GRILC: Gradient-based Reprogrammable Iterative Learning Control for Autonomous Systems

Kuan-Yu Tseng[1], **Jeff S. Shamma**[2], **and Geir E. Dullerud**[1]
[1]Mechanical Science and Engineering, [2]Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
{kuanyut2, jshamma, dullerud}@illinois.edu

Abstract

We propose a novel gradient-based reprogrammable iterative learning control (GRILC) framework for autonomous systems. Performance of trajectory following in autonomous systems is often limited by mismatch between a complex actual model and a simplifed nominal model used in controller design. To overcome this issue, we develop the GRILC framework with offline optimization using the information of the nominal model and the measured actual trajectory, and online system implementation. In addition, a partial and reprogrammable learning strategy is introduced. The proposed method is applied to the autonomous time-trialing example and the learned control policies can be stored into a library for future motion planning. The simulation results illustrate the effectiveness and robustness of the proposed approach.

1 Introduction

Over the last decades control and trajectory following of autonomous systems has been an active field of research. Typically, control systems regulating the behaviors of autonomous systems are based on mathematical models representing the system dynamics. However, dynamics of actual systems are usually complicated since they contain nonlinear drags, damping, and deformation that can be difficult to capture accurately in a mathematical model. The performance of the control systems can therefore be hindered by this mismatch between the mathematical model and actual system. In this work, we are interested in addressing this issue by introducing a new iterative learning control architecture, called *gradient-based reprogrammable iterative learning control* (GRILC), that updates control policies to achieve high performance under the presence of model errors and unmodeled disturbances. We implement the control framework on a trajectory following control task: autonomous time trialing, where repeated trials are permitted during the learning phase. At each iteration a control policy drives the system along a track trajectory aiming to minimize the elapsed time. The trajectory data of each iteration is stored and used to update the control policy in the next iteration.

Various learning-based approaches have been worked on involving control and trajectory following of autonomous systems. In [1, 2] a learning-based nonlinear Model Predictive Control (MPC) was presented and implemented on ground vehicles. The vehicle model comprises a simple nominal model and a learned disturbance model to improve path-tracking problem. In [3], the authors described a learning-based MPC approach and its real-time implementation on a quadcopter. The MPC formulation contained a nominal model to ensure safety, and a learned model to improve tracking performance. In [4], the authors proposed a learning model predictive controller for autonomous racing. The system and input trajectories of each lap are stored and used to update the controller for the next lap, and a system identification technique is utilized to update the system model. In [5] the authors proposed an iterative learning control (ILC) approach to achieve accurate steering control on

35th Conference on Neural Information Processing Systems (NeurIPS 2021), virtual.

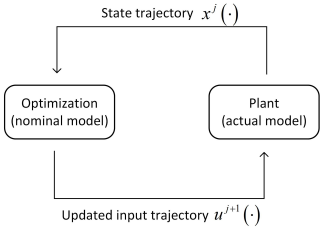

Figure 1: The architecture of the GRILC.

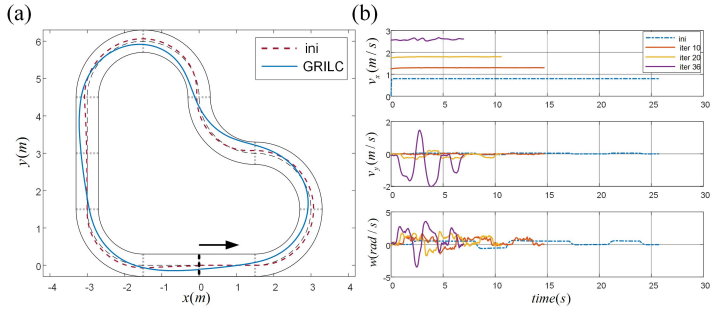

Figure 2: Learning performance of the GRILC

a full-size autonomous race car. The learning approach introduced in this paper is inspired from the work [6]. The authors in [6] proposed an iterative learning scheme to enhance tracking performance of a predefined path. The approach used in the work [7] is most relevant to our proposed method. In [7], an ILC approach is utilized to minimize the navigation time of a path without seeking to learn any reference trajectory.

In this paper, we significantly extend the idea in [7] to deal with repetitive tasks in which the initial condition of each iteration does not need to be the same. The learning control architecture is shown in Figure 1. Fed with some measured actual system trajectory, the optimization objective is to minimize some cost based on the information of actual trajectory and the simplified nominal model. After the optimization, the updated control policy (i.e., input trajectory) is implemented on the actual system. The actual realized trajectory may differ from the predicted one due to the mismatch between the nominal and actual systems. The realized trajectory is then fed into the optimization to update the input trajectory repeatedly. In addition, we introduce a partial learning strategy by partitioning the trajectory into finite segments and updating parts of the trajectory at a time. A termination condition is defined that the car will stop running and reprogram the optimization of the following segments. The concept of partial learning was discussed in [8]. The authors proposed a task decomposition method for MPC and described subtask transitions by solving controllability problems. We implement the proposed method on the autonomous time-trialing example with a closed track. The optimization objective is to minimize the trialing time of reaching a target that has a finite distance ahead of the vehicle. The proposed approach shows the decreasing lap time over iterations even with a modelling mismatch between the nominal and actual systems. The partial learning framework is also capable of storing control policies during learning and building a trajectory library. The library can be used to plan motions of the vehicle in real time. The general idea of motion planning with a library of control signals is well-established conceptually and has been used previously in different applications based on other design techniques (e.g., [9, 10]).

2 Gradient-based iterative learning control

Iterative learning in general form: We first describe the GRILC for dynamical systems in a general form. Suppose a nonlinear discrete-time function $x(t+1) = f_R(x(t), u(t))$, where $f_R(\cdot,\cdot)$ describes the actual dynamics of the system, $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ denote the system state and input, respectively. The goal of the controller is to solve the following optimization problem whose objective is to minimize some cost,

$$J = \min_{u(\cdot)} L\left(x(\cdot), u(\cdot)\right) \tag{1a}$$

$$\text{s.t.} \quad x(t+1) = f_R(x(t), u(t)) \quad \forall t = \{0, \ldots, N-1\} \tag{1b}$$

$$x_0 = x_S \tag{1c}$$

$$x(t) \in \mathcal{X}, u(t) \in \mathcal{U} \quad \forall t = \{0, \ldots, N\} \tag{1d}$$

subject to the dynamics f_R, initial conditions and constraints. L is some cost function. We assume the actual function f_R is *unknown* in the optimization but the state can be measured. Instead, we know an abstract (simplified) model of the system given by $f_A(\cdot,\cdot)$ which can capture the key dynamics of the actual system. This abstract model f_A will be used as the nominal model in the optimization. The

goal of the learning approach is to minimize the cost L using the knowledge of f_A and based on some predetermined feasible trajectory. We derive a linearized system representation which facilitates the implementation of the GRILC. Details of derivation can be found in [6]. The optimization problem (1) can be transformed to find the deviation of the input trajectory $\hat{U}$ that can minimize the cost

$$\min_{\hat{U}} L(X^j + M\hat{U} + M_0\hat{x}(0)) \tag{2}$$

subject to (1c) and (1d).

Partial and reprogrammable learning strategy: In this work, we introduce a partial and reprogrammable learning strategy based on the GRILC. Instead of optimizing the whole trajectory, the GRILC optimizes parts of the trajectory at a time. The proposed learning strategy can deal with more realistic situations such as only parts of the information of the trajectory are known. Formally, the trajectory is partitioned into finite segments (X_i^j, U_i^j) where $i = \{0, \dots, n-1\}$. Cost functions of segments are associated with the types of the segments $L(\,\cdot\,; \theta_i)$, where θ_i denotes the type of the segment i. The partial optimization problem from the start of segment i is represented as

$$\min_{\hat{U}_{i:i+k}} L(X^j_{i:i+k} + M\hat{U}_{i:i+k} + M_0\hat{x}(0); \theta_{i:i+k}) \tag{3}$$

where $k = \{0, \dots, n-1\}$. If the system works on a closed trajectory, then $X_{i+n} = X_i, U_{i+n} = U_i$, and $\theta_{i+n} = \theta_i$. Note that we can optimize multiple consecutive segments in the optimization. After the optimization, the updated input trajectory is applied to the autonomous system. A terminal condition $x \in \mathcal{X}_{\text{ter}}$ is also defined to stop the system from running and reprogram to optimize the control trajectory from the start of the segment $i + p$, where $p = \{0, \dots, k+1\}$. The generic algorithm of the GRILC is summarized as follows:

(a) Given some feasible trajectory $\{(X_i^j, U_i^j)\}_{i=0}^{n-1}$. Start from $i = 0$. Linearize the abstract model f_A about the current trajectory $(X^j_{i:i+k}, U^j_{i:i+k})$ to build the lifted system representation and matrices M and M_0 in equation (3).

(b) Solve the optimization in (3). Update the control input for the next iteration $U^{j+1}_{i:i+k} = U^j_{i:i+k} + \hat{U}$.

(c) Apply $U^{j+1}_{i:i+k}$ to the actual system and measure the state trajectory $X^{j+1}_{i:i+k}$. Stop the system when it triggers the terminal condition (i.e., $x \in \mathcal{X}_{\text{ter}}$).

(d) If $X^{j+1}_{i:i+k}$ violates the constraint, go back to step (b) with a tighter limit of $\hat{U}_{\max}$. Otherwise, set $i = i + p$ and go back to step (a).

(e) After the whole trajectory is finished (i.e., $i \geq n$), check whether the performance of the state trajectory X^{j+1} improves. If it does, then optimize the trajectory in the next iteration (i.e., $i = i - n$, $j = j + 1$). The learning terminates if no improvement can be achieved or the state trajectory keeps violating the constraints under tighter $\hat{U}_{\max}$.

Motivation and discussion: This work is developed from [7] because the iterative learning framework in [7] combining the nominal model and measured actual trajectory represents two major advantages. First, the actual trajectory is updated using the gradient information of the nominal model. The nominal model which is a poor predictor of the measured actual trajectory can still be useful in the optimization if the model captures the key features of the actual system. In this way, we can design a simplified model in the optimization to achieve high learning performance while decreasing computational burden. Second, the learning framework optimizes the system trajectory which is a neighbor of the previous feasible trajectory. This can ensure recursive feasibility that the updated trajectory does not deviate far from the previous one. Our work differs from [7] in two aspects. First, we extend the learning approach to accommodate nonfixed initial conditions. This allows the approach to be applied to a broad range of repetitive tasks, for example, updating a closed trajectory that the final condition of the iteration is the initial condition of the next iteration. Second, we propose the partial and reprogrammable learning strategy based on the GRILC. The strategy generalizes the setup of the GRILC that it allows the trajectory to be split into finite segments and updates several segments at a time. The system can then run on the patched updated trajectory and continues the learning process in a new iteration. We note that [4] and this paper both propose a learning-based

controller and present an application of the autonomous time-trialing example. However, the purposes of two works are mainly different. [4] focused on designing a learning MPC to minimize lap time and operate the vehicle at the limit of handling by storing trajectories of previous laps to update the cost-to-go and controller. The key idea of this work is to design an iterative learning controller which can deal with mismatch between the nominal and actual models by blending the low-fidelity gradient of the nominal model and the measured actual trajectory. The comparison of performance between two approaches has to be further analyzed and yet to be determined.

3 Application: autonomous time trialing

Vehicle model: The vehicle model is described as $x(t+1) = f(x(t), u(t))$, where $f(\cdot, \cdot)$ is the dynamic bicycle model. For more details of the actual model f_R and abstract model f_A dynamics, we refer to [12].

Minimum time problem: We deal with the autonomous time-trialing example using the GRILC described in Section 2. Specifically, we formulate the problem as a minimum time iterative control task. The learning control scheme uses the previous trajectory data to update the control policy to the car. The goal of the controller is to minimize the time to cross the target line at s_{target}. More formally, the controller is to solve the following constrained optimization problem,

$$J = \min_{\hat{U}} (U^j + \hat{U})^\top \Lambda_1 (U^j + \hat{U}) + \hat{U}^\top K_1 \hat{U} + 2K_2^\top \hat{U} \tag{4a}$$

$$\text{s.t.} \quad SX^j + SM\hat{U} + SM_0\hat{x}(0) \leq C \tag{4b}$$

where in (4a), $K_1 = M^\top \Lambda_2 M$ and $K_2 = M^\top \Lambda_2 (X^j + M_0\hat{x}(0) - X_d)$. Λ_1, Λ_2 are diagonal weight matrices concatenated by V_1 and V_2, respectively. X_d is concatenated vector of terminal state $x_d = [s_{\text{target}}, 0, 0, 0, 0, 0]$. In (4b) S is the selection matrix and C is the vector representing the boundary of the track.

Partial learning strategy and assembled track: We apply the partial learning strategy to the application corresponding to a more realistic situation: the vehicle has a view of finite horizon. That is, when the vehicle tries to plan the trajectory of the next iteration, it can only plan the vehicle heading to the desired point which has a finite distance ahead of its current position. To accommodate this scenario, we divide the racing track to n segments and consider three types of segments: go straight, turn left, and turn right with fixed dimension. The cost functions of three types are the same. We assume the vehicle has a view of two segments ahead of it (i.e., $k = 1$ in (3)). That is, suppose the vehicle starts from a starting point of a segment i. Its goal is to optimize a control trajectory that leads it to reach the final line of the second segment in front of it (i.e., $s_{\text{target}} = s_{\text{final},i+1}$). The terminal condition is defined as $\mathcal{X}_{\text{ter}} = \{s \geq s_{\text{final},i}\}$. The car will stop running when it reaches the final line of the first segment and starts to optimize the trajectory of the following two segments (i.e., $p = 1$). The idea of target lines lying ahead of terminal lines is that the vehicle can foresee the information of the following segment and plan the corresponding control trajectory. The learning procedure follows the algorithm described in Section 2. After the trajectory of a lap is finished, the lap time is computed by $N^{j+1} = \sum_{i=0}^{n-1} N_i^{j+1}$. The learning will terminate if no time improvement can be achieved.

Online running and library of trajectory: In the learning phase, the vehicle runs on the track and iteratively updates its control trajectories of the segments. Those trajectories can be stored into a library which can be used to generate motions of the car in real time. The concept of motion planning using a finite set of control signals was introduced as Motion Description Language (MDL) in the works [9,10]. From the learning phase we can build a three dimensional library containing trajectories with different types of segment pairs, initial longitudinal velocity v_x, and initial lateral error e_y.

4 Simulation results

In this section we present the learning performance using the GRILC approach. The quadratic optimization problem (4) is solved using the function `quadprog` built in MATLAB. The vehicle simulator is built in Python and using Robot Operating System (ROS) to connect to the optimization part and low-level controller. The discretization time $\Delta t = 0.02$s is used in the simulation. The model

parameters of the actual and abstract models are set to $m = 4.55\text{kg}$, $I_z = 0.0578\text{kg} \cdot \text{m}^2$, $L_f = L_r = 0.15\text{m}$. The tire coefficients of two models are also set to be the same. The track boundary is set as the constant $|e_y| \leq 0.3\text{m}$. The weights to penalize state are set to be $V_2 = \text{diag}\{1, 10, 10, 0, 0, 0\}$. The weights indicate that we not only want the car to reach the target point faster, but also restrain the lateral distance error e_y and heading error e_ψ with respect to the track. This is because mismatch exists between the actual model f_R and the abstract model f_A. If we did not penalize the state e_y and e_ψ, the car would travel out of the track easily in high velocity. We did not penalize the input state. The initial trajectory of the car is constructed using PID control.

The initial trajectory and the updated trajectory obtained from the GRILC are shown in Figure 2(a). The initial trajectory stayed close to the center of the track, while the learned trajectory tended to cross the corner and was close to the boundary in some segments. Figure 2(b) shows the evolution of the velocity trajectory over iterations. Initially the longitudinal velocity v_x was set to 0.8m/s. The lateral velocity v_y and yaw rate w were small. During the learning, v_x gradually increased, and v_y and w presented more aggressive trajectories over the iterations. The learning terminated at the iteration 36, when the car trajectory continued to hit the track boundary at the next iteration. The navigation time achieved 6.98s at the last iteration, which is 72.9% lower than the initial time 25.72s.

5 Conclusion

In this paper we present a novel gradient-based reprogrammable iterative learning control approach. The derivation of the general GRILC is presented. Moreover, we introduce a partial learning strategy which is more corresponding to real-world applications and allows the system to build a library of trajectories for online running. The GRILC is applied to the time-trialing example. We tested the GRILC in simulation, and showed that it is able to improve the vehicle's performance.

References

[1] Ostafew, C. J., Schoellig, A. P., Barfoot, T. D., & Collier, J. (2016). Learning-based Nonlinear Model Predictive Control to Improve Vision-based Mobile Robot Path Tracking. *Journal of Field Robotics*, 33(1), 133–152. https://doi.org/10.1002/rob.21587

[2] Ostafew, C. J., Schoellig, A. P., & Barfoot, T. D. (2014). Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 4029-4036

[3] Aswani, A., Bouffard, P., & Tomlin, C. (2012). Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. *Proceedings of the American Control Conference (ACC)*, 4661-4666. https://doi.org/10.1109/ACC.2012.6315483

[4] Rosolia, U., & Borrelli, F. (2020). Learning How to Autonomously Race a Car: A Predictive Control Approach. *IEEE Transactions on Control Systems Technology*, 28(6), 2713-2719.

[5] Kapania, N. R., & Gerdes, J. C. (2015). Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control. *Proceedings of the American Control Conference (ACC)*, 2753–2758. https://doi.org/10.1109/ACC.2015.7171151

[6] Schoellig, A. P., Mueller, F. L., & D'Andrea, R. (2012). Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots*, 33(1–2), 103–127. https://doi.org/10.1007/s10514-012-9283-2

[7] Shaqura, M., & Shamma, J. S. (2018). An iterative learning approach for motion control and performance enhancement of quadcopter UAVs. *Proceedings of the International Conference on Control, Automation and Systems (ICCAS)*, 285–290.

[8] Vallon, C., & Borrelli, F. (2020). Task Decomposition for Iterative Learning Model Predictive Control. *Proceedings of the American Control Conference (ACC)*, 2024–2029. https://doi.org/10.23919/ACC45564.2020.9147625

[9] Manikonda, V., Krishnaprasad, P. S., & Hendler, J. (1999). Languages, behaviors, hybrid architectures, and motion control. *Mathematical control theory*, 199-226. Springer, New York, NY.

[10] Egerstedt, M. (2003). Motion description languages for multi-modal control in robotics. *Control Problems in Robotics*, 75-89. Springer, Berlin, Heidelberg.

[11] Bamieh, B., Pearson, J. B., Francis, B. A., & Tannenbaum, A. (1991). A lifting technique for linear periodic systems with applications to sampled-data control. *Systems & Control Letters*, 17(2), 79–88.

[12] Rajamani, R. (2011). *Vehicle Dynamics and Control*. Springer Science & Business Media. https://doi.org/10.1007/978-1-4614-1433-9