**APPLD: Adaptive Planner Parameter Learning from Demonstration**

Xuesu Xiao\(^1\*\), Bo Liu\(^1\*\), Garrett Warnell\(^2\), Jonathan Fink\(^2\), and Peter Stone\(^1\)

---

**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 paper, 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.

---

**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. 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 paper, 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.

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 Turning

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 DWA planner [1], or designing novel systems that can leverage gradient descent to match expert demonstrations [5]. While such approaches do successfully perform automatic navigation tuning, they are thus far tightly coupled to the specific system 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 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 semantic terrain labels from human demonstrations [6], 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 [7], [8]. In particular, recent work in this space from Kahn et al. [9] 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 [10], and several researchers concerned with processing task demonstrations have proposed their own solutions [11], [12]. Our work leverages these solutions in the context of learning from human demonstrations of navigation behavior. Moreover, unlike [12], 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 fast 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 different 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 \( G : \mathcal{X} \times \Theta \to \mathcal{A} \). Here, \( \mathcal{X} \) is the state space of the planner (e.g., current robot position, sensory inputs, navigation goal, etc.), \( \Theta \) 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 \( \theta \), 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 \( \theta \) does. In addition, a human demonstration of successful navigation is recorded as time series data \( D = \{ x_i^D, a_i^D, t_i^D \}_{i=1}^N \), 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 \( D \), 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 \( B_\phi \), 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 segmentation problem can be solved by changepoint detection methods [10]. Given $D$, a changepoint detection algorithm $A_{\text{segment}}$ is applied to automatically detect how many changepoints exist and where those changepoints are within the demonstration. Denote the number of changepoints found by $A_{\text{segment}}$ 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 $D$ is then segmented into $K$ pieces $\{D_k = \{x_i^D, a_i^D, t_i^D \mid \tau_k - 1 \leq i < \tau_k\}\}_{k=1}^K$.

C. Parameter Learning

Following demonstration segmentation, we then seek to learn a suitable set of parameters $\theta_k^*$ for each segment $D_k = \{x_i^D, a_i^D, t_i^D \mid \tau_k - 1 \leq i < \tau_k\}$. To find this $\theta_k^*$, we employ behavioral cloning (BC) [13], i.e., we seek to minimize the difference between the demonstrated actions and the actions that $G_{\theta_k}$ would produce on $\{x_i^D\}$. More specifically,

$$
\theta_k^* = \arg\min_{\theta} \sum_{(x,a) \in D_k} ||a - G_{\theta}(x)||_H,
$$

where $||u||_H = v^T H v$ 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 $\theta_k^*$, the mapping $M$ is simply $M(k) = \theta_k^*$.

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. The only thing left 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 $x_i^D \in D_k$. Then, a parameterized function $f_\phi(x)$ is learned via supervised learning to classify which segment $x_i^D$ comes from, i.e.,

$$
\phi^* = \arg\max_\phi \sum_{i=1}^N \log \frac{\exp (f_\phi(x_i^D)[c_i])}{\sum_{c=1}^K \exp (f_\phi(x_i^D)[c])}.
$$

Given $f_\phi$, we define our context predictor $B$ according to

$$
B_\phi(x_i) = \text{mode} \left\{ \arg\max_c f_\phi(x_i)[c], \ t - p < i \leq t \right\}.
$$

In other words, $B_\phi$ acts as a mode filter on the context predicted by $f_\phi$ 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 $B_\phi$. 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), 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 2.0m/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.

Teleoperation commands are captured via an Xbox controller from one of the authors, 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
The 52s 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 \( A_{\text{segment}} \), a state-of-the-art Bayesian segmentation algorithm [11]. As input to CHAMP, 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. 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). \( f_\theta \) trained for online context prediction (line 14) is modeled as a two-layer neural network with ReLU activation functions.

For the purpose of finding \( \theta^*_k \) for each context, the recorded input is played to a ROS move_base node using DWA as the local planner with query parameters \( \theta \) 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 \( G_0 \) within the past \( \epsilon \) seconds (default execution time per command, 0.25s 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 \( \epsilon \) seconds (Fig. 3). 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 \( \theta^*_k \) using CMA-ES [14] 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 \( \theta \) in our experiments are: DWA’s max_vel_x \( (v) \), max_vel_theta \( (w) \), vx_samples \( (s) \), vhtheta samples \( (t) \), occlust_scale \( (o) \), pdist_scale \( (p) \), gdist_scale \( (g) \) and costmap’s inflation_radius \( (i) \). The fully parallelizable optimization takes approximately eight hours, but this time could be significantly reduced with more computational resources.

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 Tab. 1 where we show in the bottom rows the individual parameters learned by APPLD for each context. 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).

Tab. I 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 to navigate a pre-specified route and also the BC loss (Equation 1) compared to the demonstrator.

### TABLE I
**Parameters of Jackal Experiments (dwa)**

|          | 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 |

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 (vtsamples). Surprisingly, in all contexts, the navigation stack with APPLD parameters even outperforms the human demonstration in terms of time, and leads to qualitatively smoother motion than the demonstration. Average overall traversal time from start to goal, 43s, is also faster than the demonstrated 52s. The superior performance achieved by APPLD compared to DEFAULT and even the demonstrator validates our hypothesis that given access to teleoperated demonstration tuning DWA navigation parameters is possible without a roboticist. The fact that APPLD outperforms DEFAULT (NO CTX) 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 [15]. 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 by the same author who is unfamiliar with the inner workings of the E-BAND planner. The demonstration lasts 17s 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_lh (w), eband_internal_force_gain (i), eband_external_force_gain (e), and costmap_weight (c). The results are shown in Tab. III.

The first row of Tab. 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 \( \{ x_i \}_{i=1}^N \)) 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 APPLDs generality.

| Parameters and Results of BWIBot Experiments (E-BAND) |
|-----------------|----------------|----------------|----------------|----------------|----------------|
|                | v    | w    | i    | e    | c    | loss          |
| DEFAULT         | 0.75 | 1.0  | 1    | 2    | 10   | 0.1730±0.0025 |
| EXPERT          | 0.5  | 0.5  | 3    | 2    | 15   | 0.0940±0.0095 |
| APPLD           | 0.65 | 0.35 | 0.52 | 0.04 | 15.36 | 0.0669±0.0071 |

**V. SUMMARY AND FUTURE WORK**

This paper 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.

**ACKNOWLEDGEMENT**

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 (RFP-000), ARL, 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.

**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] K. Zheng, “Ros navigation tuning guide,” arxiv preprint arxiv:1706.09068, 2017.

[4] D. Teso-Fz-Betoño, E. Zuñeta, 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.

[5] M. Bhardwaj, B. Boost, and M. Mukadam, “Differentiable gaussian process motion planning,” arxiv preprint arXiv:1907.06591, 2019.

[6] M. Wigness, J. G. Rogers, and L. E. Navarro-Serment, “Robot navigation from human demonstration: Learning control behaviors,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1150–1157.

[7] 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 IEEE International Conference on Robotics and Automation. IEEE, 2017.

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

[9] G. Kahn, P. Abbeel, and S. Levine, “Badger: An autonomous self-supervised learning-based navigation system,” arxiv preprint arXiv:2002.05700, 2020.

[10] S. Aminikhanghahi and D. J. Cook, “A survey of methods for time series change point detection,” Knowledge and Information systems, vol. 51, no. 2, pp. 339–367, 2017.

[11] S. Nieckum, S. Osentoski, C. G. Atkeson, and A. G. Barto, “Online bayesian changepoint detection for articulated motion models,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 1468–1475.

[12] F. Meier, E. Theodorou, and S. Schaal, “Movement segmentation and recognition for imitation learning,” in Artificial Intelligence and Statistics, 2012, pp. 761–769.

[13] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in Advances in neural information processing systems, 1989, pp. 305–313.

[14] N. Hansen, S. D. Müller, and P. Koumoutsakos, “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es),” Evolutionary computation, vol. 11, no. 1, pp. 1–18, 2003.

[15] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz et al., “Bwibots: A platform for bridging the gap between ai and human–robot interaction research,” The International Journal of Robotics Research, vol. 36, no. 5-7, pp. 635–659, 2017.