Learning the Correct Robot Trajectory in Real-Time from Physical Human Interactions

DYLAN P. LOSEY and MARCIA K. O'MALLEY, Rice University

We present a learning and control strategy that enables robots to harness physical human interventions to update their trajectory and goal during autonomous tasks. Within the state of the art, the robot typically reacts to physical interactions by modifying a local segment of its trajectory, or by searching for the global trajectory offline, using either replanning or previous demonstrations. Instead, we explore a *one-shot* approach: here, the robot updates its entire trajectory and goal in real time without relying on multiple iterations, offline demonstrations, or replanning. Our solution is grounded in optimal control and gradient descent, and extends linear-quadratic regulator controllers to generalize across methods that locally or globally modify the robot's underlying trajectory. In the best case, this *Linear-quadratic regulator + Learning* approach matches the optimal offline response to physical interactions, and—in more challenging cases—our strategy is robust to noisy and unexpected human corrections. We compare the proposed solution against other real-time strategies in a user study and demonstrate its efficacy in terms of both objective and subjective measures.

CCS Concepts: • **Computing methodologies** → *Online learning settings*; • **Computer systems organization** → *Robotic control*;

Additional Key Words and Phrases: Learning from demonstrations, optimal control, physical human-robot interaction

ACM Reference format:
Dylan P. Losey and Marcia K. O'Malley. 2019. Learning the Correct Robot Trajectory in Real-Time from Physical Human Interactions. *ACM Trans. Hum.-Robot Interact.* 9, 1, Article 1 (December 2019), 19 pages.
https://doi.org/10.1145/3354139

1 INTRODUCTION

Our work seeks to leverage physical human-robot interaction (pHRI) to adapt robot behavior during the current task. We treat physical interactions as a method for conveying information, which the human can use to communicate his or her desired behavior to the robot. Consider a personal robot that is setting the table for a human end user: this robot may not know the right location to place a plate or the right way to move that plate to the goal. The human can physically intervene—by applying forces and torques—and *correct* the robot's behavior, such as pushing the robot towards the table or guiding the robot away from the stove. Our research develops robotic controllers that

This work was funded in part by the NSF Graduate Research Fellowship under grant GRFP-1450681.
Authors' address: D. P. Losey and M. K. O'Malley, Department of Mechanical Engineering, Rice University, 6100 Main Street, Houston, TX 77251; emails: dlosey@stanford.edu, omalleym@rice.edu.

2573-9522/2019/12-ART1
https://doi.org/10.1145/3354139

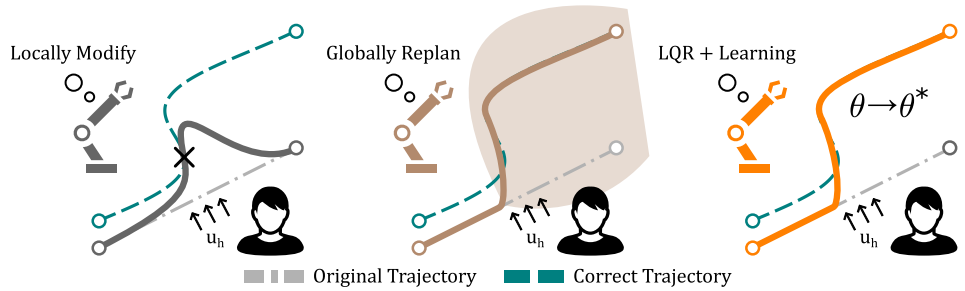

Fig. 1. Updating the robot's trajectory when responding to physical human corrections. Solid lines show the robot's updated trajectory. Today's robots either modify a local segment of their trajectory to match the human's correction (left), or search a region for the optimal trajectory offline, via replanning or demonstrations (middle). Our approach extends linear-quadratic regulator control to update either the local or global trajectory in one shot, without offline planning or previous examples (right).

adjust to these corrections in real time such that after the human lets go, the robot autonomously completes the rest of its current task in accordance with the end user's preferences.

Solving this problem requires that the robot update its underlying trajectory in response to the human's interactions. Within the state of the art, however, it is challenging for the robot to determine how to relate the human's low-level torque input to meaningful changes in its trajectory, and how to modify its trajectory in real time so that the robot's reaction is seamless and fluent.

One common pHRI strategy is for the robot to modify its trajectory to move in the direction of the human's correction [11, 15, 24, 27]. Here, the robot treats the human's interactions as local information—where the human expert pushes, pulls, or twists the robot toward a better state—and the robot adjusts its current waypoint (or next few waypoints) to match the human. But the robot never relates these local adjustments to global changes: after the human stops interacting, the trajectory soon returns to normal. Other pHRI works address this problem by learning the important aspects of the task from physical human interactions (i.e., kinesthetic demonstrations) and then replanning the robot's entire trajectory [4, 14, 19, 38]. Here, the robot successfully relates the human's low-level torque inputs to high-level trajectory parameters, but the learning and replanning occur offline, between iterations of the same task. In summary, robots either fail to meaningfully update the entire trajectory based on the current physical interaction or require computationally expensive trajectory replanning (or offline demonstrations), which prevent the robot from reacting seamlessly during the current task.

Within this work, we focus on a robot that should ideally complete its task autonomously, without any assistance or collaboration from a human partner. A nearby human notices that the robot is not performing its task in the right way; this human physically interacts and corrects the robot to better match his or her personal preferences. Harnessing the communication implicit within these physical interactions, we derive a method for the robot to meaningfully update its entire trajectory in real time. Importantly, this method is suitable for cases where the robot has never before performed this task or interacted with its current end user. Ultimately, we are interested in a *one-shot* learning process, where the robot only performs the task a single time and learns the correct trajectory for the rest of this task based on the current human input. Toward this end, we make the following insight:

The human's physical interactions guide the robot's current trajectory toward a desired trajectory that he or she would prefer the robot to follow.

Leveraging this insight, we derive a computationally efficient yet versatile learning rule that generalizes to meaningfully update either the local or global trajectory in real time based on the

current physical human interaction. Our approach extends traditional linear-quadratic regulator (LQR) control with online gradient descent to simultaneously modify both the underlying trajectory and the robot's optimal control action. Our key novelty is captured in Figure 1: the proposed approach outlines an optimal response to pHRI that is also suitable for real-time implementation. More specifically, we make the following contributions.

Extending LQR to learn from physical corrections. Based on our insight that the human's interactions guide the robot toward a better trajectory, we derive a computationally efficient learning rule for real-time implementation, where the robot meaningfully updates its underlying trajectory after each human interaction. We also provide convergence and stability guarantees—demonstrating that the learning rule safely extends LQR control—and provide two common instantiations. Under this approach, the robot can change its entire trajectory and control torques in real time, during the current task, based on the human's implicit corrections.

Generalizing across local and global trajectory updates. Our theoretical approach can be applied to either learn about a local segment of the trajectory or update the robot's global trajectory. We obtain this versatility by introducing an unknown trajectory parameter θ, where the designer chooses how to parameterize the desired trajectory based on his or her task requirements. If θ consists of low-level parameters—like waypoints—our learning rule locally modifies the trajectory by adjusting those waypoints; alternatively, if θ contains high-level parameters—like features—this same learning rule modifies the entire trajectory to match the updated features.

Analyzing best- and worst-case performance. We benchmark our extension of LQR control against comparable algorithms for updating the desired trajectory in response to human interactions. We show that, in the best case, our approach objectively performs on par with the optimal offline solution but can be implemented at the same loop rate as local trajectory modification methods. In the challenging cases, we find that our approach is robust to end users whose interactions diverge from the robot's expectations, but other, non-learning approaches eventually obtain objectively better performance when the robot has significant errors in the type of trajectory parameterization.

Comparing to the state of the art in a user study. When the robot does not know the right trajectory, we conduct a user study comparing real-time responses where (1) the robot lets the human take control as soon as an interaction is detected, (2) the robot locally modifies its trajectory in the direction of the physical interaction, and (3) the robot implements our extension of LQR control. We consider tasks both where the correct trajectory is close to or far from the initial trajectory, as well as tasks that have the right path but the wrong timing. We evaluate the objective and subjective results across both *interaction strategy* and *task type*.

Overall, this article introduces a computationally efficient solution to learning from physical interactions, grounds this solution in optimal control theory, and compares the resultant approach to state-of-the-art methods. Our learning and control strategy is suitable for one-shot applications, where the robot should adapt the rest of its current trajectory to match the end user's preferences. We explore applications of this work for robotic rehabilitation, where an expert therapist can use the proposed approach to intuitively modify the robotic training task for a participant. We also envision applications within household robotics, where end users need to adapt the robot's baseline behavior to suit his or her personal needs and environment. Within these settings, pHRI provides a natural avenue for communicating desired behavior to the robot.

2 RELATED WORK

Our research builds on prior pHRI works that consider *trajectory updates* and *control strategies*.

2.1 Trajectory Updates from Physical Human Interaction

Collaborative tasks. Several works adapt the robot's trajectory during collaborative tasks—such as sawing, assembling, or carrying—where the human and robot partners must constantly act together [14, 35, 38]. For instance, Rozo et al. [38] propose a method where the robotic assistant learns the right trajectory from offline physical demonstrations and apply this method during experiments where the robot and human assemble a table. Unlike these works, we focus on settings where the robot should complete its task *autonomously*, without collaborating with a human partner.

Avoiding end users. Other research modifies the robot's trajectory to avoid interacting with the human end user [8, 22, 23, 28]. For instance, Mainprice and Berenson [28] develop a scheme where the robot predicts the human's future motion and replans its trajectory to complete the task optimally, without interfering with the human's movement. By contrast, we consider situations where the human wants to physically interact with the robot to convey his or her preferences.

Local modifications. When the human end user applies forces and torques, the robot can respond by locally modifying its trajectory [1, 9, 11, 15, 24, 27]. Akgun et al. [1] enable the human to physically adjust important keyframes along the robot's trajectory so that—the next time the robot performs the task—it moves through these waypoints. Haddadin et al. [15] and Li et al. [24] develop reaction strategies to modify the robot's next few waypoints: the robot deforms its trajectory in the direction of the human's applied disturbance [9, 27] or to maintain a pre-defined interaction force. For each of these approaches, the robot updates a local segment of its trajectory (e.g., a keyframe or the next few waypoints), but the robot does not learn a new trajectory from start to goal. Instead, we are interested in leveraging local human interactions to update the global robot trajectory.

Adapting existing trajectories. Some prior approaches update the robot's global trajectory by adapting its current trajectory to the new scenario [13, 29, 32, 37]. In Nierhoff et al. [32], for example, the authors adapt the robot's entire trajectory in real time to satisfy changing environment constraints while remaining close to the original trajectory (without physical interactions). More related to our approach is work by Gams et al. [13]: here, the robot learns from physical coaching during periodic movements. Unlike this research, our work focuses on trajectory adaptation with optimal control and does not require periodic movements or multiple trials.

Replanning optimal trajectories. If we treat physical human interactions as observations about an underlying objective function, then the robot updates its objective after each interaction and replans an optimal trajectory with respect to this new objective [4, 5, 19]. Jain et al. [19] present one such formulation, where the human physically corrects a robot waypoint offline, and the robot solves for its optimal trajectory at the next trial. Although we have extended this method for real-time implementation [4, 5], the approach is practically limited to only a few features and simple tasks, since optimal trajectory replanning for more complex settings cannot be performed in less than a second [21, 39]. Accordingly, we treat trajectory replanning as an optimal offline solution and assess how much performance the robot loses using our real-time (i.e., sub-millisecond) approach.

Viewed together, prior works leverage physical human interactions to modify a local segment of the robot's trajectory, or to conduct offline replanning, which may rely on multiple trials or past demonstrations. Alternatively, we propose a *one-shot* approach for autonomous tasks, where the robot relates low-level interaction forces to real-time changes in the rest of its trajectory.

2.2 Control Strategies for Physical Human Interaction

Compliant responses. When a human and robot are physically interacting, compliant control is an important step toward human safety [10, 16, 17]. Here, the robot responds to physical interactions

by rendering an impedance [17]—for instance, a stiffness and damping—so that the human can locally push the robot away from its underlying trajectory. Our work continues this trend and encodes the robot to render an impedance during physical interactions.

Shared control. More generally, the human and robot can dynamically alter leader and follower roles during pHRI through shared control [12, 26, 31]. For example, Mörtl et al. [31] formalize how the human's and robot's effort should be divided during a collaborative task: intuitively, the robot lets the human take control when the human's feedback diverges from its expectations. Arbitration serves as a knob between complete robot autonomy (at one extreme) and passively following the human (at the other extreme) [12, 26]. In our work, we view shared control as a method that enables the human to easily correct the robot's state and treat it as a real-time *baseline strategy*.

Optimal control. Other research studies optimal control strategies for tasks that involve physical human interaction [20, 25, 30]. In Li et al. [25], the authors leverage game theory and optimal control to identify the correct robot behavior: like Medina et al. [30], their approach reduces the robot's stiffness when the human applies forces and torques but increases the rendered stiffness when the human does not interact, or when the human's interactions agree with the robot's prediction.

For each of the preceding control strategies, the robot returns to its original trajectory once the human stops interacting—for instance, the robot never adapts its underlying trajectory. By contrast, we develop a response that updates the robot's trajectory so that the robot completes the rest of its task in a different way after the interaction ends.

3 PROBLEM STATEMENT

Here, we formulate the problem setting for which our pHRI response strategy is appropriate. We discuss the states, actions, and dynamics; the desired trajectory; and the cost function. A key element of our formalism is the set of *trajectory parameters*, θ, which encode the trajectory that the robot will attempt to track. Importantly, the robot does not know the right trajectory parameters.

3.1 States, Actions, and Dynamics

Consider an n degree-of-freedom (DoF) robotic manipulator with joint position $q \in \mathbb{R}^n$ and joint velocity $\dot{q} \in \mathbb{R}^n$. Let the robot's state be $x \in \mathbb{R}^{2n}$, where $x = [q^T, \dot{q}^T]^T$. There are two system inputs: $u_r \in \mathbb{R}^n$ is the robot's control torque (in joint space), and $u_h \in \mathbb{R}^n$ is the disturbance torque (in joint space) caused by human-robot interaction. If we assume that the robot is gravity compensated, we can write the robot's state-space dynamics as [40]

$$\dot{x}(t) = F(x)x + G(x)(u_r + u_h). \tag{1}$$

Here, F captures the robot's passive dynamics, and G is the control matrix. In general, Equation (1) describes a class of systems where both the human and robot actions, $u_r + u_h$, alter the system state x. We will focus on robots that fall within this class of systems [4, 5, 16, 27].

3.2 Desired Trajectory

The robot has a desired trajectory (also called a *reference trajectory*) that it is attempting to track. Let the desired trajectory be $\xi \in \Xi$. Without loss of generality, we assume that ξ is parameterized by a constant vector $\theta \in \mathbb{R}^m$ such that at the current time t, the desired state is $\xi(\theta, t)$. Hence, $\xi \in \Xi$ is a function that maps the parameters θ and timestep t into a desired system state, and Ξ is the set of possible mappings $\mathbb{R}^m \times \mathbb{R}^+ \rightarrow \mathbb{R}^{2n}$. In this work, we leave the choice of ξ up to the designer and focus on learning the right value of θ given ξ. For example, the designer might choose ξ so that θ is a vector containing the waypoints along the desired trajectory and $\xi(\theta, t)$ outputs the waypoint corresponding to time t.

3.3 Cost Function

Ideally, the robot should track the correct desired trajectory while also minimizing its own effort. The correct trajectory is determined by the human end user and is parameterized by θ^*. Within our formulation, θ^* captures the human's preferences for how the robot should behave-for example, the goal the robot should reach, the speed of the robot's motion, and/or the trajectory features. Similar to recent works on optimal control for pHRI [20, 25, 30], we therefore have the following cost function:

$$J(\theta^*) = \int_{t_0}^{t_f} \mathcal{L}\Big(x(t), \xi(\theta^*, t), u_r(t)\Big)\, dt. \tag{2}$$

Here, the task starts at time t_0, ends at time t_f, and the robot incurs the total cost $J \in \mathbb{R}^+$. Let the loss function $\mathcal{L} \in \mathbb{R}^+$ be quadratic so that we reach the classical LQR control problem [41]:

$$\mathcal{L}(t) = \Big\|x(t) - \xi(\theta^*, t)\Big\|_Q^2 + \Big\|u_r(t)\Big\|_R^2. \tag{3}$$

Note that $\|x\|_A^2 = x^T A x$ within this notation. The inner product matrices Q and R are positive definite and symmetric, and determine the relative costs of errors from the correct desired trajectory and increases in the robot's effort, respectively.

3.4 Unknown Trajectory Parameters

If the robot knows θ^*, then the solution is straightforward: the robot should simply use LQR control to track $\xi(\theta^*)$ [2]. Physical human corrections are unnecessary in this case, as the robot is already following the human's preferred trajectory and minimizing the true cost function.

Our work considers cases where θ^* is not known *a priori*: instead, here the robot maintains a point estimate θ. At the current time t, the error between the correct robot trajectory (parameterized by θ^*) and the actual desired trajectory (parameterized by θ) is

$$e(t) = \xi(\theta^*, t) - \xi(\theta, t). \tag{4}$$

Intuitively, we want $\theta \to \theta^*$ so that $e \to 0$ and the robot tracks the correct trajectory. Thus, our main problem becomes learning from physical human corrections to update θ while simultaneously selecting robot actions in a way that minimizes the estimated cost function, $J(\theta)$.

4 LEARNING FROM PHYSICAL CORRECTIONS

In this section, we map the human's physical interactions u_h into real-time updates of the estimate θ so that the robot can learn the right trajectory from physical corrections. This approach is inspired by our insight from Section 1 and prior work on learning from pHRI [1, 4, 13, 19, 27]: the human interactions reduce the error between the robot's current trajectory and their preferred trajectory. We define $e = [e_p^T, e_v^T]^T$, where $e_p \in \mathbb{R}^n$ is the position error and $e_v \in \mathbb{R}^n$ is the velocity error. From our insight, we assert that the human interacts based on some linear combination of the position error and velocity error:

$$u_h(t) \propto \lambda_1 \cdot e_p(t) + \lambda_2 \cdot e_v(t), \tag{5}$$

where $\lambda_1, \lambda_2 > 0$. Hence, we model the human's actions as *observations* about the trajectory error from Equation (4). Note that the observation u_h is a lower-dimensional embedding of the error e.

4.1 Gradient Descent

To learn from physical corrections and update θ in real time, we apply online gradient descent [3, 7]. We want to find values of θ that converge toward θ^*, or, expressed more generally, we are

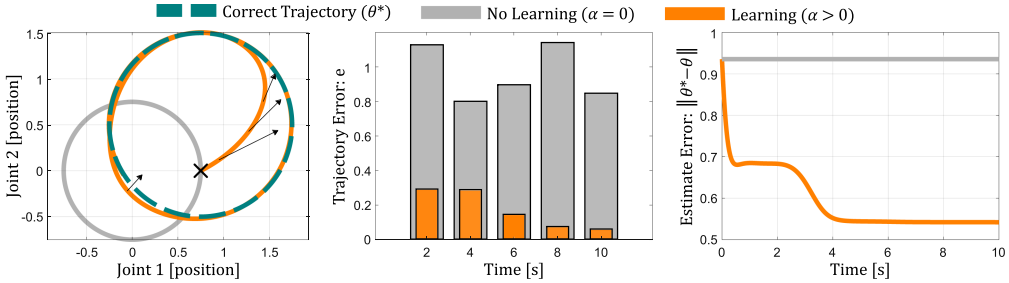

Fig. 2. Learning the correct trajectory from physical human interactions. A 2-DoF robotic arm starts in the marked position and updates its underlying trajectory based on the human's physical interactions (left, where arrows show the human's applied force). Learning from pHRI with Equation (9) reduces the error between the correct and actual trajectories (middle), and the error between the true and estimated parameters (right) when compared to a robot that does not learn.

searching for the θ that minimizes the convex loss function $V \in \mathbb{R}^+$:

$$V(t) = \frac{1}{2}\left\|e(t)\right\|^2 = \frac{1}{2}\left\|\xi(\theta^*, t) - \xi(\theta, t)\right\|_A^2, \quad A = \begin{bmatrix} \lambda_1^2 I & \lambda_1\lambda_2 I \\ \lambda_1\lambda_2 I & \lambda_2^2 I \end{bmatrix}, \tag{6}$$

where A is a positive semi-definite matrix that penalizes the linear combination $\lambda_1 e_p + \lambda_2 e_v$. Put another way, $V = 0$ when $\lambda_1 e_p + \lambda_2 e_v = 0$. Let us take the gradient of Equation (6) with respect to the vector $\theta \in \mathbb{R}^m$, where we recall that the human's preferences θ^* are assumed to be constant:

$$\nabla_\theta V(t) = -\left[\frac{\partial \xi(\theta, t)}{\partial \theta_1} \cdots \frac{\partial \xi(\theta, t)}{\partial \theta_m}\right]^T Ae(t). \tag{7}$$

In the preceding text, the bracketed term is a $2n \times m$ Jacobian matrix that relates changes in θ to changes in the current desired state, $\xi(\theta, t)$. Applying the principle of steepest descent, the robot locally decreases V by updating θ in the direction: $-\bar{\alpha} \cdot \nabla_\theta V(t)$, where $\bar{\alpha} > 0$. Substituting in Equation (7),

$$\dot{\theta}(t) = \bar{\alpha} \cdot \left[\frac{\partial \xi(\theta, t)}{\partial \theta_1} \cdots \frac{\partial \xi(\theta, t)}{\partial \theta_m}\right]^T Ae(t). \tag{8}$$

Inspecting Equation (8), we notice that—during implementation—the designer specifies $\bar{\alpha}$ and can solve for the Jacobian matrix. Put another way, the only unknown term is e (since the robot does not know θ^*), but because e is unknown, we cannot solve Equation (8) in its current form.

4.2 Leveraging Our Insight

Here, we apply our insight and treat u_h as an observation of the trajectory error e. Recall from Equation (5) that $u_h \propto [\lambda_1 I, \lambda_2 I]e$, and so $Ae \propto [\lambda_1 I, \lambda_2 I]^T u_h$. Substituting this observation model from Equation (5) into Equation (8), our *learning rule* for physical corrections becomes

$$\dot{\theta}(t) = \alpha \cdot \left[\frac{\partial \xi(\theta, t)}{\partial \theta_1} \cdots \frac{\partial \xi(\theta, t)}{\partial \theta_m}\right]^T \begin{bmatrix} \lambda_1 I \\ \lambda_2 I \end{bmatrix} u_h(t), \tag{9}$$

where $\alpha > 0$ and $\lambda_1, \lambda_2 > 0$ are parameters that the designer tunes to determine the learning rate (α) and the weight of position or velocity errors (λ_1, λ_2). For example, selecting $\lambda_1 = 1, \lambda_2 = 0$ implies that the human interacts based purely on the position error and causes the robot to update θ such that the position error e_p converges to zero. Overall, Equation (9) maps pHRI to changes in the robot's trajectory estimate. This learning rule is suitable for real-time implementation and, by changing θ, modifies the robot's entire trajectory (Figure 2).

4.3 Common Instantiations

Next, we derive instantiations of our learning rule for two common trajectory parameterizations. First, we consider a trajectory parameterized through *linear regression* so that $\xi(\theta, t) = \Phi(t)\theta$. Here, $\Phi \in \mathbb{R}^{2n \times m}$ encodes the trajectory basis functions, and θ determines their relative weights: any trajectory can be written in this form. For linear regression, our learning rule simplifies to

$$\dot{\theta}(t) = \alpha \cdot \Phi(t)^T \begin{bmatrix} \lambda_1 I \\ \lambda_2 I \end{bmatrix} u_h(t). \tag{10}$$

Second, we consider a *dynamic movement primitive* (DMP), which represents the trajectory as a second-order dynamical system [18]. Let $s > 0$ be a phase variable monotonically decreasing to zero, let $K \in \mathbb{R}^{n \times n}$ be a virtual compliance constant, and let $f(x, s, \theta) \in \mathbb{R}^n$ be a forcing term that shapes the attractor landscape [33, 34]. For DMPs, our learning rule reduces to

$$\dot{\theta}(t) = \alpha \cdot \left[\frac{\partial \mathbf{f}(x_0, t, \theta)}{\partial \theta_1} \cdots \frac{\partial \mathbf{f}(x_0, t, \theta)}{\partial \theta_m} \right]^T K^T u_h(t), \tag{11}$$

where $\mathbf{f}(x_0, t, \theta) = \int_{t_0}^{t} f(x, s, \theta)\, dt$ and the Jacobian is evaluated at the current estimate θ.

There are other implementation techniques of interest. For instance, instead of updating every parameter in θ after each interaction, an alternate approach is to only update a few parameters per timestep [5]. In this case, we first calculate Equation (9) and then update only the parameters θ_i in θ for which $|\dot{\theta}_i|$ is the largest (after normalization). This approach may be suitable when the human teaches sequentially and corrects one aspect of the desired trajectory at a time.

5 EXTENDING LQR CONTROL WITH LEARNING

In this section, we extend classical LQR control with the learning rule derived in Section 4. Combining *LQR + Learning* (*LQR+L*) provides an optimal response to pHRI and also ensures that the robot interacts safely. We discuss how this approach generalizes the state of the art and list its limitations.

5.1 Optimal Response to Human Corrections

An optimal robot should select its actions u_r to minimize the cost function $J(\theta^*)$ from Equation (2), which depends on the human's true preferences θ^*. Given all of its previous observations, the robot's best guess of the true cost function is $J(\theta)$, where the desired trajectory is parameterized by the learned estimate θ. Accordingly, the *optimal robot response* to human interactions is

$$u_r(t) = -R^{-1} G^T P(t) \Big[x(t) - \xi(\theta, t) \Big]. \tag{12}$$

Here, P is the solution to a matrix Riccati equation (either finite or infinite horizon), and we linearized the robot's dynamics about the desired trajectory. Equation (12) is an LQR control law [2, 41].

5.2 Algorithm, Safety, and Stability

When the human physically interacts, the robot updates its underlying trajectory with Equation (9) and selects the optimal action for that learned trajectory using Equation (12). This interaction algorithm is *safe*, because in practice the LQR controller behaves like a variable impedance controller and compliantly responds to the human's interaction torques [16]. Our combination of learning and LQR control is also *stable* under reasonable sufficient conditions

Sufficient conditions for stability. *LQR+L* is stable if, for any time $t \in [t_0, t_f]$ and bounded parameters θ, the trajectory $\xi(\theta, t)$ is continuous and differentiable with respect to θ.

Proof. The end user's input u_h must be bounded (i.e., the human cannot apply infinite effort). Given that $\xi(\theta, t)$ is differentiable with respect to θ, from Equation (9) $\dot{\theta}$ is bounded, and therefore θ remains bounded. Because $\xi(\theta, t)$ is continuous, it is bounded on the closed interval $[t_0, t_f]$. A robot with dynamics (1) is stable when tracking a bounded trajectory with Equation (12) [2, 41]. □

In summary, human interactions cannot cause the learning and control system to become unstable.

5.3 Generalizing Across Local and Global Learning

Our proposed approach generalizes some previous works on learning from physical interactions. Regardless of whether the designer chooses ξ such that θ parameterizes local or global trajectory qualities, we can leverage our approach to update θ and adapt the underlying trajectory.

Local. If $\xi \in \Xi$ is defined so that θ are the *waypoints* along the trajectory, then *LQR+L* is equivalent to local trajectory modifications [1, 15, 24]. The robot uses Equation (9) to deform the local waypoints in the direction of the human's input and then tracks this updated trajectory.

Global. Alternatively, if the designer selects ξ such that θ are the weights for underlying *features* or *basis functions*, *LQR+L* behaves similarly to global trajectory replanning [4, 19]. The robot meaningfully relates human inputs to global changes in its entire trajectory with Equation (9), without requiring a complete demonstration or offline replanning.

Generalizing. Thus, how the designer specifies ξ determines how *LQR+L* updates the robot's current trajectory. If θ contains low-level parameters (like waypoints or keyframes), our proposed approach results in local modifications; however, if the designer selects meaningful high-level parameters (e.g., features or basis functions), then our method updates the global trajectory. We are excited that our theoretical learning framework generalizes to both local and global settings and does so while incorporating optimal control so that the robot selects optimal responses to pHRI regardless which type of unknown parameters (low level or high level) the designer specifies.

5.4 Challenges and Limitations

Although we have shown that *LQR+L* safely optimizes the robot's best guess of its cost function, there are still practical challenges when learning from physical interactions:

(1) The robot may not have the right hypothesis space—for instance, the designer may choose ξ such that no value of θ results in the end user's desired trajectory.
(2) The human's interactions may not match our observation model—that is, $u_h \neq \lambda_1 \cdot e_p + \lambda_2 \cdot e_v$.
(3) Even when the human interacts based on the error, the actions may be noisy and imperfect.

The listed challenges are not specific to our approach: modeling errors—both in the hypothesis space and the observation model—are often unavoidable when learning from physical interactions. We will explore how robust our approach is to these limitations in simulations and a user study.

6 SIMULATIONS

In this section, we perform offline simulations to analyze our proposed *LQR+L* algorithm. These simulations consider both the best- and worst-case performance, and benchmark our approach against comparable local and global methods for responding to physical corrections. We focus on challenging settings where the human's interaction strategy diverges from the robot's observation model, and where the robot does not have the right hypothesis space. Overall, these simulations are

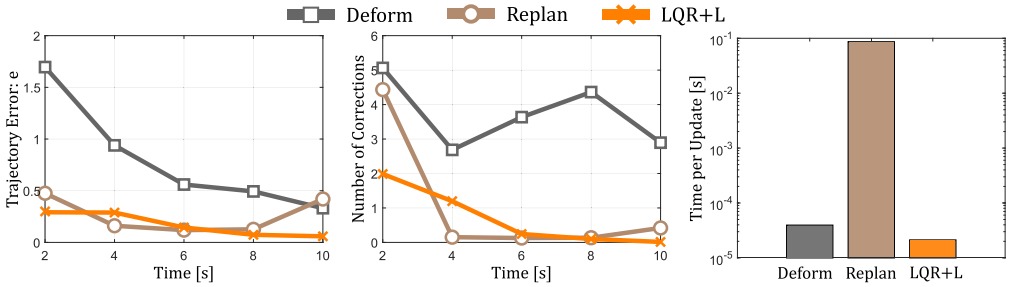

Fig. 3. Best-case performance when the robot knows the right models. Our proposed *LQR+L* algorithm learns the correct trajectory from pHRI (left) so that the human no longer needs to interact with the robot (middle) and achieves objective performance on par with the optimal *Replan* approach. However, *LQR+L* works online: both local modifications and *LQR+L* can be implemented at 10 kHz, whereas *Replan* operates at 10 Hz (right). When using local modifications in *Deform*, the end user has to repeatedly intervene to correct the robot.

meant to partially address challenges (1) and (2) from Section 5.4. We perform these simulations on a 2-DoF robotic arm, and we leverage both of the trajectory instantiations described in Section 4.3.

6.1 Best-Case Performance

Here, we compare robots that locally modify their trajectory in the direction of the interaction (*Deform*) [1, 16, 24], robots that learn the objective function from physical interactions and replan an optimal trajectory (*Replan*) [4, 19], and our proposed *LQR+L* approach. Each of these robots knows the right trajectory mapping ξ and the correct observation model relating u_h to θ. The simulated human interacts based on e, the desired trajectory error, without additional noise.

Setup. We used the same setting as shown in Figure 2: the human end user wants to alter the center and radius of a robot's stirring motion. Here, we parameterize the desired trajectory using linear regression so that θ captures the robot's radius, center position, and linear offset along the x-axis. More formally, the desired trajectory is parameterized by

$$\xi(\theta, t) = \Phi(t)\theta = \begin{bmatrix} t & \sin(t) & 1 & 0 \\ 0 & \cos(t) & 0 & 1 \end{bmatrix}\theta, \quad \theta \in \mathbb{R}^4 \tag{13}$$

where the t term in Φ is a spurious offset the robot should learn to ignore. We compared the error between the correct trajectory and the robot's learned trajectory, as well as the rate at which the trajectory was updated for *Deform*, *Replan*, and *LQR+L*. Simulations were conducted in MATLAB (MathWorks). Similar to prior works that replan an optimal trajectory [4, 19], here *Replan* had fixed start and goal positions and did not learn a new goal position from the human's interactions.

Results. We found that our proposed *LQR+L* algorithm obtains similar trajectory error to an optimal replanning method but updates its trajectory in real time, at the same rate as a local modification approach (Figure 3). Indeed, for this simple task, our *LQR+L* robot updated its trajectory every 0.022 ± 0.001 ms, but the *Replan* robot updated its trajectory every 86.3 ± 1.2 ms. We conclude that—in the best case—using *LQR+L* is advantageous because it approaches the optimal performance of global replanning approaches (in terms of trajectory error and human effort) but does so at the same loop rate as local modification methods.

Learning versus non-learning. Now that we have simulated our best-case performance against other learning approaches, we will move on to two worst-case scenarios. Here, the robot has the wrong observation model (i.e., challenge (2)) or an incorrect hypothesis space (i.e., challenge (1)). Learning is difficult in these cases, as the robot cannot correctly interpret the meaning behind the human's

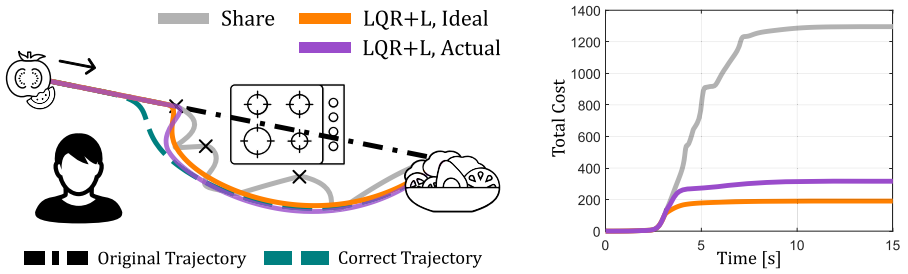

Fig. 4. Challenging case performance when the human's actual corrections diverge from the robot's observation model. The shared control baseline does not rely on any human models but also never updates the robot's trajectory, and so the end user must repeatedly guide the robot away from the stove (left). Although the robot incurs more cost $J(\theta^*)$ when the human makes decisions purely based on the robot's actual state—rather than the trajectory error—*LQR+L* still outperforms the *Share* baseline during a one-shot task (right).

interactions; hence, to assess the robustness of *LQR+L*, we will compare our proposed approach to a *non-learning*, shared control baseline.

6.2 Challenging Case Performance: Incorrect Observation Model

Next, we test *LQR+L* when the robot has the wrong observation model—that is, the human does not interact based on the trajectory error e. Instead, we simulate a human that physically interacts based on the actual error between their desired state $\xi(\theta^*)$ and the robot's current state x. This situation can arise when the robot does not accurately track its underlying trajectory $\xi(\theta)$, and—despite the haptic feedback from the LQR controller—the human is unsure about how the robot plans to move. We compare our approach with a ideal human end user who follows our observation model (*LQR+L, Ideal*), our approach when the actual human diverges from the robot's observation model by applying forces in the direction $\xi(\theta^*, t) - x(t)$ (*LQR+L, Actual*), and a baseline shared controller, which does not involve learning and is therefore robust to model errors (*Share*).

Setup. The robot is completing a pick-and-place task within the kitchen (Figure 4, left). As the robot performs this task, the human end user notices that the robot is moving above a hot stove: the human physically intervenes and corrects the robot away from the stove. We parameterized the robot's trajectory using a DMP. For *Share*, the robot cedes control to the human whenever pHRI occurs, enabling the human to guide the robot without changing its underlying trajectory [12, 31].

Results. Learning under our proposed method resulted in objectively better performance than a baseline controller without learning, even in cases where the robot had an incorrect observation model (see Figure 4, right). When the ideal human's interactions matched the robot's observation model, *LQR+L, Ideal* incurred 14.7% as much total cost as *Share*, and—even when the human's interactions were based only on the robot's actual position—*LQR+L, Actual* incurred 24.4% as much total cost as *Share*. These results suggest that *LQR+L* is robust to observation model errors and is still an effective strategy if the human makes decisions based purely on the robot's current state, as opposed to the error between the current and preferred trajectories.

6.3 Challenging Case Performance: Incorrect Hypothesis Space

Finally, we test *LQR+L* when the robot has the wrong hypothesis space ξ such that the correct trajectory cannot be accurately represented with any parameter θ. This problem arises when ξ

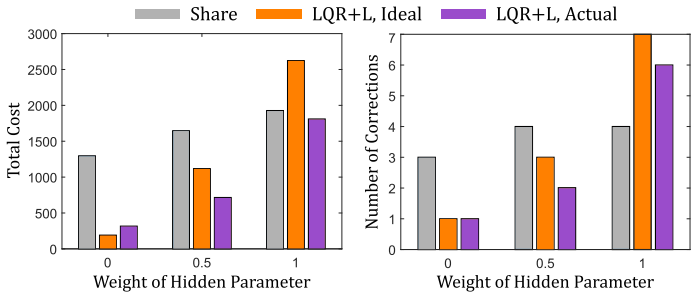

Fig. 5. Challenging case performance for tasks where the robot has a misspecified hypothesis space. When ξ is chosen such that no value of θ leads to the right behavior, our proposed approach cannot correctly interpret the meaning behind the end user's interactions. Shared control becomes advantageous as the hypothesis space error increases, both in terms of total cost (left) and the number of corrections (right). Interestingly, *LQR+L* is more robust to an incorrect hypothesis space when the human interacts based only on the actual position.

is defined such that θ consists of high-level features or basis functions, but the human has in mind a different set of features, and thus the robot requires a different trajectory parameterization to correctly interpret the human's interactions. We again compare a *Share* baseline against our algorithm with the right observation model (*LQR+L, Ideal*) and our algorithm with an incorrect observation model: note that *LQR+L, Actual* has both the wrong hypothesis space and the wrong observation model.

Setup. The robot is performing the same pick-and-place task as in Figure 4 (left), but now the end user additionally wants the robot to stay away from their body. We define ξ to include the distance-to-stove feature (which is weighted by θ), but a distance-to-user feature is not included within ξ. Hence, the robot does not know that distance to user is a possible feature: when the human guides the robot away from his or her body, the *LQR+L* robots relate those interactions to the distance to stove and cannot correctly interpret why the human is interacting. We measured the total cost and the number of corrections while varying the true importance of the hidden parameter (distance to user).

Results. Our approach for learning from pHRI outperforms a shared control baseline when the robot has small errors in the hypothesis space, but *Share* becomes preferable when the modeling errors dominate (Figure 5). *LQR+L* is robust to both hypothesis and observation errors, however; *LQR+L, Actual* received 68.9% as much total cost as *LQR+L, Ideal* in the worst case, likely because making decisions based purely on the robot's state resulted in less undoing. We recommend using a shared control approach when the hypothesis space is largely unknown—or, alternatively, we can define θ to be waypoints, which removes the possibly of an incorrect hypothesis space.

6.4 Summarizing the Simulations

In the best case, we found that *LQR+L* reaches objective performance similar to an offline replanning method but operates in real time, without any prior demonstrations. In the worst case, our approach is robust to observation model errors: we simulated end users who corrected the robot based only on its current state and found a negligible loss in performance when compared to a non-learning baseline. By contrast, if the robot has large errors within the hypothesis space, *LQR+L* performs worse than the non-learning baseline (but remains robust to observation model errors). Overall, these simulations address challenges (1) and (2) from Section 5.4.

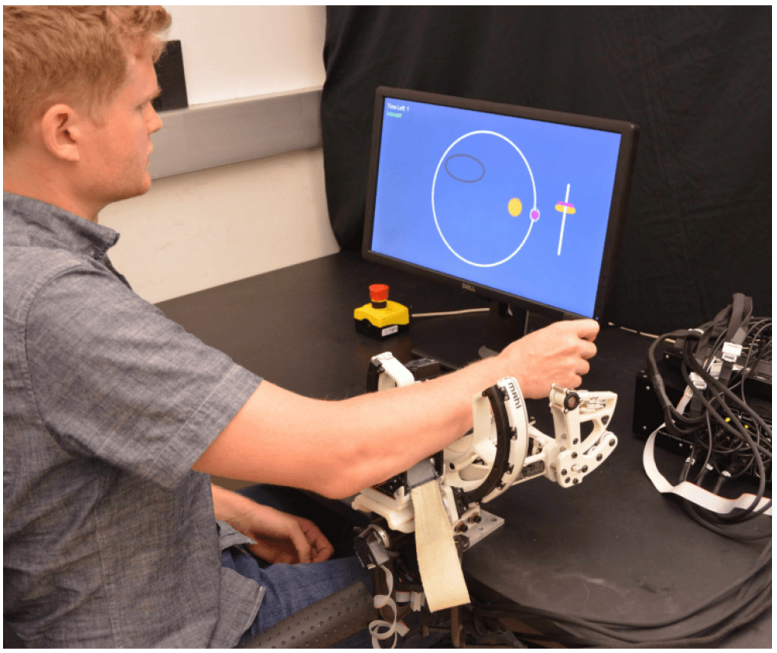

Fig. 6. Experimental setup used within our user study. Participants physically interacted with a 3-DoF robot to alter its trajectory during the current task. We displayed the correct trajectory on the screen and compared different real-time interaction strategies for updating the robot's underlying trajectory.

7 USER STUDY

To explore all three challenges from Section 5.4 when interacting with actual humans, we conducted a user study (Figure 6). Here, the robot started to perform a task with the wrong trajectory, and participants physically corrected the robot during its current task execution, without prior demonstrations or offline learning. We compared real-time shared control and local modification approaches to our *LQR+L* method across three tasks: one where only slight changes were needed, one that required global adjustments, and a final task with incorrect motion timing. To make the comparison more realistic, the *LQR+L* robot had *superfluous parameters* within its desired trajectory function ξ—so that this robot could learn unintended, erroneous movements—and the participants could not directly observe the robot's underlying trajectory. Importantly, the participants corrected the robot in noisy and imperfect ways; we assessed whether these measured corrections negatively impacted our *LQR+L* approach or the robot's movement smoothness.

7.1 Independent Variables

We varied the *interaction strategy* with three levels: one strategy let the human guide the robot during interactions without changing its underlying trajectory (*Share*), a second strategy locally modified the trajectory in the direction of interaction torques (*Deform*), and the third strategy was our proposed *LQR+L* approach. We did not include a replanning strategy, as we were focusing on real-time (sub-millisecond) response methods.

We also varied the *task type* to suit the advantages and disadvantages of each interaction strategy (Figure 7). In *Task 1*, the robot's initial trajectory was far from the correct trajectory, and so large modifications were needed. In *Task 2*, the robot's initial trajectory was close to the correct trajectory, and therefore minor adjustments were sufficient. Finally, in *Task 3*, the robot started

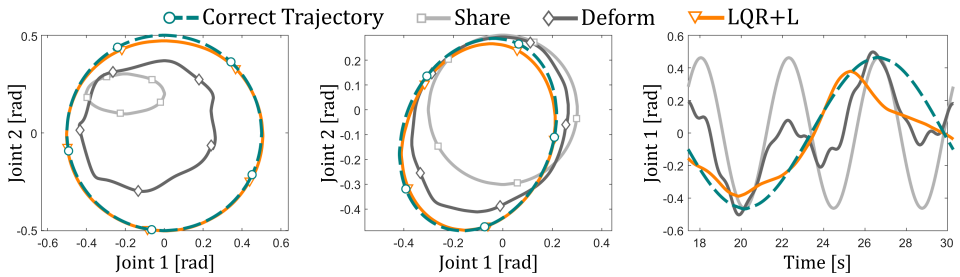

Fig. 7. The correct and learned trajectories (averaged across all participants) for each of our experimental tasks. Because *Share* does not learn a new trajectory from pHRI, here *Share* is the same as the robot's initial trajectory. We included tasks that required global changes (left), local modifications (middle), and timing adjustments (right). Although each task involved three DoFs, we only plot two joints for clarity.

with the right path but had the wrong movement timing. In all three tasks, the robot's desired trajectory ξ was parameterized using linear regression, and we provided superfluous terms that were not necessary to learn the correct trajectory. For example, in *Task 1*, the robot started with the right movement timing, but the robot could learn a different (and incorrect) timing from interactions.

7.2 Measured Outcomes

For each interaction strategy and task type, we measured the robot's total incurred cost, the amount of time the human spent interacting, and the smoothness of the robot's movement. Total cost was computed using (2), whereas movement smoothness was quantified by the spectral arc-length of the robot's speed profile [6]. In addition to these objective measures, we also recorded the participants' subjective responses to a 7-point Likert scale survey. The questions—as listed in Table 1—were selected to determine how participants felt about the robot's learning, predictability, response speed, and interaction effort.

7.3 Experimental Setup

Participants interacted with the OpenWrist [36], a 3-DoF robot developed for pHRI. Each of the interaction strategies were implemented on a desktop computer with the Mechatronics Engine and Library (https://github.com/epezent/MEL). The computer communicated with the OpenWrist robot through a Q8-USB data acquisition device (Quanser): this system updated the robot's underlying trajectory and control input at a frequency of 1 kHz. So that every participant used the same preferred trajectory, we displayed the trajectory—as well as the robot's current state—on a screen in front of each of the participants. Our user interface was developed with Unity (Unity Technologies).

7.4 Participants and Procedure

Our participant pool consisted of 12 Rice University affiliates who provided informed consent. Of the 12 total participants, 5 were female and 7 were male, with ages ranging from 19 to 31 years (25.8 ± 3.6). Ten of the 12 participants had a technical background.

We used a *within-subjects* design, where the order of the interaction strategies and task types were separately counterbalanced. Participants were not told about the different interactions strategies; however, to reduce in-task learning, we allowed participants to familiarize themselves with each strategy and task once before the nine recorded trials (three strategies $\times$ three tasks).

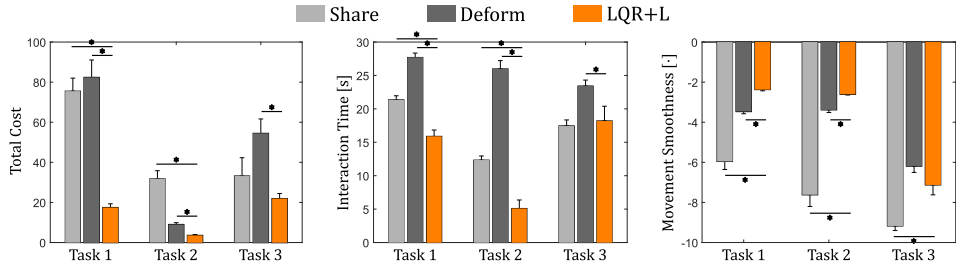

Fig. 8. Objective results from our user study for each interaction strategy and task type. We found that *LQR+L* outperformed a shared control baseline and a local modification approach in terms of total cost (left), interaction time (middle), and movement smoothness (right) for both *Task 1* and *Task 2*. During the third task—where the robot had the wrong timing—the objective results were mixed. Error bars denote the standard error about the mean (SEM), and an asterisk denotes statistical significance ($p < .05$).

7.5 Results

Objective. The objective results from our user study are displayed in Figure 8. To interpret these objective measures, we performed a two-way repeated measures analysis of variance (ANOVA), where the robot's interaction strategy and the task type were factors. We found a significant interaction effect between interaction strategy and task type for cost ($F(4, 44) = 21.6$, $p < .001$), interaction time ($F(4, 44) = 17.3$, $p < .001$), and movement smoothness ($F(4, 44) = 8.33$, $p < .001$). We therefore reran our ANOVA with simple main effects.

LQR+L resulted in significantly less total cost than either *Share* or *Deform* during both *Task 1* and *Task 2*. During *Task 3*, however, *LQR+L* yielded a significantly lower cost than *Deform* ($p < .001$), but not significantly lower than *Share* ($p = .242$). Similarly, we found that users with *LQR+L* had significantly less interaction time than with either *Share* or *Deform* during both *Task 1* and *Task 2*, and that the *LQR+L* robot also completed these tasks significantly more smoothly. In *Task 3*, we observed that *LQR+L* led to significantly less interaction time than *Deform* ($p = .020$), but there was no significant difference in interaction time between *LQR+L* and *Share* ($p = .664$). Conversely, here *LQR+L* resulted in significantly more smooth movements than *Share* ($p = .002$), but not *Deform* ($p = .131$). We therefore conclude that *LQR+L* improved total cost, interaction time, and movement smoothness when participants needed to change either the global or local trajectory, but when the trajectory timing was wrong the results were less clear cut.

Subjective. The results from our 7-point Likert scale surveys are displayed in Table 1. We first checked the reliability of each pair of related questions with Cronbach's α and found that the participants responded to every question pairing consistently ($\alpha > 0.9$). Hence, we grouped each pair of responses into a single averaged score and performed one-way repeated measures ANOVAs, where the robot's response strategy was the independent variable. The interaction strategy had a significant effect on how participants perceived the robot's learning ($F(2, 22) = 33.9, p < .01$), predictability ($F(2, 22) = 6.0, p < .01$), response speed ($F(2, 22) = 24.1, p < .01$), and interaction effort ($F(2, 22) = 27.9, p < .01$). The results of our post hoc analysis are listed in Table 1.

7.6 Discussion

Summarizing results. Our proposed *LQR+L* approach led to better objective performance during the tasks with global or local trajectory errors. Users also perceived the *LQR+L* robot as more predictable and better at learning the robot's preferred trajectory. During the task where the human

Table 1. Questions and Responses from the 7-Point Likert Scale User Survey

Questionnaire Item	Reliability	*LQR+L*	*Share, Deform*	Post Hoc
– By the end of the task, I was confident the robot had *learned* my trajectory.	.99	6.0 ± 1.2	3.0 ± 2.0	$p < .001^*$
– The robot *did not learn* my preferred trajectory.			2.0 ± 1.1	$p < .001^*$
– After each correction, I could *anticipate* how the robot would behave.	.92	5.8 ± 1.4	4.0 ± 1.9	$p = .003^*$
– The robot responded *unpredictably* to my corrections.			3.7 ± 1.8	$p = .003^*$
– The robot responded *quickly* to my corrections.	.96	6.0 ± .84	5.4 ± 1.6	$p = .199$
– It took *a while* for the robot to change its trajectory after each correction.			2.9 ± 1.5	$p < .001^*$
– Correcting the robot's trajectory required *little effort.*	.94	5.1 ± 1.9	4.9 ± 1.8	$p = .743$
– I had to apply *a lot of force* to correct the robot's trajectory.			1.4 ± .51	$p < .001^*$

Note: We grouped questions into four topics and tested their reliability using Cronbach's alpha. Questions focused on whether the robot learned, was predictable, responded quickly, and was easy to correct. *LQR+L*, *Share*, and *Deform* give the $mean \pm std$ response across users, where higher values indicate agreement (users answered with an integer $1-7$). Post hoc shows the pairwise comparisons between *LQR+L* and *Share* (top) and*LQR+L* and *Deform* (bottom), where an asterisk denotes significance.

wanted to change the movement timing, *LQR+L* resulted in less total cost and interaction time than the local modification approach and improved movement smoothness as compared to the shared control method; however, the other pairwise comparisons were not significant. Participants believed that the *LQR+L* robot required less effort to correct and responded more quickly than the *Deform* robot, but for these metrics *LQR+L* was not significantly different from *Share*.

Highlighting strengths. This user study was meant to address the challenges from Section 5.4 within a *realistic* setting. We gave the *LQR+L* robot an over-defined hypothesis space, did not provide users with the trajectory error, and measured noisy and imperfect human corrections. Despite these challenges, our proposed approach learned the right robot trajectory for local and global mistakes such that if the robot were to perform the same task again, the human would no longer need to interact and correct the robot's behavior (see Figure 7, left and middle).

Identifying shortcomings. Not all of our results were positive. Participants reported that it was difficult to alter the robot's timing with *LQR+L*. For example, "changing the speed of the robot was difficult, but changing its trajectory was much easier." We suggest that *Task 3* was particularly difficult for the *LQR+L* method because the user was attempting to make the robot track a specific sinusoidal trajectory—without any phase shift—rather than just a preferred frequency. Accordingly, participants had to iteratively speed up and slow down the robot so that it had both the right frequency and phase. The ability to adjust the robot's timing online is one of the unique aspects of our approach as compared to prior works [4, 19, 24, 27]; although it was not effective here because of the phase constraint, we are excited about this capability moving forward.

Revisiting the state of the art. Although *LQR+L* performed well within our application scenario, other methods may be preferable in different contexts. For example, if the task requires significant

interaction with the environment—such as sawing—collaborative approaches are more appropriate [14, 35, 38]. Local modification approaches become increasingly relevant if the task involves multiple repetitions: the robot can store and replay the local adjustments between iterations, enabling the end user to intuitively and iteratively correct the robot's trajectory [1, 9, 11, 13, 15, 24, 27]. For scenarios where the task has few features (i.e., two to three), global replanning methods have already been implemented online [4, 5]: thus, in practice, the time spent replanning may not be significant enough to warrant our alternate approach. Finally, as we demonstrated within the worst-case simulations, shared and optimal controllers that do not learn the correct trajectory are often sufficient when the hypothesis space is largely incorrect [12, 20, 25, 30, 31]. Our experimental results should therefore be interpreted as useful trends—demonstrating that *LQR+L* is beneficial in the right contexts—rather than claiming that this method is better for all cases.

8 CONCLUSION

We presented an extension of LQR control that enables autonomous robots to learn the right trajectory from physical human corrections in one shot, without replanning, offline demonstrations, or multiple iterations. Based on our insight—where human corrections are observations about the error from their preferred trajectory—we leveraged gradient descent to derive a computationally efficient learning rule for real-time implementation and proved that augmenting LQR control with this learning rule yields stable pHRI. Our resulting algorithm generalizes across both local and global trajectory updates: if the designer specifies a low-level hypothesis space (like waypoints), *LQR+L* locally modifies the trajectory, but if the designer identifies high-level parameters (like features), *LQR+L* modifies the entire future trajectory after each interaction.

We identified several practical challenges that often arise when learning from pHRI and explored how robust our approach is to each of these limitations within simulations and a user study. In the best case, *LQR+L* obtained objective performance on par with an offline optimal response but did so in real time, at the same loop rate as a local modification approach. In the worst case, *LQR+L* was robust to errors in the observation model; however, for large errors in the hypothesis space, a non-learning baseline outperformed the proposed learning approach. Our user study considered realistic scenarios where the robot had an incorrect hypothesis space, the participants could not directly observe the trajectory error, and the human corrections were noisy and imperfect. We found that *LQR+L* outperformed two real-time baselines (shared control and local modifications) in tasks with local or global trajectory errors: the *LQR+L* robot had less total cost and interaction time, as well as increased movement smoothness. Although these simulations and the user study support our approach, we also recognized that prior methods may be better suited for different application scenarios, such as cases where there are significant interactions with the environment, the task is iterative or captured with only a few parameters, or the designer is very unsure of the correct trajectory parameterization.

Overall, this work is only a first step toward learning from physical interactions and selecting optimal robot responses in real time for one-shot settings where the robot should complete the rest of its current task correctly. Our user study explores applications of the proposed method for robotic rehabilitation, where the therapist can interact with the robot to update its training task. The proposed method can also be applied to household robotics settings, where the end user wants to adapt the robot's underlying behavior to suit his or her preferences and environment.

ACKNOWLEDGMENTS

The authors would like to thank Evan Pezent and Craig McDonald for their help in interfacing with the OpenWrist and performing our user study. We also thank the reviewers for their thoughtful comments and valuable feedback.

REFERENCES

[1] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. 2012. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI'12)*. 391–398.

[2] Brian D. O. Anderson and John B. Moore. 2007. *Optimal Control: Linear Quadratic Methods*. Dover Publications, New York, NY.

[3] Karl Johan Åström and Björn Wittenmark. 2008. *Adaptive Control* (2nd ed.). Dover Publications, Mineola, NY.

[4] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. 2017. Learning robot objectives from physical human interaction. In *Proceedings of the Conference on Robot Learning (CoRL'17)*. 217–226.

[5] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. 2018. Learning from physical human corrections, one feature at a time. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI'18)*. 141–149.

[6] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, and Etienne Burdet. 2012. A robust and sensitive metric for quantifying movement smoothness. *IEEE Transactions on Biomedical Engineering* 59, 8 (2012), 2126–2136.

[7] Léon Bottou. 1998. Online learning and stochastic approximations. In *On-Line Learning in Neural Networks*, Vol. 17. Cambridge University Press, New York, NY, 9–42.

[8] Oliver Brock and Oussama Khatib. 2002. Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research* 21, 12 (2002), 1031–1052.

[9] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. 2016. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing* 40 (2016), 1–13.

[10] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi. 2008. An atlas of physical human–robot interaction. *Mechanism and Machine Theory* 43, 3 (2008), 253–270.

[11] Mustafa Suphi Erden and Tetsuo Tomiyama. 2010. Human-intent detection and physically interactive control of a robot without force sensors. *IEEE Transactions on Robotics* 26, 2 (2010), 370–382.

[12] Paul Evrard and Abderrahmane Kheddar. 2009. Homotopy switching model for dyad haptic interaction in physical collaborative tasks. In *Proceedings of the EuroHaptics Conference*. 45–50.

[13] Andrej Gams, Tadej Petrič, Martin Do, Bojan Nemec, Jun Morimoto, Tamim Asfour, and Aleš Ude. 2016. Adaptation and coaching of periodic motion primitives through physical and visual interaction. *Robotics and Autonomous Systems* 75 (2016), 340–351.

[14] Elena Gribovskaya, Abderrahmane Kheddar, and Aude Billard. 2011. Motion learning and adaptive impedance for robot control during physical interaction with humans. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'11)*. 4326–4332.

[15] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger. 2008. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*. 3356–3363.

[16] Sami Haddadin and Elizabeth Croft. 2016. Physical human-robot interaction. In *Springer Handbook of Robotics* (2nd ed.). Springer, 1835–1874.

[17] Neville Hogan. 1985. Impedance control: An approach to manipulation. *Journal of Dynamic Systems, Measurement, and Control* 107, 1 (1985), 1–24.

[18] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. 2013. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation* 25, 2 (2013), 328–373.

[19] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. 2015. Learning preferences for manipulation tasks from online coactive feedback. *International Journal of Robotics Research* 34, 10 (2015), 1296–1313.

[20] Nathanaël Jarrassé, Themistoklis Charalambous, and Etienne Burdet. 2012. A framework to describe, analyze and generate interactive motor behaviors. *PLoS One* 7, 11 (2012), e49945.

[21] Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30, 7 (2011), 846–894.

[22] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. 2013. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems* 61, 12 (2013), 1726–1743.

[23] Przemyslaw A. Lasota and Julie A. Shah. 2015. Analyzing the effects of human-aware motion planning on close-proximity human-robot collaboration. *Human Factors* 57, 1 (2015), 21–33.

[24] Yanan Li, Gowrishankar Ganesh, Nathanaël Jarrassé, Sami Haddadin, Alin Albu-Schaeffer, and Etienne Burdet. 2018. Force, impedance, and trajectory learning for contact tooling and haptic identification. *IEEE Transactions on Robotics* 34, 5 (2018), 1170–1182.

[25] Yanan Li, Keng Peng Tee, Rui Yan, Wei Liang Chan, and Yan Wu. 2016. A framework of human–robot coordination based on game theory and policy iteration. *IEEE Transactions on Robotics* 32, 6 (2016), 1408–1418.

[26] Dylan P. Losey, Craig G. McDonald, Edoardo Battaglia, and Marcia K. O'Malley. 2018. A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction. *Applied Mechanics Reviews* 70, 1 (2018), 010804.

[27] Dylan P. Losey and Marcia K. O'Malley. 2018. Trajectory deformations from physical human–robot interaction. *IEEE Transactions on Robotics* 34, 1 (2018), 126–138.

[28] Jim Mainprice and Dmitry Berenson. 2013. Human-robot collaborative manipulation planning using early prediction of human motion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)*. 299–306.

[29] Carlo Masone, Paolo Robuffo Giordano, Heinrich H. Bülthoff, and Antonio Franchi. 2014. Semi-autonomous trajectory generation for mobile robots with integral haptic shared control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'14)*. 6468–6475.

[30] José Ramón Medina, Tamara Lorenz, and Sandra Hirche. 2015. Synthesizing anticipatory haptic assistance considering human behavior uncertainty. *IEEE Transactions on Robotics* 31, 1 (2015), 180–190.

[31] Alexander Mörtl, Martin Lawitzky, Ayse Kucukyilmaz, Metin Sezgin, Cagatay Basdogan, and Sandra Hirche. 2012. The role of roles: Physical cooperation between humans and robots. *International Journal of Robotics Research* 31, 13 (2012), 1656–1674.

[32] Thomas Nierhoff, Sandra Hirche, and Yoshihiko Nakamura. 2016. Spatial adaption of robot trajectories based on Laplacian trajectory editing. *Autonomous Robots* 40, 1 (2016), 159–173.

[33] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. 2009. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'09)*. 763–768.

[34] Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. 2011. Skill learning and task outcome prediction for manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'11)*. 3828–3834.

[35] Luka Peternel, Tadej Petrič, Erhan Oztop, and Jan Babič. 2014. Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach. *Autonomous Robots* 36, 1–2 (2014), 123–136.

[36] Evan Pezent, Chad G. Rose, Ashish D. Deshpande, and Marcia K. O'Malley. 2017. Design and characterization of the OpenWrist: A robotic wrist exoskeleton for coordinated hand-wrist rehabilitation. In *Proceedings of the IEEE International Conference on Rehabilitation Robotics*. 720–725.

[37] Quang-Cuong Pham and Yoshihiko Nakamura. 2015. A new trajectory deformation algorithm based on affine transformations. *IEEE Transactions on Robotics* 31, 4 (2015), 1054–1063.

[38] Leonel Rozo, Sylvain Calinon, Darwin G. Caldwell, Pablo Jimenez, and Carme Torras. 2016. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics* 32, 3 (2016), 513–527.

[39] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. 2014. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research* 33, 9 (2014), 1251–1270.

[40] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. 2006. *Robot Modeling and Control*. Vol. 3. John Wiley & Sons, New York, NY.

[41] Robert F. Stengel. 1994. *Optimal Control and Estimation*. Dover Publications, New York, NY.

Received November 2018; revised May 2019; accepted July 2019