Non-Parametric Neuro-Adaptive Control Subject to Task Specifications

Christos K. Verginis
University of Texas at Austin
cverginis@utexas.edu

Zhe Xu
Arizona State University
xzhe1@asu.edu

Ufuk Topcu
University of Texas at Austin
utopcu@utexas.edu

Abstract: We develop a learning-based algorithm for the control of robotic systems governed by unknown, nonlinear dynamics to satisfy tasks expressed as signal temporal logic specifications. Most existing algorithms either assume certain parametric forms for the dynamic terms or resort to unnecessarily large control inputs (e.g., using reciprocal functions) in order to provide theoretical guarantees. The proposed algorithm avoids the aforementioned drawbacks by innovatively integrating neural network-based learning with adaptive control. More specifically, the algorithm learns a controller, represented as a neural network, using training data that correspond to a collection of different tasks and robot parameters. It then incorporates this neural network into an online closed-loop adaptive control mechanism in such a way that the resulting behavior satisfies a user-defined task. The proposed algorithm does not use any information on the unknown dynamic terms or any approximation schemes. We provide formal theoretical guarantees on the satisfaction of the task and we demonstrate the effectiveness of the algorithm in a virtual simulator using a 6-DOF robotic manipulator.

Keywords: Adaptive control, Deep learning, Trajectory tracking, Task specifications, Temporal logic

1 Introduction

Learning and control of robotic systems with uncertain dynamics is a critical and challenging topic that has been widely studied during the last decades. One can identify plenty of motivating reasons, ranging from uncertain geometrical or dynamical parameters and unknown exogenous disturbances to abrupt faults that significantly modify the dynamics. There has been, therefore, an increasing need for developing control algorithms that do not rely on the underlying robot dynamics. At the same time, such algorithms can be easily implemented on different, heterogeneous robots, since one does not need to be occupied with the tedious computation of the corresponding dynamic terms.

There has been a large variety of works that tackle the problem of control of robotic systems with uncertain dynamics, exhibiting, however, certain limitations. Most existing algorithms are based on adaptive and learning-based approaches or the so-called funnel control [1, 2, 3, 4, 5, 6]. Nevertheless, adaptive control methodologies are restricted to system dynamics that can be linearly parameterized with respect to certain unknown parameters (e.g., masses, moments of inertia), assuming the system structure perfectly known; funnel controllers employ reciprocal terms that drive the control input to infinity when the tracking error approaches a pre-specified funnel function, creating thus unnecessarily large control inputs that might damage the system actuators. Data-based learning approaches either consider some system characteristic known (e.g., a nominal model, Lipschitz constants, or global bounds), or use neural networks to learn a tracking controller or the system dynamics; the correctness of such methods, however, relies on strong assumptions on the parametric approximation by the neural network and knowledge of the underlying radial basis functions.

1.1 Contributions and Significance

This paper addresses the control of robotic systems with continuous, *unknown* nonlinear dynamics subject to task specifications expressed as signal interval temporal logic (SITL) constraints [7]. Our main contribution lies in the development of a learning-based control algorithm that guarantees the accomplishment of a given task using only mild assumptions on the system dynamics. The algorithm draws a novel connection between adaptive control and learning with neural network representations, and consists of the following steps. Firstly, it trains a neural network that aims to learn a robot controller that accomplishes a given task. Secondly, it calculates an open-loop trajectory that yields the execution of the task if followed by the system, while neglecting the dynamics. Finally, we develop an online adaptive feedback controller that uses the trained network to guarantee convergence to the open-loop trajectory and hence satisfaction of the task. In contrast to the majority of works in the related literature (e.g., [5, 8, 9]), the proposed algorithm does not use *any* information on the robot's dynamic terms or any related approximation schemes. In fact, although it does use a neural network to approximate a controller, its correctness does not rely on sufficiently small approximation errors, as opposed to previous works (e.g., [2, 10]). Instead, it requires only a mild growth condition on the closed-loop system that is driven by the learned controller.

The major significance of our contribution is twofold. Firstly, we guarantee the theoretical correctness of the proposed algorithm by considering only mild conditions on the neural network, removing the long-standing assumptions on parametric approximations and boundedness of the estimation error. Secondly, we demonstrate via the experimental results the generality of the algorithm with respect to different tasks and robot parameters. That is, the training data that we generate for the training of the neural network in the first step correspond to tasks that are different, in terms of spatiotemporal specifications, from the given one to be executed[1]. Additionally, we employ robots with different dynamic parameters to generate these data. We evaluate the proposed algorithm in numerous scenarios comprising different variations of the given task and different robot dynamic parameters, which do not necessarily match the training data. We show that the algorithm, owing to its adaptation properties, is able to guarantee the satisfaction of the respective tasks in all the aforementioned scenarios by using the same neural network.

1.2 Related work

A large variety of previous works considers neural-network-based adaptive control (neuro-adaptive control) with stability guarantees, focusing on the optimal control problem [2, 11, 12, 10, 13, 14, 15, 16, 17, 18, 19, 5]. Nevertheless, the related works draw motivation from the neural network density property (see, e.g., [20])[2] and assume sufficiently small approximation errors and linear parameterizations of the unknown terms (dynamics, optimal controllers, or value functions), which is also the case with standard adaptive control methodologies [1, 21, 22, 8]. This paper relaxes the aforementioned assumptions and proposes a *non-parametric* neuro-adaptive controller, whose stability guarantees rely on a mild boundedness condition of the closed-loop robot state that is driven by the learned controller.

Other learning-based related works include modeling with Gaussian processes [6, 23, 24, 25], or use neural networks [26, 27, 28, 29, 30, 31, 32, 33, 30, 34, 35, 36] to accomplish reachability, verification or temporal logic specifications. Nevertheless, the aforementioned works either use partial information on the underlying robot dynamics, or do not consider them at all. In addition, works based on Gaussian processes usually propagate the dynamic uncertainties, possibly leading to conservative results. Similarly, data-driven model-predictive control techniques [9, 37] use data to over-approximate unknown additive disturbance terms or are restricted to linear systems.

Control of unknown nonlinear systems has been also tackled in the literature by using funnel control, without necessarily using off-line data or dynamic approximations [3, 4, 38, 39, 40]. Nevertheless, funnel controllers usually depend on reciprocal time-varying barrier functions that drive the control input to infinity when the error approaches a pre-specified funnel function, creating thus unnecessarily large control inputs that might damage the system actuators.

[1] The task difference is illustrated in Section 4.

[2] A sufficiently large neural network can approximate a continuous function arbitrarily well in a compact set.

2 Preliminaries and Problem Formulation

2.1 Notation

The set of non-negative reals is denoted by $\mathbb{R}_{\geq 0}$. For a signal $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, $x^{(k)}$ denotes its kth derivative, for an integer $k \geq 0$, with $x^{(0)} \coloneqq x$. The 2-norm of a vector $x \in \mathbb{R}^n$ is denoted by $\|x\|$. Given a function $\mathsf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, with $\mathsf{f}(x, y) \coloneqq [\mathsf{f}_1(x, y), \ldots, \mathsf{f}_n(x, y)]^\top$, we use

$$\mathsf{D}\mathsf{f}^x(x, y) \coloneqq \begin{bmatrix} \frac{\partial \mathsf{f}_1(x,y)}{\partial x_1} & \cdots & \frac{\partial \mathsf{f}_1(x,y)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathsf{f}_n(x,y)}{\partial x_1} & \cdots & \frac{\partial \mathsf{f}_n(x,y)}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

2.2 Signal interval temporal logics

Let $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ be a continuous-time signal. Signal interval temporal logic (SITL) consists of predicates μ that are obtained after evaluation of a continuously differentiable predicate function $h : \mathbb{R}^n \to \mathbb{R}$ [7]. For $\zeta \in \mathbb{R}^n$, let $\mu \coloneqq \top$ if $h(\zeta) \geq 0$ and $\mu \coloneqq \bot$ if $h(\zeta) < 0$. Hence, h maps from $\mathbb{R}^n$ to $\mathbb{R}$, while μ maps from $\mathbb{R}^n$ to $\{\top, \bot\}$. The SITL syntax is given by

$$\phi \coloneqq \top \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[a,b]} \phi_2$$

where ϕ_1 and ϕ_2 are SITL formulas, $[a, b] \subset \mathbb{R}_{\geq 0}$, with $b > a$ is a time interval, and $U_{[a,b]}$ encodes the until operator. Define the eventually and always operators as $F_{[a,b]}\phi \coloneqq \top U_{[a,b]}\phi$ and $G_{[a,b]}\phi \coloneqq \neg F_{[a,b]}\neg\phi$. The satisfaction relation $(y, t) \models \phi$ denotes that the signal $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ satisfies ϕ at time t. The SITL semantics are recursively given by $(y, t) \models \mu$ if and only if $h(y(t)) \geq 0$, $(y, t) \models \neg\phi$ if and only if $\neg((y, t) \models \phi)$, $(y, t) \models \phi_2 \wedge \phi_2$ if and only if $(x, t) \models \phi_1$ and $(y, t) \models \phi_2$, and $(y, t) \models \phi_1 U_{[a,b]} \phi_2$ if and only if there exists a $t_1 \in [t + a, t + b]$ such that $(y, t_1) \models \phi_2$ and there exists a $t_2 \in [t, t_1]$ such that $(y, t_2) \models \phi_1$. A formula ϕ is satisfiable if there exists a $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ such that $(y, 0) \models \phi$.

2.3 Problem Statement

Consider a continuous-time dynamical system governed by the kth-order continuous dynamics

$$\dot{x}_i = x_{i+1}, \quad \forall i \in \{1, \ldots, k-1\} \tag{1a}$$

$$\dot{x}_k = f(x, u(x, t), t), \tag{1b}$$

$$y = x_1 \tag{1c}$$

where $x_i \in \mathbb{R}^n$, $n \in \mathbb{N}$, is the system state, with $x \coloneqq [x_1^\top, \ldots, x_k^\top]^\top \in \mathbb{R}^{kn}$, $u : \mathbb{R}^{kn} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the time-varying feedback-control input, and $y :\in \mathbb{R}^n$ is the system output. Moreover, the term $f(\cdot)$ is a vector field that is continuously differentiable in x over $\mathbb{R}^{kn}$ for each fixed $t \geq t_0$, and uniformly bounded in t over $[t_0, \infty)$ for each fixed $x \in \mathbb{R}^{kn}$. We consider that $f(\cdot)$ is completely unknown; we do not assume any knowledge of the structure, Lipschitz constants, or bounds of $f(\cdot)$ and we do not use any scheme to approximate it. Nevertheless, we make the following assumption:

Assumption 1. *The matrix*

$$\widetilde{\mathsf{D}f}^u(x, u, t) \coloneqq \mathsf{D}f^u(x, u, t) + \mathsf{D}f^u(x, u, t)^\top \tag{2}$$

is positive definite, for all $(x, u, t) \in \mathbb{R}^{kn} \times \mathbb{R}^n \times \mathbb{R}_{\geq 0}$.

Assumption 1 is a sufficiently controllability condition for (1); intuitively, it states that the coefficients of u do not change the direction imposed to the system. Note that standard holonomic Lagrangian systems satisfy this condition. Examples include robotic manipulators, omnidirectional mobile robots, and fully actuated aerial vehicles. Systems not covered by (1) consist of underactuated or non-holonomic robots, such as unicycles, underactuated aerial or underwater vehicles. For each one of these systems, there exist transformations that can bring the respective dynamics to the form (1) studied here (see, e.g., [41]).

Consider now a time-constrained task expressed as an SITL formula ϕ over y. The objective of this paper is to construct a time-varying feedback-control algorithm $u(x, t)$ such that the output of the closed-loop system (1) satisfies ϕ, i.e., $(y(t), t) \models \phi$.

3 Main Results

This section describes the proposed algorithm, which consists of three steps. Firstly, it learns a controller, represented as a neural network, using training data that correspond to a collection of different tasks and robot parameters. Secondly, it uses formal methods tools to compute an *open-loop* trajectory that satisfies the given task. Finally, we design an adaptive, time-varying feedback controller that uses the neural-network approximation and guarantees tracking of the open-loop trajectory, consequently achieving satisfaction of the task.

3.1 Neural network approximation

We assume the existence of offline data from a finite set of T system trajectories that satisfy a collection of SITL tasks, possibly produced by robots with different dynamic parameters. The data from each trajectory $i \in \{1, \dots, T\}$ comprise a finite set of triplets $\{x_s(t), t, u_s(t)\}_{t \in \mathcal{T}_i}$, where $\mathcal{T}_i$ is a finite set of time instants, $x_s(t) \in \mathbb{R}^{kn}$ are system states, and $u_s(t) \in \mathbb{R}^n$ are the respective control inputs, compliant with the dynamics (1). We use the data to train a neural network in order to approximate the respective controller $u(t)$. More specifically, we use the pairs $(x_s(t), t)_{t \in \mathcal{T}_i}$ as input to a neural network, and $u_s(t)_{t \in \mathcal{T}_i}$ as the respective targets, for all trajectories $i \in \{1, \dots, T\}$. For given $x \in \mathbb{R}^{kn}, t \in \mathbb{R}_{\geq 0}$, we denote by $u_{\text{nn}}(x, t)$ the output of the neural network.

3.2 Open-loop trajectory

The next step is the computation of an open-loop trajectory $y_{\text{d}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ that satisfies ϕ, i.e., $(y_{\text{d}}(t), t) \models \phi$. For this computation, we use the recent work [7], which proposes an efficient planning algorithm using automata to construct a set of bounded trajectories that satisfy an SITL formula. In order to accommodate the temporal aspect of an SITL formula, such a trajectory can be represented as a finite prefix followed by the infinite repetition of a finite suffix, i.e.,

$$y_{\text{d}}(t) = y_{\text{d}}(0 : t_{f_1}) \Big| [y_{\text{d}}(t_{f_1} : t_{f_2})]^{\omega}$$

Here, $y_{\text{d}}(0 : t_{f_1})$ and $y_{\text{d}}(t_{f_1} : t_{f_2})$ denote the trajectories $y_{\text{d}}(t)$ from 0 to t_{f_1} and from t_{f_1} to t_{f_2} respectively, for some time instants $t_{f_1} > 0$, $t_{f_2} > t_{f_1}$. The operator $\cdot|\cdot$ denotes trajectory concatenation and the superscript ω denotes infinite repetition. Note that the training data of Section 3.1 are assumed to follow this prefix-suffix form; each trajectory is assumed to consist of a finite prefix followed by some repetitions of a finite suffix, i.e., $\max\{\mathcal{T}_i\} > \kappa t_{f_2}$ for some integer $\kappa > 2$ for all trajectories $i \in \{1, \dots, T\}$.

The procedure of computing the open-loop trajectory $y_{\text{d}}(t)$ does *not* take into account the dynamics (1); unlike that training data used in Section 3.1, $y_{\text{d}}(t)$ is a geometric trajectory in $\mathbb{R}^n$ that satisfies the given task. More details regarding the procedure are out of scope of this paper and can be found in [7].

3.3 Control design

We now design a time-varying feedback controller to track the trajectory $y_{\text{d}}(t)$ by using the output of the trained neural network. Let the error $e \coloneqq y - y_{\text{d}}$ and its sliding mode version

$$\begin{aligned} s &\coloneqq \left(\frac{d}{dt} + \lambda\right)^{k-1} e \\ &= \sum_{j=0}^{k-1} \frac{(k-1)!}{(k-1-j)!j!} \lambda^j e^{(k-1-j)}, \end{aligned} \tag{3}$$

for a positive constant λ, which represents a stable linear filter with input s and output e (i.e., $e(p) = \frac{s(p)}{(p+\lambda)^{k-1}}$, with p denoting the Laplace frequency variable). Therefore, the problem of driving $e(t)$ to zero reduces to the problem of driving $s(t)$ to zero, since the aforementioned stable linear filter has a unique equilibrium point $e = 0$ when the input is zero, i.e., $s = 0$.

We impose next an assumption on the closed-loop system trajectory that is driven by the neural network's output.

Assumption 2. *The output $u_{\text{nn}}(x,t)$ of the trained neural network satisfies*

$$\frac{1}{\underline{g}}\left\|\sum_{j=1}^{k-1}\frac{(k-1)!}{(k-1-j)!j!}\lambda^j\left(x_{k-j+1}-y_{\text{d}}^{(k-j)}\right)-y_{\text{d}}^{(k)}+f(x,0,t)+\mathsf{D}f^u(x,u,t)u_{\text{nn}}(x,t)\right\|\leq d\|s\|+B \tag{4}$$

for unknown positive constants d, B, for all $x\in\mathbb{R}^{kn}$, $u\in\mathbb{R}^n$, $t\geq t_0$, where $\underline{g}$ is the minimum eigenvalue of $\widetilde{\mathsf{D}f}^u(x,u,t)$, which is positive according to Assumption 1.

Intuitively, Assumption 2 states that the neural-network controller $u_{\text{nn}}(x,t)$ is able to maintain the *boundedness* of the system state with respect to the error signal by the *unknown* constants d, B. The assumption is motivated by the property of neural networks to approximate a continuous function arbitrarily well in a compact domain for a large enough number of neurons and layers [20]. Contrary to the works in the related literature (e.g., [2, 11, 12, 10]), however, we do not adopt approximation schemes for the system dynamics and we do not impose restrictions on the size of d, B. Moreover, Assumption 2 does not imply that the neural network controller $u_{\text{nn}}(x,t)$ guarantees tracking of the open-loop trajectory y_{d}. Instead, it is merely a growth condition. Additionally, note that the open-loop trajectories that are computed in Section 3.2, as well as the robot dynamic parameters that appear in $f(\cdot)$ are bounded. Hence, condition (4) would hold even if $y_{\text{d}}(t)$ was replaced by a trajectory that corresponds to a different task and $f(\cdot)$ contained different robot parameters. We demonstrate this property in the experimental results of Section 4, where we test several tasks and robots with different dynamic parameters using a single neural network.

We now define the signals $\hat{d}$, $\hat{B}$ as estimates of the constants d and B, respectively, from (4), and the respective errors $\widetilde{d}\coloneqq\hat{d}-d$, $\widetilde{B}\coloneqq\hat{B}-B$. We design the adaptive feedback controller as

$$u(x,\hat{d},\hat{B})=u_{\text{nn}}(x,t)-K_1 s-\hat{d}s-\hat{B}\hat{s} \tag{5a}$$

and the evolution of $\hat{d}$, $\hat{B}$ as

$$\dot{\hat{d}}=k_d\|s\|^2,\qquad \dot{\hat{B}}=k_B\|s\|, \tag{5b}$$

where $\hat{s}=\frac{s}{\|s\|}$ if $s\neq 0$, and $\hat{s}=0$ if $s=0$; k_d, k_B are positive gain constants and $K_1\in\mathbb{R}^{n\times n}$ is a diagonal positive definite gain matrix. The control design is inspired by adaptive control methodologies [1], where the time-varying gains $\hat{d}(t)$, $\hat{B}(t)$, used to estimate the unknown bounds d, B of (4), adapt to the unknown dynamics in order to ensure closed-loop stability. The discontinuity imposed by $\hat{s}$ was also employed in the recent work [42], which, however, uses a reciprocal control term, possibly producing unnecessarily large control inputs, as explained in Section 1. Note that (5) does not use any information on the system dynamics $f(\cdot)$ or the constants B, d. The tracking of $y_{\text{d}}(t)$ is guaranteed by the following theorem, whose proof is given in the supplementary material file.

Theorem 1. *Let the system evolving according to (1) and an open-loop trajectory $y_{\text{d}}(t)$ that satisfies a given SITL task. The control algorithm (5) guarantees $\lim_{t\to\infty}s(t)=0$, as well as the boundedness of all closed-loop signals.*

Note that, contrary to works in the related literature (e.g., [39]), we do not impose reciprocal terms in the control input that grow unbounded in order to guarantee closed-loop stability. The resulting controller is essentially a simple linear feedback on $s(t)$ with time-varying adaptive control gains, accompanied by the neural network output that ensures the growth condition 4.

4 Experimental Results

We conduct experiments using a 6-dof UR5 robotic manipulator in the CoppeliaSim robotic simulation [43] to demonstrate the effectiveness of the theoretical results. The robot dynamics are of the form

$$\dot{x}_1=x_2 \tag{6a}$$

$$\dot{x}_2=B(x_1)^{-1}\left(u-C(x)x_2-g(x_1)+d(t)\right), \tag{6b}$$

where $x_1\in\mathbb{R}^6$ is vector robot joint angles, and $x_2\in\mathbb{R}^6$ is the vector of the respective joint velocities; $B(x_1)\in\mathbb{R}^{6\times 6}$ is the positive definite inertia matrix, $C(x)\in\mathbb{R}^{6\times 6}$ is the Coriolis

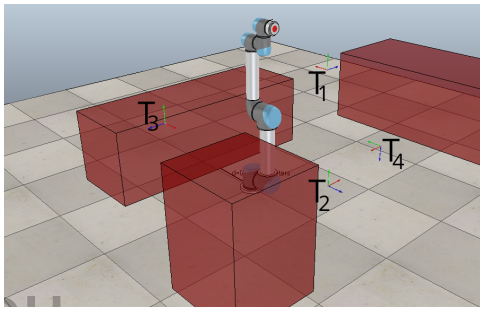


Figure 1: The initial configuration of the UR5 robot with four points of interest T_i, $i \in \{1, \dots, 4\}$.

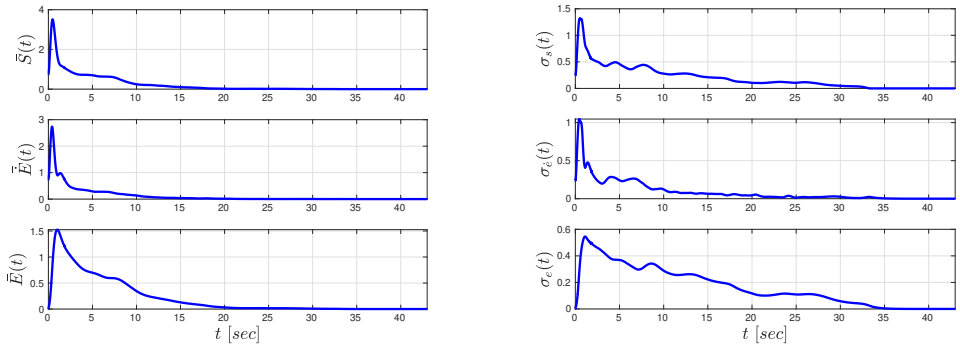
matrix, $g(x_1) \in \mathbb{R}^6$ is the gravity vector, and $d(t) \in \mathbb{R}^6$ is a vector of external time-varying disturbances. The workspace consists of four points of interest T_i, $i \in \{1, \dots, 4\}$ and three obstacles $\mathcal{O}_i \subset \mathbb{R}^3$, $i \in \{1, 2, 3\}$, as depicted in Fig. 1. The SITL task we consider has the form $\phi = \big(\bigwedge_{i\in\{1,2,3\}} G_{[0,\infty)}\mathcal{R} \cap \mathcal{O}_i = \emptyset\big) \bigwedge \big(\bigwedge_{i\in\{1,\dots,4\}} G_{[0,\infty)} F_{I_i}(\|x_1 - c_i\| \leq 0.1)\big)$, i.e., visit of x_1 to $c_i \in \mathbb{R}^6$ (within the radius 0.1) infinitely often within the time intervals dictated by I_i, for $i \in \{1, \dots, 4\}$, while avoiding the obstacles. The points are $c_1 = [-0.07, -1.05, 0.45, 2.3, 1.37, -1.33]^\top$, $c_2 = [1.28, 0.35, 1.75, 0.03, 0.1, -1.22]^\top$, $c_3 = [-0.08, 0.85, -0.23, 2.58, 2.09, -2, 36]^\top$, $c_4 = [-0.7, -0.76, -1.05, -0.05, -3.08, 2.37]^\top$ (radians), corresponding to the end-effector poses $T_1 = [-0.15, -0.475, 0.675, \frac{\pi}{2}, 0, 0]^\top$, $T_2 = [-0.6, 0, 2.5, 0, -\frac{\pi}{2}, -\frac{\pi}{2}]^\top$, $T_3 = [-0.025, 0.595, 0.6, -\frac{\pi}{2}, 0, \pi]^\top$, and $T_4 = [-0.525, -0.55, 0.28, \pi, 0, -\frac{\pi}{2}]^\top$, respectively, illustrated in Fig. 1.

We set a nominal value for the time intervals as $I_i = [0, 43]$ (seconds), and we train a neural network on 150 trajectories generated by system runs that satisfy different variations of ϕ and from different initial conditions; the variations are produced by adding uniformly random offsets in $[-0.3, 0.3]$ to the elements of all c_i, and in $[-3, 3]$ to the end points of the intervals I_i. In all the training trajectories, the system navigates to the points c_i in the prescribed time in the sequence dictated by the indices for one cycle, i.e., $((x_1(0), 0) \to (c_1, t_{1_1}) \to (c_2, t_{1_2}) \to (c_3, t_{1_3}) \to (c_4, t_{1_4})$. Moreover, the system runs are generated via a nominal model-based controller and using different robot dynamic parameters; that is, we add uniformly random offset values in $[0, 0.3]$ (kg and $kg \cdot m^2$) to the inertial parameters of the robot, i.e., masses and moments of inertia of the UR5 motors and links. The neural network consists of 4 fully connected layers of 128 neurons; each layer is followed by a batch normalization module and a ReLU activation function. For the training we use the adam optimizer and the mean-square-error loss function.

In order to test the control algorithm (5), we generate, using the procedure described in Section 3.2, a set of 100 new open-loop trajectories $y_{\text{d}}^k(t)$, that satisfy different variations of ϕ, where the superscript denotes trajectory $k \in \{1, \dots, 100\}$. Similar to the training data, we produce the variations by adding uniformly random offsets in $[-0.3, 0.3]$ (radians) to the elements of all c_i, and $[-3, 3]$ to the end points of the intervals I_i. In order to demonstrate the effectiveness of the proposed algorithm, we add an extra degree of variation; namely, for each of the 100 test trajectories, we select randomly the *sequence* of visits to the points c_i in the prescribed time instants. For example, a trajectory cycle might be $(y_{\text{d}}(0), 0) \to (c_3, 8) \to (c_1, 15) \to (c_4, 30) \to (c_2, 42)$ or $(y_{\text{d}}(0), 0) \to (c_4, 10) \to (c_3, 19) \to (c_1, 32) \to (c_2, 44)$. In order to render the open-loop trajectories $y_{\text{d}}^k(t)$ collision-free, we use the rapidly-exploring random tree (RRT) algorithm [44] after obtaining the timed sequence of points to be visited. By choosing $\lambda = 1$, the sliding error (3) becomes $s = \dot{e} + e$, where $e(t) = x_1(t) - y_{\text{d}}(t)$. Moreover, we choose the control gains in (5) as $K_1 = \text{diag}\{15, 100, 100, 15, 10, 0.01\}$, $k_d = k_B = 0.1$. Finally and similar to the training data, we add uniformly random offset values in $[0, 0.3]$ (kg and $kg \cdot m^2$) to the inertial parameters of the robot in order to test the robustness of the algorithm.

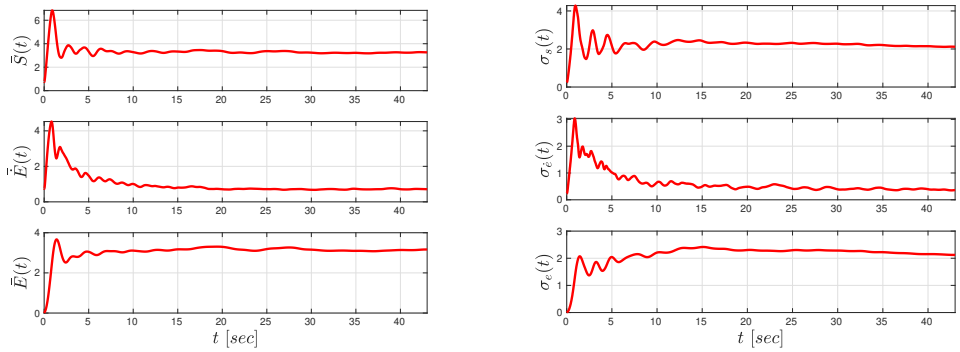
We first test the algorithm using a python ODE simulator of (6), where we set $d(t)$ as a vector of sinusoidal functions of time. We compare our algorithm with the simple non-adaptive controller

$$u_c(x, t) = u_{\text{nn}}(x, t) - K_1 s \tag{7}$$



(a) (b)

Figure 2: The evolution of the mean error norms $\bar{S}(t)$, $\bar{E}(t)$, and $\bar{\dot{E}}(t)$ in (a), and the standard deviations $\sigma_s(t)$, $\sigma_e(t)$, $\sigma_{\dot{e}}(t)$ in (b), for the proposed control algorithm for $t \in [0, 43]$ seconds.



(a) (b)

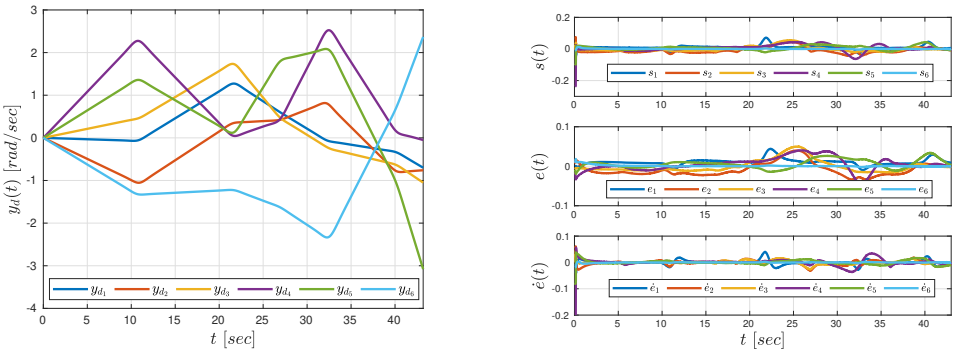
Figure 3: The evolution of the mean error norms $\bar{S}(t)$, $\bar{E}(t)$, and $\bar{\dot{E}}(t)$ in (a), and the standard deviations $\sigma_s(t)$, $\sigma_e(t)$, $\sigma_{\dot{e}}(t)$ in (b), for the controller (7) for $t \in [0, 43]$ seconds.

The comparison results are depicted in Figs. 2 and 3, which depict the mean error norms and standard deviations for the 100 trajectories for the two controllers, i.e., $\bar{S}(t) = \frac{1}{100}\sum_{k=1}^{100} \|s^k(t)\|$, $\bar{E}(t) := \frac{1}{100}\sum_{k=1}^{100} \|e^k(t)\|$, $\bar{\dot{E}}(t) := \frac{1}{100}\sum_{k=1}^{100} \|\dot{e}^k(t)\|$, and $\sigma_S(t) := \sqrt{\frac{\sum_{k=1}^{100}(s^k(t)-\bar{S}(t))^2}{100}}$, $\sigma_e(t) := \sqrt{\frac{\sum_{k=1}^{100}(e^k(t)-\bar{E}(t))^2}{100}}$, $\sigma_{\dot{e}}(t) := \sqrt{\frac{\sum_{k=1}^{100}(\dot{e}^k(t)-\bar{\dot{E}}(t))^2}{100}}$ for $t \in [0, 43]$ sec. It is clear from the two figures that the proposed algorithm outperforms the simple controller (7).

We next validate the proposed algorithm using one of the test trajectories $y_{\mathrm{d}}(t)$, which is shown in Fig. 4a, in the CoppeliaSim simulator for $t \in [0, 43]$ sec. The results are depicted in Figs. 4b and 5. In particular, 4b shows the tracking errors $s(t)$, $e(t)$, $\dot{e}(t)$, which evolve very close to zero, and Fig. 5 depicts screenshots of the robot visits to T_i, $i \in \{1, \ldots, 4\}$. Plots of more results (distance from obstacles, control input, adaptation variables) as well as a video of the CoppeliaSim experiment are provided in the supplementary material. For the CoppeliaSim experiment, the control algorithm was implemented in MATLAB environment and communicated to the CoppeliaSim client via ROS messages. The neural networks were trained in Python environment using the Pytorch library.

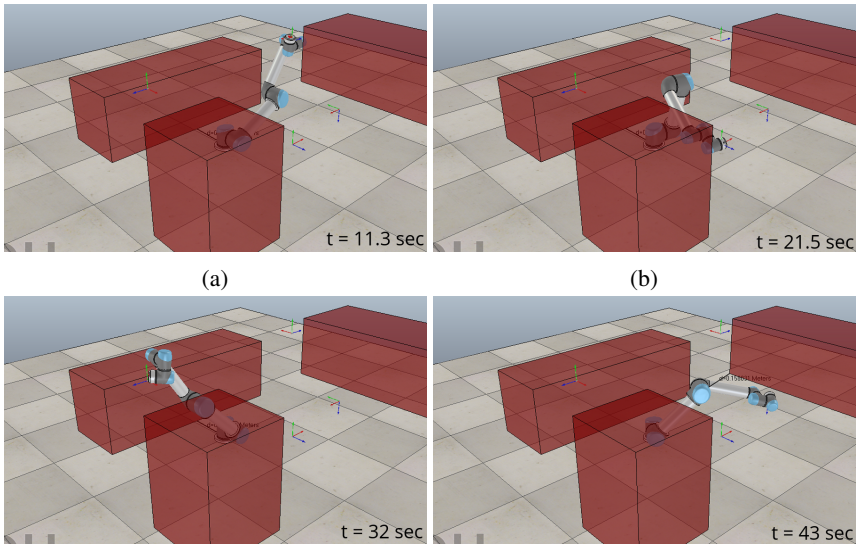
5 Discussion

As shown in the experimental results of Sec. 4, the control algorithm is able to accomplish tasks which were not considered when generating the training data. Similarly, the trajectories used in the training data were generated using robots with different dynamical parameters, and not specifically the ones used in the tests. The aforementioned attributes signify the ability of the proposed algorithm to generalize to different tasks and robots with different parameters. Nevertheless, it should be noted



(a) (b)

Figure 4: (a) The open-loop reference trajectory $y_{\mathrm{d}}(t)$. (b) The evolution of the error signals $s(t)$, $e(t)$, $\dot{e}(t)$ for $t \in [0, 43]$ sec.



(a) (b)

(c) (d)

Figure 5: Snapshots of the closed-loop robot trajectory at $t \in \{11.3, 21.5, 32, 43\}$ sec, corresponding to the respective visit to the target points T_i, $i \in \{1, \ldots, 4\}$.

that the control algorithm (5) guarantees asymptotic stability properties, without characterizing the system's transient state. Hence, task specifications that require fast system response (e.g., a robot needs to travel a long distance in a short time horizon) would possibly necessitate large control gains (K_1, k_d, k_B in (5)) in order to achieve fast convergence to the derived open-loop trajectory. The derivation of transient-state bounds and their relation with the given task specification (as for instance in [39, 40]) is left for future work. Finally, the discontinuity of (5) might be problematic and create chattering when implemented in real actuators. A continuous approximation that has shown to yield satisfying performance is the boundary-layer technique [45].

6 Conclusions and Future Work

We develop a novel control algorithm for the control of robotic systems with unknown nonlinear dynamics subject to task specifications expressed as SITL constraints. The algorithm integrates neural network-based learning and adaptive control. We provide formal guarantees and perform extensive experimental results. Future directions will focus on extending the proposed methodology to underactuated and non-holonomic systems as well as performing experiments on real robots.

References

[1] M. Krstic, I. Kanellakopoulos, and P. Kokotovic. Nonlinear and Adaptive Control Design. *Publisher: Wiley New York*, 1995.

[2] K. G. Vamvoudakis and F. L. Lewis. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[3] T. Berger, H. H. Lê, and T. Reis. Funnel control for nonlinear systems with known strict relative degree. *Automatica*, 87:345–357, 2018.

[4] C. P. Bechlioulis and G. A. Rovithakis. A low-complexity global approximation-free control scheme with prescribed performance for unknown pure feedback systems. *Automatica*, 50(4): 1217–1226, 2014.

[5] G. Joshi, J. Virdi, and G. Chowdhary. Asynchronous deep model reference adaptive control. *Conference on Robot Learning*, 2020.

[6] M. Capotondi, G. Turrisi, C. Gaz, V. Modugno, G. Oriolo, and A. De Luca. An online learning procedure for feedback linearization control without torque measurements. *Conference on Robot Learning*, pages 1359–1368, 2020.

[7] L. Lindemann and D. V. Dimarogonas. Efficient automata-based planning and control under spatio-temporal logic specifications. *American Control Conference (ACC)*, pages 4707–4714, 2020.

[8] J. Huang, W. Wang, C. Wen, and J. Zhou. Adaptive control of a class of strict-feedback time-varying nonlinear systems with unknown control coefficients. *Automatica*, 93:98–105, 2018.

[9] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe. Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. *IEEE Robotics and Automation Letters*, 5(2):3050–3057, 2020.

[10] B. Fan, Q. Yang, X. Tang, and Y. Sun. Robust adp design for continuous-time nonlinear systems with output constraints. *IEEE transactions on neural networks and learning systems*, 29(6): 2127–2138, 2018.

[11] Y. Yang, K. G. Vamvoudakis, and H. Modares. Safe reinforcement learning for dynamical games. *International Journal of Robust and Nonlinear Control*, 30(9):3706–3726, 2020.

[12] T. Cheng, F. L. Lewis, and M. Abu-Khalaf. A neural network solution for fixed-final time optimal control of nonlinear systems. *Automatica*, 43(3):482–490, 2007.

[13] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6):2042–2062, 2017.

[14] J. Zhao and M. Gan. Finite-horizon optimal control for continuous-time uncertain nonlinear systems using reinforcement learning. *International Journal of Systems Science*, 51(13):2429–2440, 2020.

[15] D. Vrabie and F. Lewis. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks*, 22(3):237–246, 2009.

[16] C. Sun and K. G. Vamvoudakis. Continuous-time safe learning with temporal logic constraints in adversarial environments. *American Control Conference (ACC)*, pages 4786–4791, 2020.

[17] R. Kamalapurkar, H. Dinh, S. Bhasin, and W. E. Dixon. Approximate optimal trajectory tracking for continuous-time nonlinear systems. *Automatica*, 51:40–48, 2015.

[18] X. Huang, Y. Song, and J. Lai. Neuro-adaptive control with given performance specifications for strict feedback systems under full-state constraints. *IEEE transactions on neural networks and learning systems*, 30(1):25–34, 2018.

[19] L. Mo, X. Yuan, and Y. Yu. Neuro-adaptive leaderless consensus of fractional-order multi-agent systems. *Neurocomputing*, 339:17–25, 2019.

[20] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[21] F. Hong, S. Ge, B. Ren, and T. Lee. Robust adaptive control for a class of uncertain strict-feedback nonlinear systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 19(7):746–767, 2009.

[22] Z. Chen. Nussbaum functions in adaptive control with time-varying unknown control coefficients. *Automatica*, 102:72–79, 2019.

[23] K. Leahy, E. Cristofalo, C.-I. Vasile, A. Jones, E. Montijano, M. Schwager, and C. Belta. Control in belief space with temporal logic specifications using vision-based localization. *The International Journal of Robotics Research*, 38(6):702–722, 2019.

[24] A. Jain, T. Nghiem, M. Morari, and R. Mangharam. Learning and control using gaussian processes. *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 140–149, 2018.

[25] F. Berkenkamp and A. P. Schoellig. Safe and robust learning control with gaussian processes. *European Control Conference (ECC)*, pages 2496–2501, 2015.

[26] M. Ma, J. Gao, L. Feng, and J. Stankovic. Stlnet: Signal temporal logic enforced multivariate recurrent neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] A. J. Shah, P. Kamath, S. Li, and J. A. Shah. Bayesian inference of temporal task specifications from demonstrations. 2018.

[28] R. Yan and A. Julius. Neural network for weighted signal temporal logic. *arXiv preprint arXiv:2104.05435*, 2021.

[29] W. Liu, N. Mehdipour, and C. Belta. Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*, 2021.

[30] C. Hahn, F. Schmitt, J. U. Kreber, M. N. Rabe, and B. Finkbeiner. Teaching temporal logics to neural networks. *arXiv preprint arXiv:2003.04218*, 2020.

[31] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *arXiv preprint arXiv:2102.12855*, 2021.

[32] C. Wang, Y. Li, S. L. Smith, and J. Liu. Continuous motion planning with temporal logic specifications using deep neural networks. *arXiv preprint arXiv:2004.02610*, 2020.

[33] A. Camacho and S. A. McIlraith. Towards neural-guided program synthesis for linear temporal logic specifications. *arXiv preprint arXiv:1912.13430*, 2019.

[34] R. Riegel, A. Gray, F. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[35] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. *IEEE Conference on Decision and Control (CDC)*, pages 5929–5934, 2020.

[36] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[37] E. T. Maddalena, C. d. S. Moraes, G. Waltrich, and C. N. Jones. A neural network architecture to learn explicit mpc controllers from data. *IFAC-PapersOnLine*, 53(2):11362–11367, 2020.

[38] L. Lindemann, C. K. Verginis, and D. V. Dimarogonas. Prescribed performance control for signal temporal logic specifications. *IEEE Conference on Decision and Control (CDC)*, pages 2997–3002, 2017.

[39] C. K. Verginis and D. V. Dimarogonas. Timed abstractions for distributed cooperative manipulation. *Autonomous Robots*, 42(4):781–799, 2018.

[40] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki. Kdf: Kinodynamic motion planning via geometric sampling-based algorithms and funnel control. 2021.

[41] C. P. Bechlioulis, G. C. Karras, S. Heshmati-Alamdari, and K. J. Kyriakopoulos. Trajectory tracking with prescribed performance for underactuated underwater vehicles under model uncertainties and external disturbances. *IEEE Transactions on Control Systems Technology*, 25 (2):429–440, 2016.

[42] C. Verginis and D. V. Dimarogonas. Asymptotic tracking of second-order nonsmooth feedback stabilizable unknown systems with prescribed transient response. *IEEE Transactions on Automatic Control*, 2020.

[43] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[44] S. M. LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[45] J.-J. Slotine and S. S. Sastry. Tracking control of non-linear systems using sliding surfaces, with application to robot manipulators. *International journal of control*, 38(2):465–492, 1983.

[46] B. Paden and S. Sastry. A calculus for computing filippov's differential inclusion with application to the variable structure control of robot manipulators. *IEEE transactions on circuits and systems*, 34(1):73–82, 1987.

[47] N. Fischer, R. Kamalapurkar, and W. E. Dixon. Lasalle-yoshizawa corollaries for nonsmooth systems. *IEEE Transactions on Automatic Control*, 58(9):2333–2338, 2013.

[48] A. Zemouche, M. Boutayeb, and G. I. Bara. Observers for a class of lipschitz systems with extension to h_∞ performance analysis. *Systems & Control Letters*, 57(1):18–27, 2008.

Appendix A Proof of Theorem 1

We provide first the proof of Theorem 1, after giving some necessary notation and preliminary background.

Notation

Given a function $f : \mathbb{R}^n \to \mathbb{R}^k$, its Filippov regularization is defined as [46]

$$\mathsf{K}[f](x) \coloneqq \bigcap_{\delta>0} \bigcap_{\mu(\bar{N})=0} \overline{\text{co}}(f(\mathcal{B}(x,\delta)\backslash\bar{N}), t), \tag{8}$$

where $\bigcap_{\mu(\bar{N})=0}$ is the intersection over all sets $\bar{N}$ of Lebesgue measure zero, and $\overline{\text{co}}(E)$ is the closure of the convex hull $\text{co}(E)$ of the set E. The Filippov regularization of $\text{sgn}(x) \in \mathbb{R}$ is denoted by $\mathsf{K}[\text{sgn}](x) = \text{SGN}(x)$ where $\text{SGN}(x) = -1$, if $x < 0$, $\text{SGN}(x) = 1$, if $x > 0$, and $\text{SGN}(x) \in [-1, 1]$, if $x = 0$. For a vector $x = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$, we use the notation $\text{SGN}(x) \coloneqq [\text{SGN}(x_1), \ldots, \text{SGN}(x_n)]^\top \in \mathbb{R}^n$.

Nonsmooth Analysis

Consider the following differential equation with a discontinuous right-hand side:

$$\dot{x} = f(x, t), \tag{9}$$

where $f : \mathcal{D} \times [t_0, \infty) \to \mathbb{R}^n$, $\mathcal{D} \subset \mathbb{R}^n$, is Lebesgue measurable and locally essentially bounded.

Definition 1 (Def. 1 of [47]). *A function $x : [t_0, t_1) \to \mathbb{R}^n$, with $t_1 > t_0$, is called a Filippov solution of (9) on $[t_0, t_1)$ if $x(t)$ is absolutely continuous and if, for almost all $t \in [t_0, t_1)$, it satisfies $\dot{x} \in \mathsf{K}[f](x, t)$, where $\mathsf{K}[f](x, t)$ is the Filippov regularization of $f(x, t)$.*

Lemma 1 (Lemma 1 of [47]). *Let $x(t)$ be a Filippov solution of (9) and $V : \mathcal{D} \times [t_0, t_1) \to \mathbb{R}$ be a locally Lipschitz, regular function[3]. Then $V(x(t), t)$ is absolutely continuous, $\dot{V}(x(t), t) = \frac{\partial}{\partial t} V(x(t), t)$ exists almost everywhere (a.e.), i.e., for almost all $t \in [t_0, t_1)$, and $\dot{V}(x(t), t) \overset{a.e}{\in} \dot{\tilde{V}}(x(t), t)$, where*

$$\dot{\tilde{V}} \coloneqq \bigcap_{\xi \in \partial V(x,t)} \xi^\top \begin{bmatrix} \mathsf{K}[f](x, t) \\ 1 \end{bmatrix},$$

and $\partial V(x, t)$ is Clarke's generalized gradient at (x, t) [47].

Corollary 1 (Corollary 2 of [47]). *For the system given in (9), let $\mathcal{D} \subset \mathbb{R}^n$ be an open and connected set containing $x = 0$ and suppose that f is Lebesgue measurable and $x \mapsto f(x, t)$ is essentially locally bounded, uniformly in t. Let $V : \mathcal{D} \times [t_0, t_1) \to \mathbb{R}$ be locally Lipschitz and regular such that $W_1(x) \leq V(x, t) \leq W_2(x)$, $\forall t \in [t_0, t_1)$, $x \in \mathcal{D}$, and*

$$z \leq -W(x(t)), \;\; \forall z \in \dot{\tilde{V}}(x(t), t), \; t \in [t_0, t_1), \; x \in \mathcal{D},$$

where W_1 and W_2 are continuous positive definite functions and W is a continuous positive semi-definite on $\mathcal{D}$. Choose $r > 0$ and $c > 0$ such that $\bar{\mathcal{B}}(0, r) \subset \mathcal{D}$ and $c < \min_{\|x\|=r} W_1(x)$. Then for all Filippov solutions $x : [t_0, t_1) \to \mathbb{R}^n$ of (9), with $x(t_0) \in \mathbb{D} \coloneqq \{x \in \bar{\mathcal{B}}(0, r) : W_2(x) \leq c\}$, it holds that $t_1 = \infty$, $x(t) \in \mathbb{D}$, $\forall t \in [t_0, \infty)$, and $\lim_{t\to\infty} W(x(t)) = 0$.

Differential Mean Value Theorem (DMVT)

Proposition 1 (DMVT). *[48] Let $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}^n$ be a differentiable vector valued function in an open set $\mathsf{S} \subset \mathbb{R}^n$. Let $a, b \in \mathsf{S}$. Then, there exist constant vectors $z_1^*, \ldots, z_n^* \in \mathsf{Co}(a, b)$, with $z_i^* \notin a, b$, $\forall i \in \{1, \ldots, n\}$, such that*

$$\mathsf{f}(a) - \mathsf{f}(b) = \mathsf{Df}(z^*)(a - b),$$

with $z^* \coloneqq [z_1^*, \ldots, z_n^*]^\top$.

[3] See [47] for a definition of regular functions.

Proof of Theorem 1. Since the control algorithm is discontinuous, we use the notion of Filippov solutions. Let $\widetilde{x} \coloneqq [s^\top, \widetilde{d}, \widetilde{B}]^\top \in \mathbb{R}^{n+2}$. The Filippov regularization of u is $\mathsf{K}[u] = u_{\text{nn}}(x,t) - k_1 s - \hat{d}s - \hat{B}\hat{\mathsf{S}}$, where $\hat{\mathsf{S}} \coloneqq \frac{s}{\|s\|}$ if $s \neq 0$ and $\hat{\mathsf{S}} \in (-1,1)$ otherwise. Note that, in any case, it holds that $s^\top \hat{\mathsf{S}} = \|s\|$.

Let now the continuously differentiable function

$$V(\widetilde{x}) \coloneqq \frac{1}{2\underline{g}}\|s\|^2 + \frac{1}{2k_d}\widetilde{d}^2 + \frac{1}{2k_B}\widetilde{B},$$

which satisfies $W_1(\widetilde{x}) \leq V(\widetilde{x}) \leq W_1(\widetilde{x})$ for $W_1(\widetilde{x}) \coloneqq \min\{\frac{1}{2\underline{g}}, \frac{1}{2k_d}, \frac{1}{2k_B}\}\|\widetilde{x}\|^2$, $W_2(\widetilde{x}) \coloneqq \max\{\frac{1}{2\underline{g}}, \frac{1}{2k_d}, \frac{1}{2k_B}\}\|\widetilde{x}\|^2$. According to Lemma 1, $\dot{V}(\widetilde{x}) \overset{a.e.}{\in} \dot{\widetilde{V}}(\widetilde{x})$, with $\dot{\widetilde{V}} \coloneqq \bigcap_{\xi \in \partial V(\widetilde{x})} \mathsf{K}[\dot{\widetilde{x}}]$. Since $V(\widetilde{x})$ is continuously differentiable, its generalized gradient reduces to the standard gradient and thus it holds that $\dot{\widetilde{V}}(\widetilde{x}) = \nabla V^\top \mathsf{K}[\widetilde{x}]$, where $\nabla V = [\frac{1}{\underline{g}}s^\top, \frac{1}{k_d}\widetilde{d}, \frac{1}{k_B}\widetilde{B}]^\top$. By differentiating V and using (3), one obtains

$$\begin{aligned}
\dot{V} \subset \dot{\widetilde{W}}_s \coloneqq & \frac{1}{\underline{g}}s^\top \left(\sum_{j=0}^{k-1} \frac{(k-1)!}{(k-1-j)!j!} \lambda^j e^{(k-j)} \right) + \frac{1}{k_d}\widetilde{d}\dot{\hat{d}} + \frac{1}{k_B}\widetilde{B}\dot{\hat{B}} \\
= & \frac{1}{\underline{g}}s^\top \left(\sum_{j=1}^{k-1} \frac{(k-1)!}{(k-1-j)!j!} \lambda^j \left(x^{(k-j+1)} - y_{\text{d}}^{(k-j)} \right) + f(x,u,t) - y_{\text{d}}^{(k)} \right) + \frac{1}{k_d}\widetilde{d}\dot{\hat{d}} + \frac{1}{k_B}\widetilde{B}\dot{\hat{B}}
\end{aligned}$$

Since f is continuously differentiable in x, u and uniformly bounded in t, we conclude from the DMVT (Proposition 1) that there exist constant vectors $z_1^*, \ldots, z_{n_k}^* \in \mathsf{Co}(0,u)$, such that

$$f(x,u,t) = f(x,0,t) + \mathsf{D}f^u(x,z^*,t)u,$$

and hence $\dot{\widetilde{W}}_s$ becomes

$$\begin{aligned}
\dot{\widetilde{W}}_s = & \frac{1}{\underline{g}}s^\top \left(\sum_{j=1}^{k-1} \frac{(k-1)!}{(k-1-j)!j!} \lambda^j \left(x^{(k-j+1)} - y_{\text{d}}^{(k-j)} \right) + f(x,0,t) + \mathsf{D}f^u(x,z^*,t)u - y_{\text{d}}^{(k)} \right) \\
& + \frac{1}{k_d}\widetilde{d}\dot{\hat{d}} + \frac{1}{k_B}\widetilde{B}\dot{\hat{B}},
\end{aligned}$$

and by substituting (5)

$$\begin{aligned}
\dot{\widetilde{W}}_s \coloneqq & \frac{1}{\underline{g}}s^\top \Big(\sum_{j=1}^{k-1} \frac{(k-1)!}{(k-1-j)!j!} \lambda^j \left(x_{k-j+1} - y_{\text{d}}^{(k-j)} \right) + f(x,0,t) + \mathsf{D}f^u(x,z^*,t)u_{\text{nn}}(x,t) - y_{\text{d}}^{(k)} \Big) \\
& - \frac{1}{\underline{g}}K_1 s^\top \mathsf{D}f^u(x,z^*,t)s - \frac{\hat{d}}{\underline{g}}s^\top \mathsf{D}f^u(x,z^*,t)s - \frac{\hat{B}}{\underline{g}}s^\top \mathsf{D}f^u(x,z^*,t)\hat{\mathsf{S}} + \widetilde{d}\|s\|^2 + \widetilde{B}\|s\|.
\end{aligned}$$

By using Assumption 2 and the definition of $\underline{g}$, we further obtain

$$\dot{\widetilde{W}}_s \leq d\|s\|^2 + B\|s\| - K_1\|s\|^2 - \hat{d}\|s\|^2 - \hat{B}\|s\| + \widetilde{d}\|s\|^2 + \widetilde{B}\|s\|,$$

and by using $\widetilde{d} = \hat{d} - d$, $\widetilde{B} = \hat{B} - B$, we finally get

$$\dot{\widetilde{W}}_s \leq -K_1\|s\|^2 =: -Q(\widetilde{x}),$$

which implies that $\zeta \leq -Q(\widetilde{x})$, for all $\zeta \in \dot{\widetilde{V}}$, with Q being a continuous and positive definite function in $\mathbb{R}^{n+2}$. Choose now any finite $r > 0$ and let $c < \min_{\|\widetilde{x}\|=r} W_1(\widetilde{x})$. Hence, all the conditions of Corollary 1 are satisfied and hence, all Filippov solutions starting from $\widetilde{x}(0) \in \Omega_f \coloneqq \{\widetilde{x} \in \mathcal{B}(0,r) : \widetilde{W}_2(\widetilde{x}) \leq c\}$ are bounded and remain in Ω_f, satisfying $\lim_{t\to\infty} Q(\widetilde{x}(t)) = 0$.

Note that r, and hence c, can be arbitrarily large allowing and finite initial condition $\widetilde{x}(0)$. Moreover, the boundedness of $\widetilde{x}$ implies the boundedness of s, $\widetilde{d}$, and $\widetilde{B}$, and hence of $\hat{d}(t)$ and $\hat{B}(t)$, for all $t \in \mathbb{R}_{\geq 0}$. In view of (5), we finally conclude the boundendess of $u(\cdot)$, $\dot{\hat{d}}$, and $\dot{\hat{B}}$, for all $t \in \mathbb{R}_{\geq 0}$, leading to the conclusion of the proof.

□

Appendix B Further Experimental Details

In this section we provide more results for the experiment in the CoppeliaSim environment. In particular, Fig. 6 depicts the distance, denoted by $d_o(t)$ of the UR5 robot to the obstacles, which is always positive, verifying thus the safety of the experiment. In addition, Fig. 7 shows the evolution of the adaptation variables $\hat{d}(t)$, $\hat{B}(t)$. Note that, although $\hat{B}(t)$ seems to be continuously increasing (due to the very small but non-zero values of $\|s\|$ - see eq. (5b)), both signals are confined in very small values for the entire trajectory. Finally, Fig. 8 shows the control inputs of the experiment.

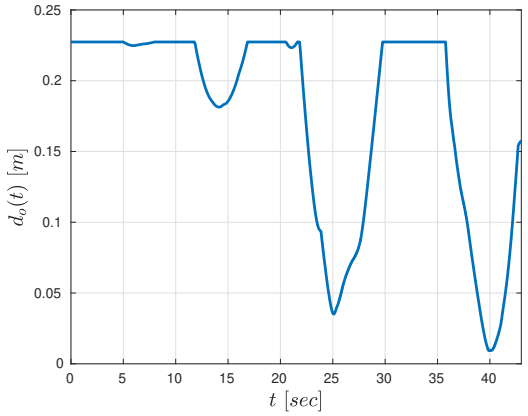


Figure 6: The evolution of the distance $d_o(t)$ of the UR5 robot to the environment obstacles in the CoppeliaSim experiment for $t \in [0, 43]$ seconds.

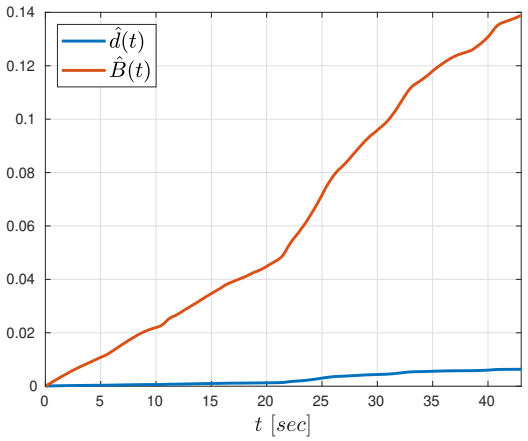


Figure 7: The evolution of the adaptation variables $\hat{d}(t)$, $\hat{B}(t)$ in the CoppeliaSim experiment for $t \in [0, 43]$ seconds.

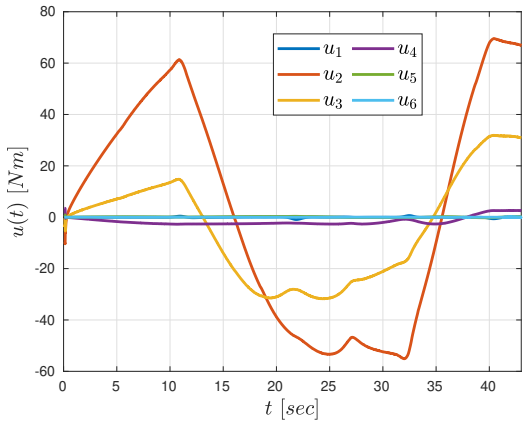


Figure 8: The resulting control inputs $u(t)$ in the CoppeliaSim experiment for $t \in [0, 43]$ seconds.