Reservoir Computing for Drone Trajectory Intent Prediction: A Physics Informed Approach

Adolfo Perrusquía, *Member, IEEE*, and Weisi Guo, *Senior Member, IEEE*

***Abstract*—The design of accurate trajectory prediction algorithms is crucial to implement adequate countermeasures against drones with anomalous performances. Wrong predictions may cause high-false-positives that compromise safety in national infrastructures. In this article, a physics informed reservoir computing (PIRC) scheme for drone trajectory prediction is proposed. The approach is comprised of two main complementary learning algorithms that enhance the prediction and generalization capabilities: 1) a standard reservoir computing scheme for high-dimensional encoding exploitation and 2) a nonlinear control scheme that gives a physical feedback to the reservoir weights to ensure the prediction error is minimized. The nonlinear control scheme is modeled by the prediction error dynamics and a feedback linearization controller. Two different PIRC schemes are proposed which preserve the reservoir properties and enhance the prediction robustness. Lyapunov stability theory is used to verify the boundedness and convergence of the proposed algorithms. Simulation studies and comparisons are given to verify the proposed approach.**

***Index Terms*—Drones, nonlinear control, physics informed model, reservoir computing (RC), trajectory prediction.**

I. Introduction

DRONE detection and prediction has become paramount in the last decade due to the proliferation of cheaper drone technology that magnifies the threat in the airspace [1], [2], [3]. In terms of intent prediction, two classes can be distinguished: 1) high-level intent and 2) trajectory intent. Whilst high-level intent defines the purpose of use of the drone (e.g., surveillance, delivery, etc.), trajectory intent defines the mission profile that the drone aims to achieve [4], [5], [6]. In this article, we will focus on trajectory intent prediction. Several technologies have been developed to address the challenge of predicting the future trajectory intent which encompasses imagery data, time-series data, and either physics informed or data-driven learning models [7], [8], [9].

Manuscript received 25 August 2023; revised 12 November 2023, 14 January 2024, and 28 February 2024; accepted 16 March 2024. Date of publication 4 April 2024; date of current version 23 August 2024. This work was supported in part by the Engineering and Physical Sciences Research Council under Grant EP/V026763/1, and in part by the Royal Academy of Engineering and the Office of the Chief Science Adviser for National Security under the U.K. Intelligence Community Postdoctoral Research Fellowship Programme. This article was recommended by Associate Editor H. Takagi. *(Corresponding author: Adolfo Perrusquía.)*

The authors are with the School of Aerospace, Transport and Manufacturing, Cranfield University, MK43 0AL Bedford, U.K. (e-mail: adolfo.perrusquia-guzman@cranfield.ac.uk; weisi.guo@cranfield.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2024.3379381.

Digital Object Identifier 10.1109/TCYB.2024.3379381

Data obtained from radar, lidar or GPS are generally used to predict the future trajectory of the drone's hidden mission profile; specifically, position and linear velocity measurements [10], [11]. One of the simplest trajectory prediction algorithm consists in the design of simple state transition models to estimate future trajectories one-step ahead; which are common in state estimator techniques, such as Kalman filters and their variants [12], [13], [14]. However, the simplistic nature of this model can lead to biased predictions and they are not useful for long-term predictions. This issue has been addressed by different authors using several machine learning techniques which are briefly discussed as follows.

A. Related Work

Machine Learning models are widely used for regression tasks and thus, for prediction [15]. Linear or nonlinear regression architectures are data driven methods that are commonly used for prediction. In the linear case, a set of polynomial basis functions are used to adjust the model throughout the input data which can be computational expensive due to its solution in the least-squares sense or, equivalently, to the curse of dimensionality [16]. On the other hand, support vector machines (SVMs) can overcome this issue by using the Kernel trick that enables nonlinear regression tasks [17]. Here, SVM is a data-hungry algorithm that requires large amount of data to construct an adequate predictive model. However, drone's trajectories are, in most cases, high-nonlinear such that predicting its future trajectory is difficult in a low-dimensional feature space.

Neural networks and deep learning architectures have also been used in the literature for regression tasks with interesting results [5], [7]. The main advantage of these models is the extraction of high-dimensional features of the time-series data to increase the prediction precision. Some of the most famous approaches are: multilayer perceptron (MLP) [18], residual network (ResNet) [19], fully and multiscale convolutional neural network (FCN, MCNN) [20], [21], recurrent neural networks (RNNs), and reservoir computing (RC). Here, feedforward architectures, such as MLP, ResNet, FCN, and MCNN, are suitable for nontemporal data since they cannot capture time-dependencies, however some trajectories with linear profiles can be analysed with these networks. On the other hand, RNNs are useful for temporal-data by incorporating memory in the network [22], [23]. The most famous approaches are long-short term memory (LSTM) networks [24], [25], gate recurrent units (GRUs),

and transformers [26]. These models exhibit prediction improvements, however the computational power is increased due to the recurrent units training.

Reservoir computing (RC) networks (also known as echo-state neural networks) are a kind of recurrent networks that reduces the computational effort with competitive results [27], [28]. It has an encoder-decoder architecture where the encoder possesses a reservoir module with sparse connections. Here, its main advantage is that the input and reservoir weight are set as random and left untrained whilst the decoder weights are trained using any linear regression model. One of its main advantages in comparison with other RNNs is that RC methods are explainable by its nature of design, that is, we are able to analyse the network properties by mainly studying the eigenvalues of the reservoir module that provides rich information of the input trajectories. These good properties have been exploited in the literature in other machine learning settings, including reinforcement learning, RNNs, CNNs, and generative adversarial networks [29], [30]. This architecture achieves good results if the reservoir module provides of high-dimensional and heterogeneous representation of the trajectories. Otherwise, some authors suggest to modify the linear decoder module with a nonlinear architecture, such as a MLP or SVM, to compensate the lack of richness of the reservoir module [28], [31], [32]. However, there is no evidence of real improvement using nonlinear decoder architectures instead of linear ones. Here, the challenge is how we can design a RC method that obtains a rich high-dimensional representation of the input data to ensure good generalization. Therefore, in this article we aim to provide a solution to this problem by providing a physical interpretation to the reservoir weights.

In the recent years, improvements for regression models have been developed. One major improvement consists in the incorporation of physics informed architectures [33] to increase the precision of the predictions. Physics informed neural networks (PINNs) [34] and novel trajectory inference algorithms [35] exploit the physical properties of the system to infer accurately the trajectory with noise attenuation capabilities. However, knowledge of the exact model of the system (in this case of the drone) is not available which compromises the inference results. Hence, an open gap is how we can provide a physical knowledge to enhance the prediction capabilities when the drone's model is unknown. In this article, we provide an elegant mechanism to incorporate a physics informed model from the prediction error of a RC network.

B. Contributions

In view of the above, this article proposes a physics informed RC (PIRC) framework for drone's trajectory intent prediction. The approach consists in exploiting the capabilities of the standard RC scheme for trajectory prediction and enhance its precision and robustness by incorporating a physics informed model. In this approach, the reservoir weights are improved by a feedback linearization controller obtained from a nonlinear physics informed model. This physics informed model is constructed from the prediction error dynamics between the real drone's trajectories and the predicted trajectory. Two novel PIRC schemes are proposed which maintain the RC properties and enhance its robustness. Stability and boundedness of the proposed approach is assessed using Lyapunov stability theory. Simulation studies are carried out with different drone's trajectory profiles to demonstrate the effectiveness of the proposed approach.

The contributions of this work with respect to previous developments for trajectory inference of drones based on data-driven algorithms are the following.

1) The proposed RC scheme combines the merits of data-driven methods with physics informed models to increase the prediction precision.
2) Only a linear decoder/readout model is required instead of nonlinear models.
3) The prediction error dynamics is used as a physics informed model to improve the reservoir weights.
4) Two different PIRC that preserve noise attenuation capabilities and high precision are proposed.
5) The proposed approach requires low-computational expense in comparison with recurrent networks.
6) The proposed algorithms are simple to put to work since only data collected from both the drone's and reservoir's trajectories are used.

The outline of this article is as follows. Section II defines the design and properties of RC schemes. Section III introduces the proposed PIRC architectures. Sections IV and V reports simulation and experimental studies using different drone's trajectories. The conclusions are presented in Section VI.

Throughout this article, $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$ denote the spaces of natural numbers, real numbers, real n-vectors, and real $n \times m$-matrices, respectively; $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ denotes an identity matrix of $n\times n$; $\lambda_{\min}(\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{A})$ denote the minimum and maximum eigenvalues of matrix $\boldsymbol{A}$, $\otimes$ denotes the Kronecker product, $\text{vec}(\boldsymbol{A})$ is the vectorization of matrix $\boldsymbol{A}$, $\text{mat}(\boldsymbol{x})$ is the matrization of the vector $\boldsymbol{x}$, the norms $\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}$ and $\|\boldsymbol{X}\|_{\mathcal{F}} = \sqrt{\text{tr}\{\boldsymbol{X}^\top \boldsymbol{X}\}}$ stand for the Euclidean and Frobenius norms, respectively; $\text{tr}\{\cdot\}$ defines the trace function, where $x \in \mathbb{R}$ is a scalar, $\boldsymbol{x} \in \mathbb{R}^n$ is a vector, and $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ is a matrix with $n, m \in \mathbb{N}$.

II. Background—Reservoir Computing Scheme

The standard RC scheme for prediction is giving in Fig. 1. The scheme is divided in three main parts: 1) an encoder layer that transforms the drone's trajectories in a low-dimensional space into a heterogeneous high-dimensional space; 2) a reservoir layer that contains a rich pool of heterogeneous dynamics that encapsulate the high-dimensional features of the drone's trajectories; and 3) a decoder layer, also known as readout, that transforms these high-dimensional features into a low-dimensional representation that matches with the actual (regression) or future trajectories (prediction).

The reservoir dynamics is modeled as the following differential neural network:

$$\dot{\boldsymbol{r}} = \sigma(\boldsymbol{A}\boldsymbol{r} + \boldsymbol{W}_{\text{in}}\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is an input vector of the drone's trajectories, $\boldsymbol{r} \in \mathbb{R}^r$ is the reservoir state, $\boldsymbol{A} \in \mathbb{R}^{r\times r}$ defines the reservoir weights, $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{r\times n}$ are the input weights that transform the low-dimensional representation of the drone's trajectories in

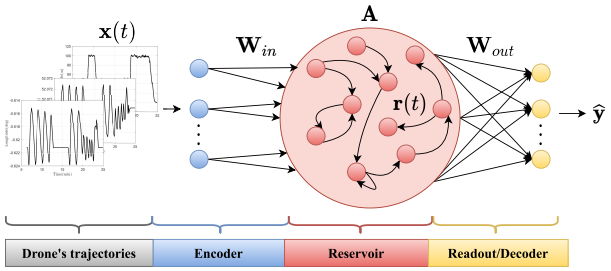


Fig. 1. Standard RC scheme for trajectory prediction.

$\mathbb{R}^n$ into a high-dimensional representation in $\mathbb{R}^r$, $\sigma(\cdot) : \mathbb{R}^r \to \mathbb{R}^r$ is a nonlinear activation function, for example, sigmoid or hyperbolic tangent. In this article, we choose $\sigma(\cdot) = \tanh(\cdot)$ because it is a monotonic differentiable function that regulates the reservoir states whose image is between -1 and 1, which is useful for prediction purposes in contrast to other S-shaped functions. The prediction output in k future steps is then easily computed by

$$\widehat{\boldsymbol{y}} = \phi(\boldsymbol{W}_{\text{out}}\boldsymbol{r} + \boldsymbol{w}) \tag{2}$$

where $\widehat{\boldsymbol{y}} \in \mathbb{R}^n$ is the future prediction output in k steps of the drone's trajectories, $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{n\times r}$ is the decoder/readout weights that returns the high-dimensional representation of the reservoir states $\boldsymbol{r}$ into the original low-dimensional representation, $\boldsymbol{w} \in \mathbb{R}^n$ defines the weights of the bias terms, $\phi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^n$ can be a feedforward neural network or a linear model. Therefore, the encoder network possesses $r(n + r)$ units, whilst the decoder has $n(r + 1)$ units. Here, the number of reservoir units r is a user-design hyperparameter that depends on the number of input trajectories and the richness that we aim to inject to the reservoir module.

Remark 1: In standard RC schemes, both the input weights $\boldsymbol{W}_{\text{in}}$ and reservoir weights $\boldsymbol{A}$ are randomly generated and left untrained, whilst the decoder weights $\boldsymbol{W}_{\text{out}}$ are trained using a ridge regression loss function.

Remark 2: Three main hyperparameters [36] in standard RC schemes need to be initialized to increase the generalization capabilities: 1) the input-scaling parameter w_{in} where $\boldsymbol{W}_{\text{in}} \in [-w_{\text{in}}, w_{\text{in}}]$; 2) the sparsity of the reservoir weights α that defines the proportion of nonzero elements in the reservoir matrix $\boldsymbol{A}$; and 3) the processing units in the recurrent layer r, and the spectral radius parameter $\rho(\boldsymbol{A})$ that describes the largest eigenvalue of $\boldsymbol{A}$ and that fulfils the next equality

$$\boldsymbol{A} = \rho(\boldsymbol{A}) \cdot \frac{\boldsymbol{A}_0}{\lambda_{\max}(\boldsymbol{A}_0)} \tag{3}$$

for some reservoir matrix $\boldsymbol{A}_0$ generated randomly in $[-1, 1]$.

Remark 3: The properties stated in Remark 2 hold for discrete-time RC schemes where the eigenvalues lie in the unit circle. However, in continuous time the reservoir weights matrix must have negative eigenvalues to ensure stability of the network, that is, $\mathcal{R}e\{\lambda(\boldsymbol{A})\} < 0$. One possible solution, and the one we adopted in this article, is to construct $\boldsymbol{A}$ as a random negative definite matrix that verifies

$$\begin{aligned} \boldsymbol{A}_1 &= \frac{1}{2}(\boldsymbol{A}_0 + \boldsymbol{A}_0^\top), \\ \boldsymbol{A} &= \frac{\boldsymbol{A}_1}{\lambda_{\max}(\boldsymbol{A}_1)} - \alpha\lambda_{\max}(\boldsymbol{A}_1)\boldsymbol{I}_r \end{aligned} \tag{4}$$

for some random generated matrix $\boldsymbol{A}_0$ in $[0, 1]$ and α is a scaling factor that increases/decreases the eigenvalues of $\boldsymbol{A}$.

Remark 4: The decoder/readout is usually a linear model, that is, $\phi(\cdot) = \boldsymbol{I}_n$. Other representations can be adopted, such as SVM or a MLP network, to exploit the heterogeneous dynamics offered by the reservoir module. However, the literature does not report relevant improvements since they depend on the richness of the reservoir trajectories.

In this article, we adopt a linear model of the form

$$\widehat{\boldsymbol{y}} = \boldsymbol{W}_{\text{out}}\boldsymbol{r} + \boldsymbol{w}. \tag{5}$$

Define $\boldsymbol{W}_{\text{dec}} := \left[\boldsymbol{W}_{\text{out}}|\boldsymbol{w}\right] \in \mathbb{R}^{n\times(r+1)}$ which can be computed by the minimization of the following convex optimization problem:

$$\boldsymbol{W}^*_{\text{dec}} = \underset{\{\boldsymbol{W}_{\text{out}},\boldsymbol{w}\}}{\text{argmin}} \frac{1}{2}\|\boldsymbol{W}_{\text{dec}}\bar{\boldsymbol{r}} - \boldsymbol{y}\|_2^2 \tag{6}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ stands for the exact future trajectory and $\bar{\boldsymbol{r}} = [\boldsymbol{r}^\top, 1]^\top \in \mathbb{R}^{r+1}$. Assume that both the reservoir trajectories and future drone's trajectories are stored in the following matrices $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k] \in \mathbb{R}^{n\times k}$ and $\boldsymbol{R} = [\bar{\boldsymbol{r}}_1, \bar{\boldsymbol{r}}_2, \ldots, \bar{\boldsymbol{r}}_k] \in \mathbb{R}^{(r+1)\times k}$. Then the convex optimization problem can be rewritten as

$$\boldsymbol{W}^*_{\text{dec}} = \underset{\{\boldsymbol{W}_{\text{out}},\boldsymbol{w}\}}{\text{argmin}} \frac{1}{2}\|\boldsymbol{W}_{\text{dec}}\boldsymbol{R} - \boldsymbol{Y}\|_{\mathcal{F}}^2. \tag{7}$$

Thus, its solution is computed with a standard least-squares algorithm

$$\boldsymbol{W}^*_{\text{dec}} = \boldsymbol{Y}\boldsymbol{R}^\dagger. \tag{8}$$

One of the main weakness of standard RC schemes is their poor representation capabilities due to the random initialization of both the input and reservoir weights. This lack of robustness is usually compensated by designing nonlinear decoder architectures to improve the prediction capabilities of the network. However, the decoder maintain the robustness problem if the reservoir does not pose a rich enough heterogeneous dynamics. In this article, the incorporation of physical knowledge into the network is proposed to ensure that the reservoir module is rich enough and it encompasses a high-dimensional representation of the drone's trajectories.

III. Physics Informed Reservoir Computing

A nonlinear control approach is adopted to incorporate physical knowledge into the RC scheme. Here, the physical properties of the system are inferred to the reservoir weights instead to the decoder weights to exploit the high heterogeneity of the reservoir structure. The general scheme of the proposed PIRC is given in Fig. 2.

The diagram is composed of the standard RC scheme where an additional feedback loop is added to enhance the prediction capabilities of the reservoir network. Here, the prediction error $\boldsymbol{e}$ between the output of the network and the training input trajectories feeds a nonlinear control scheme whose output improves the reservoir weights $\boldsymbol{A}$ and, in consequence, the prediction error $\boldsymbol{e}$ is minimized.

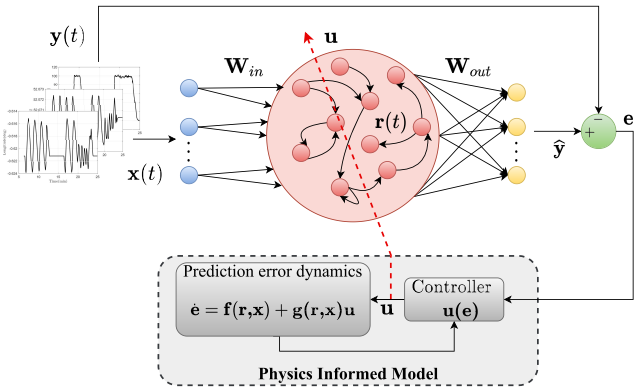


Fig. 2. PIRC scheme.

Remark 5: The feedback loop is only used for the network training phase in order to improve the reservoir weights $\boldsymbol{A}$. After training, the new improved weights remain fixed.

According to the Weierstrass approximation theorem [37], the prediction dynamics of the drone's trajectories can be exactly approximated by the following RC network:

$$\dot{\boldsymbol{y}} = \boldsymbol{W}_{\text{out}}\sigma(\boldsymbol{A}^{*\boldsymbol{r}*} + \boldsymbol{W}_{\text{in}}\boldsymbol{x}) + \boldsymbol{\varepsilon} \tag{9}$$

where $\boldsymbol{A}^* \in \mathbb{R}^{r\times r}$ defines the optimal reservoir weights matrix, $\boldsymbol{r}^* \in \mathbb{R}^r$ stands to the optimal reservoir states using r units, and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is a bounded approximation error that can be decreased as the number of units in the reservoir layer increases. In addition, assume that the optimal reservoir weights can be written as $\boldsymbol{A}^* = \boldsymbol{A} + \boldsymbol{B}^*$ for some unknown matrix $\boldsymbol{B}^* \in \mathbb{R}^{r\times r}$. Then, the RC dynamics is slightly modified to

$$\hat{\dot{\boldsymbol{y}}} = \boldsymbol{W}_{\text{out}}\sigma((\boldsymbol{A}+\boldsymbol{B})\boldsymbol{r} + \boldsymbol{W}_{\text{in}}\boldsymbol{x}) \tag{10}$$

where $\boldsymbol{B} \in \mathbb{R}^{r\times r}$ is a matrix that will be constructed from the physics informed model. Define the prediction error between the RC scheme and the input trajectories as $\boldsymbol{e} = \hat{\boldsymbol{y}} - \boldsymbol{y} \in \mathbb{R}^n$. Then, the error dynamics is given by

$$\dot{\boldsymbol{e}} = \boldsymbol{W}_{\text{out}}[\sigma(\boldsymbol{W}_{\text{in}}\boldsymbol{x} + (\boldsymbol{A}+\boldsymbol{B})\boldsymbol{r}) - \sigma(\boldsymbol{W}_{\text{in}}\boldsymbol{x} + \boldsymbol{A}^*\boldsymbol{r}^*)] - \boldsymbol{\varepsilon}. \tag{11}$$

From (11) we can observe that the drone's dynamics has been incorporated in the error dynamics, which gives a physical meaning to the prediction error that can be exploited to improve the reservoir weights.

A. Physics Informed Reservoir Computing

Taylor series expansion are used in the reservoir dynamics around the vector $z_0 := (\boldsymbol{A}+\boldsymbol{B})\boldsymbol{r} + \boldsymbol{W}_{\text{in}}\boldsymbol{x}$ as

$$\sigma(\boldsymbol{A}^*\boldsymbol{r}^* + \boldsymbol{W}_{\text{in}}\boldsymbol{x}) \equiv \sigma(z_0) + \boldsymbol{D}_\sigma(z_0)(z - z_0) + \boldsymbol{\varepsilon}_\sigma \tag{12}$$

where $\boldsymbol{D}_\sigma(z_0) = [(\partial\sigma(z))/(\partial z)]|_{z=z_0} \in \mathbb{R}^{r\times r}$ and $\boldsymbol{\varepsilon}_\sigma \in \mathbb{R}^r$ is a second order approximation error. Then, the error dynamics (11) is equivalently written as

$$\begin{aligned}\dot{\boldsymbol{e}} &= \boldsymbol{W}_{\text{out}}\big[\boldsymbol{D}_\sigma(z_0)((\boldsymbol{A}+\boldsymbol{B}^*)\boldsymbol{r}^* - (\boldsymbol{A}+\boldsymbol{B})\boldsymbol{r} + \boldsymbol{\varepsilon}_\sigma\big] - \boldsymbol{\varepsilon} \\ &= -\boldsymbol{W}_{\text{out}}\boldsymbol{D}_\sigma(z_0)[\boldsymbol{A}\tilde{\boldsymbol{r}} + \boldsymbol{B}\tilde{\boldsymbol{r}}] + \bar{\boldsymbol{\varepsilon}}\end{aligned} \tag{13}$$

where $\bar{\boldsymbol{\varepsilon}} = -\boldsymbol{W}_{\text{out}}\boldsymbol{D}_\sigma(z_0)[\tilde{\boldsymbol{B}}\boldsymbol{r}^* - \boldsymbol{\varepsilon}_\sigma] - \boldsymbol{\varepsilon} \in \mathbb{R}^n$, $\tilde{\boldsymbol{B}} = \boldsymbol{B} - \boldsymbol{B}^* \in \mathbb{R}^{r\times r}$ and $\tilde{\boldsymbol{r}} = \boldsymbol{r} - \boldsymbol{r}^* \in \mathbb{R}^r$ stand for the error matrix and the error of the reservoir states. Here, $\boldsymbol{r}^*$ can be computed with $\boldsymbol{r}^* = \boldsymbol{W}_{\text{out}}^{\dagger}(\boldsymbol{y} - \boldsymbol{w})$.

Define

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{r},\boldsymbol{x}) &= -\boldsymbol{W}_{\text{out}}\boldsymbol{D}_\sigma(z_0)\boldsymbol{A}\tilde{\boldsymbol{r}} \in \mathbb{R}^n \\ \boldsymbol{g}(\boldsymbol{r},\boldsymbol{x}) &= -\boldsymbol{W}_{\text{out}}\boldsymbol{D}_\sigma(z_0) \otimes \tilde{\boldsymbol{r}}^\top \in \mathbb{R}^{n\times r^2} \\ \boldsymbol{u} &= \text{vec}(\boldsymbol{B}) \in \mathbb{R}^{r^2}.\end{aligned} \tag{14}$$

Then the error dynamics (13) can be equivalently written as

$$\dot{\boldsymbol{e}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{r}) + \boldsymbol{g}(\boldsymbol{x},\boldsymbol{r})\boldsymbol{u} + \bar{\boldsymbol{\varepsilon}}. \tag{15}$$

If $\boldsymbol{u}$ is computed as [38]

$$\boldsymbol{u} = -\boldsymbol{g}^{\dagger}(\boldsymbol{x},\boldsymbol{r})(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{r}) + \boldsymbol{K}\boldsymbol{e}) \tag{16}$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is a diagonal matrix gain which is tuned small enough to avoid noise excitation. Therefore, the error dynamics (15) in closed-loop with the control input (16) is

$$\dot{\boldsymbol{e}} = -\boldsymbol{K}\boldsymbol{e} + \bar{\boldsymbol{\varepsilon}}. \tag{17}$$

The following theorem establishes the uniform ultimately boundedness (UUB) [39] of the prediction error trajectories under the proposed PIRC.

Theorem 1: The prediction error trajectories (15) under the control input (16) exhibit semi-global asymptotic stability and converge to a bounded set S_μ of radius $\mu = [(\|\bar{\boldsymbol{\varepsilon}}\|_2)](\lambda_{\min}(\boldsymbol{K}))]$ as $t \to \infty$ and therefore, the prediction error trajectories $\boldsymbol{e}$ are UUB.

Proof: Consider the following Lyapunov function:

$$V = \frac{1}{2}\boldsymbol{e}^\top\boldsymbol{e}. \tag{18}$$

Taking the time-derivative of (18) along the error trajectories (17) gives

$$\begin{aligned}\dot{V} &= -\boldsymbol{e}^\top\boldsymbol{K}\boldsymbol{e} + \boldsymbol{e}^\top\bar{\boldsymbol{\varepsilon}} \\ &\le -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{e}\|_2^2 + \|\bar{\boldsymbol{\varepsilon}}\|_2\|\boldsymbol{e}\|_2 \\ &= -\lambda_{\min}(\boldsymbol{K})\|\boldsymbol{e}\|_2\left(\|\boldsymbol{e}\|_2 - \frac{\|\bar{\boldsymbol{\varepsilon}}\|_2}{\lambda_{\min}(\boldsymbol{K})}\right).\end{aligned} \tag{19}$$

$\dot{V}$ is negative definite if

$$\|\boldsymbol{e}\|_2 > \frac{\|\bar{\boldsymbol{\varepsilon}}\|_2}{\lambda_{\min}(\boldsymbol{K})} \equiv \mu. \tag{20}$$

There exists a large enough $\boldsymbol{K}$ that ensures that the error trajectories (17) converges into a bounded set S_μ of radius $\mu = [(\|\bar{\boldsymbol{\varepsilon}}\|_2)/(\lambda_{\min}(\boldsymbol{K}))]$, that is, $\|\boldsymbol{e}\|_2 \le \mu$ as $t \to \infty$ and therefore, the trajectories of $\boldsymbol{e}$ are UUB. This completed the proof. ■

After $\boldsymbol{u}$ is computed we need to return it to a $r \times r$ matrix using

$$\boldsymbol{B} = \text{vec}^{-1}(\boldsymbol{u}) \equiv \text{mat}(\boldsymbol{u}). \tag{21}$$

Remark 6: In the training phase we have access to the future trajectories $\boldsymbol{y}$ such that we can easily compute $\boldsymbol{u}$. On the other hand, in the testing phase we do not have access to the future trajectories and $\boldsymbol{B}$ remains constant throughout the testing trajectory.

TABLE I
OPEN-ACCESS DATASETS OF DIFFERENT DRONES' MISSION PROFILES

Dataset	Task
UAV Attack [40]	Waypoint mapping flights
ALFA [41]	Perimeter flights
Drone Identification and Tracking [42]	Waypoint flights
Package delivery [43]	Package delivery flights
Ardupilot data	Custom flights

B. Modified Reservoir Computing

One of the main issues of standard RC schemes is their nonlinear structure that hinders the design of mechanisms to improve the reservoir richness. To solve this issue we propose the following modified RC (MRC):

$$\begin{aligned}\dot{\boldsymbol{r}} &= \boldsymbol{A}\sigma(\boldsymbol{r}) + \sigma(\boldsymbol{W}_{\text{in}}\boldsymbol{x})\\ \widehat{\boldsymbol{y}} &= \boldsymbol{W}_{\text{out}}\boldsymbol{r} + \boldsymbol{w}.\end{aligned} \tag{22}$$

This structure is proposed to preserve the noise attenuation properties of the standard RC schemes, whilst writing the reservoir dynamics as a linear parametrizable model in terms of the reservoir weights $\boldsymbol{A}$.

Assume the real drone's trajectories can be exactly represented under the proposed structure (22) as

$$\begin{aligned}\dot{\boldsymbol{r}}^* &= \boldsymbol{A}^*\sigma(\boldsymbol{r}^*) + \sigma(\boldsymbol{W}_{\text{in}}\boldsymbol{x}) + \boldsymbol{\varepsilon}_r\\ \boldsymbol{y} &= \boldsymbol{W}_{\text{out}}\boldsymbol{r}^* + \boldsymbol{w}\end{aligned} \tag{23}$$

where $\boldsymbol{\varepsilon}_r \in \mathbb{R}^r$ is a small approximation error. Assume that the optimal reservoir weights $\boldsymbol{A}^*$ can be written as $\boldsymbol{A}^* = \boldsymbol{A} + \boldsymbol{B}^*$ for some matrix $\boldsymbol{B}^* \in \mathbb{R}^{r\times r}$.

C. Physics Informed Modified Reservoir Computing

A similar approach can be adopted in the proposed RC structure (22) to integrate a physics informed model. For this purpose, consider that (22) is slightly modified to

$$\begin{aligned}\dot{\boldsymbol{r}} &= (\boldsymbol{A} + \boldsymbol{B})\sigma(\boldsymbol{r}) + \sigma(\boldsymbol{W}_{\text{in}}\boldsymbol{x})\\ \widehat{\boldsymbol{y}} &= \boldsymbol{W}_{\text{out}}\boldsymbol{r} + \boldsymbol{w}.\end{aligned} \tag{24}$$

Consider the prediction error dynamics between the proposed RC model (24) and the real drone's trajectories (23)

$$\begin{aligned}\dot{\boldsymbol{e}} &= \boldsymbol{W}_{\text{out}}[(\boldsymbol{A} + \boldsymbol{B})\sigma(\boldsymbol{r}) - \boldsymbol{A}^*\sigma(\boldsymbol{r}^*) - \boldsymbol{\varepsilon}_r]\\ &= \boldsymbol{W}_{\text{out}}(\boldsymbol{A} + \boldsymbol{B})[\sigma(\boldsymbol{r}) - \sigma(\boldsymbol{r}^*)] + \boldsymbol{\eta}\end{aligned} \tag{25}$$

where $\boldsymbol{\eta} = \boldsymbol{W}_{\text{out}}[\widetilde{\boldsymbol{B}}\sigma(\boldsymbol{r}^*) - \boldsymbol{\varepsilon}_r] \in \mathbb{R}^n$. Then define

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{r}) &= \boldsymbol{W}_{\text{out}}\boldsymbol{A}[\sigma(\boldsymbol{r}) - \sigma(\boldsymbol{r}^*)] \in \mathbb{R}^n\\ \boldsymbol{g}(\boldsymbol{r}) &= \boldsymbol{W}_{\text{out}} \otimes [\sigma(\boldsymbol{r}) - \sigma(\boldsymbol{r}^*)]^\top \in \mathbb{R}^{n\times r^2}.\end{aligned} \tag{26}$$

If we design a control input of the form (16) and from the results of Theorem 1, then semi-global stability can be concluded and the prediction error trajectories $\boldsymbol{e}$ are UUB. Algorithm 1 summarizes the pseudo-code of the proposed PIRC.

Algorithm 1 PIRC for Drone's Trajectory Intent Prediction

Input: Random generated matrix $\boldsymbol{A}$ and $\boldsymbol{W}_{in}$, number of reservoir units r, gain $\boldsymbol{K}$, prediction window to construct the training data $\boldsymbol{X}$ and $\boldsymbol{Y}$ from trajectories of length k.

1: Implement the reservoir dynamics (1) or (22).
2: Collect k samples of the reservoir states $\boldsymbol{r}$ and construct the matrix $\boldsymbol{R}$
3: Compute the decoder weights $\boldsymbol{W}_{dec}$ using (8).
4: Fix the decoder weights $\boldsymbol{W}_{dec}$.
5: Construct the physics informed models using (14) or (26).
6: Compute the control $\boldsymbol{u}$ using (16) and reshape the vector into a matrix using (21).
7: Implement the physics informed reservoir computing using (10) or (24).

Output: $\widehat{\boldsymbol{y}}$.

IV. SIMULATION STUDIES

Several drones' trajectories with different mission profile are tested to verify the effectiveness of the approach. The trajectories are obtained from open-access datasets that cover different real-world mission profiles (see Table I).

These datasets are already preprocessed such that the amount of noise is small. In the experiments conducted in this article, we add some noise to the measurements to model raw measurements from sensors. The datasets contain telemetry data over time, such as longitude, latitude, and altitude. These coordinates are converted into local Cartesian coordinates for each flight. In addition, the sampling time of the GPS measurements across each dataset was not consistent, and so all flights are up-sampled or down-sampled as required to standardize the sampling time to 100 Hz. We consider different scenarios and implementations by modifying the number of input position trajectories. Despite the amount of data is considerable, we only report the results of specific cases of study that encompasses the majority of the drones' mission profiles. Python 3.9.0 and MATLAB 2023a are used to code the proposed algorithms in an XPS Laptop endowed with NVIDIA GeForce RTX 2060 with Max-Q Design.

A. Single Random Trajectory

For visualization simplicity and comparison purposes, consider the raw measurements of a drone's altitude trajectory. First, we test some traditional approaches for prediction under different prediction windows, that is, we predict the next step, the next 100 steps (equivalent to 1 s), and the next 1000 steps (equivalent to 10 s) of the trajectory. We build the training data and the targets in accordance to the prediction capability that we want to inject to each model. The models used are: 1) support vector regression (SVR); 2) MLP; 3) Gaussian process regressor (GPR); 4) convolutional neural networks with attention layer (CNN-A); 5) LSTM; 6) GRU; 7) bidirectional LSTM (BLSTM); 8) convolutional BLSTM (CBLSTM); 9) CBLSTM with attention layer (CBLSTM-A); 10) RC with linear decoder (RClin); 11) RC with SVR decoder (RCSVR); 12) RC with MLP decoder (RCMLP); 13) PIRC; 14) MRC; and 15) physics informed MRC (PIMRC).

The SVR models use a Gaussian kernel with the default parameters, the MLP possesses 3 hidden layers with 50, 30 and 10 neurons, respectively. The GPR uses a square exponential kernel function with hyperparameter optimization (which highly increases the training time). For the RC networks, we choose $r = 5$ processing units and the reservoir weights

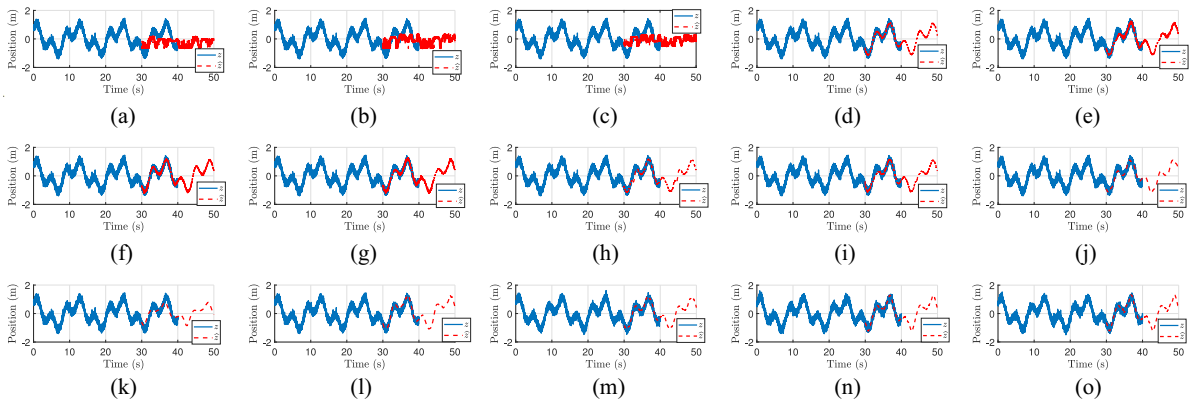


Fig. 3. Comparison results of the Trajectory intent prediction for a time window of 1000 steps. (a) SVR. (b) MLP. (c) GPR. (d) CNN-A. (e) LSTM. (f) GRU. (g) BLSTM. (h) CBLSTM. (i) CBLSTM-A. (j) RClin. (k) RCSVR. (l) RCMLP. (m) PIRC. (n) MRC. (o) PIMRC.

TABLE II
NUMBER OF TRAINABLE PARAMETERS

Model	CNN-A	LSTM	GRU	BLSTM	CBLSTM/CBLSTM-A	RClin
Parameters	150,101	70,501	65,401	90,951	190,951	5

TABLE III
MSE AND COMPUTATION OF DIFFERENT PREDICTORS. THE BEST RESULTS ARE IN BOLD

Method	Prediction					
	1 step		100 steps		1000 steps	
	MSE	Time(s)	MSE	Time(s)	MSE	Time(s)
SVR	0.0382	**0.1094**	0.3067	**0.0938**	0.5383	**0.0469**
MLP	0.038	3.7656	0.2909	3.2969	0.4302	3.0156
GPR [44]	0.0377	604.481	0.2820	561.2126	0.4060	115.1689
CNN-A [45]	0.0137	0.5929	0.0163	1.1012	0.0282	3.8951
LSTM [46]	0.0373	2.7243	0.0217	2.9671	0.0301	3.0727
GRU [47]	0.0374	2.1782	0.0218	2.4030	0.0231	2.3810
BLSTM [48]	0.0374	5.2730	0.0217	5.5058	**0.0264**	4.2507
CBLSTM [49]	**0.0107**	3.6168	0.0169	3.5191	0.05	5.3497
CBLSTM-A [50]	**0.0107**	5.7927	**0.0146**	5.9891	0.0375	7.0823
RClin	0.0213	0.125	0.2081	0.125	0.0277	0.1406
RCSVR	0.0315	0.1562	0.2589	0.2188	0.1017	0.125
RCMLP	0.0267	4.0781	0.5568	3.5469	0.0647	2.4531
PIRC	0.0235	4.25	0.1192	4.1406	0.0291	3.625
MRC	0.0214	0.7344	0.0681	0.8125	0.0303	0.8281
PIMRC	0.0217	6.0625	0.0634	6.0312	**0.026**	6.2969

are initialized to satisfy (4). For this case study, we used 50 neurons for each hidden layer of the LSTM, GRU and CNN networks. For the bidirectional networks, the output layer contains 100 neurons in order to be consistent with the weights training. The main task is to predict the future altitude trajectory of the next 20 s (test data). Here, the number of the proposed neurons are the same for each model such that the prediction error is small and the predicted trajectory exhibits noise attenuation.

Table II shows the number of parameters for this simple experiment. Notice that the number of parameters of each network is notably high in comparison to the RClin method. The number of parameters can be reduced by incorporating regularisation techniques, for example, dropout and batch normalization. However, the number of parameters is still high in contrast to the RClin method which only needs to compute the decoder weights. The prediction results in a time window of 1000 steps are given in Fig. 3. The training computation time of each algorithm and the mean-squared error (MSE) of the predicted data are summarized in Table III.

Different performances can be observed for each approach. All the methods can predict accurately the future trajectory by only considering 1-step ahead. However, both the SVR, MLP, and GRP tend to have an overfitting problem since these models also predict the noise. The same phenomena occurs to the GRU, LSTM and its variations, such that the MSE is small but the predicted trajectory contains large noise. On the other hand, RC models are high accurate and exhibit noise attenuation capabilities. The prediction results for 100–1000 steps show that SVR, MLP and GPR cannot predict precisely the future trajectory which is consistent with the reported in the literature since these models cannot capture time-dependencies. One important result is that the RClin has better results in comparison to the RCSVR and RCMLP, which allows to conclude that nonlinear decoder architectures do not necessarily improve the performance of the RC despite the fact that the input reservoir trajectories present enough richness. For the 100 prediction case, all the recurrent networks outperform the RClin since the reservoir cannot capture adequately the dynamic properties of the input trajectory for better predictions. The physics informed approaches notably improves the RClin results where the MRC and PIMRC outperform the classic RClin and PIRC. For 1000-steps prediction, the proposed models obtain similar results in comparison with the models with attention and bidirectional layers. Here, the reported results demonstrate that RC methods offer competitive results with less computational effort in comparison with deep models that require training of a high number of parameters, whilst in the RC methods the number of parameters are much smaller. In view of these results, the sequel of this article will focus mainly on the proposed architectures to clearly indicate some of their challenges, advantages, and disadvantages.

From the RC results, one can conclude that the RClin has better results and we only need to choose a random negative definite matrix $\boldsymbol{A}$, however this is a misleading result (e.g., see the 100-steps prediction results). Here, the eigenvalues of matrix $\boldsymbol{A}$ play a fundamental role in the richness of the reservoir trajectories, that is, large eigenvalues may lead to a large prediction error due to the convergent reservoir inner dynamics. Conversely, small eigenvalues may improve the richness of the reservoir trajectories but with high probability to destabilize them in presence of a dominant input dynamics $\boldsymbol{x}$. To motivate this fact, consider that the reservoir

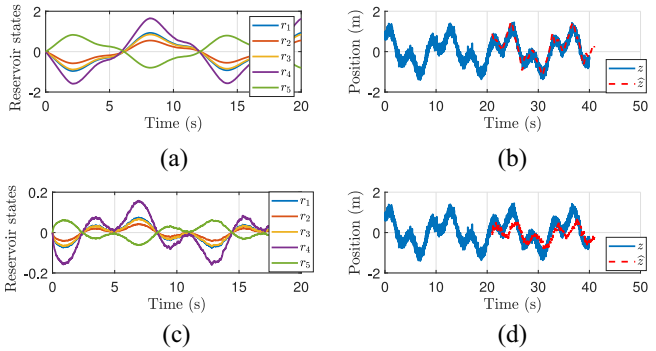


Fig. 4. RClin prediction results with different reservoir weights. (a) Reservoir States: $\boldsymbol{A} = -0.1\boldsymbol{I}_5$. (b) Prediction: $\boldsymbol{A} = -0.1\boldsymbol{I}_5$. (c) Reservoir States: $\boldsymbol{A} = -10\boldsymbol{I}_5$. (d) Prediction: $\boldsymbol{A} = -10\boldsymbol{I}_5$.

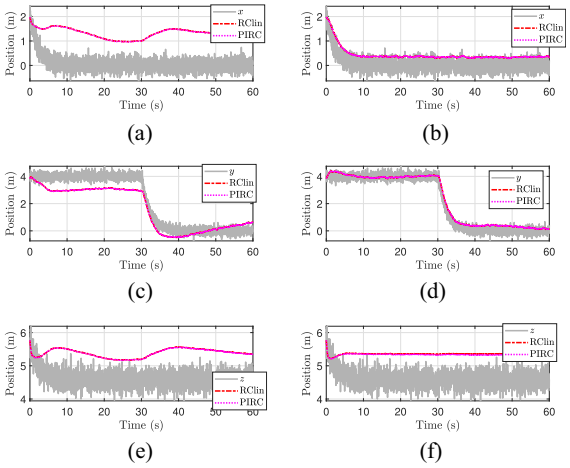


Fig. 5. Predictions of a random package delivery trajectory. (a) Position in X: Joint prediction. (b) Position in X: Single prediction. (c) Position in Y: Joint prediction. (d) Position in Y: Single prediction. (e) Position in Z: Joint prediction. (f) Position in X: Single prediction.

weights are initialized as $\boldsymbol{A} = -0.1\boldsymbol{I}_5$ and $\boldsymbol{A} = -10\boldsymbol{I}_5$ and we want to predict the next 100 steps. The results are given in Fig. 4. It can be observed that the lack of richness in the reservoir trajectories yields to poor prediction results. On the other hand, the prediction precision is enhanced by decreasing the eigenvalues of $\boldsymbol{A}$. Therefore, the richness problem can be improved by adjusting the reservoir weights which can be achieved using the proposed physics informed architectures. Here, the PIRC (10), the MRC (22), and the PIMRC (24) models are compared to show their robustness and prediction enhancement. The results are shown in Fig. 3 and Table III.

The results show that the prediction precision is maintained for both the 1-step and 1000-steps cases. For the 100-steps case, a clear improvement in the prediction results is observed in comparison to the standard RClin (see Table III). However, the time-consuming of our approach is increased due to the computation of $\boldsymbol{u}$.

B. Independent Trajectories

The main issues of the proposed methodology appear when the trajectories are independent from each other and the lack of richness. Here, the reservoir states aim to combine the trajectory features to create a rich pool of trajectories that enhances the prediction capabilities of the network. However, when the trajectories are not related to each other and the richness of information is poor, then the prediction performance is degraded even for short window predictions. One way to alleviate this issue is by implementing single trajectory prediction instead of a joint trajectory prediction. Here, the reservoir states will contain mainly information of a single trajectory and avoids to combine information from the other input trajectories. To show this fact, we draw a random trajectory from the package delivery dataset and predict the future trajectory only one-step ahead. In this experiment, we only test the RClin and PIRC methods to avoid any biased conclusions. The results are shown in Fig. 5 where the first column shows the one-step joint prediction results and the second column exhibits the one-step single prediction results. The results clearly demonstrate that independent trajectories degrades the performance of the RC predictors due to the nonlinear combination between the input trajectories. On the other hand, the single prediction results show better results, however they are affected by the richness of the input trajectory. Specifically in the positions in X and Z, the trajectory is almost constant throughout the length of the trajectory and thus, the predicted model will not be accurate. To fix this issue, the input trajectories must be rich enough in order to excite the modes of the reservoir states that enable a good generalization of the RC predictor.

Joint prediction of trajectories of independent trajectories that do not pose enough richness is a challenging task and topic for further work.

V. Experimental Studies

A personal drone is used to conduct real-world testing in a controlled environment. The control algorithm is designed in MATLAB with an interface with Beagle-Bone-Blue (BBB) chip processor. The VICON camera system, composed of 25 well-distributed cameras with different resolutions, is used to track the position of the drone.

A. Multi-Input Trajectories

To further motivate the effectiveness of the proposed approach, consider a 3-D-perimeter flight trajectory composed of 10 000 data samples. We use 5000 samples to train the RClin, MRC, PIRC, and PIMRC predictors. We use $r = 10$ units to obtain a high-dimensional and heterogeneous representation of the input trajectories. We predict the future trajectory for 1, 100, and 1000 steps.

The predicted trajectories are shown in Figs. 6 and 7. The MSE values for each predictor are summarized in Table IV. The multi-input trajectory results are informative. In contrast to the single trajectory results, here the MRC and PIMRC exhibit more prediction error. After several experiments, we observe that the richness of the reservoir structure is slightly degraded by using the linear parametrizable reservoir structure (22). More units at the reservoir are required to overcome this issue. On the other hand, the RClin and PIRC show

TABLE IV
MSE OF DIFFERENT PREDICTORS UNDER THE NOISE-FREE TRAJECTORY. THE BEST RESULTS ARE IN BOLD

Method	MSE Prediction		
	1 step	100 steps	1000 steps
RClin	**2.124e-5**	**2.794e-4**	**2.20e-5**
PIRC	2.132e-5	2.825e-4	**2.20e-5**
MRC	0.0157	0.0506	0.0133
PIMRC	0.0157	0.0534	0.0133

TABLE V
MSE OF DIFFERENT PREDICTORS UNDER THE NOISY TRAJECTORY. THE BEST RESULTS ARE IN BOLD

Method	MSE Prediction		
	1 step	100 steps	1000 steps
RClin	2.943e-4	0.0025	2.565e-4
PIRC	**2.7097e-4**	**0.0020**	**2.544e-4**
MRC	0.0160	0.0505	0.0134
PIMRC	0.0158	0.05	0.0135

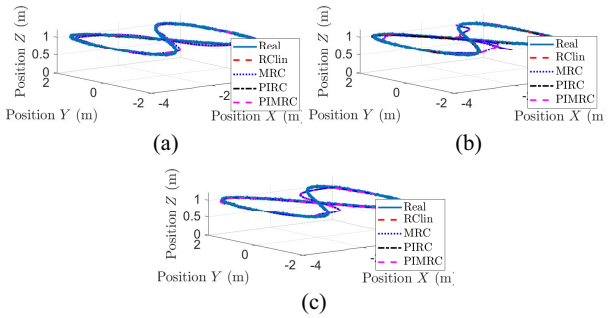


Fig. 6. RC trajectory prediction results. (a) 1-step. (b) 100-steps. (c) 1000-steps.

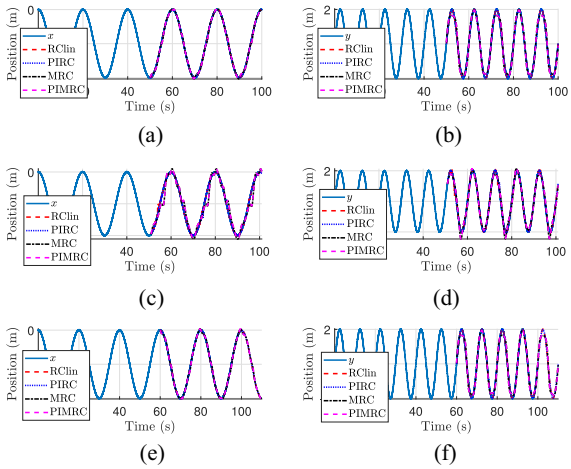


Fig. 7. Individual flight trajectory prediction results. (a) x: 1-step. (b) y: 1-step. (c) x: 100-steps. (d) y: 100-steps. (e) x: 1000-steps. (f) y: 1000-steps.

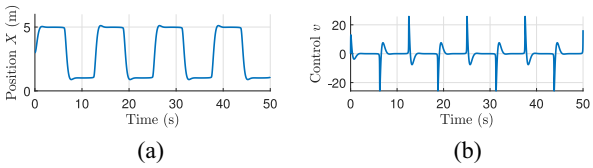


Fig. 8. Custom longitude trajectory. (a) Longitude trajectory x. (b) Control input v.

good trajectory prediction results. It is observed that input trajectories with high frequencies are difficult to accurately predict their future values even in the training phase. On the other hand, if an input trajectory is constant and the reservoir possesses a large number of units, then the output is a sinusoidal function with small amplitude centered in the constant value of the input trajectory. This fact can be observed in the z results of Fig. 7.

The used trajectory is smooth and noise-free such that only the RClin (under adequate reservoir weights $\boldsymbol{A}$) can solve the prediction problem. We add artificially some additive noise to the flight trajectory to demonstrate the robustness of the proposed approach. The results are summarized in Table V. Notice that both PIRC architectures outperformed the results of RClin and MRC. In this scenario, the physics informed model improves the richness of the reservoir to achieve better predictions. Increasing the number of units decreases the MSE of all the methods, however the computational complexity is also increased.

B. Prediction Improvement

One of the main challenges in drone's intent prediction is the randomness of the trajectory throughout the mission profile. In RC schemes, the network is capable to predict the future trajectories (in an acceptable future time) when these trajectories exhibit a repetitive pattern, for example, in surveillance, perimeter flights, waypoint flights, etc. However, when the drone changes randomly its trajectory then the predictions of the RC schemes will be poor. One solution to incorporate knowledge of the decision making process is by means of the input signal $\boldsymbol{v} \in \mathbb{R}^m$ trajectories that drive the drone to a specific desired performance and destination [51], [52], [53]. Here, the control input does not only improve the richness of the input signals, but also gives information about the immediate decision making process. To show the above fact, consider the custom flight trajectory in the X direction and its respective control signal v shown in Fig. 8.

In this experiment, we use $r = 10$ units and the results of the RClin are provided. In this case, we do not compare the physics informed models to avoid biased conclusions. Two scenarios are considered: 1) *Case A*—only the longitude trajectory is used as input data and 2) *Case B*—both the longitude and control input trajectories are used as input data to train the RClin. The trajectory is composed of 5000 data samples where 1500 samples are used to train the RClin and the rest of data are used for testing purposes. The prediction-window is of 1-step. In addition, we test the trained RClin with a simple waypoint trajectory to show the prediction improvement by incorporating the control input trajectory. The results of Case A and Case B are shown in Fig. 9.

Notice that the results of Case A show that the RClin learns that approximately every 5 s the trajectory changes due to repetitive frequency of the training data. In the proposed waypoint trajectory, the RClin prediction starts to oscillate due to this learned pattern. Here, the amplitude of the oscillations

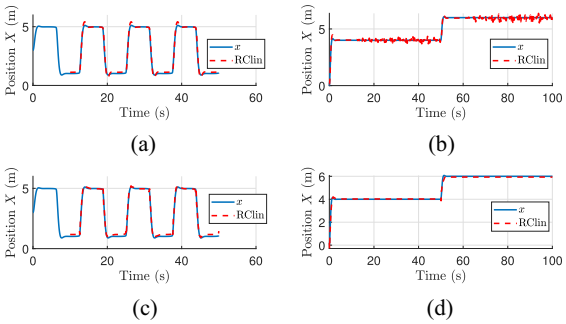


Fig. 9. Prediction results with/without control trajectories. (a) Case A: Custom trajectory. (b) Case A: Waypoint trajectory. (c) Case B: Custom trajectory. (d) Case B: Waypoint trajectory.

increases as time increases. For Case B the control input serve as an additional input to incorporate the decision making process. The MSE error results for the custom and waypoint trajectories are: 0.0196 and 0.0135 for Case A, and 0.0245 and 0.0053 for Case B. The results show that Case A has a better-MSE result for the custom trajectory in comparison to Case B because the control input is subject to high overshoots due to the change of the input trajectory that slightly affects the prediction. However, a considerable improvement can be observed for the waypoint trajectory. Here, the control input adds stability and robustness in the prediction.

The main drawback of using the control input is that is only suitable for short prediction-windows but not for large ones. Further work will study the incorporation of model predictive control techniques to enhance the prediction power of RC networks under control input signals.

VI. Conclusion

In this article, the trajectory intent prediction of drones is addressed using continuous-time RC schemes. The main contribution of this article lies in the incorporation of a physics informed model that enhances the robustness of standard RC schemes, whilst maintaining their richness and noise attenuation capabilities. The physics model is based on the prediction error dynamics between the reservoir prediction and the real drone's trajectories. A feedback linearization controller is used to update the reservoir weights to increase the reservoir richness and improve the prediction precision. Stability and boundedness of the proposed techniques are assessed using Lyapunov stability theory. Simulation studies using different open-access data are provided to show the advantages and disadvantages of the proposed methodology. It is demonstrated that linear decoder models exhibit better performance in comparison to nonlinear decoder architectures, such as Gaussian SVR and MLP. The reservoir weights play a major role in the final prediction results which show competitive results compared with deep models based on CNN and LSTM architectures. On the other hand, the incorporation of a physics informed model in the reservoir weights provides robustness into the network. In addition, it is shown that the addition of the control input trajectories can enhance the prediction robustness for random destination profiles.

Future research vectors will focus on RC based on model predictive control and high-level intent classification of drone's trajectories. Further work is interested in incorporating additional signals to increase the prediction accuracy, such as airspeed and external disturbances.

References

[1] P. Wendt, A. Voltes-Dorta, and P. Suau-Sanchez, "Estimating the costs for the airport operator and airlines of a drone-related shutdown: An application to Frankfurt international airport," *J. Transp. Security*, vol. 13, pp. 93–116, Jul. 2020.

[2] G. Lykou, D. Moustakas, and D. Gritzalis, "Defending airports from UAS: A survey on cyber-attacks and counter-drone sensing technologies," *Sensors*, vol. 20, no. 12, p. 3537, 2020.

[3] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100218.

[4] R. Rezaie and X. R. Li, "Destination-directed trajectory modeling, filtering, and prediction using conditionally Markov sequences," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 2, pp. 820–833, Apr. 2021.

[5] H. Zhang, Y. Yan, S. Li, Y. Hu, and H. Liu, "UAV Behavior-intention estimation method based on 4-D flight-trajectory prediction," *Sustainability*, vol. 13, no. 22, p. 12528, 2021.

[6] Z. Sheng, Y. Xu, S. Xue, and D. Li, "Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17654–17665, Oct. 2022.

[7] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 75–81, Apr. 2018.

[8] S. Samaras et al., "Deep learning on multi sensor data for counter UAV applications—A systematic review," *Sensors*, vol. 19, no. 22, p. 4837, 2019.

[9] A. Perrusquía and W. Guo, "Drone's objective inference using policy error inverse reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 22, 2023, doi: 10.1109/TNNLS.2023.3333551.

[10] J. Liang, B. I. Ahmad, M. Jahangir, and S. Godsill, "Detection of malicious intent in non-cooperative drone surveillance," in *Proc. Sensor Signal Process. Defence Conf. (SSPD)*, 2021, pp. 1–5.

[11] B. Fraser, A. Perrusquía, D. Panagiotakopoulos, and W. Guo, "A deep mixture of experts network for drone trajectory intent classification and prediction using non-cooperative radar data," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2023, pp. 1–6.

[12] G. Revach, N. Shlezinger, R. J. Van Sloun, and Y. C. Eldar, "KalmanNet: Data-driven Kalman filtering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2021, pp. 3905–3909.

[13] A. Perrusquía and W. Guo, "Closed-loop output error approaches for drone's physics informed trajectory inference," *IEEE Trans. Autom. Control*, vol. 68, no. 12, pp. 7824–7831, Dec. 2023.

[14] C. Wang, H. Han, J. Wang, H. Yu, and D. Yang, "A robust extended Kalman filter applied to ultrawideband positioning," *Math. Problems Eng.*, vol. 2020, May 2020, Art. no. 1809262.

[15] X. Ye, A. Radovanovic, and J. V. Milanovic, "The use of machine learning for prediction of post-fault rotor angle trajectories," *IEEE Trans. Power Syst.*, early access, Feb. 19, 2024, doi: 10.1109/TPWRS.2024.3367183.

[16] A. Perrusquia and W. Yu, "Robust control under worst-case uncertainty for unknown nonlinear systems using modified reinforcement learning," *Int. J. Robust Nonlin. Control*, vol. 30, no. 7, pp. 2920–2936, 2020.

[17] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020.

[18] C.-K. Ngan, *Time Series Analysis: Data, Methods, and Applications*. Norderstedt, Germany: BoD–Books on Demand, 2019.

[19] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 1578–1585.

[20] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series," in *Proc. CCIA*, 2018, pp. 120–129.

[21] W. Chen and K. Shi, "Multi-scale attention convolutional neural network for time series classification," *Neural Netw.*, vol. 136, pp. 126–140, Apr. 2021.

[22] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann, "Recurrent Kalman networks: Factorized inference in high-dimensional deep feature spaces," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 544–552.

[23] K. Saleh, M. Hossny, and S. Nahavandi, "Intent prediction of pedestrians via motion trajectories using stacked recurrent neural networks," *IEEE Trans. Intell. Veh.*, vol. 3, no. 4, pp. 414–424, Dec. 2018.

[24] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, "Quantum long short-term memory," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 8622–8626.

[25] J. Yang, Y. Chen, S. Du, B. Chen, and J. C. Principe, "IA-LSTM: Interaction-aware LSTM for pedestrian trajectory prediction," *IEEE Trans. Cybern.*, early access, Feb. 21, 2024, doi: 10.1109/TCYB.2024.3359237.

[26] S. Guo, M. Xia, H. Xue, S. Wang, and C. Liu, "OceanCrowd: Vessel trajectory data-based participant selection for mobile crowd sensing in ocean observation," *IEEE Trans. Sustain. Comput.*, early access, Feb. 23, 2024, doi: 10.1109/TSUSC.2024.3369092.

[27] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.

[28] F. M. Bianchi, S. Scardapane, S. Løkse, and R. Jenssen, "Reservoir computing approaches for representation and classification of multivariate time series," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2169–2179, May 2021.

[29] C. Liu, H. Zhang, Y. Luo, and H. Su, "Dual heuristic programming for optimal control of continuous-time nonlinear systems using single echo state network," *IEEE Trans. Cybern.*, vol. 52, no. 3, pp. 1701–1712, Mar. 2022.

[30] Q. Ma, E. Chen, Z. Lin, J. Yan, Z. Yu, and W. W. Ng, "Convolutional multitimescale echo state network," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1613–1625, Mar. 2021.

[31] A. Rodan, A. F. Sheta, and H. Faris, "Bidirectional reservoir networks trained using SVM+ privileged information for manufacturing process modeling," *Soft Comput.*, vol. 21, no. 22, pp. 6811–6824, 2017.

[32] X. Chen, X. Luo, L. Jin, S. Li, and M. Liu, "Growing echo state network with an inverse-free weight update strategy," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 753–764, Feb. 2023.

[33] A. Perrusquía and W. Guo, "Physics informed trajectory inference of a class of nonlinear systems using a closed-loop output error technique," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 12, pp. 7583–7594, Dec. 2023.

[34] C. Legaard et al., "Constructing neural network based models for simulating dynamical systems," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–34, 2023.

[35] A. Perrusquía and W. Guo, "A closed-loop output error approach for physics-informed trajectory inference using online data," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1379–1391, Mar. 2023.

[36] C. Sun, M. Song, D. Cai, B. Zhang, S. Hong, and H. Li, "A systematic review of echo state networks from design to application," *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 23–37, Jan. 2024.

[37] A. Perrusquía and W. Yu, "Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview," *Neurocomputing*, vol. 438, pp. 145–154, May 2021.

[38] W. Yu and A. Perrusquia, *Human-Robot Interaction Control Using Reinforcement Learning*. Hoboken, NJ, USA: Wiley, 2021.

[39] A. Perrusquía, R. Garrido, and W. Yu, "Stable robot manipulator parameter identification: A closed-loop input error approach," *Automatica*, vol. 141, Jul. 2022, Art. no. 110294.

[40] J. Whelan, T. Sangarapillai, O. Minawi, A. Almehmadi, and K. El-Khatib, 2020, "UAV attack dataset," Dataset, IEEEDataPort. [Online]. Available: https://dx.doi.org/10.21227/00dg-0d12

[41] A. Keipour, M. Mousaei, and S. Scherer, "ALFA: A dataset for UAV fault and anomaly detection," *Int. J. Robot. Res.*, vol. 40, no. 2-3, pp. 515–520, 2021.

[42] M. Street. "Drone identification and tracking." 2021. [Online]. Available: https://kaggle.com/competitions/icmcis-drone-tracking

[43] T. Rodrigues et al., 2020, "Data collected with package delivery quadcopter drone," Dataset. [Online]. Available: https://kilthub.cmu.edu/articles/dataset/Data_Collected_with_Package_Delivery_Quadcopter_Drone/12683453

[44] C. Puri, G. Kooijman, B. Vanrumste, and S. Luca, "Forecasting time series in healthcare with gaussian processes and dynamic time warping based subset selection," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 12, pp. 6126–6137, Dec. 2022.

[45] T. Tayeh, S. Aburakhia, R. Myers, and A. Shami, "An attention-based ConvLSTM autoencoder with dynamic thresholding for unsupervised anomaly detection in multivariate time series," *Mach. Learn. Knowl. Extract.*, vol. 4, no. 2, pp. 350–370, 2022.

[46] X. Wen and W. Li, "Time series prediction based on LSTM-attention-LSTM model," *IEEE Access*, vol. 11, pp. 48322–48331, 2023.

[47] M. Pirani, P. Thakkar, P. Jivrani, M. H. Bohara, and D. Garg, "A comparative analysis of ARIMA, GRU, LSTM and BiLSTM on financial time series forecasting," in *Proc. IEEE Int. Conf. Distrib. Comput. Elect. Circuits Electron.*, 2022, pp. 1–6.

[48] B. Fraser, A. Perrusquía, D. Panagiotakopoulos, and W. Guo, "Hybrid deep neural networks for drone high level intent classification using non-cooperative radar data," in *Proc. 3rd Int. Conf. Elect., Comput., Commun. Mechatron. Eng. (ICECCME)*, 2023, pp. 1–6.

[49] A. Bhoomika, S. N. S. Chitta, K. Laxmisetti, and B. Sirisha, "Time series forecasting and point anomaly detection of sensor signals using LSTM neural network architectures," in *Proc. 10th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2023, pp. 1257–1262.

[50] A. S. Raihan and I. Ahmed, "A bi-LSTM autoencoder framework for anomaly detection—A case study of a wind power dataset," 2023, *arXiv:2303.09703*.

[51] Z. Sheng, L. Liu, S. Xue, D. Zhao, M. Jiang, and D. Li, "A cooperation-aware lane change method for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 3236–3251, Mar. 2023.

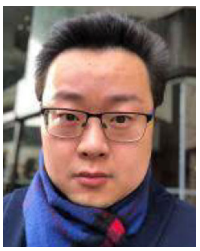
[52] A. Perrusquía and W. Guo, "Performance objective extraction of optimal controllers: A hippocampal learning approach," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, 2022, pp. 1545–1550.

[53] B. Lian et al., "Anomaly detection and correction of optimizing autonomous systems with inverse reinforcement learning," *IEEE Trans. Cybern.*, vol. 53, no. 7, pp. 4555–4566, Jul. 2023.

Adolfo Perrusquía (Member, IEEE) received the M.Sc. and Ph.D. degrees in automatic control from the Automatic Control Department, CINVESTAV-IPN, Mexico City, Mexico, in 2016 and 2020, respectively.

He is currently a Lecturer with the School of Aerospace, Transport and Manufacturing, Cranfield University, Bedford, U.K., and a RAEng Alumni. His main research of interest focuses on reinforcement learning, data-driven control, nonlinear control, machine learning, and system identification.

Dr. Perrusquía is a member of the IEEE Computational Intelligence Society and an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.



Weisi Guo (Senior Member, IEEE) received the M.Eng. degree in engineering and the M.A. and Ph.D. degrees in computer science from the University of Cambridge, Cambridge, U.K., in 2005, 2011, and 2011, respectively.

He is a Chair Professor of Human Machine Intelligence with Cranfield University, Cranfield, U.K. He has published over 180 papers and is a PI on a number of molecular communication research grants.

Prof. Guo's research has won him several international awards. He was a Turing Fellow at the Alan Turing Institute.