2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

A Predictive Deep Learning Approach to Output Regulation: The Case of Collaborative Pursuit Evasion

S. Shivam[1], A. Kanellopoulos[2], K. G. Vamvoudakis[2], and Y. Wardi[1]

Abstract— **In this paper, we consider the problem of controlling an underactuated system in unknown, and potentially adversarial environments. The emphasis will be on autonomous aerial vehicles, modelled by Dubins dynamics. The proposed control law is based on a variable integrator via online prediction for target tracking. To showcase its efficacy we analyze a pursuit evasion game between multiple autonomous agents. To obviate the need for perfect knowledge of the evader's future strategy, we use a deep neural network that is trained to approximate the behavior of the evader based on measurements gathered online during the pursuit.**

I. Introduction

Output tracking in dynamical systems is the practice of designing decision makers which ensure that a system's output tracks a given signal [1], [2]. Well-known existing methods for nonlinear output regulation and tracking include control techniques based on nonlinear inversions [3], high-gain observers [4], and the framework of model predictive control (MPC) [5], [6].

Recently a new approach to output tracking has been proposed by authors of this paper, based on the Newton-Raphson flow for solving algebraic equations [7]. Subsequently it has been tested on various applications, including controlling an inverted pendulum and position control of platoons of mobile robotic vehicles [7], [8]. While perhaps not as general as the aforementioned established tracking techniques, it seems to hold out promise of efficient computations and large domains of stability. However, this approach has two glaring limitations: It is model based, and its application requires that the system's input and output have the same dimension. The objective of this paper is to circumvent these limitations. To this end we analyze a particular but challenging example which serves to illustrate initial ideas, whose further developments and expositions in more-general settings is currently under investigation.

The example in question consists of the pursuit-evasion problem investigated in [9], where the strategies of both pursuers and evader are based on respective games. In [9], the pursuers know the game of the evader ahead of time, and an MPC technique is used to determine their trajectories. This paper considers the case where the pursuers do not have an a-priori knowledge of the evader's game or its structure, and they employ a neural network (NN) in real time to identify its input-output mapping. We use our tracking-control technique [7] which arguably can require simpler computations than MPC, and obtain similar results to [9]. Furthermore, the considered problem has lower-dimension input than output, and we demonstrate an approach to overcome this limitation which may have a broad scope in applications.

It is hard to exaggerate the importance of learning techniques in control. The successful deployment of complex control systems increasingly depends on their ability to operate on highly unstructured – even adversarial – settings, where *a-priori* knowledge of the evolution of the environment is impossible to acquire. Moreover, due to the increasing interconnection between the physical and the cyber domains, control systems become more intertwined with human operators, making model-based solutions fragile to unpredictable. Towards that, methods that augment low-level control techniques with intelligent decision making mechanisms have been extensively investigated in the past three decades (see [10]). Machine learning [11], [12] offers a suitable framework to allow control systems to autonomously adapt by leveraging data gathered from their environment. To enable data-driven solutions for autonomy, learning algorithms use artificial neural networks.

NNs have been used extensively to implement reinforcement learning and machine-learning techniques in various control application; see [13] for an early work, and [14] for a recent survey. Prediction has been in the forefront of research conducted on machine learning, with applications ranging from cyber security [15], [16] to pursuit-evasion games [17].

The rest of the paper is structured as follows. Section II describes our proposed control technique and some preliminary results on NN, and it formulates the pursuers-evader problem. Section III describes our approach to the problem, based on model-based and learning-based strategies. Simulation results are presented in Section IV, and Section V concludes the paper. An extended version of this paper appeared in [18].

II. Preliminaries and Problem Formulation

A. Tracking Control Technique

This subsection recounts results published in our previous work in which prediction-based output tracking was used for fully-actuated systems [7], [8]. Consider a system as shown in Figure 1 with $r(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^m$, $u(t) \in \mathbb{R}^m$, and

[1]S. Shivam, and Y. Wardi are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA. e-mail: (sshivam6@gatech.edu, ywardi@gatech.edu).

[2]A. Kanellopoulos, and K. G. Vamvoudakis are with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA. e-mail: (ariskan@gatech.edu, kyriakos@gatech.edu).

This work was supported in part, by ONR Minerva under grant No. $N00014-18-1-2160$, and by NSF under grants No. SaTC-1801611 and CPS-1851588.

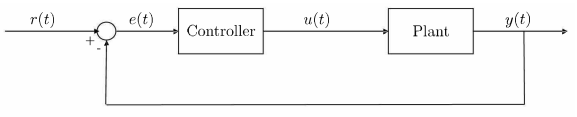

Fig. 1. Basic control system scheme.

$e(t) := r(t) - y(t)$. The objective of the controller is to ensure that

$$\lim_{t\to\infty} ||r(t) - y(t)|| < \varepsilon,$$

for a given (small) $\varepsilon \in \mathbb{R}^+$.

To illustrate the basic idea underscoring the controller, let us first assume that (i) The plant subsystem is a memoryless nonlinearity of the form

$$y(t) = g(u(t)), \tag{1}$$

for a continuously-differentiable function $g : \mathbb{R}^m \to \mathbb{R}^m$, and (ii) the target reference $\{r(t) : t \in [0, \infty)\}$ is a constant, $r(t) \equiv r$ for a given $r \in \mathbb{R}^m$.[1] These assumptions will be relaxed later. In this case, the tracking controller is defined by the following equation,

$$\dot{u}(t) = \Big(\frac{\partial g}{\partial u}(u(t))\Big)^{-1}\big(r - y(t)\big), \tag{2}$$

assuming that the Jacobian matrix $\frac{\partial g}{\partial u}(u(t))$ is nonsingular at every point $u(t)$ computed by the controller via (2). Observe that (2) defines the Newton-Raphson flow for solving the algebraic equation $r - g(u) = 0$, and hence (see [19]) the controller converges in the sense that $\lim_{t\to\infty}\big(r(t) - y(t)\big) = 0$. Next, suppose that the reference target is time-dependent, while keeping the assumption that the plant is a memoryless nonlinearity. Suppose that $\{r(t)\}$ is bounded, continuous, piecewise-continuously differentiable, and $\{\dot{r}(t)\}$ is bounded. Define

$$\eta := \limsup_{t\to\infty} ||\dot{r}(t)||, \tag{3}$$

then (see [19]), with the controller defined by (2), we have that

$$\lim_{t\to\infty} ||r(t) - y(t)|| \leqslant \eta. \tag{4}$$

Note that Eqs. (1) and (2) together define the closed-loop system as a dynamical system in the variable $\{u(t)\}$. Its stability, in the sense that $\{u(t)\}$ and $\{y(t)\}$ are bounded whenever $\{r(t)\}$ and $\{\dot{r}(t)\}$ are bounded, is guaranteed by (4) as long as the control trajectory $\{u(t)\}$ does not pass through a point $u(t)$ where the Jacobian matrix $\frac{\partial g}{\partial u}(u(t))$ is singular.

Finally, let us dispense with the assumption that the plant subsystem is a memoryless nonlinearity. Instead, suppose that it is a dynamical system modeled by the following two equations,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) := x_0 \tag{5}$$

$$y(t) = h(x(t)), \tag{6}$$

[1]Henceforth we will use the notation $\{x(t)\}$ For a generic signal $\{x(t),\ t \in [0, \infty)\}$, to distinguish it from its value at a particular point t, $x(t)$.

where the state variable $x(t)$ is in $\mathbb{R}^n$, and the functions $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ satisfy the following assumption.

Assumption 1. (i). The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is continuously differentiable, and for every compact set $\Gamma \subset \mathbb{R}^m$ there exists $K \in \mathbb{R}^+$ such that, for every $x \in \mathbb{R}^n$ and $u \in \Gamma$, $||f(x, u)|| \leqslant K\big(||x|| + 1\big)$. (ii). The function $h : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable. □

This assumption ensures that whenever the control signal $\{u(t)\}$ is bounded and continuous, the state equation (5) has a unique solution $x(t)$ on the interval $t \in [0, \infty)$.

In this setting, $y(t)$ is no longer a function of $u(t)$, but rather of $x(t)$ which is a function of $\{u(\tau) : \tau < t\}$. Therefore (1) is no longer valid, and hence the controller cannot be defined by (2). To get around this conundrum we pull the feedback not from the output $y(t)$ but from a predicted value thereof. Specifically, fix a look-ahead time $T \in \mathbb{R}^+$, and suppose that at time t the system computes a prediction of $y(t + T)$, denoted by $\tilde{y}(t + T)$. Suppose also that $\tilde{y}(t + T)$ is a function of $(x(t), u(t))$, hence can be written as $\tilde{y}(t + T) = g(x(t), u(t))$, where the function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be continuously differentiable.

Now the feedback law is defined by the following equation,

$$\dot{u}(t) = \Big(\frac{\partial g}{\partial u}(x(t), u(t))\Big)^{-1}\big(r(t + T) - g(x(t), u(t))\big). \tag{7}$$

The state equation (5) and control equation (7) together define the closed-loop system. This system can be viewed as an $(n + m)$-dimensional dynamical system with the state variable $(x(t)^{\mathrm{T}}, u(t)^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{n+m}$ and input $r(t) \in \mathbb{R}^m$. We are concerned with a variant of Bounded-Input-Bounded-State (BIBS) stability whereby if $\{r(t)\}$ and $\{\dot{r}(t)\}$ are bounded then $\{x(t)\}$ and $\{u(t)\}$ are bounded as well. Such stability no-longer can be taken for granted as in the case where the plant is a memoryless nonlinearity.

We remark that a larger T means larger prediction errors, and these translate into larger asymptotic tracking errors. On the other hand, analyses of various second-order systems in [7] reveals that they all were unstable if T is too small, and stable if T is large enough. Therefore, a requirement for a restricted prediction error stands in contradiction with the stability requirement. This issue was resolved by speeding up the controller in the following manner. Consider $\alpha > 1$, and modify (7) by multiplying its right hand side by α, resulting in the following control equation:

$$\dot{u}(t) = \alpha\Big(\frac{\partial g}{\partial u}(x(t), u(t))\Big)^{-1}\big(r(t + T) - g(x(t), u(t))\big).$$

It was verified in [7], [8] on several examples that regardless of the value of $T \in \mathbb{R}^+$, a large-enough α stabilizes the closed-loop system; this statement seems to have a broad scope and does not require the plant to be a minimum-phase system. Furthermore, if the closed-loop system is stable then

the following bound holds (see [19]),

$$\limsup_{t\to\infty} ||r(t) - \tilde{y}(t)|| \leqslant \frac{\eta}{\alpha},$$

where η is defined by (3). Thus, a large gain α can stabilize the closed-loop system and reduce the asymptotic tracking error.

B. Problem Formulation

The application example consists of controlling the trajectory of a planar Dubins vehicle providing a dynamic model for fixed-wing aircraft at a constant elevation [20]. The model is a three-dimensional dynamical system of the form

$$\begin{aligned}
\dot{z}_1^p(t) &= V^p \cos\theta^p(t),\\
\dot{z}_2^p(t) &= V^p \sin\theta^p(t),\\
\dot{\theta}^p(t) &= u(t),
\end{aligned}$$

where $(z_1^p(t), z_2^p(t))^{\mathrm{T}}$ denotes the planar position of the vehicle, $\theta^p(t)$ its heading, and $u(t)$ is the angular acceleration. The vehicles speed, V^p, is a given positive constant. $u(t)$ satisfies the constraint $\|u(t)\| \leqslant u_{\max}$ for a given $u_{\max} > 0$, and this enforces the minimum turning radius of $V^p/u_{\max}$. In the considered control application such a vehicle, henceforth referred to as the pursuer, is tasked with tracking an evading vehicle. The evader's motion has the following single (two-dimensional) integrator model,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} z_1^{\mathrm{e}}(t)\\ z_2^{\mathrm{e}}(t)\end{bmatrix} = \begin{bmatrix} V^{\mathrm{e}}\cos\theta^{\mathrm{e}}\\ V^{\mathrm{e}}\sin\theta^{\mathrm{e}}\end{bmatrix},$$

where $(z_1^e(t), z_2^e(t))^\top$ denote the planar position of the evader, and V^e is its speed, assumed a constant smaller than V^p. We consider two cases; one where the evader is agnostic to the pursuer and follows a known trajectory, and the other where the evader is adversarial in nature and its trajectory is not known to the pursuer. The next section will provide two solutions for the problem of estimating the evader's trajectory based, respectively, on a model-based approach and a learning-based approach.

III. Predictive Framework

A. Model-Based Pursuit Evation

The considered system is underactuated because the two-dimensional position of each pursuer , $(z_1^p(t), z_2^p(t))^\top$, is controlled by a one-dimensional input, $u(t)$. In order to apply the tracking framework described in Subsection II A, which requires equal dimensions, we define a suitable function $F : R^2 \to R^+$ and set $g(x(t), u(t)) := \int_t^{t+T} F(\tilde{y}^p(\tau) - \tilde{y}^e(\tau))\mathrm{d}\tau$, where $\tilde{y}^p(\tau)$ and $\tilde{y}^e(\tau)$ are the predicted positions of the pursuer and the evader at time τ; we then apply the Newton-Raphson flow to the equation $g(x(t), u(t)) = 0$. The modified controller becomes

$$\dot{u}(t) = -\alpha\Big(\frac{\partial g}{\partial u}(x(t), u(t))\Big)^{-1}\big(g(x(t), u(t))\big),\ t \geqslant 0. \quad (8)$$

Since $g(x, u)$ is a scalar, the modified algorithm works similarly to the base case described in Section II.

Assume general nonlinear system dynamics as in Eqs. (5) - (6). The predicted state trajectory, $\xi(\tau)$, $\tau \in [t, t+T]$, is defined by the state equation (6) with a constant input, $u(\tau) \equiv u(t)$, and the initial state $\xi(t) = x(t)$. Thus,

$$\dot{\xi}(\tau) = f(\xi(\tau), u(t)),\ \tau \in [t, t+T], \quad (9)$$

with the initial condition $\xi(t) = x(t)$. The predicted output at $\tau = t+T$ is defined as $\tilde{y}^p(\tau) = h(\xi(\tau))$. Furthermore, by taking the partial derivative of (9) with respect to u(t), we obtain

$$\frac{\partial\dot{\xi}}{\partial u(t)}(\tau) = \frac{\partial f}{\partial \xi}(\xi(\tau), u(t))\frac{\partial \xi}{\partial u(t)}(\tau) + \frac{\partial f}{\partial u}(\xi(\tau), u(t)),$$

with the initial condition $\frac{\partial \xi}{\partial u(t)}(t) = 0$. This is a differential equation in $\frac{\partial \xi}{\partial u(t)}(\tau)$; $\tau \in [t, t+T]$, which can be solved numerically concurrently with (11). Eqs. (11) and (12) give $g(x(t), u(t))$ and $\frac{\partial g}{\partial u}(x(t), u(t))$ thereby providing all the elements required for the control law defined by (10).

In the adversarial problem formulation the trajectory of the evader is not known in advance, which can be overcome in two ways. In the first approach, the pursuer(s) use game theory to predict the approximate direction of evasion. In the case of a single pursuer, [21] proved that the evader's optimal strategy is to move away from the pursuer in the opposite direction from it if the distance between the two is greater than the pursuer's minimum turning radius, and in a perpendicular direction from the line connecting them if the distance is reduced to the minimum turning radius. The problem-setting described below somewhat modifies this setting by having the evader move towards a given stationary goal instead of straight away from the pursuer, but we keep the same non-holonomic evasive manoeuvre as in [21]. In the case of multiple pursuers, it is assumed that the evader applies the same strategy only to one pursuer at a time, the one closest to it.

The second approach involves learning the evader's behavior over time using NN. The pursuers take their positions and the position of the evader as input, and the NN gives the estimated evasion direction as the output.

We consider a pursuit evasion problem involving two pursuing agents, but possible extensions to scenarios involving more pursuers are clear from the presentation. Such problems are typically formulated as zero-sum differential games [21]. Instead of solving the Hamilton-Jacobi-Isaacs (HJI) equations formulated in [22], we apply the tracking technique described in Subsection II-A, augmented by learning structures, to approximate the desired behavior.

In order to formulate the pursuit evasion problem, we define a global state space system consisting of the dynamics of the pursuers and the evader. The global state dynamics

become,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} z_1^{\mathrm{p}_1}(t) \\ z_2^{\mathrm{p}_1}(t) \\ \theta^{\mathrm{p}_1}(t) \\ z_1^{\mathrm{p}_2}(t) \\ z_2^{\mathrm{p}_2}(t) \\ \theta^{\mathrm{p}_2}(t) \\ z_1^{\mathrm{e}}(t) \\ z_2^{\mathrm{e}}(t) \end{bmatrix} = \begin{bmatrix} V^{\mathrm{p}_1}\cos\theta^{\mathrm{p}_1} \\ V^{\mathrm{p}_1}\sin\theta^{\mathrm{p}_1} \\ u_1^{\mathrm{p}} \\ V^{\mathrm{p}_1}\cos\theta^{\mathrm{p}_2} \\ V^{\mathrm{p}_2}\sin\theta^{\mathrm{p}_2} \\ u_2^{\mathrm{p}} \\ V^{\mathrm{e}}\cos\theta^{\mathrm{e}} \\ V^{\mathrm{e}}\sin\theta^{\mathrm{e}} \end{bmatrix}, \tag{10}$$

where the superscripts indicate the autonomous agents. For compactness, we denote the global state vector by $x(t) \in \mathbb{R}^8$, and the pursuers' control vector by $u(t) \in \mathbb{R}^2$ Thus, given the initial states of the agents $x_0 \in \mathbb{R}^8$, the evolution of the pursuit evasion game is described by $\dot{x}(t) = f(x(t), u, u_{\mathrm{e}})$, $x(0) = x_0$, $t \geqslant 0$.

Subsequently, this zero-sum game can be described as a minimax optimization problem through the cost index,

$$\begin{aligned} J(x, u, u_{\mathrm{e}}) &= \int_0^\infty e^{-\gamma t} L(x)\mathrm{d}t \\ &:= \int_0^\infty e^{-\gamma t}\left(\beta_1(d_1^2 + d_2^2) + \beta_2 \frac{d_1^2 d_2^2}{d_1^2 + d_2^2}\right)\mathrm{d}t, \end{aligned} \tag{11}$$

where $d_i = \sqrt{(z_1^i - z^{\mathrm{e}})^2 + (z_2^i - z_2^{\mathrm{e}})^2}$, $i \in \{\mathrm{p}_1, \mathrm{p}_2\}$ is the distance between the i-th pursuer and the evader, $\beta_1,\ \beta_2 \in \mathbb{R}^+$ are user defined contants, and $\gamma \in \mathbb{R}^+$ is a discount factor. The first term ensures that the pursuers remain close to the evader, while the second term encourages cooperation between the agents. The cost decreases exponentially to ensure that the integral has a finite value in the absence of equilibrium points.

Let $V(x) : \mathbb{R}^8 \to \mathbb{R}$ be a smooth function quantifying the value of the game when specific policies are followed starting from state $x(t)$. Then, we can define the corresponding Hamiltonian of the game as

$$H\big(x, u, u_e, \frac{\partial V}{\partial x}\big) = L(x) + \frac{\partial V}{\partial x}^{\mathrm{T}} f(x, u, u_e) + \gamma V. \tag{12}$$

The optimal feedback policies of the pursuer and evader of this game are $u^\star(x)$, $u_e^\star(x)$ of this game are known to constitute a saddle point [22] such that,

$$u^\star(x) = \arg\min_u H(x, u, u_e), \tag{13}$$

$$u_e^\star(x) = \arg\max_{u_e} H(x, u, u_e). \tag{14}$$

Under these optimal policies, the following HJI equation is satisfied,

$$H\big(x, u^\star, u_e^\star, \frac{\partial V^\star}{\partial x}\big) = 0. \tag{15}$$

Evaluating the optimal pursuit policies, yields the singular optimal solutions described by, $V_{\theta_{p1}} u_1 = V_{\theta_{p2}} u_2 = 0$, where V_{x_i} is the partial derivative of the value function with respect to the state x_i, calculated by solving (15). To obviate the need for bang-bang control, as is derived by (13) and (14) we shall employ the predictive tracking technique described in Section II-A to derive approximate, easy to implement, feedback controllers for the pursuing autonomous agents. Furthermore, by augmenting the predictive controller with learning mechanisms, the approximate controllers will have no need for explicit knowledge of $u_e^\star(x)$, the evader's policy.

The following theorem presents bounds on the optimality loss induced by the use of the look-ahead controller approximation.

Theorem 1. Let the pursuit evasion game evolve according to the dynamics given by (10), where the evader is optimal with respect to (11) and the pursuers utilize the learning-based predictive tracking strategy given (8). Then, the tracking error of the pursuers and the optimality loss due to the use of the predictive controller are bounded if $\exists \bar{\Delta} \in \mathbb{R}^+$, such that, $\Delta(x(t), \hat{u}(t), \hat{u}(t)_{\mathrm{e}}) \leqslant \bar{\Delta},\ \forall t \geqslant 0$, where $\Delta(x, \hat{u}, \hat{u}_{\mathrm{e}}) = V_{x_{\mathrm{e}}} v_{\mathrm{e}}(\cos\hat{u}_{\mathrm{e}} - \cos u_{\mathrm{e}}^\star) + V_{y_{\mathrm{e}}} v_{\mathrm{e}}(\sin\hat{u}_{\mathrm{e}} - \sin u_{\mathrm{e}}^\star) + V_{\theta_{\mathrm{p}}1}(u_1^\star - \hat{u}_1) + V_{\theta_{\mathrm{p}}2}(u_2^\star - \hat{u}_2)$, with V_ξ denoting the partial derivative of the game value with respect to the state component $\xi(t)$.

Proof: Consider the Hamiltonian function when the approximate controller, denoted $\hat{u}(t)$ and the NN-based prediction of the evader's policy, $\hat{u}_{\mathrm{e}}(t)$ are used,

$$H(x, \hat{u}, \hat{u}_{\mathrm{e}}) = L(x) + \big(\frac{\partial V}{\partial x}\big)^{\mathrm{T}} f(x, \hat{u}, \hat{u}_{\mathrm{e}}) + \gamma V. \tag{16}$$

Taking into account the nonlinear dynamics of the system (10), one can rewrite (16) in terms of the optimal Hamiltonian as,$H(x, \hat{u}, \hat{u}_{\mathrm{e}}) = H(x, u^\star, u_{\mathrm{e}}^\star) + \Delta(\hat{u}, \hat{u}_{\mathrm{e}})$, where $H(x, u^\star, u_{\mathrm{e}}^\star) = 0$ is the HJI equation that is obtained after substituting (13) and (14) in (12). Now, take the orbital derivative of the value function along the trajectories using the approximate controllers as, $\dot{V} = \big(\frac{\partial V}{\partial x}\big)^{\mathrm{T}} f(x, \hat{u}, \hat{u}_{\mathrm{e}})$. Substituting (16) yields $\dot{V} = -L(x) - \gamma V + \Delta(x, \hat{u}, \hat{u}_{\mathrm{e}})$. Thus, since $L(x) > 0$, $\forall x \in \mathbb{R}^8 \backslash \{0\}$,

$$\dot{V} < -\gamma V + \Delta(x, \hat{u}, \hat{u}_{\mathrm{e}}) \Rightarrow \dot{V} < -\gamma V + \bar{\Delta}.$$

Hence for $V \geqslant \bar{\Delta}/\gamma$, we have $\dot{V} \leqslant 0$. Thus $\{x \in \mathbb{R}^8 \mid V(x) \leqslant \bar{\Delta}/\gamma\}$ is a forward invariant set, which implies that the tracking error and the optimality loss over any finite horizon is bounded. ■

B. Deep Learning-Based Pursuit Evasion

A deep NN, consisting of $L > 2$ hidden layers, describes a nonlinear mapping between its input space $\mathbb{R}^n$ and output space $\mathbb{R}^p$. Each layer receives the output of the previous layer as an input and, subsequently, feeds its own output to the next layer. Each layer's output consists of the weighted sum of its input alongside a bias term, filtered through an application-specific activation function [11].

Specifically, let $\mathbb{R}^{n_l}$ be the input space of a specific layer, and $\mathbb{R}^{p_l}$ the corresponding output space. Then the layer's output is,

$$Y_i(x) = \sigma\Big(\sum_{j=1}^{n_l} v_{ij} X_j + v_{i0}\Big),\ i = 1, 2, \ldots, p_l,$$

where $X' = \begin{bmatrix} X_1 & \ldots & X_{n_l} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{n_l}$ is the input vector, gathered from training data or from the output of previous

layers, $v_{ij} \in \mathbb{R}$ is a collection of n_l weights for each layer, $v_{i0} \in \mathbb{R}$ the bias term and $\sigma : \mathbb{R}^{n_l} \to \mathbb{R}$ is the layer's activation function. We note that it is typical to write the output of layer compactly, with slight abuse of notation, as,

$$Y = \sigma(W^{\mathrm{T}}\sigma'(X)),$$

where $Y = \begin{bmatrix} Y_1 & \dots & Y_{p_l} \end{bmatrix} \in \mathbb{R}^{p_l}$, $W = \begin{bmatrix} v_{ij} \end{bmatrix} \in \mathbb{R}^{(n_l+1)\times p_l}$ and $\sigma' : \mathbb{R}^{n_l'} \to \mathbb{R}^{n_l}$ is the activation function of the previous layer, taking as input the vector $X = \begin{bmatrix} X'^{\mathrm{T}} & 1 \end{bmatrix}^{\mathrm{T}}$.

It is known [23], that two-layer NNs possess the universal approximation property, according to which, any smooth function can be approximated arbitrarily close by an NN of two or more layers. Let $\mathbb{S} \subset \mathbb{R}^n$ be a simply connected compact set and consider the nonlinear function $\kappa : \mathbb{S} \to \mathbb{R}^p$. Given any $\epsilon_b \geqslant 0$, there exists a NN such structure such that,

$$\kappa(x) = \sigma\big(W^{\mathrm{T}}\sigma'(x)\big) + \epsilon, \ \forall x \in \mathbb{S},$$

where $\|\epsilon\| \leqslant \epsilon_b$. We note that, typically, the activation function of the output layer $\sigma(\cdot)$ is taken to be linear.

Evaluating the weight matrix W in a network is the main concern of the area of machine learning. In this work, we employ the gradient descent based backpropagation algorithm. Given a collection of N_d training data, stored in the tuple $\{x_k, \kappa_k\}_k$, where $x_k \in \mathbb{R}^n$, $\kappa_k \in \mathbb{R}^p$, $\forall k = 1, \dots, N_d$, we denote the output errors as $r_k = \kappa(x_k) - \kappa_k$. Then, the update equation for the weights at each optimization iteration t_k is given by,

$$w_{ij}(t_k + 1) = w_{ij}(t_k) - \eta \frac{\partial (r_k^{\mathrm{T}} r_k)}{\partial w_{ij}}, \ \forall t_k \in \mathbb{N},$$

where $\eta \in \mathbb{R}^+$ denotes the learning rate. We note that the update index t_k need not correspond to the sample index k, since different update schedules leverage the gathered data in different ways [23]. It can be seen that in order for the proposed method to compute the pursuers' control inputs, an accurate prediction of the future state of the evader is required. However, this presupposes that the pursuers themselves have access to the evader's future decisions; an assumption that is, in most cases, invalid. Thus, we augment the pursuers' controllers with a NN structure, that learns to predict the actions of the evader, based on past recorded data.

Initially, we assume that the evader's strategy is computed by a feedback algorithm, given her relative position to the pursuers. This way, the unknown function we wish to approximate is $f : \mathbb{R}^{2N} \to \mathbb{R}^2$, with, $u^e = f(\delta z_1^{p_1}, \delta z_2^{p_1}, \dots, \delta z_1^{p_N}, \delta z_2^{p_N})$, where, $(\delta z_1^{p_i}, \delta z_2^{p_i})$ denote the distance of pursuer i to the evader in the X and Y axes, respectively. In order to train the network, we let the pursuers gather data regarding the fleet's position with respect to the evader, as well as her behavior over a predefined time window $T_l > 0$. We point out that the choice of T_l comprises a balance between the effectiveness of the learning procedure and its computational efficiency.

Subsequently, we denote by $\hat{u}^e(x)$, the current prediction function for the evader's strategy, i.e., $\hat{u}^e(x) = \sigma\big(\hat{W}^{\mathrm{T}}\hat{\sigma}'(\chi)\big)$, where $\chi = \begin{bmatrix} \delta z_1 & \delta y_1 & \dots & \delta x_N & \delta y_N \end{bmatrix} \in \mathbb{R}^{2N}$, $\hat{W}$ denotes the current weight estimate of the NNs output layer, and $\hat{\sigma}'(\cdot)$ is the current estimate of the hidden layers, parametrized by appropriate hidden weights. We remark that the learning algorithm for the evader's behavior is implemented sequentially, in batches (rather than continuous time) throughout the duration of the pursuit.

IV. Simulation Results

This section presents results for the problems described in the previous section. First, the agnostic evader case is considered followed by the adversarial case. For the second case, single and multiple pursuer systems are considered separately. The controller is implemented on a Dubins vehicle. For the purpose of tracking, we define the system output to be $y^i = \begin{bmatrix} z_1^i & z_2^i \end{bmatrix}^{\mathrm{T}}$, $i \in \{p_1, p_2, e\}$.

A. Single Pursuer - Agnostic Target

In this subsection an agnostic evader moves along a known trajectory without practicing an evasion strategy. In the simulation we set the pursuer's speed to $V^p = 2$ m/s, and its input saturation to 2π rad/s. The evader moves along two semicircular curves with a constant speed of 0.4π m/s.

Figure 2 depicts the trajectories of the pursuer and evader. It is seen that whenever the pursuer catches up with the evader, it goes around a whole circle at its minimum turning radius; this is due to the fact that it has a constant speed. A graph of the tracking error vs. time is shown in Figure 3, and its periodic nature is due to the circles along its trajectory.

A better tracking performance can be obtained if the pursuer has a smaller minimum turning ratio. We tried that on a simulation where the saturation input is $\frac{\pi}{2}$ rad/s, a quarter of what it was in the previous example. The results, not shown here but depicted in [18], exhibit radii and maximum errors at 25% of those shown in Figure 2 and Figure 3.

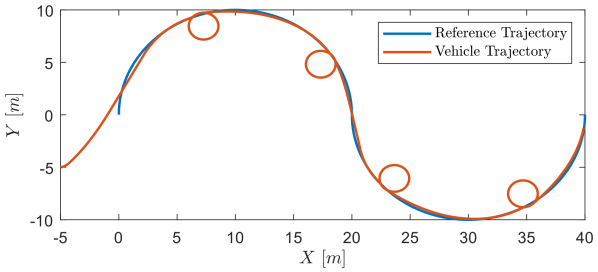

Fig. 2. Agnostic evader.

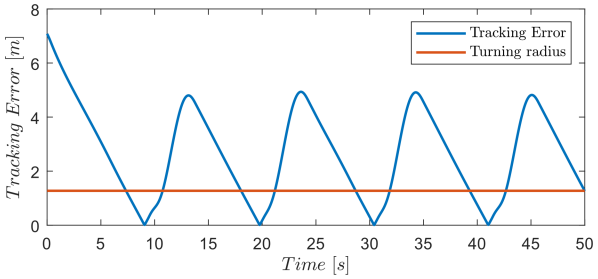

Fig. 3. Evolution of an agnostic evader tracking error.

B. Single Pursuer - Adversarial Evader

The pursuer is again modelled as a Dubins vehicle, while the evader is modelled as a single integrator with a maximum velocity less than the speed of the pursuer. Hence, while the pursuer is faster, the evader is more agile, and can instantly change its direction of motion. In this and subsequent cases, the evader is considered adversarial in nature and uses game theory to choose evasion direction.

Let $y^p(t)$ and $y^e(t)$ be the position vector of the pursuer and evader respectively at time t. First, the pursuer makes an estimate of the optimal evasion direction based on the relative position of the evader and itself at time t using its knowledge of the evader's game. Assuming this direction of evasion to be fixed over the prediction window from t to $t+T$ gives the predicted position of the evader at all time instances in this interval, denoted as $\tilde{y}^e(\tau), \tau \in [t, t+T]$. Next, the pursuer estimates its own predicted position if its input is kept constant, called $\tilde{y}^p(\tau), \tau \in [t, t+T]$. Finally, $g(t)$ is set as $||\tilde{y}^e(t+T) - \tilde{y}^p(t+T)||^2$ and the value of $\frac{\partial g}{\partial u}(x(t), u(t))$ ($x(t)$ being the ensemble vector of the states of the pursuer and the evader) is used to compute the input differential equation (8).

Figures 4 shows the trajectories of the pursuer and the evader, with the goal for the evader set to the point $(150, 60)$. It can be observed that the evader moves towards the goal while the pursuer is far away and starts evasive maneuvers when it gets close to it, by entering its non-holonomic region. Figure 5 displays the tracking error, defined as the distance between the pursuer and the evader, which is almost periodic. This is because the evader's maneuver forcing the pursuer to circle back. The peak tracking error after the pursuer catches up is slightly more than twice the turning radius, as expected.

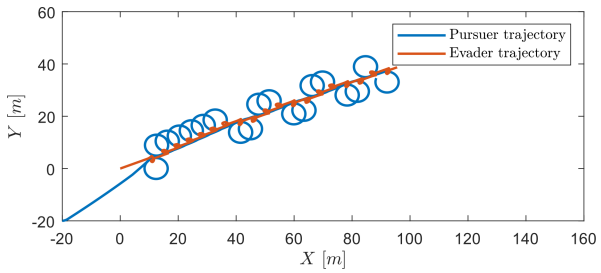

Fig. 4. Trajectories for a single pursuer-evader system.

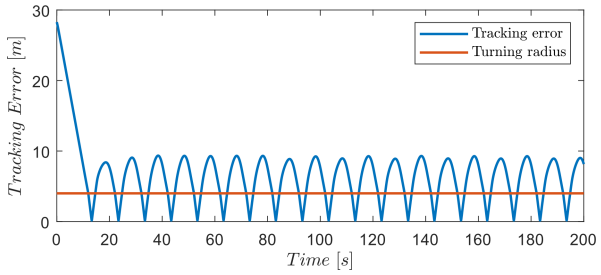

Fig. 5. Evolution of the tracking error for a single pursuer-evader system.

C. Multiple Pursuers - Adversarial Evader

Consider the case of two pursuers and a single evader. Having multiple pursuers requires a cooperation between them for an effective utilization of resources. Thus, a pursuer can no longer make decisions solely based on the position of the evader relative to itself, but it must take into account the positions of the other pursuers. To achieve that, we redefine the expression for $g(x, u)$ as shown below for the case of two pursuers. Let d_{p} be the distance between the two pursuers, and let

$$g(x(t), u(t)) := \int_t^{t+T} \Big\{ \beta_1 (d_1^2(\tau) + d_2^2(\tau)) + \beta_2 \frac{d_1^2(\tau) d_2^2(\tau)}{d_1^2(\tau) + d_2^2(\tau)} + \beta_3 e^{-\gamma d_{\text{p}}(\tau)} \Big\} \mathrm{d}\tau, \ \forall t \geqslant 0.$$

The first term ensures that the pursuers remain close to the evader, while the second term encourages cooperation between agents [9]. The last term is added to repel pursuers apart if they come close to each other.

Figure 6 shows the trajectories of the pursuers and the evader when the goal for the evader is set to the point $(15, -1)$. In this case, the pursuers close in on the evader and trap it away from its goal due to their cooperative behavior. The evader is forced to continuously perform an evasive maneuver on one pursuer at a time while the other pursuer makes a turn. This can be seen more clearly in the tracking error plot given in Figure 7, where after catching up with the evader, one pursuer is at its maximum distance when the other one is close to its minimum distance. This result, reflecting on the coordination between the pursuers, is qualitatively comparable to those obtained in [9].

Lastly, we present the results under the learning-based prediction described in Section III. Figure 8 depicts the trajectories of the pursuers and evader, and Figure 9 presents a comparative result of the tracking errors of the model-based algorithm vis-à-vis the NN-based control. Figure 10 compares the performance quality defined by (11) for the NN-based control and the model-based control. From these figures, it can be seen that the NN structure offers predictive capabilities to the controller, and a comparable tracking-performance to the model based control.

V. Conclusion and Future Work

This work extends the framework of prediction-based nonlinear tracking in the context of pursuit evasion games. We present results for vehicle pursuit of agnostic targets, modeled as moving along known trajectories, as well as adversarial target tracking, where the evader evolves according to game-theoretic principles. Furthermore, to obviate the need for explicit knowledge of the evader's strategy, we employ learning algorithms alongside the predictive controller. The overall algorithm is shown to produce comparable results to those in the literature, while it precludes the need for solving optimal control problems in real time.

Future work will focus on developing robustness guarantees will allow for more realistic scenarios, where noise and external disturbances are taken into consideration.

References

[1] S. Devasia, D. Chen, and B. Paden, "Nonlinear inversion-based output tracking," *IEEE Transactions on Automatic Control*, vol. 41, no. 7, pp. 930–942, 1996.

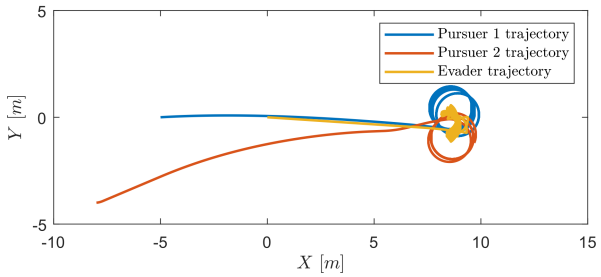

Fig. 6. Trajectories for the two pursuer-single evader system.

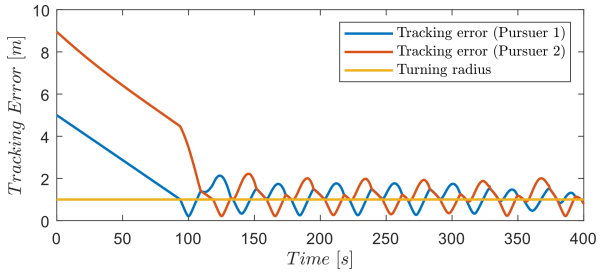

Fig. 7. Evolution of the tracking error for the two pursuer-single evader system.

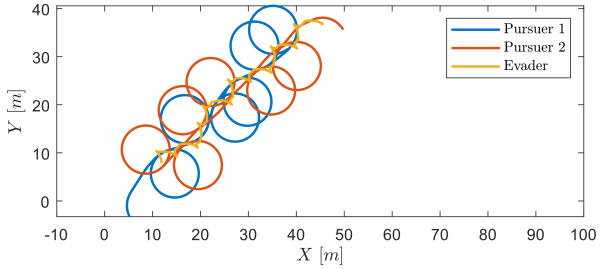

Fig. 8. Trajectories for two pursuers-single evader system with learning.

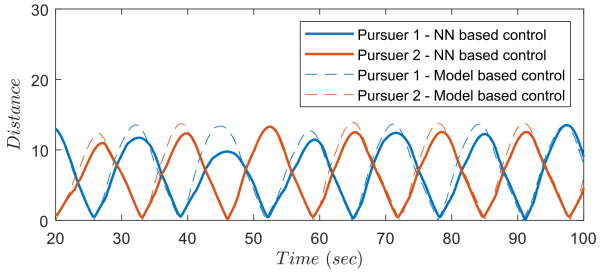

Fig. 9. Evolution of the tracking error for the systems with and without learning.

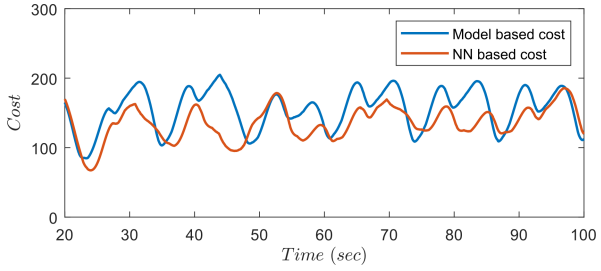

Fig. 10. Total cost for the system with and without learning.

[2] P. Martin, S. Devasia, and B. Paden, "A different look at output tracking: control of a vtol aircraft," *Automatica*, vol. 32, no. 1, pp. 101–107, 1996.

[3] A. Isidori and C. Byrnes, "Output regulation of nonlinear systems," *IEEE Trans. Automat. Control*, vol. 35, pp. 131–140, 1990.

[4] H. Khalil, "On the design of robust servomechanisms for minimum phase nonlinear systems," *Proc. 37th IEEE Conf. Decision and Control,* Tampa, FL, pp. 3075–3080, 1998.

[5] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.

[6] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design, 2nd Edition*. Nob Hill, LLC, 2017.

[7] Y. Wardi, C. Seatzu, M. Egerstedt, and I. Buckley, "Performance regulation and tracking via lookahead simulation: Preliminary results and validation," in *56th IEEE Conf. on Decision and Control,* Melbourne, Australia, December 12-15, 2017.

[8] S. Shivam, I. Buckley, Y. Wardi, C. Seatzu, and M. Egerstedt, "Tracking control by the newton-raphson flow: Applications to autonomous vehicles," in *European Control Conference,* Naples, Italy, June 25-28, 2018.

[9] S. A. Quintero, D. A. Copp, and J. P. Hespanha, "Robust uav coordination for target tracking using output-feedback model predictive control with moving horizon estimation," in *American Control Conference,* Chicago, Illinois, July 1-3, 2015.

[10] G. Saridis, "Intelligent robotic control," *IEEE Transactions on Automatic Control*, vol. 28, no. 5, pp. 547–557, 1983.

[11] S. S. Haykin, *Neural networks and learning machines*. Pearson Upper Saddle River, 2009, vol. 3.

[12] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal adaptive control and differential games by reinforcement learning principles*. IET, 2013, vol. 2.

[13] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[14] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Systems & Control Letters*, vol. 100, pp. 14–20, 2017.

[15] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2009, pp. 140–147.

[16] T. Alpcan and T. Başar, *Network security: A decision and game-theoretic approach*. Cambridge University Press, 2010.

[17] H. J. Pesch, I. Gabler, S. Miesbach, and M. H. Breitner, "Synthesis of optimal strategies for differential games by neural networks," in *New Trends in Dynamic Games and Applications*. Springer, 1995, pp. 111–141.

[18] S. Shivam, A. Kanellopoulos, K. Vamvoudakis, and Y. Wardi, "A predictive deep learning approach to output regulation: The case of collaborative pursuit evasion," in *Arxiv,* https://arxiv.org/abs/1909.00893, 2019.

[19] Y. Wardi, C. Seatzu, and M. Egerstedt, "Tracking control via variable-gain integrator and lookahead simulation: Application to leader-follower multiagent networks," in *Sixth IFAC Conference on Analysis and Design of Hybrid Systems (2018 ADHS)l,* Oxford, the UK, July 11-13, 2018.

[20] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[21] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[22] T. Basar and G. J. Olsder, *Dynamic noncooperative game theory*. Siam, 1999, vol. 23.

[23] F. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.