An Adaptive Framework for Manipulator Skill Reproduction in Dynamic Environments

Ryan Donald,[1] Brendan Hertel,[1] Stephen Misenti,[1,2] Yan Gu[2] and S. Reza Ahmadzadeh[1]

***Abstract*— Robot skill learning and execution in uncertain and dynamic environments is a challenging task. This paper proposes an adaptive framework that combines Learning from Demonstration (LfD), environment state prediction, and high-level decision making. Proactive adaptation prevents the need for reactive adaptation, which lags behind changes in the environment rather than anticipating them. We propose a novel LfD representation, Elastic-Laplacian Trajectory Editing (ELTE), which continuously adapts the trajectory shape to predictions of future states. Then, a high-level reactive system using an Unscented Kalman Filter (UKF) and Hidden Markov Model (HMM) prevents unsafe execution in the current state of the dynamic environment based on a discrete set of decisions. We first validate our LfD representation in simulation, then experimentally assess the entire framework using a legged mobile manipulator in 36 real-world scenarios. We show the effectiveness of the proposed framework under different dynamic changes in the environment. Our results show that the proposed framework produces robust and stable adaptive behaviors.**

I. Introduction

As robots become intertwined with human environments, they must robustly negotiate the difficulties of these environments. It may be easy to navigate and manipulate in a structured warehouse, but with clutter and debris the complexity of such a task increases [1]. One domain which has been under-explored is manipulation in dynamic environments that are characterized by time-varying changes in the robot's surroundings. Such perturbations in the environment could be caused by moving targets and obstacles. Another important factor that can result in unstructured and dynamic environments in the movement of the ground in the inertial frame. This type of movement has been investigated in the control of legged locomotion [2]–[6]. While achieving reliable locomotion in such environments remains a challenging problem, mobile manipulation in such environments have shown to be yet more complex and poses many issues [7]. First, the base must be stabilized, and a manipulator should be able to smoothly execute the task despite perturbations in the base. Additionally, a reactive system should be operating to provide the robot with the ability to react and safely avoid obstacles if the perturbations increase dramatically.

To achieve safe and robust robot manipulation in dynamic environments, we propose an adaptive skill learning framework consisting of three main modules. First, a novel Learning from Demonstration (LfD) representation is proposed that can quickly adapt online to perturbations in the environment. Our LfD representation, Elastic-Laplacian Trajectory Editing (ELTE), combines ideas from Laplacian Trajectory Editing (LTE) [8] and elastic maps [9]. This representation is able to smoothly deform trajectories online while maintaining the shapes of demonstrated skill. Shape preservation allows the robot to mimic important features of the skill and to adapt seamlessly throughout the execution, preventing jagged or rapid changes. The next part of the framework consists of a state estimation module, using the Unscented Kalman Filter (UKF) [10], that can provide prediction of the environment state at a future time resulting in a proactive adaptation. This module allows the framework to generate smoother reproductions, as changes in the environment are predicted and incorporated into execution before happening. The final part of the framework is a high-level decision making system utilizing a Hidden Markov Model (HMM) [11] that allows the robot to react to changes in the state of the environment to avoid collision and adapt the robot movement when the environment is unstable or otherwise unsuitable for execution.

To validate the robustness and accuracy of our framework, we used a legged mobile manipulator system that combines a Kinova Jaco2 arm and a Ghost Robotics Vision 60 legged base (shown in Fig 1). Using this system, we conducted experiments including inspecting a moving target when the

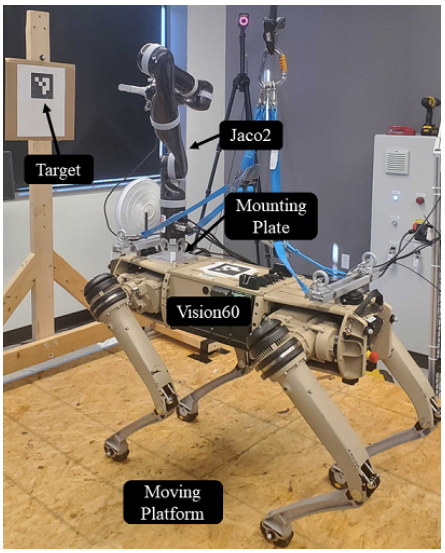

Fig. 1: Experimental setup with Kinova Jaco2 mounted on a Ghost Robotics Vision 60.

[1] Persistent Autonomy and Robot Learning (PeARL) Lab, University of Massachusetts Lowell, MA, 01854. Email: {ryan_donald, brendan_hertel, stephen_misenti}@student.uml.edu, reza@cs.uml.edu

[2] Terrain Robotics Advanced Control and Experimentation (TRACE) Lab, Purdue University, IN, 47907. Email: yangu@purdue.edu

base is static and inspecting a target marker when the base is self-stabilizing on a dynamic floor (a custom-built wooden platform on top of a Motek M-Gait treadmill).

II. Related Work

There are a variety of approaches for online adaptation for manipulators in robotics. In learning from demonstrations, previous approaches have shown their robustness against perturbations [12]. An LfD representation which is robust against perturbations will still continue execution even if the robot or goal is perturbed during execution. This is usually achievable because the LfD representation is modeled as a dynamical system [13], [14], and the system dynamics change with the environment. In some cases, the goal is perturbed, and the dynamical system incorporates the goal and adapts accordingly, such as Dynamic Movement Primitives (DMPs) [13]. In other cases, a stable dynamic system like Stable Estimator of Dynamical Systems (SEDS) [14] is used, allowing the robot to return to the path after a perturbation. These LfD systems provide only trajectory adaptation based on current environment information, and have no ability to proactively adjust the trajectory, anticipate a future change in the endpoint, or react to the environment to prevent unsafe execution. Some works have included a reactive system on top of DMPs such as [15], which can react to unsafe execution by retracting from the target. A reactive fuzzy decision maker is used to determine if execution is safe. However, this work still does not present a proactive solution to changes in the environment.

Other approaches use a variety of control policies for reactive behavior. A common method of control is visual servoing [16], where image or video feedback is used to control robot execution. Often, images are used to estimate the state of the robot and its environment which informs a separate robot controller that handles execution in the environment. Visual servoing can be a simple yet powerful tool, but other techniques can provide more information. For example, [17] uses multiple sensors such as vision, proximity, and force/torque to semi-autonomously execute an inspection task in a variable environment. While this system reacts to different environments, it does not react to disturbances in the environment during execution.

Other methods learn control policies which react to the environment. A common domain where reactions are necessary is Unmanned Aerial Vehicle (UAV) navigation [18], [19]. In [18], a reactive control policy is learned from human demonstrations and applied to navigation in real-world environments. Alternatively, [19] replaces the human demonstrator with a computational trajectory optimization teacher, and the policy is learned in simulation. This method employs Model Predictive Control (MPC) as the teacher, which has recently been applied to manipulators [20], [21]. MPC allows robots to rapidly react to changes in the environment, as shown in [21], where reactive MPC-generated movements are used with scenarios such as obstacle avoidance. Alternatively, reinforcement learning has been used to react to changes in the environment and switch between trained controllers [7]. Reactive policies for obstacle avoidance have also been learned using deep reinforcement learning [22]. These methods only react in the short-term, and do not exhibit any proactive behaviors. Additionally, these models assume vision of the target while reacting, which may not be valid under heavy perturbations.

Additionally, work has been done to incorporate the non-inertial motion of a platform into robot control systems. A simple form of this is shown in [23], where the motion of the non-inertial platform is included in the formulation of the control system, allowing for efficient control of a robot when compared to standard methods, as long as the non-inertial platform's motion is known. This was then expanded upon in [24], where a predictive measure is used to predict the motion of the platform, in this case a ship. They propose an auto-regressive predictor, as well as a superposition of sine-waves as two methods to provide the prediction of the ship's motion. This is shown to improve the control of the system, as long as the motion is predictable.

Few methods incorporate proactive movements, or predictive reactive movements. Proactive human-robot collaboration has been proposed as a future manufacturing paradigm where a robot is proactively planning movements [25]. However, for predictive control of manipulators, especially in dynamic environments, few works exist. Woolfrey et al. [26] create a model for disturbances in the environment, and adjust execution based on the model. This is limited to environments with motion that can easily be modeled, such as periodic motion. Our framework uses a more general model to predict future movements, and does not require disturbances to be periodic. Additionally, we can adapt online to changes in the environment and react to the environment to prevent unsafe execution.

III. Methodology

The proposed adaptive skill learning framework must be able to deal with the unstable and dynamic changes in the environment. In other words, such a framework must be able to perform not only based on its observations, but it must be able to predict the environment based on previous and current information.

As depicted in Fig 2, the proposed framework includes several modules. First, sensors are used to perceive the current state of the dynamic environment. From this perception, we extract important features from the environment. In the case of manipulation on a dynamic environment, these features are the current robot and goal states. These

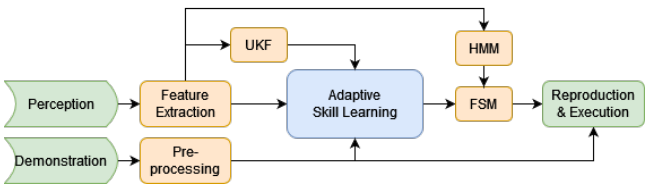

Fig. 2: An overview of all components of the proposed adaptive skill learning and execution framework.

features are then used as input to several other modules. To predict the state of the environment in future timesteps, we included a state estimation module, namely, the Unscented Kalman Filter (UKF) [10]. This prediction of the state is used to generate/update the skill execution. We developed an adaptive skill learning representation to encode and reproduce trajectories that also allows for online adaptation of the movement according to the changing states and predictions. To react to the perturbations in the environment in high-level, we design a Hidden Markov Model (HMM) [11]. This HMM is based off the changing state of the environment and determines if it is safe to execute the encoded skill in the current state of the environment. If execution is considered safe, execution proceeds normally. Otherwise, the robot reacts according to the environment state by halting or retracting from the goal. We developed a Finite State Machine (FSM) to handle the decision making of the HMM.

A. Adaptive Skill Learning

We propose a novel skill learning from demonstration method, Elastic-Laplacian Trajectory Editing (ELTE), that combines ideas from elastic maps [9] and Laplacian Trajectory Editing (LTE) [8]. Elastic maps are created by finding a set of nodes $\boldsymbol{y}$ which represent the data, and reduce the stretching and bending energies of the "spring" connections between nodes [27]. Overall, an elastic map is found by minimizing three energies: (i) the approximation energy U_Y which penalizes a bad fit to the data, (ii) the stretching energy U_E which penalizes high distance between adjacent nodes, and (iii) the bending energy U_R which penalizes the curvature of nodes. The constructed elastic map is then used for movement reproduction, as it has desirable features such as smoothness, incorporation of initial, final, or via-point constraints, and can generalize to one or more demonstrations. In this work, we modify the approximation energy, U_Y, to change how trajectories adapt, as well as the optimization method of the map, resulting in an adaptive skill learning representation. With the original elastic map optimization, reproductions are rewarded for following the given demonstration. However, we wish to deform trajectories such that they maintain the shape of the given demonstration and have no incentive for converging to the given demonstration, as in trajectory editing methods [8]. Additionally, it is necessary that we optimize the map during execution, and previously executed portions of the map must not be changed. Therefore, we modify the optimization of the map to allow for online adaptation.

We define a demonstration $\boldsymbol{\zeta}$ as a vector of points $[\zeta_1, \zeta_2, ..., \zeta_T]^\top$ with an individual point $\zeta_i \in \mathbb{R}^d$. A reproduction, $\boldsymbol{y} = [y_1, y_2, ..., y_T]^\top$, is found by minimizing the energy objectives listed above. We use a convex formulation for efficient optimization with flexible constraints [28]. These convex objectives are formulated as

$$U_Y = ||\boldsymbol{L}\boldsymbol{y} - \boldsymbol{L}\boldsymbol{\zeta}||_2^2 \tag{1}$$

$$U_E = w_E||\boldsymbol{E}\boldsymbol{y}||_2^2 \tag{2}$$

$$U_R = w_R||\boldsymbol{R}\boldsymbol{y}||_2^2, \tag{3}$$

where w_E, w_R are weight parameters, $||\cdot||_n$ is the L^n-norm, $\boldsymbol{L}$ is the graph Laplacian [29], and the matrices $\boldsymbol{E}$ and $\boldsymbol{R}$ are the first and second order finite difference matrices, respectively [28]. These matrices are defined as:

$$\boldsymbol{L} = \frac{1}{2}\begin{bmatrix} 2 & -2 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -2 & 2 \end{bmatrix},$$

$$\boldsymbol{E} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix},$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix}.$$

The convex optimization involving these constraints is formulated as

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{minimize}}\ f_0(\boldsymbol{y}) = U_Y + U_E + U_R \\ &\text{subject to}\ f_i(\boldsymbol{y}) = ||p - y_j||_1 - r \leq 0 \end{aligned} \tag{4}$$

where $f_i(\boldsymbol{y})$ is the ith constraint, constraining some point of the reproduction y_j within a radius r of a point in space p. The inequality constraint here is used as the perception system may have low confidence in the goal position until later in the trajectory. Therefore, we can begin modifying the trajectory early with large radii and refine the constraint as the trajectory executes. Additionally, using this formulation we can provide other constraints such as obstacle avoidance and via-point constraints if necessary (see [28] for details).

According to the context of the environment (i.e., output of the perception and feature extraction systems that detects the target object), the encoded movement must be generalized to new situations. In this paper, we explore moving targets, meaning that the endpoint must be modified. Therefore, during execution, we modify the trajectory online. Given that we have already executed some fraction of the trajectory from timesteps $0:t$, we must modify the rest of the trajectory from timesteps $t+1:T$. Therefore, we modify (4) for online adaptation as

$$\begin{aligned} &\underset{\boldsymbol{y}_{t+1:T}}{\text{minimize}}\ f_0(\boldsymbol{y}) = U_Y + U_E + U_R \\ &\text{subject to}\ f_1(\boldsymbol{y}) = ||p - y_T||_1 - r \leq 0 \end{aligned} \tag{5}$$

where p is the current prediction of the target final location and r is some region around that target. Note that while $\boldsymbol{y}_{0:t}$ is not included in the solution to the problem, they are still included in the optimization as they affect the energies associated with the map. The updated problem formulation remains convex and can be solved efficiently, allowing for smooth online adaptation which maintains the shape of given

demonstrations. Additionally, to increase the efficiency of optimizing the elastic map, we do not use the clustering process and Expectation-Maximization (EM) algorithm as shown in [9], [27]. Instead, we skip the clustering step by connecting each node to a corresponding data point from the demonstration, then solving (5). In fact, EM would be unable to solve the online adaptation problem, as it optimizes the complete map instead of a portion.

B. Environment Prediction

For short-term prediction, we utilize the Unscented Kalman Filter (UKF) [10]. This is done through a short term linear approximation of the input, similar to the Extended Kalman Filter (EKF) [30], with the exception of using sigma points to provide a more accurate estimation. In our framework, features of the environment's state are input to the UKF, and a prediction is generated for a future time-step. As a short-term prediction, this method allows for accurate prediction of the environment's movement, but also allows for prediction at variable time-steps in the future with different levels of certainty. The longer-term prediction provides the robot with a general goal at the beginning of execution which becomes more refined over the execution duration.

The UKF can be defined with the steps below. First, we define the state space, $\boldsymbol{x}$, as a vector of m features, in our case this is the vector $\boldsymbol{x} = [x_{\hat{i}}, x_{\hat{j}}, x_{\hat{k}}, \dot{x}_{\hat{i}}, \dot{x}_{\hat{j}}, \dot{x}_{\hat{k}}]$, where x is the position of the target and $\dot{x}$ is the velocity of the target. Similarly we have the observation, $\boldsymbol{z}$, which is the current position of the marker in the coordinate frame of the UR5e. Alongside this, the process and observation models at time step k are defined as $\boldsymbol{x}_k = F(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_k$ and $\boldsymbol{z}_k = H(\boldsymbol{x}_k) + \boldsymbol{n}_k$, respectively. These equations introduce noise to the process and observation models, where $\boldsymbol{v}_k$ is the process noise and $\boldsymbol{n}_k$ is the observation noise. The process model is defined as a constant-acceleration kinematics system along each axis. We then define the sigma-points matrix $\boldsymbol{\chi}$ as a $2m+1$ matrix, where m is the number of dimensions in the state vector, as a parameterized set of sigma points, where sigma point χ_i has weight W_i. For full details, see [31].

The state estimations from the UKF can be used to predict future states at each timestep. For target tracking, the final timestep T is predicted, where $\boldsymbol{x}_T = [y_T, \dot{y}_T]$. This prediction is constantly updating with each new observation, thus updating the trajectory generated by our adaptive skill learning module formulated by (5).

C. Reactive System and High-Level Decision Making

To allow higher-level decision making, we utilize a Hidden Markov Model (HMM) [11]. We define this as a model with the following discrete set of hidden states: $S = [Forward, Pause, Reverse]$ and continuous set of observable states $V = [\dot{x}]$. These states are chosen as to appropriately react to the velocity of the target. If the target is moving too fast, it is likely unstable, and execution should either pause or reverse for safety. Once the velocity slows, trajectory execution can continue as normal, returning to the $Forward$ state.

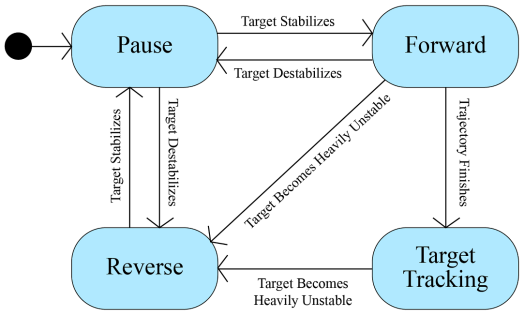

Fig. 3: A diagram of the Finite State Machine (FSM) that controls how high-level states transition based on changes in the environment.

For the HMM, we define the state transition probability distribution, represented by the matrix $\boldsymbol{A}$ with an element

$$\boldsymbol{A}_{ij} = P[S_{t+1} = s_i | S_t = s_j] \tag{6}$$

where S_t is the state at time t and $s_i, s_j \in S$. The observation probabilities are defined as a matrix $\boldsymbol{B}$ where an element $\boldsymbol{B}_i(v_k)$ is a function of the observed state as

$$\boldsymbol{B}_i(v_k) = P[V_t = v_k | S_t = s_i] \tag{7}$$

where V_t is the observed state at time t. Additionally, an initial state distribution is defined as a matrix $\boldsymbol{\pi}$ where and element

$$\boldsymbol{\pi}_i = P[S_1 = s_i] \tag{8}$$

defines the initial probability of state s_i. With these three sets of probabilities, the entire HMM can be parameterized as a tuple of $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. The HMM was trained with a set of data with three distinct states. The first state is conditions where the system is able to operate successfully and safely. The second state is when the perturbations in the environment reach near the limits of the system's capabilities to operate, and the third and final state is when these perturbations exceed the capabilities of the system to operate in. We then use the Baum-Welch [32] algorithm to train the HMM on this data.

Given the change in state from the HMM, we design a Finite State Machine (FSM) to govern execution, shown in Fig. 3. The FSM receives the trajectory from the skill reproduction module and the current state of execution from the HMM. The FSM internally records where in the duration of execution the current time-step is, and can use these time-steps to either move forward, pause, or reverse execution. If perturbations in the environment are low, execution continues as normal. However, if perturbations are too high and considered unsafe according to the trained HMM, execution can either be paused or reversed. If execution is reversed, the trajectory generation overwrites previous execution to allow for safer and new adaptive motion. Once execution finishes, the FSM moves into a "target tracking" state where the manipulator continues to follow the target. However, the

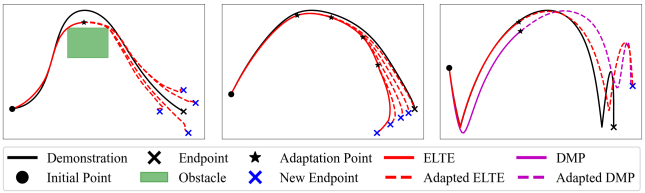

Fig. 4: (Left) An example of optimal continuations found for a given execution of a 2D reaching trajectory. The reproduction must smoothly approach a given perturbed goal while avoiding the obstacle. (Middle) An example of optimal adaptive continuations found for a given execution of a 2D reaching trajectory. The reproduction smoothly changes with the changing endpoint. (Right) A comparison of ELTE and DMPs for an endpoint changing partially through execution.

robot can still exit out of this state and begin retracting from the target if it becomes unstable.

IV. Experiments

A. Validation of the Adaptive Skill Learning in Simulation

We first validate our trajectory generation approach in a simulated 2D reaching environment. The results of are shown in Fig. 4 (left). In this experiment, a demonstration is given which approaches a given target while avoiding an obstacle. An obstacle avoidance constraint is included in the optimization problem. The target in this experiment moves around its initial endpoint, while still avoiding the obstacle constraint. Several possible continuations of the trajectory are shown to various novel positions around the endpoint. Each continuation smoothly continues to approximate the shape of the trajectory before approaching the desired endpoint. Additionally, all continuations maintain smoothness across the current point, avoiding discontinuities which could cause jerk in the robot execution.

Additionally, we examine a continuously changing endpoint in simulation. As shown in Fig. 4 (middle), a demonstration is given, and a reproduction is found for the demonstrated endpoint. During execution, the endpoint moves to a new location, adapting the reproduction. Again, the endpoint is changed, and a new adaptation is found. The time at which the adaptation is made is shown using opacity, with higher opacities adapting later in the execution. This shows that our method can adapt online to a continuously changing endpoint. However, because of the continuously changing endpoint the adaptation struggles to maintain the shape of the demonstrated trajectory. For continuous perturbations, predictive adaptation of perturbations could be used to better reproduce a perturbed trajectory.

As shown in Fig. 4 (right), we also compare our trajectory generation against Dynamic Movement Primitives (DMPs) [13]. This demonstration has sharp corners that causes representations like DMPs to overshoot. However, ELTE reproduces corners correctly, maintaining the shape which is important for tasks such as writing or welding where corners must be met exactly to successfully complete the task.

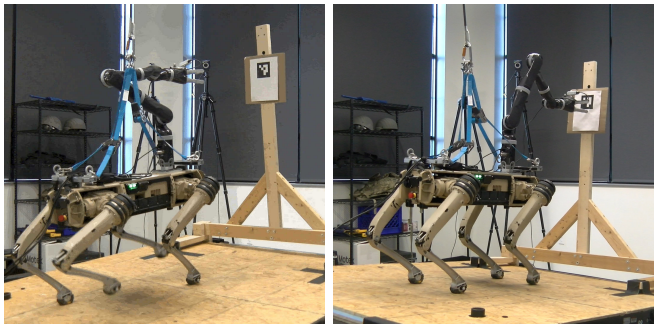

Fig. 5: The integrated legged mobile manipulator stepping in place on a dynamic platform while successfully inspecting the target.

B. Experimental Setup

We validated our approach in several real world experiments. Our robotic platform consists of the Kinova Jaco2 7DOF manipulator arm and the Ghost Robotics Vision 60 (V60) legged robot. We mounted the Jaco2 on the V60 using a custom-designed and fabricated plate (shown in Fig 1). In the first set of experiments, the goal was to approach an AR marker within an electrical box (shown in Fig 6). A camera was affixed to the end-effector for inspection of the electrical box. The AR marker was placed within the box and was used as the target for inspection. A demonstration using kinesthetic teaching was taken with this setup without perturbations in the base or target. The electrical box was placed on a cart with wheels such that it may be moved around during task execution. The V60 remained standing during these experiments, but did move slightly due to the shifting payload.

Additionally, we used this system on a dynamic moving platform. Our moving platform consists of a wooden base atop a Motek M-Gait treadmill with the ability for generating periodic pitch and sway motions (denoted as "Moving Platform" in Fig. 1). Connected to this platform, we mounted an AR marker to represent the electrical box for inspection. The marker was fixed to the platform such that the marker was fixed within the moving platform frame of view. The camera remained affixed to the robot end-effector. We experimented with the V60 standing and stepping in place. During the stepping in place trials, a human user teleoperated the V60 to keep it in the center of the platform. Note that we do not consider communication between the Jaco2 and V60 in any experiment. The V60 is running the default off-the-shelf walking controller made for stable joystick control and does not implement obstacle avoidance or gait control. The Jaco2 runs our proposed framework. For all experiments, the parameters used for the sigma-points of the UKF were set to $\alpha = 1$, $\beta = 2 \times 10^{-6}$, and $\kappa = 0$. The parameters for the adaptive skill learning were $w_E = w_R = 0.001$. Skill reproductions were generated in task space and executed using closed-form inverse kinematics with a low-level controller.

C. Real-world Experiments

We first validated our approach in an environment with a dynamically moving target. In these experiments, shown in

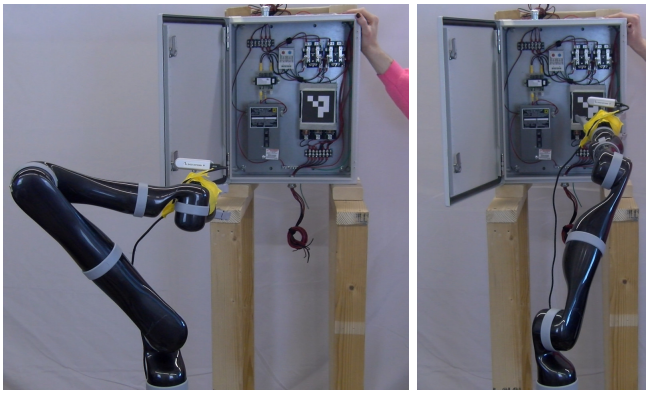

Fig. 6: Execution of an inspection task before (left) and after (right) the environment changes and the target is moved. Our framework reacts appropriately to the change in the environment and successfully executes the task.

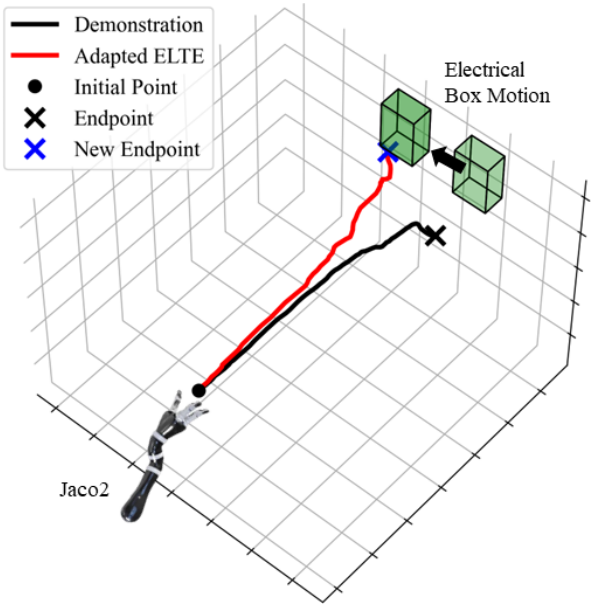

Fig. 7: Results of adapting online to a moving target as shown in Fig. 6. The electrical box (shown in green) is originally in front of the robot (as shown on the left of Fig. 6) then is moved to the left during execution (shown on the right of Fig. 6).

Fig. 6, the electrical box was moved by a human to simulate a dynamic surface. The base is not controlled and remains mostly still, but there is slight movement as it balances in response to the shifting weight of the arm. Fig. 7 shows the original demonstration given as well as the reproduced trajectory. The demonstration given was with a still base on a higher platform, therefore the arm has to adapt initially be reaching up, then adapts to the movement of the electrical box over the course of execution. The box was moved to the left during execution, and the reproduction adapts accordingly as shown on the right in Fig. 6 and in the adapted trajectory in Fig. 7. This validates our framework in a real-world dynamic environment, where the arm is still able to approach the target successfully.

We performed experiments where our framework was

TABLE I: Results of testing with and without the proactive-reactive framework (PRF) for the dynamic surface with varying levels of motion (*S*: Success, *P*: Partial fail, *F*: Fail)

		Without PRF			With PRF		
		Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
No Motion	Standing	*S*	*S*	*S*	*S*	*S*	*S*
	Step in place	*S*	*S*	*S*	*S*	*S*	*S*
Light Motion	Standing	*S*	*S*	*S*	*S*	*S*	*S*
	Step in place	*S*	*S*	*P*	*S*	*S*	*S*
Heavy Motion	Standing	*S*	*S*	*S*	*S*	*S*	*S*
	Step in place	*P*	*F*	*F*	*S*	*P*	*P*

expected to react to real-world movements in real-time using the moving platform setup described in the previous section. We tested using a dynamic moving platform with pitch and sway motions at various speeds, and with the V60 either standing or stepping in place for various levels of instability, shown in Fig 5. We compared motions with and without adaptation. The light motion was programmed to have the treadmill pitch $\pm 3°$at 1.5 Hz and with a sway of ± 0.5 m at 2 Hz. The heavy motion was programmed to have the treadmill pitch $\pm 8°$and sway ± 0.5 m both at 2.4 Hz. The results of this are shown in Table I and the accompanying video[1]. In this table, *S* for *Success* denotes tests in which the robot successfully reaches the box, *F* for *Fail* denotes tests which did not successfully reach the box, for instance because of poor adaptation or an emergency stop intervention to prevent unsafe execution, and *P* for *Partial fail* denotes tests which still completed the task but had issues during execution, such as obstacle collisions (obstacle avoidance constraints were not included in this experiment). The results show that for light or no motion, there is no significant difference in performance with and without the adaptive framework. However, in heavier motions, especially when the base is more unstable due to stepping in place, the performance with our framework is more successful. The proactive adaptation is able to track the moving target and predict its motion, and the reactive adaptation stops unsafe execution, preventing hard fails.

The performance of the reactive system is shown in the three plots of Fig. 8, where the real time position of the AR tag is tracked, and the UKF and HMM results are shown for this run. This run is in the configuration where the base is stepping in place, and the platform is moving with light motion. As you can see, the UKF is able to track the position of the tag to provide a reliable short term prediction of the motion of the AR tag. Additionally, the HMM provides state transitions when the tag begins to become unstable, and the robot reacts appropriately by pausing or reversing execution, and continues only when stability returns.

Finally, we compared our LfD method, Elastic-Laplacian Trajectory Editing (ELTE), with Dynamic Movement Primitives (DMPs) [13]. The framework shown in Fig. 2 was used for both methods, only the skill learning module was changed. Both executed on the moving platform while the

[1]Accompanying video: `https://youtu.be/H342Y0Hxl_0`

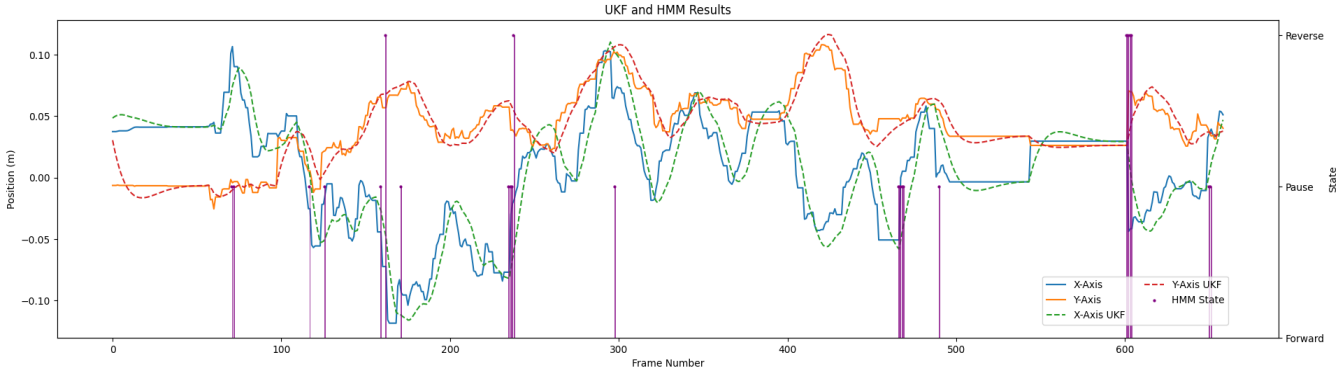

Fig. 8: The X and Y position of the target relative to the camera, the UKF predicted output of these positions, and the HMM output state during execution. This run was performed with light motion on the treadmill while the V60 is stepping in place.

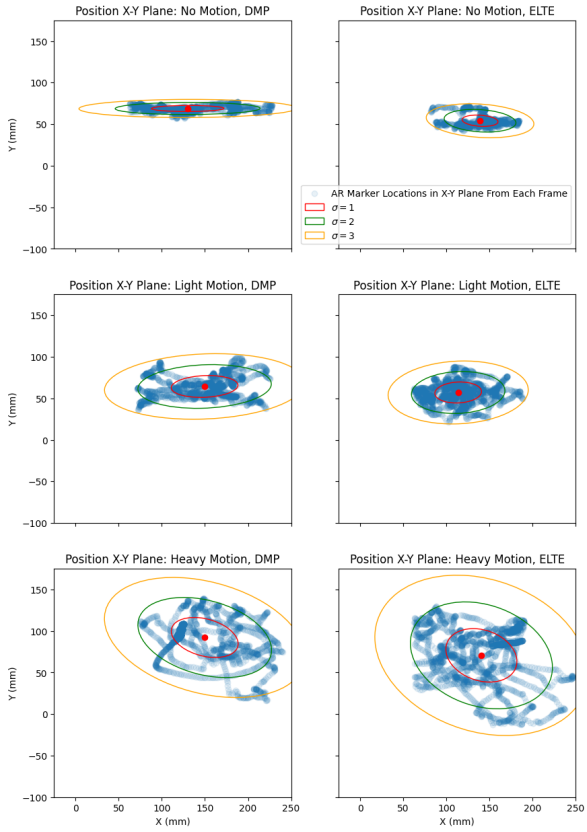

Fig. 9: Difference in position between the camera and the AR tag in the X-Y plane for each of the tests in Section IV-C, with our framework with DMP on the left column and our framework with ELTE on the right column.

legged base was stepping in place. As reported in Table II, it can be seen that ELTE provides a more stable execution as ELTE consistently has a lower average distance from target and a lower standard deviation of the distance. This indicates better tracking and adaptation ability, as the target was kept in the camera frame with more stability. This performance can also be seen in Fig. 9, where each plot showcases the deviations between the camera and the AR tag in the horizontal and vertical axes, during the last third

TABLE II: The mean and standard deviation of the distance (mm) between the target and the center of the camera when comparing ELTE and DMP methods in varying levels of surface motion.

	DMP		ELTE	
	mean distance	std	mean distance	std
No Motion	124.84	42.51	112.99	37.34
Light Motion	117.61	48.07	107.44	34.03
Heavy Motion	130.67	56.94	125.75	51.24

of the movement in each test. These distributions are plotted alongside their mean, and three confidence-ellipses at $\sigma = 1, 2, 3$. These plots showcase a portion of the distributions of the data shown in Table II. This is done to allow the reader to clearly see the differences between the algorithms, by showcasing the performance during the portion of the movement in which the inspection is most likely to occur. The first row of Fig. 9 showcases the condition in which there is no motion. Here, the ELTE algorithm is superior, as the DMP algorithm's distribution has greater variances in the x-axis, with similar variances in the y-axis. The second row showcases the condition for light motion. In this case, ELTE algorithm again has smaller variances in the distribution. Lastly, the final row contains the results for the heavy motion. In this case, the ELTE algorithm has larger variances during the final third of motion (although ELTE presents lower variances overall as shown in Table II), however the DMP algorithm has produced a shift in the distribution, as it was unable to properly compensate for the motion of the environment.

V. Conclusions and Future Work

In this work we have proposed and validated an adaptive skill learning framework for manipulation in dynamic environments, as well as designed and experimented a novel Learning from Demonstration (LfD) representation for adaptive skill learning and generation in unstructured environments. This framework combines skill adaptation with a reactive system for safe execution, while the LfD representation provides fast and smooth adaptation which maintains the shape of a given demonstration. We evaluate

the validity of our framework through 36 real-world experiments using a legged mobile manipulator under a variety of disturbances. Our framework is shown to provide safer execution compared to trials without any reactive behaviors, and our LfD representation provides more robust and stable adaptation compared to other adaptive LfD representations.

There are a myriad of opportunities for future work in this under-explored field of manipulation in dynamic environments. Firstly, creating more proactive adaptation is a possible avenue. Here, we use an Unscented Kalman Filter for state prediction, but other filters could be used or possibly a neural network trained for predictive movements. Additionally, in the case where the manipulator is attached to a dynamic base, investigating full-body control for proactive or reactive movements may yield better results.

ACKNOWLEDGEMENTS

This work was supported in part by by the U.S. Office of Naval Research (N00014-21-1-2582) and National Science Foundation (FRR- 2237463).

REFERENCES

[1] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Learning generalizable robot skills from demonstrations in cluttered environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4655–4660.

[2] A. Iqbal, S. Veer, and Y. Gu, "Asymptotic stabilization of aperiodic trajectories of a hybrid-linear inverted pendulum walking on a vertically moving surface," in *Proc. of American Control Conference*, 2023, pp. 3030–3035.

[3] A. Iqbal, Y. Gao, and Y. Gu, "Provably stabilizing controllers for quadrupedal robot locomotion on dynamic rigid platforms," *IEEE/ASME Trans. Mechatron.*, vol. 25, no. 4, pp. 2035–2044, 2020.

[4] Y. Gao, C. Yuan, and Y. Gu, "Invariant filtering for legged humanoid locomotion on a dynamic rigid surface," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 4, pp. 1900–1909, 2022.

[5] A. Iqbal and Y. Gu, "Extended capture point and optimization-based control for quadrupedal robot walking on dynamic rigid surfaces," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 72–77, 2021.

[6] Y. Gao, Y. Gong, V. Paredes, A. Hereid, and Y. Gu, "Time-varying ALIP model and robust foot-placement control for underactuated bipedal robotic walking on a swaying rigid surface," in *Proc. of American Control Conference*, 2023, pp. 3282–3287.

[7] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. v. Wichert, and W. Burgard, "Planning reactive manipulation in dynamic environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 136–143.

[8] T. Nierhoff, S. Hirche, and Y. Nakamura, "Spatial adaption of robot trajectories based on laplacian trajectory editing," *Autonomous Robots*, vol. 40, no. 1, pp. 159–173, 2016.

[9] B. Hertel, M. Pelland, and S. R. Ahmadzadeh, "Robot learning from demonstration using elastic maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[10] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE, 2000, pp. 153–158.

[11] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[13] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[15] S. R. Ahmadzadeh, P. Kormushev, R. S. Jamisola, and D. G. Caldwell, "Learning reactive robot behavior for autonomous valve turning," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 366–373.

[16] D. Kragic, H. I. Christensen *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, p. 2002, 2002.

[17] M. A. Abidi, R. O. Eason, and R. C. Gonzalez, "Autonomous robotic inspection and manipulation using multisensor feedback," *Computer*, vol. 24, no. 4, pp. 17–31, 1991.

[18] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 1765–1772.

[19] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3342–3349.

[20] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[21] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 750–759.

[22] E. Aljalbout, J. Chen, K. Ritt, M. Ulmer, and S. Haddadin, "Learning vision-based reactive policies for obstacle avoidance," in *Conference on Robot Learning*. PMLR, 2021, pp. 2040–2054.

[23] P. J. From, V. Duindam, J. T. Gravdahl, and S. Sastry, "Modeling and motion planning for mechanisms on a non-inertial base," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3320–3326.

[24] P. J. From, J. T. Gravdahl, T. Lillehagen, and P. Abbeel, "Motion planning and control of robotic manipulators on seaborne platforms," *Control Engineering Practice*, vol. 19, no. 8, pp. 809–819, 2011.

[25] S. Li, R. Wang, P. Zheng, and L. Wang, "Towards proactive human–robot collaboration: A foreseeable cognitive manufacturing paradigm," *Journal of Manufacturing Systems*, vol. 60, pp. 547–552, 2021.

[26] J. Woolfrey, W. Lu, and D. Liu, "Predictive end-effector control of manipulators on moving platforms under disturbance," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2210–2217, 2021.

[27] A. N. Gorban and A. Y. Zinovyev, "Principal graphs and manifolds," *Handbook of Research on Machine Learning Applications and Trends*, p. 28–59, 2008.

[28] B. Hertel and S. R. Ahmadzadeh, "Confidence-based skill reproduction through perturbation analysis," in *20th International Conference on Ubiquitous Robots (UR)*. IEEE, 2023.

[29] H. Ravichandar, S. R. Ahmadzadeh, M. A. Rana, and S. Chernova, "Skill acquisition via automated multi-coordinate cost balancing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7776–7782.

[30] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented kalman filters," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2011.

[31] R. van der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, Oregon Health & Science University, 2004.

[32] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.