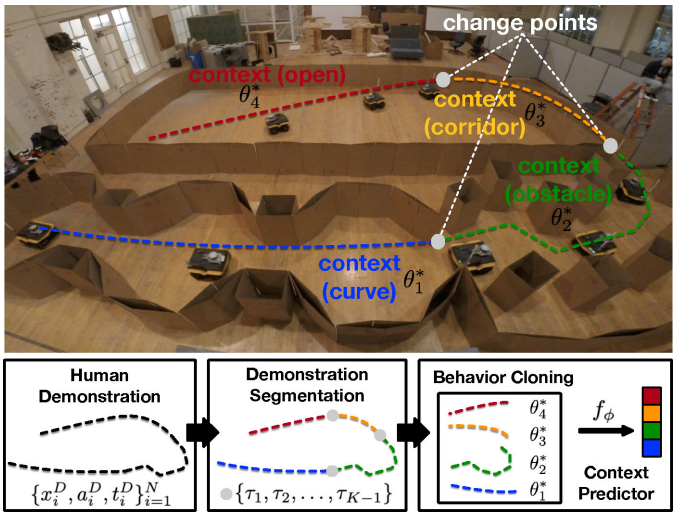

Fig. 1. Overview of APPLD: human demonstration is segmented into different contexts, for each of which, a set of parameters θ_k^* is learned via Behavior Cloning. During deployment, proper parameters are selected by an online context predictor.

II. Related Work

This section summarizes related work on parameter tuning, machine learning for robot navigation, and task demonstration segmentation, also known as changepoint detection.

A. Parameter Tuning

Broadly speaking, APPLD seeks to tune the high-level parameters of existing robot navigation systems. For this task, Zheng's guide [3] describes the current common practice of *manual* parameter tuning, which involves robotics experts using intuition, experience, or trial-and-error to arrive at a reasonable set of parameters. As a result, some researchers have considered the problem of automated parameter tuning for navigation systems, e.g., dynamically finding trajectory optimization weights [4] for a Dynamic Window Approach (DWA) planner [1], optimizing two different sets of DWA parameters for straight-line and U-turn scenarios [5], or designing novel systems that can leverage gradient descent to match expert demonstrations [6]. While such approaches do successfully perform automatic navigation tuning, they are thus far tightly coupled to the specific system or scenario for which they are designed and typically require hand-engineered features. In contrast, the proposed automatic parameter tuning work is more broadly applicable: APPLD treats the navigation system as a black box, and it does not require hand-engineering of features.

B. Machine Learning for Navigation

Researchers have also considered using machine learning, especially Learning from Demonstration [7] or Imitation Learning [8], more generally in robot navigation, i.e., beyond tuning the parameters of existing systems. One such approach is that of using inverse reinforcement learning to estimate costs over driving styles [9], social awareness [10]–[12], and semantic terrain labels [13] from human demonstrations, which can then be used to drive classical planning systems. Other work has taken a more end-to-end approach, performing navigation by learning functions that map directly from sensory inputs to robot actions [14], [15]. In particular, recent work in this space from Kahn *et al.* [16] used a neural network to directly assign costs to sampled action sequences using camera images. Because these types of approaches seek to replace more classical approaches to navigation, they also forgo the robustness, reliability, and generality of those systems. For example, Kahn *et al.* reported the possibility of catastrophic failure (e.g., flipping over) during training. In contrast, the work we present here builds *upon* traditional robot navigation approaches and uses machine learning to improve them only through parameter tuning, which preserves critical system properties such as safety.

C. Temporal Segmentation of Demonstrations

APPLD leverages potentially lengthy human demonstrations of robotic navigation. In order to effectively process such demonstrations, it is necessary to first segment these demonstrations into smaller, cohesive components. This problem is referred to as *changepoint detection* [17], and several researchers concerned with processing task demonstrations have proposed their own solutions [18]–[22]. Our work leverages these solutions in the context of learning from human demonstrations of navigation behavior. Moreover, unlike [20], we use the discovered segments to then train a robot for—and deploy it in—a target environment.

III. Approach

To improve upon existing navigation systems, the problem considered here is that of *determining a parameter-selection strategy that allows a robot to move quickly and smoothly to its goal*.

We approach this problem as one of learning from human demonstration. Namely, we assume that a human can provide a teleoperated demonstration of desirable navigation behavior in the deployment environment and we seek to find a set of planner parameters that can provide a good approximation of this behavior. As we will show in Section IV, when faced with a complex environment, a human demonstrator naturally drives differently in each regions of the environment such that no single set of planner parameters can closely approximate the demonstration in all states. To overcome this problem, the human demonstration is divided into pieces that include consistent sensory observations and navigation commands. By segmenting the demonstration in this way, each piece—which we call a *context*—corresponds to a relatively cohesive navigation behavior. Therefore, it becomes more feasible to find a single set of planner parameters that imitates the demonstration well for each context.

A. Problem Definition

We assume we are given a robot with an existing navigation planner $G : \mathcal{X} \times \Theta \rightarrow \mathcal{A}$. Here, $\mathcal{X}$ is the state space of the planner (e.g., current robot position, sensory inputs, navigation goal, etc.), Θ is the space of free parameters for G (e.g., sampling

density, maximum velocity, etc.), and $\mathcal{A}$ is the planner's action space (e.g., linear and angular velocity). Using G and a particular set of parameters θ, the robot performs navigation by repeatedly estimating its state x and applying action $a = G(x;\theta) = G_\theta(x)$. Importantly, we treat G as a black-box, e.g., we do not assume that it is differentiable, and we need not even understand what each component of θ does. In addition, a human demonstration of successful navigation is recorded as time series data $\mathcal{D} = \{x_i^D, a_i^D, t_i^D\}_{i=1}^N$, where N is the length of the series, and x_i^D and a_i^D represent the robot state and demonstrated action at time t_i^D. Given G and $\mathcal{D}$, the particular problem we consider is that of finding two functions: **(1)** a mapping $M : \mathcal{C} \to \Theta$ that determines planner parameters for a given *context* c, and **(2)** a parameterized context predictor $B_\phi : \mathcal{X} \to \mathcal{C}$ that predicts the context given the current state. Given M and B_ϕ, our system then performs navigation by selecting actions according to $G(x; M(B_\phi(x)))$. Note that since the formulation presented here involves only changing the *parameters* of G, the learned navigation strategy will still possess the benefits that many existing navigation systems can provide, such as assurance of safety.

B. Demonstration Segmentation

Provided with a demonstration, the first step of the proposed approach is to segment the demonstration into pieces—each of which corresponds to a single context only—so that further learning can be applied for each specific context. This general segmentation problem can be, in principle, solved by any changepoint detection method [17]. Given $\mathcal{D}$, a changepoint detection algorithm A_{segment} is applied to automatically detect how many changepoints exist and where those changepoints are within the demonstration. Denote the number of changepoints found by A_{segment} as $K-1$ and the changepoints as $\tau_1, \tau_2, \ldots, \tau_{K-1}$ with $\tau_0 = 0$ and $\tau_K = N+1$, the demonstration $\mathcal{D}$ is then segmented into K pieces $\{\mathcal{D}_k = \{x_i^D, a_i^D, t_i^D \mid \tau_{k-1} \le i < \tau_k\}\}_{k=1}^K$.

C. Parameter Learning

Following demonstration segmentation, we then seek to learn a suitable set of parameters θ_k^* for each segment $\mathcal{D}_k = \{x_i^D, a_i^D, t_i^D \mid \tau_{k-1} \le i < \tau_k\}$. To find this θ_k^*, we employ *behavioral cloning* (BC) [23], i.e., we seek to minimize the difference between the demonstrated actions and the actions that G_{θ_k} would produce on $\{x_i^D\}$. More specifically,

$$\theta_k^* = \underset{\theta}{\operatorname{argmin}} \sum_{(x,a)\in\mathcal{D}_k} ||a - G_\theta(x))||_H, \tag{1}$$

where $||v||_H = v^T H v$ is the induced norm by a diagonal matrix H with positive real entries, which is used for weighting each dimension of the action. A black-box optimization method $A_{\text{black-box}}$ is then applied to solve Equation (1). Having found each θ_k^*, the mapping M is simply $M(k) = \theta_k^*$.

Algorithm 1: APPLD.

```
// Training
Input: the demonstration D = {x_i^D, a_i^D, t_i^D}_{i=1}^N,
       space of possible parameters Θ, and the navigation
       stack G.
Call A_segment on D to detect changepoints
       τ_1, ..., τ_{K-1} with τ_0 = 0 and τ_K = N + 1.
Segment D into
       {D_k = {x_i^D, a_i^D, t_i^D | τ_{k-1} ≤ i < τ_k}}_{k=1}^K.
Train a classifier f_φ on {x_i^D, c_i}_{i=1}^N, where c_i = k if
       x_i^D ∈ D_k.
for k = 1 : K do
    Call A_black-box with objective defined in Equation (1)
       on D_k to find parameters θ_k^* for context k.
end for
Form the map M(k) = θ_k^*, ∀1 ≤ k ≤ K.
// Deployment
Input: the navigation stack G, the mapping M from
       context to parameters, and the context predictor B_φ.
for t = 1 : T do
    Identify the context c_t = B_φ(x_t) according to
       Equation (3).
    Navigate with G(x_t; M(c_t)).
end for
```

D. Online Context Prediction

At this point, we have a library of learned parameters and the mapping M that is responsible for mapping a specific context to its corresponding parameters. All that remains is a scheme to dynamically infer which context the robot is in during *execution*. To do so, we form a supervised dataset $\{x_i^D, c_i\}_{i=1}^N$, where $c_i = k$ if $x_i^D \in \mathcal{D}_k$. Then, a parameterized function $f_\phi(x)$ is learned via supervised learning to classify which segment x_i^D comes from, i.e.,

$$\phi^* = \underset{\phi}{\operatorname{argmax}} \sum_{i=1}^{N} \log \frac{\exp\left(f_\phi(x_i^D)[c_i]\right)}{\sum_{c=1}^K \exp\left(f_\phi(x_i^D)[c]\right)}. \tag{2}$$

Given f_ϕ, we define our context predictor B according to

$$B_\phi(x_t) = \operatorname{mode}\left\{\underset{c}{\operatorname{argmax}}\, f_\phi(x_i)[c],\ t-p < i \le t\right\}. \tag{3}$$

In other words, B_ϕ acts as a mode filter on the context predicted by f_ϕ over a sliding window of length p.

Taken together, the above steps constitute our proposed APPLD approach. During training, the above three stages are applied sequentially to learn a library of parameters $\{\theta_k^*\}_{k=1}^K$ (hence the mapping M) and a context predictor B_ϕ. During execution, Equation (3) is applied online to pick the right set of parameters for navigation. Algorithm 1 summarizes the entire pipeline from offline training to online execution.

IV. Experiments

In this section, APPLD is implemented to experimentally validate our hypothesis that *existing* autonomous navigation systems

can be successfully applied to complex environments given *(1)* access to a human demonstration from teleoperation, and *(2)* an appropriate high-level control strategy that dynamically adjusts the existing system's parameters based on context. To perform this validation, APPLD is applied on two different robots—a Jackal and a BWIBot—that each operate in a different environment with different underlying navigation methods. The results of APPLD are compared with those obtained by the underlying navigation system using *(a)* its default parameters (DEFAULT) from the robot platform manufacturer, and *(b)* parameters we found using behavior cloning but without context (APPLD (NO CONTEXT)). We also compare to the navigation system as tuned by robotics experts in the second experiment. In all cases, we find that APPLD outperforms the alternatives.

A. Jackal Maze Navigation

In the first experiment, a four-wheeled, differential-drive, unmanned ground vehicle—specifically a Clearpath Jackal—is tasked to move through a custom-built maze (Fig. 1). The Jackal is a small and agile platform with a top speed of 2.0 m/s. To leverage this agility, the complex maze consists of four qualitatively different areas: *(i)* a pathway with curvy walls (curve), *(ii)* an obstacle field (obstacle), *(iii)* a narrow corridor (corridor), and *(iv)* an open space (open) (Fig. 1). A Velodyne LiDAR provides 3D point cloud data, which is transformed into 2D laser scan for 2D navigation. The Jackal runs Robot Operating System (ROS) onboard, and APPLD is applied to the local planner, DWA [1], in the commonly-used `move_base` navigation stack. Other parts of the navigation stack, e.g. global planning with Dijkstra's algorithm, remain intact.

Teleoperation commands are captured via an Xbox controller from one of the authors with previous experience with video games, who is unfamiliar with the inner workings of the DWA planner and attempts to operate the platform to quickly traverse the maze in a safe manner. The teleoperator follows the robot and controls the robot from a third person view. This viewpoint, different from the robot's first person view, may provide the human demonstrator with different contextual information, but our experiments will show that the robot's limited onboard LiDAR input suffices for online context identification. The 52 s demonstration is recorded using `rosbag` configured to record all joystick commands and all inputs to the `move_base` node.

For changepoint detection (Algorithm 1, line 3), we use CHAMP as A_{segment}, a state-of-the-art Bayesian segmentation algorithm [18]. The recorded LiDAR range data statistics (mean and standard deviation) from X_i^D and the recorded demonstrated actions $a_i^D = (v_i^D, w_i^D)$ are provided as input to CHAMP. CHAMP outputs a sequence of changepoints $\tau_1, \tau_2, \ldots, \tau_{K-1}$ that segment the demonstration into K segments, each with uniform context (line 4). As expected, CHAMP determines $K = 4$ segments in the demonstration, each corresponding to a different context (line 5). f_ϕ trained for online context prediction (line 14) is modeled as a two-layer neural network with ReLU activation functions.

For the purpose of finding θ_k^* for each context, the recorded input is played to a ROS `move_base` node using DWA as the local planner with query parameters θ and the resulting output navigation commands are compared to the demonstrator's actions. Ideally, the DWA output and the demonstrator commands would be aligned in time, but for practical reasons (e.g., computational delay), this is generally not the case—the output frequency of `move_base` is much lower than the frequency of recorded joystick commands. To address this discrepancy, we match each a_i^D with the *most recent* queried output of G_θ within the past ϵ seconds (default execution time per command, 0.25 s for Jackal), and use it as the augmented navigation at time t_i^D. If no such output exists, augmented navigation is set to zero since the default behavior of Jackal is to halt if no command has been received in the past ϵ seconds (Fig. 3). This condition may occur due to insufficient onboard computation to perform sampling at the requested density. For the metric in Equation (1), we use mean-squared error, i.e., H is the identity matrix.

Following the action-matching procedure, we find each θ_k^* using CMA-ES [24] as our black-box optimizer (Algorithm 1, line 7). The optimization runs on a single Dell XPS laptop (Intel Core i9-9980HK) using 16 parallel threads. The elements of θ in our experiments are: DWA's *max_vel_x* (v), *max_vel_theta* (w), *vx_samples* (s), *vtheta_samples* (t), *occdist_scale* (o), *pdist_scale* (p), *gdist_scale* (g) and costmap's *inflation_radius* (i). We intentionally select parameters here that directly impact navigation behavior and exclude parameters which are robot-model-specific, e.g., physical acceleration limit (*acc_lim_x* and *acc_lim_theta*), or unrelated to the behaviors being studied, e.g. goal tolerance (*xy_goal_tolerance*). Note that *max_vel_x* and *max_vel_theta* are not the physical velocity limit of the robot, but rather the maximum velocity commands that are allowed to be executed. They interact with the sampling density parameters, *vx_samples* and *vtheta_samples*, in a way that affects whether finding a reasonable motion command through sampling can be performed in real time. The parameters *occdist_scale*, *pdist_scale*, and *gdist_scale*, are optimization weights for distance to obstacle, distance to path, and distance to goal, respectively. The inflation radius, *inflation_radius*, specifies the physical safety margin to be used around obstacles. All parameters are initialized at the midpoint between their lower- and upper-bound. The fully parallelizable optimization takes approximately eight hours, but this time could be significantly reduced with more computational resources and engineering effort.

The action profiles of using the parameters discovered by DEFAULT, APPLD (NO CONTEXT), and APPLD are plotted in Fig. 2, along with the single-shot demonstration segmented into four chunks by CHAMP. Being trained separately based on the segments discovered by CHAMP, the APPLD parameters (green) perform most closely to the human demonstration (black), whereas the performance of both DEFAULT (red) and APPLD (NO CONTEXT) (orange) significantly differs from the demonstration in most cases (Fig. 2).

The specific parameter values learned by each technique are given in Table I, where we show in the bottom rows the individual parameters learned by APPLD for each context. The learned parameters relative to the default values are intuitive in many ways. For example, APPLD found that Curve requires a larger value for

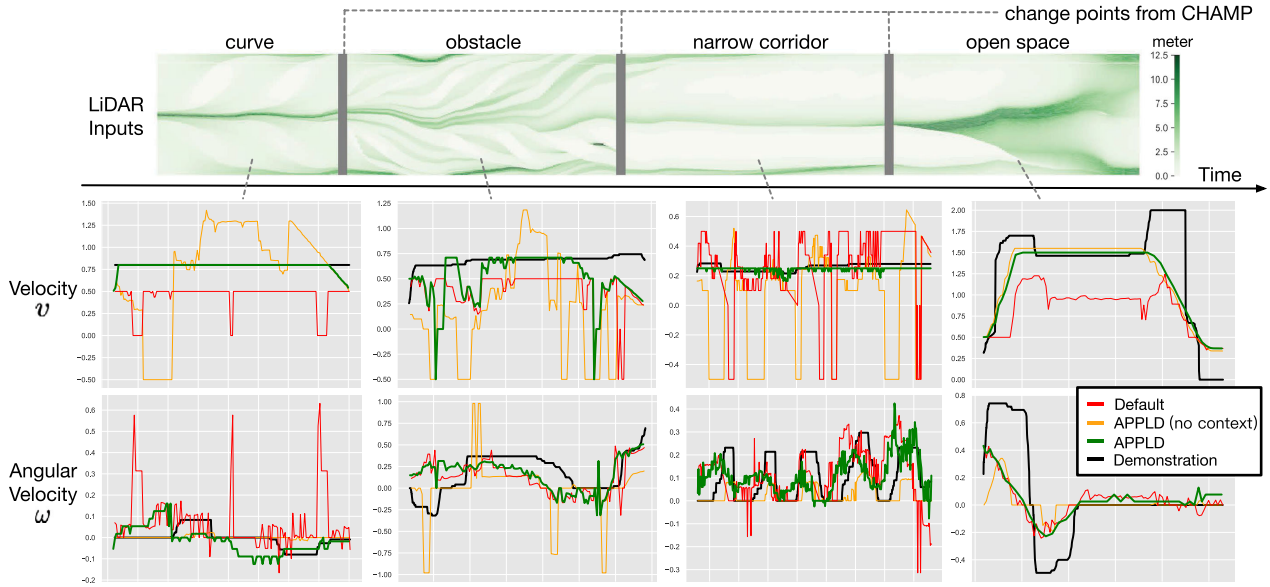

Fig. 2. Jackal Trajectory in Environment Shown in Fig. 1: heatmap visualization of the LiDAR inputs over time is displayed at the top and used for segmentation by CHAMP. For each region divided by CHAMP changepoints, CMA-ES finds a set of parameters that best imitates the human demonstration. Velocity and angular velocity profiles from DEFAULT (red), APPLD (NO CONTEXT) (orange), and APPLD (green) parameters, along with the human demonstration (black), are displayed with respect to time. Plots are scaled to best demonstrate performance differences between different parameters.

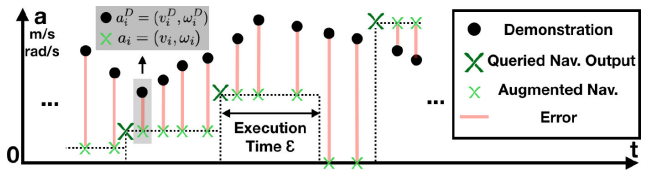

Fig. 3. Action-Matching and Loss Metric.

TABLE I
PARAMETERS OF JACKAL EXPERIMENTS (DWA): *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
DEF.	0.50	1.57	6	20	0.10	0.75	1.00	0.30
NO CTX.	1.55	0.98	10	3	0.01	0.87	0.99	0.46
Curve	0.80	0.73	6	42	0.04	0.98	0.94	0.19
Obstacle	0.71	0.91	16	53	0.55	0.54	0.91	0.39
Corridor	0.25	1.34	8	59	0.43	0.65	0.98	0.40
Open	1.59	0.89	18	18	0.40	0.46	0.27	0.42

the parameters p and g and a lower value for the parameter i, i.e., the platform needs to place a high priority on sticking to the straight global path so that it can avoid extraneous motion due to the proximity of the curvy walls. It is similarly intuitive that APPLD found that Obstacle Field requires higher sampling rates (s and t) and more consideration given to obstacle avoidance (higher o) in order to find feasible motion through the irregular obstacle course. Corridor is extremely tight, and, accordingly, APPLD found that a smaller linear velocity (v) was necessary in order to compensate for the larger computational load associated with the necessary higher angular velocity sampling rate (t) required to find feasible paths. In Open, APPLD appropriately learned that the maximum velocity (v) should be increased in order to match the demonstrator's behavior. In addition to these intuitive properties, APPLD was also able to capture other, more subtle, parameter interactions that are more difficult to describe. At run time, APPLD's trained context classifier selects in which mode the navigation stack is to operate and adjusts the parameters accordingly (Fig. 1).

Table II shows the results of evaluating the overall navigation system using the different parameter-selection schemes along with the demonstrator's performance as a point of reference. We report both the time it takes for each system navigate a pre-specified route and also the BC loss (Equation (1)) compared to the demonstrator. We choose to study traversal time since most suboptimal navigation behavior will cause stop-and-go motions, induce recovery behaviors, cause the robot to get stuck, or collide with obstacles (termination) – each of which will result in a higher traversal time.

For each metric, lower is better, and we compute mean and standard deviation over 10 independent trials. For trials that end in failure (e.g., the robot gets stuck), we add an asterisk (*) to the reported results and use penalty time value of 60 s. The results show that, for every context, APPLD achieves the lowest BC loss and fastest real-world traverse time, compared to DEFAULT and APPLD (NO CONTEXT). In fact, while APPLD is able to successfully navigate in every trial, DEFAULT fails in 8/10 trials in the narrow corridor due to collisions in `recovery_behaviors` after getting stuck, and APPLD (NO CONTEXT) fails in 9/10, 10/10, and 10/10 trials in curve, obstacle field, and narrow corridor, respectively. In open space, APPLD (NO CONTEXT) is able to navigate quickly at first, but is not able to precisely and quickly reach the goal due to low angular sample density (*vtheta_samples*). Surprisingly, in all contexts, the navigation stack with APPLD parameters even outperforms the human demonstration in terms

TABLE II
LOSS AND TIME COMPARISON OF JACKAL EXPERIMENTS (DWA)

Context	BC Loss	Real-world Time (s)
(a) Curve		
Demonstration	N/A	12.10
DEFAULT	0.1755±0.0212	30.20±3.87
APP. (NO CTX.)	0.1856±0.0030	*55.14±13.84
APPLD	**0.0780±0.0002**	**9.39±0.73**
(b) Obstacle Field		
Demonstration	N/A	9.00
DEFAULT	0.2061±0.0540	12.32±0.59
APP. (NO CTX.)	0.2537±0.0083	*60.00±0.00
APPLD	**0.1586±0.0216**	**7.69±0.35**
(c) Narrow Corridor		
Demonstration	N/A	24.06
DEFAULT	0.0953±0.0916	*49.52±19.88
APP. (NO CTX.)	0.0566±0.0419	*60.00±0.00
APPLD	**0.0198±0.0010**	**19.11±0.82**
(d) Open Space		
Demonstration	N/A	7.28
DEFAULT	0.8597±0.0013	15.07±0.61
APP. (NO CTX.)	0.2094±0.0013	15.08±7.42
APPLD	**0.2071±0.0021**	**7.19±0.42**

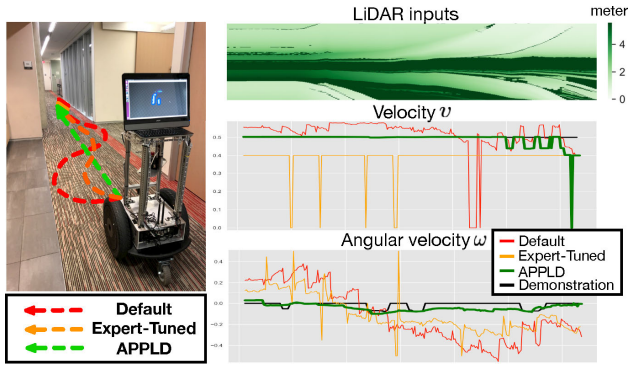

Fig. 4. BWIBot Navigates in GDC Hallway.

TABLE III
PARAMETERS AND RESULTS OF BWIBOT EXPERIMENTS (E-BAND): *MAX_VEL_LIN* (V), *MAX_VEL_TH* (W), *EBAND_INTERNAL_FORCE_GAIN* (I), *EBAND_EXTERNAL_FORCE_GAIN* (E), *COSTMAP_WEIGHT* (C)

	v	w	i	e	c	loss
DEF.	0.75	1.0	1	2	10	0.1730±0.0025
EXP.	0.5	0.5	3	2	15	0.0940±0.0095
APP.	0.65	0.35	0.52	0.04	15.36	**0.0669±0.0071**

of time, and leads to qualitatively smoother motion than the demonstration. Average overall traversal time from start to goal, 43 s, is also faster than the demonstrated 52 s. The superior performance achieved by APPLD compared to DEFAULT and even the demonstrator validates our hypothesis that given access to a teleoperated demonstration, tuning DWA navigation parameters is possible without a roboticist. We notice that, in some challenging situations, even the human demonstrator suffered from suboptimal navigation, e.g., stop-and-go, overshoot, etc. Even in these cases, APPLD can produce smooth, stable, and sometimes even faster navigation due to the benefit of a properly-parameterized autonomous planner. The fact that APPLD outperforms APPLD (NO CONTEXT) indicates the necessity of the high-level context switch strategy.

B. BWIBot Hallway Navigation

Whereas we designed the Jackal experiments to specifically test all aspects of APPLD, in this section, we evaluate APPLD's generality to another robot in another environment running another underlying navigation system. Specifically, we evaluate our approach using a BWIBot (Fig. 4 left)—a custom-built robot that navigates the GDC building at The University of Texas at Austin every day as part of the Building Wide Intelligence (BWI) project [25]. The BWIBot is a nonholonomic platform built on top of a Segway RMP mobile base, and is equipped with a Hokuyo LiDAR. A Dell Inspiron computer performs all computation onboard. Similar to the Jackal, the BWIBot uses the ROS architecture and the `move_base` navigation framework. However, unlike the Jackal, the BWIBot uses a local planner based on the elastic bands technique (E-BAND) [2] instead of DWA.

As in the Jackal experiments, teleoperation is performed using an Xbox controller from a third person view by the same author who is unfamiliar with the inner workings of the E-BAND planner. The demonstration lasts 17 s and consists of navigating the robot through a hallway, where the demonstrator seeks to move the robot in smooth, straight lines at a speed appropriate for an office environment. Unlike the Jackal demonstration, quick traversal is not the goal of the demonstration.

In this setting, the APPLD training procedure is identical to that described for the Jackal experiments. In this case, however, CHAMP did not detect any changepoints based on the LiDAR inputs and demonstration (Fig. 4 right), indicating the hallway environment is relatively uniform and hence one set of parameters is sufficient.

The BC phase takes about two hours with 16 threads on the same laptop used for the Jackal experiments. The parameters learned for the E-BAND planner are *max_vel_lin* (v), *max_vel_th* (w), *eband_internal_force_gain* (i), *eband_external_force_gain* (e), and *costmap_weight* (c). The results are shown in Table III.

The first row of Table III shows the parameters of the BWIBot planner used in the DEFAULT system. Because CHAMP does not discover more than a single context, APPLD and APPLD (NO CONTEXT) are equivalent for this experiment. Therefore, we instead compare to a set of expert-tuned (EXPERT) parameters that is used on the robot during everyday deployment, shown in the second row of the table. These parameters took a group of roboticists several days to tune by trial-and-error to make the robot navigate in relatively straight lines. Finally, the parameters discovered by APPLD are shown in the third row. The last column of the table shows the BC loss induced by DEFAULT, EXPERT, and APPLD parameters (again averaged over 10 runs). Real-world time is not reported since a quick traversal is not the purpose of the demonstration in the indoor office space. The action profiles from these three sets of parameters (queried on the demonstration trajectory $\{x_i^D\}_{i=1}^N$) are compared with the demonstration and plotted in Fig. 4 lower right, where the learned trajectories are the closest

to the demonstration. When tested on the real robot, the APPLD parameters achieve qualitatively superior performance, despite the fact that the experts were also trying to make the robot navigate in a straight line (Fig. 4 left).

The BWIBot experiments further validate our hypothesis that parameter tuning for existing navigation systems is possible based on a teleoperated demonstration instead of expert roboticist effort. More importantly, the success on the E-BAND planner without any modifications from the methodology developed for DWA supports APPLD's generality.

V. Summary and Future Work

This letter presents APPLD, a novel learning from demonstration framework that can autonomously learn suitable planner parameters and adaptively switch them during execution in complex environments. The first contribution of this work is to grant non-roboticists the ability to tune navigation parameters in new environments by simply providing a single teleoperated demonstration. Secondly, this work allows mobile robots to utilize existing navigation systems, but adapt them to different contexts in complex environments by adjusting their navigation parameters on the fly. APPLD is validated on two robots in different environments with different navigation algorithms. We observe superior performance of APPLD's parameters compared with all tested alternatives, both on the Jackal and the BWIBot. An interesting direction for future work is to investigate methods for speeding up learning by clustering similar contexts together. It may also be possible to perform parameter learning and changepoint detection jointly for better performance.

Acknowledgment

This work was supported in part by the Learning Agents Research Group (LARG) at UT Austin, in part by LARG research, and in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARL, DARPA, Lockheed Martin, GM, and Bosch. The work of P. Stone was supported by 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.

References

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

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

[3] K. Zheng, "Ros navigation tuning guide," 2017, *arXiv:1706.09068*.

[4] 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, pp. 935–953, 2019.

[5] 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. Autom.*, 2020, pp. 3937–3943.

[6] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable Gaussian process motion planning," 2019, *arXiv:1907.09591*.

[7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[8] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–35, 2017.

[9] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2641–2646.

[10] K. Shiarlis, J. Messias, and S. Whiteson, "Rapidly exploring learning trees," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1541–1548.

[11] N. Pérez-Higueras, F. Caballero, and L. Merino, "Teaching robot navigation behaviors to optimal RRT planners," *Int. J. Soc. Robot.*, vol. 10, no. 2, pp. 235–249, 2018.

[12] N. Pérez-Higueras, F. Caballero, and L. Merino, "Learning human-aware path planning with fully convolutional networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–5.

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

[14] 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. Autom.*, 2017, pp. 1527–1533.

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

[16] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," 2020, *arXiv:2002.05700*.

[17] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, 2017.

[18] S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto, "Online Bayesian changepoint detection for articulated motion models," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1468–1475.

[19] P. Fearnhead and Z. Liu, "On-line inference for multiple changepoint problems," *J. Roy. Statistical Soc.: Series B (Statistical Methodology)*, vol. 69, no. 4, pp. 589–605, 2007.

[20] F. Meier, E. Theodorou, and S. Schaal, "Movement segmentation and recognition for imitation learning," in *Proc. Artif. Intell. Statist.*, 2012, pp. 761–769.

[21] S. Krishnan *et al.*, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *Int. J. Robot. Res.*, vol. 36, nos. 13-14, pp. 1595–1618, 2017.

[22] T. Iqbal, S. Li, C. Fourie, B. Hayes, and J. A. Shah, "Fast online segmentation of activities from partial trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 5019–5025.

[23] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313.

[24] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, 2003.

[25] P. Khandelwal *et al.*, "BWIbots: A platform for bridging the gap between AI and human–robot interaction research," *Int. J. Robot. Res.*, vol. 36, nos. 5-7, pp. 635–659, 2017.