EVOLVER: Online Learning and Prediction of Disturbances for Robot Control

Jindou Jia, Wenyu Zhang, Kexin Guo, Jianliang Wang, Xiang Yu, Yang Shi, and Lei Guo

***Abstract*—In nature, when encountering unexpected uncertainty, animals tend to react quickly to ensure safety as the top priority, and gradually adapt to it based on recent valuable experience. We present a framework, namely EVOLVER, to mimic the bio-behavior for robotics to achieve rapid transient reaction ability and high-precision steady state performance simultaneously. In particular, the Koopman operator is leveraged to explore the latent structure of internal and external disturbances, which is subsequently utilized in an *evolutionary* model-based disturbance observer to estimate the eventual disturbance. The resulting observer can guarantee a provable convergence in optimal conditions. Several practical considerations, including construction of a training dataset, data noise handling, and lifting functions selection, are elaborated in pursuit of the theoretical optimality in real applications. The lightweight feature of our framework enables online computation, even on a microprocessor (STM32F7 with 100 Hz control frequency). The framework is thoroughly evaluated by one simulation and three experiments. The experimental scenarios include: 1) trajectory prediction of an irregular free-flying object subject to aerodynamic drag, 2) indoor and outdoor agile flights of a quadrotor subject to wind gust, and 3) high-precision end-effector control of a manipulator subject to base moving disturbance. Comparison results show that the performance of our proposed EVOLVER is superior to several state-of-the-art model-based and learning-based schemes.**

***Index Terms*—Disturbance observer, Koopman operator, online learning for control, disturbance prediction**

I. Introduction

ANALYTICALLY established dynamical models seldom conform exactly to the control demand in real-world robotics applications. This is especially true for a nonlinear robotic system operating in an unstructured environment (e.g., a quadrotor maneuvering in an unknown windy field or a manipulator operating on a floating base). Without any doubt, how to eliminate system disturbances/uncertainties is one of the central tasks in the robot control community.

In the traditional control community, a plethora of *model-based* schemes have been designed to tackle different uncertainties in the past decades such as adaptive control [1], H_∞ control [2], Sliding Mode Control (SMC) [3], Neural Network-based Control (NNBC) [4], and Composite Hierarchical Anti-disturbance Control (CHADC) [5]. A sudden unexpected uncertainty can be quickly (not completely) suppressed at the transient stage by real-time feedback or feedforward methods, ensuring the initial safety of the robot. However, focusing on uncertainty with less *a priori* information, it is difficult to eliminate the adverse effect completely in the steady state, inducing a bounded convergence set [6].

In sharp contrast, the *data-driven-based* techniques are leveraged to fit the uncertainty offline or online [7] in the learning community. The final approximation accuracy can achieve decent performance when the learned data is enough and the designed policy is efficient. Despite of being appealing, since the learning procedure takes time to construct the initial training dataset and train the model [7], the ability to promptly respond to unexpected uncertainty is limited, which may result in an unsafe operation initially [8]. Whereas initial safety is crucial for many robotics, e.g., a quadrotor maneuvering under an abrupt wind disturbance has to make sure that it will not be blown over initially. Moreover, the stability of most learning methods is difficult to guarantee from a theoretical perspective [9]. Another ever-rising learning strategy is to explore the latent uncertainty structure offline and learn the eventual uncertainty adaptively online.

An interesting question arises: Is it possible to possess rapid transient reaction ability and high-precision steady state tracking performance simultaneously in one interpretable framework? This paper provides an affirmative answer to this challenge. In nature, the behavior of integrating reactive control and active learning/prediction for creatures has been investigated. In [10], researchers study the optomotor response of the larval zebrafish under multiple perturbations. At the transient stage, the feedback reactive control with a proven integration part can deal with abrupt perturbations. If the suffered perturbation is long-lasting and predictable, the larval zebrafish can learn the new environmental condition in the cerebellum and induce no more perception delay.

Inspired by the bio-behavior, we present an **EVOL**utionary

This work was supported in part by the Defense Industrial Technology Development Program (Grant Number JCKY2020601C016), the National Natural Science Foundation of China (Grant Numbers 61833013, 61973012, 62388101, 62173024, 62273023), the National Key Research and Development Program of China (Grant Number 2022YFB4701301), the Natural Science Foundation of Zhejiang Province (Grant Number LD21F030001), the Key Research and Development Program of Zhejiang (Grant Number 2021C03158), the Major Science and Technology Innovation Program of Hangzhou (Grant Number 2022AIZD0137), the Beijing Nova Program (Grant Number 20230484266) and the Outstanding Research Project of Shen Yuan Honors College, BUAA (Grant Number 230122104). (*Corresponding author: Xiang Yu.*)

J. J. Jia, W. Y. Zhang, X. Yu, and L. Guo are with the School of Automation Science and Electrical Engineering, Beihang University, 100191, Beijing, China (e-mail: {jdjia, wyzhang, xiangyu_buaa, lguo}@buaa.edu.cn).

J. J. Jia is also with Shenyuan Honors College, Beihang University, 100191, Beijing, China.

K. X. Guo is with the School of Aeronautic Science and Engineering, Beihang University, 100191, Beijing, China (e-mail: kxguo@buaa.edu.cn).

J. L. Wang is with the Collaborative Flying Robotics, Hangzhou Innovation Institute, Beihang University, 310051, Hangzhou, China (e-mail: wjl-180@hotmail.com).

Y. Shi is with the Department of Mechanical Engineering, University of Victoria, BC V8W 3P6, Canada (e-mail: yshi@uvic.ca).

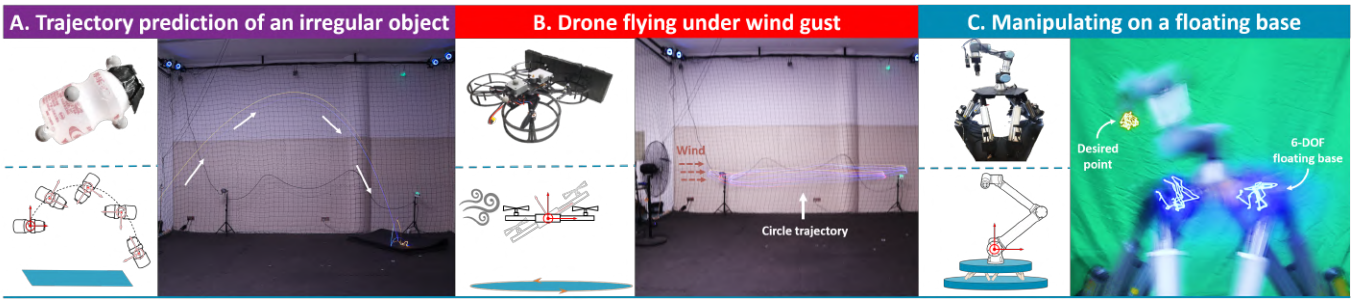

Fig. 1. We demonstrate our approach through (**A**) trajectory prediction of an irregular free-flying object under aerodynamic drag, (**B**) agile flight of a quadrotor under wind disturbance, and (**C**) end-effector control of a manipulator under unwanted base moving disturbance. Long-exposure photos of the above three tasks are taken by placing several LED lamps on the experimental subjects. Experimental video and supplementary materials can be found at https://sites.google.com/view/buaa-evolver.

model-based uncertainty obser**VER** (EVOLVER) by merging reactive control and active learning to achieve the abovementioned objective. At the transient stage, the initial uncertainty observer can guarantee rapid reaction performance without *a priori* uncertainty information. By virtue of the learned model of uncertainty, the uncertainty observer can achieve high-precision estimation in the steady state. Evolutionary contains two meanings. One is that the uncertainty observer in the steady state is evolutionary relative to the one at the transient stage with the help of the learned uncertainty model. The other one is that the learned uncertainty model can continue to evolve with the arrival of additional evaluated valuable data.

A. Contributions

The main contributions are summarized as follows:

A framework of online learning and prediction of disturbances for robot control, that appropriately integrates *model-based* estimation with *data-driven-based* learning, is presented to handle uncertainties (Section IV). The EVOLVER is able to achieve rapid transient reaction ability and high-precision steady state tracking performance simultaneously. The zero-error convergence analysis of the EVOLVER is provided in optimal conditions: the uncertainty is predictable, infinite independent data samples, and orthonormal lifting functions. For uncertainties influenced by unknown factors (i.e., unpredictable), it is intractable to learn accurately, while the adverse effect can be alleviated by further integrating historical states of uncertainty into the lifting functions (Section IV-E).

Practical considerations, including training dataset, data noise, and lifting functions selection that affect estimation performance, are analyzed in pursuit of theoretical optimality. 1) A data selection method considering the *timeliness* and *correlation* of the data is designed (Section V-A), to make the dataset cover more learning-promoting samples. 2) A robust differentiator with a predefined convergence time is exploited to construct a noiseless and accurate training dataset (Section V-B). 3) Manual and automatic selection strategies for the lifting functions are utilized in response to different scenarios (Section V-C). Our framework enables online computation, even on a STM32F7 with 100 Hz control frequency.

Extensive evaluations in various scenarios are conducted to demonstrate the applicability of the EVOLVER (Section VI). In simulation, the chaotic *Lorenz* uncertainty for a second-order *Newton* system is studied. In experiments, as shown in Fig. 1, trajectory prediction of an irregular free-flying object under aerodynamic drag, indoor and outdoor agile flights of a quadrotor under wind disturbance, and end-effector control of a manipulator under base disturbance are implemented.

One interesting finding is that some common characteristics exist among the proposed EVOLVER and the larval zebrafish [10]. Both of them can not only completely overcome the uncertainty at the transient stage when the suffered disturbance is constant, but also eliminate action delays in the steady state after the learned uncertainty model is employed.

B. Organization and Notation

Organization: The article is organized as follows. Section II surveys related work. Section III formulates the uncertainty learning problem. Section IV presents the main theoretical results. Section V details several specific practical considerations. The applicability of the EVOLVER is demonstrated in one simulation and three experiments in Section VI. Section VII concludes this article.

Notation: Throughout the paper, $\mathbb{R}$ denotes the real set; the column vector is denoted with a bold lowercase letter (e.g. $\boldsymbol{a}$), and the matrix is denoted with bold capital (e.g. $\boldsymbol{A}$); $\lambda_M(\cdot)$ represents the maximum eigenvalue of a symmetric matrix; $\circ$ denotes the composition operation; $\boldsymbol{A}^\dagger$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{A}$; $\|\cdot\|$ denotes the 2-norm of a vector; $\hat{\boldsymbol{a}}$ denotes the estimation of $\boldsymbol{a}$ and $\tilde{\boldsymbol{a}}$ denotes the estimation error $\tilde{\boldsymbol{a}} = \boldsymbol{a} - \hat{\boldsymbol{a}}$.

II. Related Work

Broadly, the literature on handling uncertainty can be classified into two categories: *attenuation* by robust control schemes and *compensation* by a hierarchical controller equipped with specific uncertainty estimation techniques [11]. A concise review is presented below.

A. Uncertainty Attenuation Scheme

Attenuation methods usually consider the worst case that the dynamic system may suffer from, resulting in performance conservativeness. Robust SMC [3] utilizes a sign function

term related to the upper bound of disturbance, to attenuate the disturbance. Prescribed performance control (PPC) [12] guarantees a predefined convergence rate and arbitrarily small convergence residual set by a series of error transformations. In essence, PPC amplifies the small error at the beginning and uses larger control input in consequence. The ideal anti-disturbance performances of SMC and PPC depend on the extremely fast response speed and unlimited amplitude of the actuator. This assumption limits their applications in real robotics with physical constraints. Robust H_∞ control [2] aims to suppress the H_∞ norm of the transfer function from disturbance/uncertainty to output. The *attenuation* structure is a single-degree-of-freedom control, where contradictions exist among different performance requirements (e.g., tracking *vs* disturbance rejection, nominal performance *vs* robustness) [13]. In addition, the *attenuation* scheme works after the disturbance has induced output error, namely, a post-reaction scheme, which is inferior to the pre-reaction scheme (i.e., feedforward *compensation*).

B. Uncertainty Compensation Scheme

The basic idea behind *compensation* methods is to employ an observer to estimate the uncertainty by way of its influence on the system performance, and subsequently compensate its influence by using feedforward control. This control scheme is able to act before uncertainty induced error is present as long as the prediction is accurate. *Compensation* method is a two-degree-of-freedom control [5], [14], providing a promising solution for addressing the drawbacks existing in most robust control techniques [13]. With proper integration with a baseline controller, the tracking and uncertainty rejection performances can be achieved simultaneously. There are mainly two design philosophies for the observer: *model-based* construction and *data-driven-based* learning. *Model-based* observers utilize the system model and *a priori* information of uncertainty to construct a convergent observer. In contrast, *data-driven-based* ones intend to learn the uncertainty in terms of datasets extracted from the robotic system.

1) Model-based Methods: In the last four decades, extensive *model-based* uncertainty observer schemes have been developed. With respect to linear systems, Frequency Domain Disturbance Observer (FDDO) [15], Extended State Observer (ESO) [16], Unknown Input Observer (UIO) [17], Generalized Proportional Integral Observer (GPIO) [18], and Disturbance Prediction Observer [14] are presented; Extended High-gain State Observer [19] and Time Domain Nonlinear Disturbance Observer (TDNDO) [20] are designed for nonlinear systems. Although different methods are developed independently, the ideas behind the results are similar. For example, under the proper selection of observer parameters, ESO, UIO, and GPIO are equivalent [13]. Most disturbance observers can achieve zero estimation error in the event of constant disturbances.

For more complicated disturbances, *model-based* disturbance observer usually requires *a priori* disturbance features. For example, UIO [17] and TDNDO [21] can accurately estimate the disturbance produced by exogenous linear systems, e.g., harmonic disturbance with known frequencies. For time-varying disturbances that can be represented by a high-order polynomial of time, GPIO [18] and higher-order TDNDO [22] can achieve zero asymptotic convergence of the disturbance estimation. A feedforward control method is proposed in [14] with disturbance prediction, which assumes the disturbance can be represented by a high-order polynomial of time series or the historical states. For multi-disturbances, simultaneous attenuation and compensation mechanism is proposed in [5]. In [4] and [23], Neural Networks (NNs) are exploited to represent uncertainty, where the weights are adjusted by an adaptive feedback scheme online. However, the adaptive NN-based scheme cannot handle external disturbances independent of the system states. Note that we classify such adaptive NN-based methods as *model-based* observer because no data-mining of historical data is involved and the selection of basis functions relies on *a priori* information of system model.

Most *model-based* observers are limited to a specific type of disturbances and lack universality. For disturbances with a bounded variation rate, the disturbance estimation error can converge to a bounded set [24], [25]. The upper bound of the set can be regulated by observer gains, which involves the trade-off between estimation accuracy and noise amplification.

2) Data-driven-based Methods: Benefiting from improvement in computing power and learning algorithms, *data-driven-based* methods appear to be another option for handling uncertainty. There are two general classes: full dynamics learning [26], [27] and partial dynamics learning [28]. We mainly focus on the latter where partially known dynamics can be utilized. In [29], a stochastic regime is designed to capture the uncertainty and reproduce the variability observed in experiments, whereas the learned uncertainty has not been used in control. In [30], the *Gaussian* belief propagation strategy is employed to compute the uncertainty satisfying *Gaussian* distribution and subsequently tighten the constraints of Model Predictive Control (MPC). In [31], a hierarchical framework is proposed by merging traditional feedback control and residual Reinforcement Learning (RL) for contact-intensive tasks. In a similar manner, the hierarchical framework is adopted by [32], where the additive unknown dynamics is approximated by *Gaussian* processes. Although this hierarchy scheme can learn the residual model with fewer samples and rollouts compared to pure RL, the learning procedure is still time-consuming. For example, 3 hours are needed for the block assembly task [31], and 15 s are needed for variable stiffness actuator pendulum swing-up [32]. In addition, one cannot ensure the outputs of the aforementioned learning methods are rigorously continuous, which is harmful to the actuators.

In summary, with respect to *model-based* uncertainty observers, abrupt uncertainty can be handled promptly due to the rapid online adaptive features. However, the steady state performance of most *model-based* observers is limited to a certain type of uncertainties and lacks generalization. By contrast, *data-driven-based* methods can achieve decent steady state performance, whereas the transient anti-disturbance performance (learning phase) is usually unsatisfactory, as the construction procedure of the initial training dataset takes time. A random policy is frequently adopted at the first stage of learning [27], which may be dangerous for robots (e.g., under-actuated aerial robots). To obtain the rapid transient

reaction ability of *model-based* methods and high-precision steady state tracking performance of *data-driven-based* methods simultaneously, a preliminary attempt combines the two methods by directly superimposing the outputs of both [33]. However, such schemes have not handled the discontinuity of learning methods well and the two methods may contradict when estimating uncertainties with similar features.

III. Problem Formulation

Consider an affine nonlinear control system as,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_x(\boldsymbol{x}) + \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{\Delta}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$ denote the state and the control input, respectively; $\boldsymbol{f}_x(\cdot) \in \mathbb{R}^n$ and $\boldsymbol{f}_u(\cdot) \in \mathbb{R}^{n\times m}$ are smooth nonlinear mappings, and continuously differentiable; $\boldsymbol{f}_u(\cdot)$ is of full row rank to ensure uncertainty compensability; $\boldsymbol{\Delta} \in \mathbb{R}^n$ represents the unknown portion, including external disturbance and internal model uncertainty. The compensation of $\boldsymbol{\Delta}$ is assumed within the capability of the control input.

A general type of uncertainty is considered in this work,

$$\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta}, \boldsymbol{x}, \boldsymbol{\ell}_k, \boldsymbol{\ell}_u), \tag{2}$$

where $\boldsymbol{h}(\cdot) \in \mathbb{R}^n$ is an unknown nonlinear mapping to be approximated by available input and output data; $\boldsymbol{\ell}_k \in \mathbb{R}^k$ and $\boldsymbol{\ell}_u \in \mathbb{R}^u$ denote other known and unknown external disturbance vectors, respectively. For example, in the case of the quadrotor (Section VI-C), $\boldsymbol{\Delta}$ represents the wind disturbance which depends on the system state and external wind speed [34]–[36], where $\boldsymbol{\ell}_u$ can represent the external wind speed that cannot be obtained. Unlike $\boldsymbol{x}$ and $\boldsymbol{\ell}_k$, $\boldsymbol{\ell}_u$ cannot be regarded as a learning feature to fit the uncertainty model (2) in the online learning paradigm. In this work, an alternative strategy is designed in Section IV-E to attenuate the influence of $\boldsymbol{\ell}_u$. Let $s = n + k + u$. Denote

$$\boldsymbol{z} = \left[\boldsymbol{x}^T, \boldsymbol{\ell}_k{}^T, \boldsymbol{\ell}_u\right]^T \in \mathbb{R}^s, \tag{3}$$

Assumption 1. *(Uncertainty Rate Boundness [16]) There exists an unknown positive value $\bar{\Delta}$ such that $\left\|\dot{\boldsymbol{\Delta}}\right\| \leq \bar{\Delta}$.*

The reasonability and limitation of the above assumption are analyzed in Appendix A.

Problem Statement: Consider system (1). The objective is to develop an evolutionary *model-based* uncertainty observer to deal with uncertainty (2). By resorting to the observer, the rapid transient reaction performance and the high precision steady state tracking performance are required to be achieved.

Overall Solution: Fig. 2 presents the proposed EVOLVER framework. In the transient state, an initial uncertainty observer is designed to handle the unknown uncertainty, and the initial training dataset is constructed meanwhile. In the steady state, the Koopman operator is leveraged to explore the unknown model of internal and external uncertainties, which is subsequently utilized in the uncertainty observer. The whole framework features a provable convergence guarantee in optimal conditions. The modules constituting Fig. 2 are detailed in followed sections.

IV. Theoretical Framework

In this section, the EVOLVER is developed to handle the unmodeled uncertainty of (2).

A. Initial Uncertainty Observer

To maintain the rapid transient reaction performance, the initial uncertainty observer is designed. The objective of the nonlinear uncertainty observer is to achieve

$$\dot{\tilde{\boldsymbol{\Delta}}} = \boldsymbol{L}\tilde{\boldsymbol{\Delta}}, \tag{4}$$

where the symmetric matrix $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ denotes the observer gain matrix. The disturbance estimation error $\tilde{\boldsymbol{\Delta}}$ converges exponentially to zero, as all of the eigenvalues of $\boldsymbol{L}$ have negative real parts.

From (1) and (4), the dynamics of the disturbance estimate $\hat{\boldsymbol{\Delta}}$ can be derived as follows

$$\begin{aligned}\dot{\hat{\boldsymbol{\Delta}}} =& \dot{\boldsymbol{\Delta}} - \boldsymbol{L}\left(\boldsymbol{\Delta} - \hat{\boldsymbol{\Delta}}\right)\\ =& \dot{\boldsymbol{\Delta}} - \boldsymbol{L}\left(\dot{\boldsymbol{x}} - \boldsymbol{f}_x(\boldsymbol{x}) - \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u} - \hat{\boldsymbol{\Delta}}\right).\end{aligned} \tag{5}$$

Note that $\tilde{\boldsymbol{\Delta}} = \boldsymbol{\Delta} - \hat{\boldsymbol{\Delta}}$. However, in most robotic applications $\dot{\boldsymbol{x}}$ is unknown or noise-filled. To circumvent this issue, define an auxiliary function $\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{L}\boldsymbol{x}$ and an auxiliary variable $\boldsymbol{\xi} = \hat{\boldsymbol{\Delta}} + \boldsymbol{p}(\boldsymbol{x})$. Without considering $\dot{\boldsymbol{\Delta}}$, the initial nonlinear uncertainty observer is formalized as

$$\begin{cases}\dot{\boldsymbol{\xi}} = \boldsymbol{L}\left(\boldsymbol{f}_x(\boldsymbol{x}) + \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u} + \hat{\boldsymbol{\Delta}}\right),\\ \hat{\boldsymbol{\Delta}} = \boldsymbol{\xi} - \boldsymbol{p}(\boldsymbol{x}).\end{cases} \tag{6}$$

Lemma 1. *(Initial Observer Convergence) Consider the nonlinear system (1) with the uncertainty (2). Under Assumption 1, the estimation error of the initial nonlinear uncertainty observer (6) can exponentially enter into a bounded set, if $\lambda_M(\boldsymbol{L}) < -\frac{1}{4\epsilon}$ is satisfied where ϵ is an arbitrary positive number related to the bounded set.*

Proof. See Appendix B. ■

The uncertainty observer (6) is firstly proposed in [20] for constant disturbance, i.e., $\dot{\boldsymbol{\Delta}} = \boldsymbol{0}$. The convergence analysis under Assumption 1 is further extended here. The detailed steps of proof in Appendix B will recur in the remainder of this article (i.e., Lemma 3 and Theorem 1). The strength of (6) is the exponential convergence property in transient state, even for the unknown uncertainty.

B. Learning Uncertainty Model with Koopman Operator

The uncertainty model $\dot{\boldsymbol{\Delta}}$ is not included in the initial observer (6). To achieve high precision steady state estimation performance, a learning approach is needed to explore the uncertainty model. In order to learn the uncertainty model, an easy-to-implement and explicit *data-driven-based* method is required in our work. Easy-to-implement requires the chosen method to be executed online. Explicit model expression, instead of a black-box input-output mapping, is demanded due to the learned structure with sound interpretability used in the evolutionary uncertainty observer. Koopman operator [37] is such a method to meet these requirements.

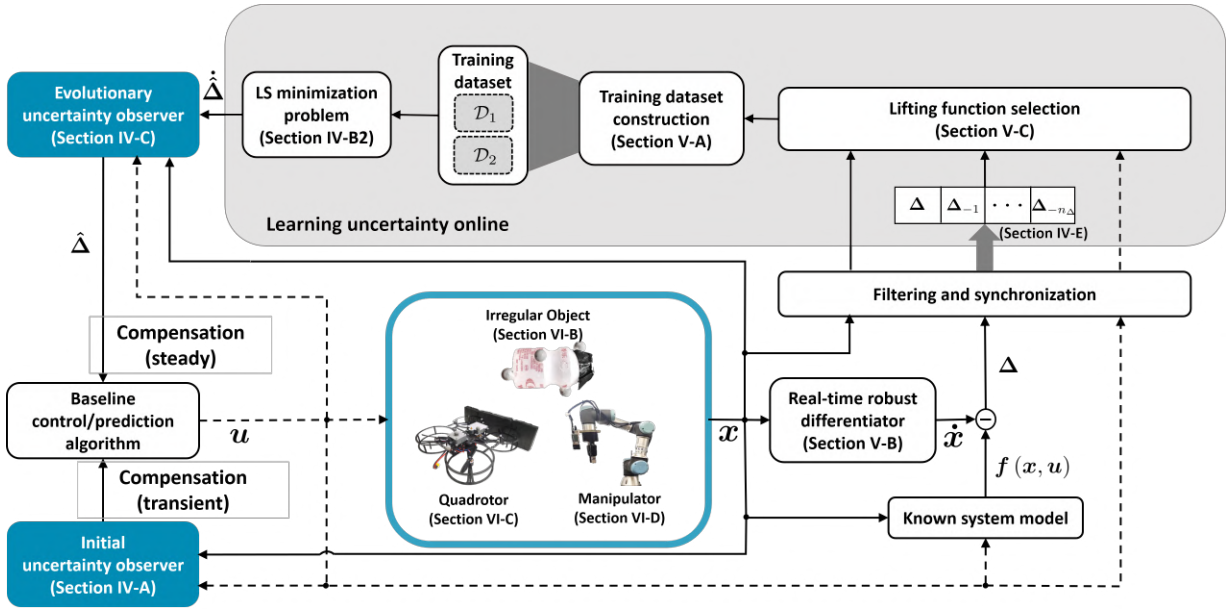

Fig. 2. Overall framework of our proposed EVOLVER. In the trajectory prediction example, the signal flow denoted by the dotted line does not exist.

1) Koopman Operator: In the Koopman operator framework [37], the behavior of a nonlinear system can be captured by an infinite dimensional linear operator, thereby allowing the use of powerful and comprehensive analysis techniques for linear systems. This feature enables a wide variety of applications, such as nonlinear observability analysis [38], state observer design [39], [40], and shared control [41]. In [42], [43], researchers extend the Koopman operator to dynamical systems with exogenous inputs. For ease of exposition, we recall the Koopman operator without considering the factor $\boldsymbol{z}$ in (3) for the time being, and subsequently describe how to modify the algorithm to ensure its integrity.

Consider the uncertainty model

$$\dot{\boldsymbol{\Delta}} = \boldsymbol{h}\left(\boldsymbol{\Delta}\right). \tag{7}$$

Define the lifting function $\varphi\left(\boldsymbol{\Delta}\right): \mathbb{R}^n \to \mathbb{R}$, which belongs to an infinite-dimensional function space $\mathcal{F}$. Then the Koopman operator $\mathcal{K}: \mathcal{F} \to \mathcal{F}$ is defined as [37]

$$\mathcal{K}\varphi\left(\boldsymbol{\Delta}\left(t_0\right)\right) = \varphi \circ \boldsymbol{\Delta}\left(t_0 + t\right), \tag{8}$$

where t_0 represents the initial moment and $\boldsymbol{\Delta}\left(t_0 + t\right) = \boldsymbol{\Delta}\left(t_0\right) + \int_{t_0}^{t_0+t} \boldsymbol{h}\left(\boldsymbol{\Delta}\right) dt$.

Due to the infinite-dimensional feature of the Koopman operator $\mathcal{K}$, the direct implementation is impossible. Extended Dynamic Mode Decomposition (EDMD) [44] offers a data-driven option of approximating $\mathcal{K}$ to a finite-dimensional transfer matrix by a series of nonlinear lifting functions.

2) EDMD for Uncertainty System: The observer functions of the Dynamic Mode Decomposition (DMD) [45] depend on the linear measurements of system states. EDMD can capture nonlinear transient behavior. By combining various linear control techniques, EDMD greatly promotes the application of the Koopman operator in robotics [46]–[48]. In this study, the EDMD is applied to the nonlinear uncertainty system (7).

To begin with, define a set of linearly independent lifting functions as

$$\boldsymbol{\Phi}\left(\boldsymbol{\Delta}\right) := \left[\boldsymbol{\Delta}^T, \varphi_1\left(\boldsymbol{\Delta}\right), \cdots \varphi_{N-n}\left(\boldsymbol{\Delta}\right)\right]^T \in \mathbb{R}^N, \tag{9}$$

where $\varphi_i\left(\cdot\right) \in \mathcal{F}$ and N represents the number of lifting functions. For the convenience of extracting original uncertainty from lifting functions, the first n functions are selected as the uncertainty $\boldsymbol{\Delta}$. Define a subspace $\mathcal{F}_N \subset \mathcal{F}$, which is spanned by the lifting functions in $\boldsymbol{\Phi}\left(\boldsymbol{\Delta}\right)$, i.e., $\left\{\boldsymbol{\Delta}^T, \varphi_1, \cdots \varphi_{N-n}\right\}$.

The EDMD constructs a finite-dimensional approximation $\boldsymbol{K}_{NM} \in \mathbb{R}^{N\times N} : \mathcal{F}_N \to \mathcal{F}_N$ of the Koopman operator $\mathcal{K}$ by solving the least-squares minimization problem

$$\min_{\boldsymbol{K}_{NM}} \frac{1}{2} \sum_{j=1}^{M} \left\| \boldsymbol{\Phi}_{t_s}^j - \boldsymbol{K}_{NM} \boldsymbol{\Phi}^j \right\|^2, \tag{10}$$

where M represents the samples number, $\boldsymbol{\Phi}^j$ is the lifting function corresponding to the j-th sample $\boldsymbol{\Delta}^j$ ($j = 1, 2, \cdots, M$), t_s denotes the sampling time, and $\boldsymbol{\Phi}_{t_s}^j$ is the evolution of $\boldsymbol{\Phi}^j$ after one sampling time.

Fortunately, the problem (10) has the closed-form solution

$$\boldsymbol{K}_{NM} = \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\Phi}_{t_s}^j \left(\boldsymbol{\Phi}^j\right)^T \left[\frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\Phi}^j \left(\boldsymbol{\Phi}^j\right)^T \right]^{\dagger}. \tag{11}$$

The linearized model of (7) in its lifting space $\mathcal{F}_N$ can be obtained

$$\boldsymbol{\Phi}\left(\boldsymbol{\Delta}\left(t_0 + t_s\right)\right) = \boldsymbol{K}_{NM} \boldsymbol{\Phi}\left(\boldsymbol{\Delta}\right) + \boldsymbol{r}\left(t_0\right), \tag{12}$$

where $\boldsymbol{r} \in \mathbb{R}^N$ is the residual error function, by considering that $\mathcal{F}_N$ may not be an invariant subspace of $\mathcal{F}$. If the selected finite lifting functions do form an invariant subspace, the resulting linear model can fully describe the system behavior, i.e., $\boldsymbol{r}\left(t_0\right) = 0$ [49].

The continuous-time dynamics (7) requires reformulation as the discrete-time function (12). The approximate continuous differential equation can be obtained

$$\begin{cases} \dot{\boldsymbol{\Phi}}(\boldsymbol{\Delta}) = \boldsymbol{A}_{\Delta}\boldsymbol{\Phi}(\boldsymbol{\Delta}) + \mathfrak{r}_{\mathbf{1}}, \\ \boldsymbol{\Delta} = \boldsymbol{C}\boldsymbol{\Phi}(\boldsymbol{\Delta}), \end{cases} \tag{13}$$

where $\mathfrak{r}_{\mathbf{1}} \in \mathbb{R}^N$ denotes the residual function error,

$$\boldsymbol{A}_{\Delta} = \log(\boldsymbol{K}_{NM})/t_s \in \mathbb{R}^{N\times N}, \tag{14}$$

with natural logarithm function $\log(\cdot)$, and $\boldsymbol{C} = \left[\boldsymbol{I}_{n\times n}, \boldsymbol{0}_{n\times(N-n)}\right] \in \mathbb{R}^{n\times N}$. The principal matrix logarithm $\log(\cdot)$ demands all of the eigenvalues of the matrix have nonnegative real parts, which may fail when the dataset is insufficient in practice. Without increasing more system measurements like [48], the designed dataset construction method in Section V-A can effectively handle this issue as shown in Fig. 6**C** and Fig. 6**D**.

Lemma 2. *(EDMD Convergence [50]) The sequence of operators $\boldsymbol{K}_{NM}$ converges to $\mathcal{K}$ as $M \to \infty$ and $N \to \infty$, if the following optimal conditions are satisfied.*

(i) *The lifting functions are selected from an orthonormal basis of $\mathcal{K}$.*
(ii) *The samples are drawn independently.*
(iii) *The Koopman operator $\mathcal{K}$ is bounded.*

The proof procedure of Lemma 2 is detailed in [50] and [51]. The descriptions of the conditions are rephrased for ease of understanding. If condition (i) is satisfied, the positive measure μ of $\boldsymbol{\Delta}$ to make $\boldsymbol{c}^H\boldsymbol{\Phi} = 0$ held is zero. $\boldsymbol{c} \in \mathbb{C}^N$ denotes any nonzero vector and $\boldsymbol{c}^H$ denotes the Hermitian transpose of $\boldsymbol{c}$. Conditions (i) and (ii) lead to that $\boldsymbol{K}_{NM}$ converges to $\mathcal{K}_N$ as $M \to \infty$ [51], where $\mathcal{K}_N$ is the $L_2(\mu)$-orthogonal projection of the Koopman operator $\mathcal{K}$ on the subspace $\mathcal{F}_N$. Condition (i) can be satisfied using Hermite polynomial-based lifting functions [52]. Conditions (i) and (iii) result in that $\boldsymbol{K}_N$ converges to $\mathcal{K}$ as $N \to \infty$ [50]. Condition (iii) can be satisfied if the learned system is invertible, Lipschitz with Lipschitz inverse and the measure is absolutely continuous.

Take the factor $\boldsymbol{z}$ of (3) into account, which contributes to the evolution of $\boldsymbol{\Delta}$. This can be treated in an analogous way by expanding the states $\boldsymbol{\Phi}(\boldsymbol{\Delta})$. Following [53] and [46], design additional lifting functions $\boldsymbol{\Psi}(\boldsymbol{\Delta}, \boldsymbol{z}) : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}^{N_2}$, which take the coupling terms of $\boldsymbol{\Delta}$ and $\boldsymbol{z}$ as variables. By concatenating with the previous lifting functions $\boldsymbol{\Phi}(\boldsymbol{\Delta})$ as $\mathbf{S}(\boldsymbol{\Delta}, \boldsymbol{z}) := \left[\boldsymbol{\Phi}(\boldsymbol{\Delta})^T, \boldsymbol{\Psi}(\boldsymbol{\Delta}, \boldsymbol{z})^T\right]^T$, it can be rendered

$$\dot{\mathbf{S}}(\boldsymbol{\Delta}, \boldsymbol{z}) = \boldsymbol{A}\mathbf{S}(\boldsymbol{\Delta}, \boldsymbol{z}) + \mathfrak{r}_{\mathbf{2}}, \tag{15}$$

with $\boldsymbol{A} \in \mathbb{R}^{(N+N_2)\times(N+N_2)}$ and residual error $\mathfrak{r}_{\mathbf{2}} \in \mathbb{R}^{(N+N_2)}$. Similar to (10)-(14), the Koopman operator $\boldsymbol{A}$ in continuous differential equation can be obtained and partitioned as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\Delta} & \boldsymbol{A}_z \\ * & * \end{bmatrix}, \tag{16}$$

where $\boldsymbol{A}_{\Delta} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{A}_z \in \mathbb{R}^{N\times N_2}$. Focusing on the evolution of $\boldsymbol{\Phi}(\boldsymbol{\Delta})$, it follows that

$$\begin{cases} \dot{\boldsymbol{\Phi}}(\boldsymbol{\Delta}) = \boldsymbol{A}_{\Delta}\boldsymbol{\Phi}(\boldsymbol{\Delta}) + \boldsymbol{A}_z\boldsymbol{\Psi}(\boldsymbol{\Delta}, \boldsymbol{z}) + \mathfrak{r}_{\mathbf{3}}, \\ \boldsymbol{\Delta} = \boldsymbol{C}\boldsymbol{\Phi}(\boldsymbol{\Delta}). \end{cases} \tag{17}$$

where $\mathfrak{r}_{\mathbf{3}} \in \mathbb{R}^N$ is the first N rows of $\mathfrak{r}_2$ in (15).

In practice, the residual $\mathfrak{r}_{\mathbf{3}}$ is mainly caused by the unknown factor $\boldsymbol{\ell}_u$, the monotony of the dataset, data noise, and improper choice of lifting functions. The strategies for reducing $\mathfrak{r}_{\mathbf{3}}$ are provided in Section IV-E and Section V.

Based on the above analysis, a corollary of the uncertainty learnability (i.e., $\mathfrak{r}_{\mathbf{3}} = 0$) can be obtained.

Corollary 1. *(Uncertainty Learnability) The uncertainty $\boldsymbol{\Delta}$ described in (2) is fully learnable by the EDMD, i.e., $\mathfrak{r}_{\mathbf{3}} = 0$, if the following conditions are satisfied.*

(i) *$\boldsymbol{\Delta}$ does not depend on $\boldsymbol{\ell}_u$.*
(ii) *The optimal conditions of Lemma 2 are satisfied.*

Until now, the procedure of learning the uncertainty model (2) by virtue of the Koopman operator has been presented. Direct employment of the learned model (17) to estimate uncertainty faces two challenges: how to guarantee the transient performance while the initial training dataset is being constructed and how to obtain the initial estimation of uncertainty. Instead of directly employing the learned model (17), a *model-based* uncertainty observer, taking advantage of the learned model, intends to overcome the challenges in the next section.

C. Evolutionary Uncertainty Observer

If the uncertainty model (17) has been learned, (5) can be adjusted to

$$\begin{aligned} \dot{\hat{\boldsymbol{\Delta}}} =& \dot{\boldsymbol{\Delta}} - \boldsymbol{L}\left(\boldsymbol{\Delta} - \hat{\boldsymbol{\Delta}}\right) \\ =& \boldsymbol{C}\boldsymbol{A}_{\Delta}\boldsymbol{\Phi} + \boldsymbol{C}\boldsymbol{A}_z\boldsymbol{\Psi} \\ & - \boldsymbol{L}\left(\dot{\boldsymbol{x}} - \boldsymbol{f}_x(\boldsymbol{x}) - \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u} - \hat{\boldsymbol{\Delta}}\right). \end{aligned} \tag{18}$$

Therefore, the nonlinear uncertainty observer is formalized as

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{C}\boldsymbol{A}_{\Delta}\boldsymbol{\Phi} + \boldsymbol{C}\boldsymbol{A}_z\boldsymbol{\Psi} + \boldsymbol{L}\left(\boldsymbol{f}_x(\boldsymbol{x}) + \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u} + \hat{\boldsymbol{\Delta}}\right), \\ \hat{\boldsymbol{\Delta}} = \boldsymbol{\xi} - \boldsymbol{p}(\boldsymbol{x}). \end{cases} \tag{19}$$

Lemma 3. *(Observer Convergence) Consider the nonlinear system (1) with the uncertainty (2). Suppose the conditions in Lemma 2 are all satisfied and $\boldsymbol{\ell}_u$ is neglected. The estimation error of the designed nonlinear uncertainty observer (19) can exponentially go to zero regardless of $\boldsymbol{x}$ and $\boldsymbol{u}$, if $\boldsymbol{L}$ is negative definite.*

Proof. See Appendix C. ■

Notice that the calculations of $\boldsymbol{\Phi}(\boldsymbol{\Delta})$ and $\boldsymbol{\Psi}(\boldsymbol{\Delta}, \boldsymbol{z})$ in (19) need current uncertainty $\boldsymbol{\Delta}$. Because accurate $\dot{\boldsymbol{x}}$ requires good differentiating and filtering tools which induce non-negligible delays, $\boldsymbol{\Delta}$ cannot be obtained precisely from the current state by using $\boldsymbol{\Delta} = \dot{\boldsymbol{x}} - \boldsymbol{f}_x(\boldsymbol{x}) - \boldsymbol{f}_u(\boldsymbol{x})\boldsymbol{u}$. Nevertheless, it can be obtained by prediction from previous accurate fitting data with the learned uncertainty model. Moreover, even though the learned part $(\boldsymbol{C}\boldsymbol{A}_{\Delta}\boldsymbol{\Phi} + \boldsymbol{C}\boldsymbol{A}_z\boldsymbol{\Psi})$ is discontinuous with the update of $\boldsymbol{A}_{\Delta}$ and $\boldsymbol{A}_z$, the output of the uncertainty observer (19) is still continuous by virtue of its integral feature.

Algorithm 1 EVOLVER framework.

Initialize: Observer gain $\boldsymbol{L}$; lifting functions $\mathbf{S}$; step size t_s; $j \leftarrow 0$; threshold T_1 and T_2; internal clocking $t_{in} \leftarrow 0$;
Repeat: Detect the presence of uncertainty;
1: **for** $i = 0$ to ∞ **do**
2: **if** $i \cdot t_s \leq T_1$ **then**
3: Execute the initial observer (6);
4: Get current sample $(\boldsymbol{\Delta}, \boldsymbol{z})$ and lift it to $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$;
5: Add last $\mathbf{S}^{i-1}(\boldsymbol{\Delta}, \boldsymbol{z})$ and current $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$ to datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively;
6: **return** $\hat{\boldsymbol{\Delta}}$;
7: **else**
8: Use dataset $\mathcal{D}_1$ and $\mathcal{D}_2$ to fit model $\boldsymbol{A}$ by (11);
9: $t_{in} \leftarrow t_{in} + t_s$;
10: **if** $j == 0$ **then**
11: $j \leftarrow j + 1$;
12: $\boldsymbol{A}_j \leftarrow \boldsymbol{A}$;
13: **else**
14: **if** $t_{in} > T_2$ **then**
15: $j \leftarrow j + 1$;
16: $\boldsymbol{A}_j \leftarrow \boldsymbol{A}$;
17: $t_{in} \leftarrow 0$;
18: **end if**
19: **end if**
20: Execute the observer (19) with updated $\boldsymbol{A}_j$;
21: **return** $\hat{\boldsymbol{\Delta}}$;
22: Get current sample $(\boldsymbol{\Delta}, \boldsymbol{z})$ and lift it to $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$;
23: Add last $\mathbf{S}^{i-1}(\boldsymbol{\Delta}, \boldsymbol{z})$ and current $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$ to datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively;
24: **end if**
25: Compensate $\hat{\boldsymbol{\Delta}}$ by the baseline controller.
26: **end for**

D. Overall Theoretical Framework

Regard $\mathcal{D} = \left\{\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z}) \mid i = 1, 2, \cdots, M+1\right\}$ as the training dataset with $M+1$ samples, where $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$ represents the sample of i-th sampling period before the current time. Then, the dataset tuple $\mathcal{D}_1 = \left\{\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z}) \mid i = 2, 3, \cdots, M+1\right\}$ and $\mathcal{D}_2 = \left\{\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z}) \mid i = 1, 2, \cdots, M\right\}$ are constructed from $\mathcal{D}$, which implies that $\mathcal{D}_1$ is the evolution of $\mathcal{D}_2$ after one sample period.

Algorithm 1 illustrates the overall online learning, prediction, and compensation procedure. In some cases, an auto-detective mechanism is needed to judge the presence of the uncertainty, such that the EVOLVER can be activated. It can be achieved by enabling the initial uncertainty observer (6) all the time without compensation, and detecting if the average estimated uncertainty over a period of time exceeds a certain threshold. See supplementary materials for details. The EVOLVER mainly consists of two stages: the transient phase ($lines$ $3-6$) and the steady state phase ($lines$ $8-23$). In the transient phase, the initial uncertainty observer (6) works while the initial training dataset is constructed. After $M+1$ samples are collected ($line$ 2), the system enters the steady state phase. In the steady state phase, the EDMD is used to fit the uncertainty model and subsequently the evolutionary uncertainty observer (19) takes over. Especially, in order to accurately capture the uncertainty of interest (i.e., the uncertainty around the current operating point) with system (1) evolving, the learned uncertainty model is refreshed in an inherent renewal cycle T_2 or by a difference judgment ($lines$ $14-18$). Note that the switch from the initial uncertainty observer (6) to the evolutionary uncertainty observer (19) is continuous, which is benefited from the integral feature of the uncertainty observer.

Theorem 1. *(Convergence of Algorithm 1) For nonlinear system (1) with unmodeled uncertainty (2), under Algorithm 1, if the conditions of Lemma 2 are all satisfied and $\boldsymbol{\ell}_u$ is neglected, the uncertainty estimation error $\tilde{\boldsymbol{\Delta}}$ in Algorithm 1 converges to zero asymptotically. Furthermore, if $\boldsymbol{\ell}_u$ is considered, $\tilde{\boldsymbol{\Delta}}$ will converge to an arbitrarily small bounded set by decreasing $\lambda_M(\boldsymbol{L})$.*

Proof. See Appendix D. ■

The EVOLVER framework is able to extend the application range of the normal disturbance observer which is first proposed in [20] and is intended for dealing with the constant disturbance only. The presented scheme in Theorem 1 can handle both time-varying unknown dynamics and environmental disturbances.

E. Attenuating the Unpredictable Uncertainty

As stated in Theorem 1, the zero convergence error neglects the influence of $\boldsymbol{\ell}_u$. This effect can be attenuated by decreasing $\lambda_M(\boldsymbol{L})$. However, the effectiveness of such an approach is limited, because noise and actuator delay in real robotics limit the magnitude of $\boldsymbol{L}$. Hence, further solutions for attenuating the uncertainty caused by $\boldsymbol{\ell}_u$ are needed.

Due to the lack of relevant knowledge, it is intractable to completely predict the impact of $\boldsymbol{\ell}_u$. Before proceeding, it is assumed that the motion exhibits specific autocorrelation such that it is amenable to forecasting. We further mine the historical data to find valuable patterns by regarding

$$\boldsymbol{\ell}_u = \left[\boldsymbol{\Delta}_{-1}^T, \boldsymbol{\Delta}_{-2}^T, \cdots, \boldsymbol{\Delta}_{-n_\Delta}^T\right]^T, \tag{20}$$

where $\boldsymbol{\Delta}_{-i}$ ($i \in \{1, 2, \cdots, n_\Delta\}$) denotes the historical uncertainty data of the past i sampling cycles, and n_Δ represents the chosen number of historical samples. The guideline for choosing n_Δ in real applications is provided in supplementary materials.

To achieve disturbance prediction, in [14], [54], the disturbance in each dimension is parameterized with historical states linearly. The coefficients are optimized using least squares (LS). In such a way, the continuity of the predicted uncertainty cannot be guaranteed, which may induce a chattering phenomenon (see Fig. 10**G**). In the EVOLVER framework, this drawback is avoided as a result of the natural integration of the uncertainty observer (19).

Note that for some cases, the evolution of $\boldsymbol{\Delta}$ depends on the higher derivative of the system state which cannot be measured. One way is to treat it as $\boldsymbol{\ell}_u$ and apply the above attenuation strategy. Another way is to utilize the following

robust differentiator (22) to obtain the higher derivative with accurate and robust performance.

V. Practical Consideration

Without a doubt, the requirements $M \to \infty$ and $N \to \infty$ in Lemma 2 are unreachable in practice. The practical problems, including the information richness in the training dataset, data noise, and lifting function selection are analyzed. The corresponding solutions are designed to pursue optimality.

A. Dataset Construction

As stated in Lemma 2, zero-error learning demands infinite independent samples. Moreover, the calculation of the principal matrix logarithm in (14) demands a sufficient dataset. However, the EDMD framework is executed over a given sliding-time window of limited historical state data, in which the data may be strongly correlated. In particular, the latest dataset can be more monotonous as the robot operates at a fixed point. Very few works take this point into account. In [46], an active learning framework is proposed to learn the Koopman operator by driving the robot to explore informative state space. By contrast, the original robotics task is expected unchanged in this study. We next construct a diverse dataset to cover more states with valuable information by evaluating each new sample.

Define a vector $\mathfrak{F} \in \mathbb{R}^M$, which contains the evaluated values (defined in (21) below) of all M samples of $\mathcal{D}_2$ in the sequence. Denote $\Im_i$ as the i-th sample $\mathbf{S}^i$ (abbreviation of $\mathbf{S}^i(\boldsymbol{\Delta}, \boldsymbol{z})$) in $\mathcal{D}_2$ and its evaluated value, respectively. We assess the value of a sample by considering two factors: how new the sample is and how different the sample is from the others, referred to as *timeliness* and *correlation*, respectively. Focusing on *timeliness*, the value is discounted as time passes (i.e., the part $e^{-\beta\cdot|\Delta t_i|}$ in (21)). With respect to *correlation*, a probabilistic technique is utilized (i.e., the part $\mathfrak{f}(p_i)$ in (21)). The previous dataset $\mathcal{D}_2$ is fitted by a multivariate *Gaussian* distribution $\mathcal{G}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ firstly, and then the probability of each incoming sample is calculated. The lower the probability, the more valuable the new sample (i.e., including more additional information), and vice versa. To ease the computational burden, the fitted variables can choose a few principal ones. In addition, the outlier problem needs to be considered as well [55], which may have a very low probability in $\mathcal{G}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. By combing the foregoing *timeliness* and *correlation*, the evaluation index is formalized

$$\Im_i = \mathfrak{f}(p_i)\, e^{-\beta\cdot|\Delta t_i|}, \tag{21}$$

where e is the natural constant, $\beta > 0$ is a time discount factor, Δt_i is the time difference from the current time to the i-th sampling time, p_i represents the probability of i-th sample in the multivariate *Gaussian* distribution, denoted by $p_i = \mathcal{G}(\mathrm{S_i})$, and $\mathfrak{f}(p_i)$ represents a nonlinear mapping defined as $\mathfrak{f}(p_i) = \frac{\alpha_1(p_i-\alpha_2)}{e^{\alpha_1(p_i-\alpha_2)}} e$ with positive parameters α_1 and α_2. The *correlation* degree and outlier rejection can be adjusted by α_1 and α_2, respectively. Fig. 3**A** and Fig. 3**B** illustrate the characteristics of the resulting index. Given the information value of each sample, the sample with the lowest value in the dataset is replaced by the incoming one with a higher value.

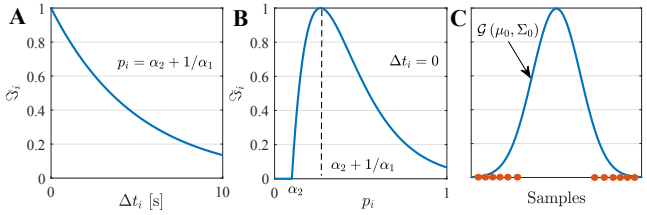

Fig. 3. Illustration of information value $\Im_i$. (**A**) *Timeliness*. The information value $\Im_i$ is discounted over time with a fixed *Correlation*. (**B**) *correlation*. When $p_i > \alpha_2 + \frac{1}{\alpha_1}$, the sample $\mathrm{S_i}$ becomes more and more valuable with probability p_i decreasing. When $p_i < \alpha_2 + \frac{1}{\alpha_1}$, the lower the probability p_i, the more likely $\mathrm{S_i}$ is to be regarded as the outlier. (**C**) *Marginalization* phenomenon (not uniform distribution) when employing only one multivariate *Gaussian* distribution.

Experiments reveal that using one multivariate *Gaussian* distribution may lead to *marginalization* as depicted in Fig. 3**C**. To avoid this issue, in the process of *Gaussian* distribution fitting, we can shuffle the data in $\mathcal{D}_2$, divide $\mathcal{D}_2$ into $n_\mathcal{G}$ segments, and fit them with $n_\mathcal{G}$ ($n_\mathcal{G} > 1$) *Gaussian* distributions (denoted as $\{\mathcal{G}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_1^{n_\mathcal{G}}$) in the sequence.

The dataset construction flow is summarized in Algorithm 2. The lowest evaluated value and its corresponding sample in $\mathcal{D}_2$ are denoted by $\Im_m$ and $\mathrm{S_m}$, while $\mathrm{S_n}$ represents the new incoming sample and $\mathcal{G}_m(\mathrm{S_n})$ represents the maximum probability of $\mathrm{S_n}$ in $\{\mathcal{G}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_1^{n_\mathcal{G}}$. As $t > T_1$, $\mathrm{S_n}$ will be evaluated by (21) with $\Delta t = 0$ ($lines\ 10-12$), denoted as $\Im_n$. If $\Im_n > \Im_m$, $\mathrm{S_n}$ will replace $\mathrm{S_m}$ ($lines\ 14$), otherwise, $\mathrm{S_n}$ is discarded ($line\ 16$). Notice that for saving computing resources, the refresh cycle of $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ is decided by T_3 ($lines\ 3-8$). The efficiency of our designed dataset construction method is demonstrated in Section VI-B, as shown in Fig. 6**C**. It can be seen that after considering the information index $\Im$, the dataset covers more states with valuable information. One limitation of this strategy is the lack of rigorous theoretical guarantees. Future work will pursue improving this dataset construction method from a more theoretical aspect.

B. Data Noise Handling

The dataset of snapshot pairs in the EDMD is constructed from noisy states. In order to save computing resources, a lightweight processing approach is adopted in this work. At the data differential stage, the derivative of a signal $s \in \mathbb{R}$ is obtained by a real-time robust differentiator (RD) [56], which is formalized as

$$\begin{cases} \dot{y}_1 = k_1\nu_1(s - y_1) + y_2, \\ \dot{y}_2 = k_2\nu_2(s - y_1), \\ \dot{s} = y_2, \end{cases} \tag{22}$$

with auxiliary variables $y_1 \in \mathbb{R}$, $y_2 \in \mathbb{R}$; auxiliary functions $\nu_1(x) = k_3^{-1}\Theta(k_3^2 x) \in \mathbb{R}$, $\nu_2(x) = 2\Theta(k_3^2 x)\Theta(k_3^2 x)' \in \mathbb{R}$; and tuning parameters $k_1 \in \mathbb{R}$, $k_2 \in \mathbb{R}$, $k_3 \in \mathbb{R}$. $\Theta(\cdot)$ is called the differentiator generating function and $\Theta(\cdot) = sign(\cdot)\left(|\cdot|^{\frac{1}{2}} + |\cdot|^{\frac{3}{2}}\right)$ is chosen. For a signal with a bounded

Algorithm 2 Dataset Construction Flow.

Initialize: Construct initial $\mathcal{D}$ as $t \leq T_1$ and fit initial $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0$; threshold T_3; internal clocking $t_i \leftarrow 0$;
while $t > T_1$ **do**
  $t_{in}++$;
  if $t_{in} > T_3$ **then**
    $t_{in} \leftarrow 0$;
    Reorganize newest $\mathcal{D}_1$ and $\mathcal{D}_2$;
    Refresh $\{\mathcal{G}_i(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_1^{n_\mathcal{G}}$ with $\mathcal{D}_2$;
    Refresh $\mathfrak{S}$ by (21);
  end if
  Find $\Im_m, \mathrm{S_m}$;
  Get the new incoming sample $\mathrm{S_n}$;
  Calculate the probability $p_n = \mathcal{G}_m(\mathrm{S_n})$;
  Calculate $\Im_n$ by (21) with $\Delta t = 0$;
  if $\Im_n > \Im_m$ **then**
    Replace $\Im_m$ by $\Im_n$; replace $\mathrm{S_m}$ by $\mathrm{S_n}$ in $\mathcal{D}_2$; replace corresponding sampling in $\mathcal{D}_1$;
  else
    Discard $\mathrm{S_n}$;
  end if
  return $\mathcal{D}$.
end while

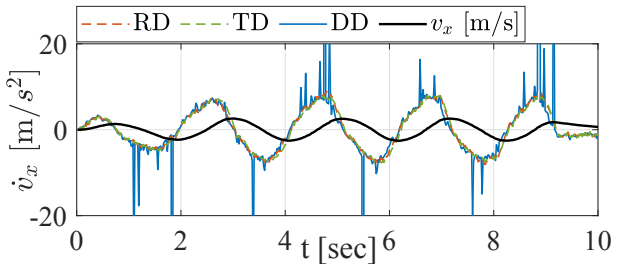

Fig. 4. The performance of the employed differentiator. The solid black curve represents the real flight data of the quadrotor velocity that contains velocities between 0 to 3 m/s. The performance of three kinds of differentiators: robust differentiator (RD) (22), tracking differentiator (TD) [16], and discrete differentiator (DD) (23) are shown in dashed red, dashed green, and solid blue curves, respectively. The parameters of the robust one are set as $k_1 = 16.97, k_2 = 36, k_3 = 1.48$.

second derivative, the differentiator can converge to the derivative of such a signal within a predefined bounded time, which is robust to measurement noise.

Fig. 4 shows the performance of the RD (22) with the real velocity data of the quadrotor used in Section VI-C. The discrete differentiator (DD) is formalized as

$$\dot{s} = \frac{s - s_{-3t_s}}{3t_s}, \tag{23}$$

with sampling time t_s. s_{-3t_s} denotes the historical s before 3 sampling cycles. Although the DD has been smoothed by averaging the last three differences, the obtained $\dot{s}$ contains serious noise. In sharp contrast, the employed RD shows accurate and robust performance.

Moreover, the performance of tracking differentiator (TD) proposed by [16] is also shown in Fig. 4. Its performance can reach the level of the RD after careful parameter adjustment. Due to the provable convergence guarantee and clear guidance on parameter adjustment [56], the RD is chosen in our work.

At the data filtering stage, the obtained $\dot{s}$ is filtered by a second-order low-pass filter. In this process, we can tolerate millisecond signal delay to some extent in order to achieve more accurate filtering.

C. Lifting Functions Selection

Another factor to be considered is the selection of lifting functions, which heavily influence the EDMD performance. In this work, there are two guidelines for choosing lifting functions. On the one hand, the underlying knowledge about robotics may provide a reasonable choice of lifting functions [57], e.g., the aerodynamic drag for an irregular free-flying object or a flying quadrotor [34], [35]. On the other hand, if the considered system structure is unknown, the lifting functions can be chosen by leveraging on Hermite polynomial [58] and Kronecker product to construct orthonormal basis [52] to satisfy the condition (i) of Lemma 2 (see Section VI-A).

VI. Simulation and Experimental Validations

In this section, one simulation and three experiments with different types of uncertainties are conducted. Notice that, with a little abuse of symbols, the adopted symbols in each experiment are independent.

Moreover, to facilitate quantitative analysis, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are defined as $MAE = \frac{1}{n_d}\sum_{i=1}^{n_d}\|\boldsymbol{x}_i - \boldsymbol{x}_{d,i}\|$, $RMSE = \sqrt{\frac{1}{n_d}\sum_{i=1}^{n_d}\|\boldsymbol{x}_i - \boldsymbol{x}_{d,i}\|^2}$, where n_d is the size of collected data, $\boldsymbol{x}_i$ and $\boldsymbol{x}_{d,i}$ are i-th evaluated variable and its desired state.

A. Handling Chaotic Lorenz Uncertainty

The presented EVOLVER is applied to a second-order dynamical system with chaotic *Lorenz* uncertainty, which belongs to the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta})$.

1) Problem Formulation: The considered second-order *Newton* system is modeled as

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{a}, \\ m\boldsymbol{a} = \boldsymbol{u} + \boldsymbol{\Delta}, \end{cases} \tag{24}$$

with position $\boldsymbol{p} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$ (i.e., $\boldsymbol{x}$ in (1)), acceleration $\boldsymbol{a} \in \mathbb{R}^3$, mass $m \in \mathbb{R}$, and control input $\boldsymbol{u} \in \mathbb{R}^3$. The uncertainty $\boldsymbol{\Delta} = [\Delta_1, \Delta_2, \Delta_3]^T$ is generated by the following chaotic *Lorenz* system [57]

$$\begin{cases} \dot{\Delta}_1 = 10(\Delta_2 - \Delta_1), \\ \dot{\Delta}_2 = \Delta_1(28 - \Delta_3) - \Delta_2, \\ \dot{\Delta}_3 = \Delta_1\Delta_2 - \frac{8}{3}\Delta_3, \end{cases} \tag{25}$$

as shown in Fig. 5, **D** to **F**, by solid black curves. The reason for selecting the *Lorenz* uncertainty lies in its chaotic feature and rapid dynamic change, which can be very challenging to eliminate. From (25), it can be seen that the *Lorenz* uncertainty only depends on its own state, which belongs to the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta})$.

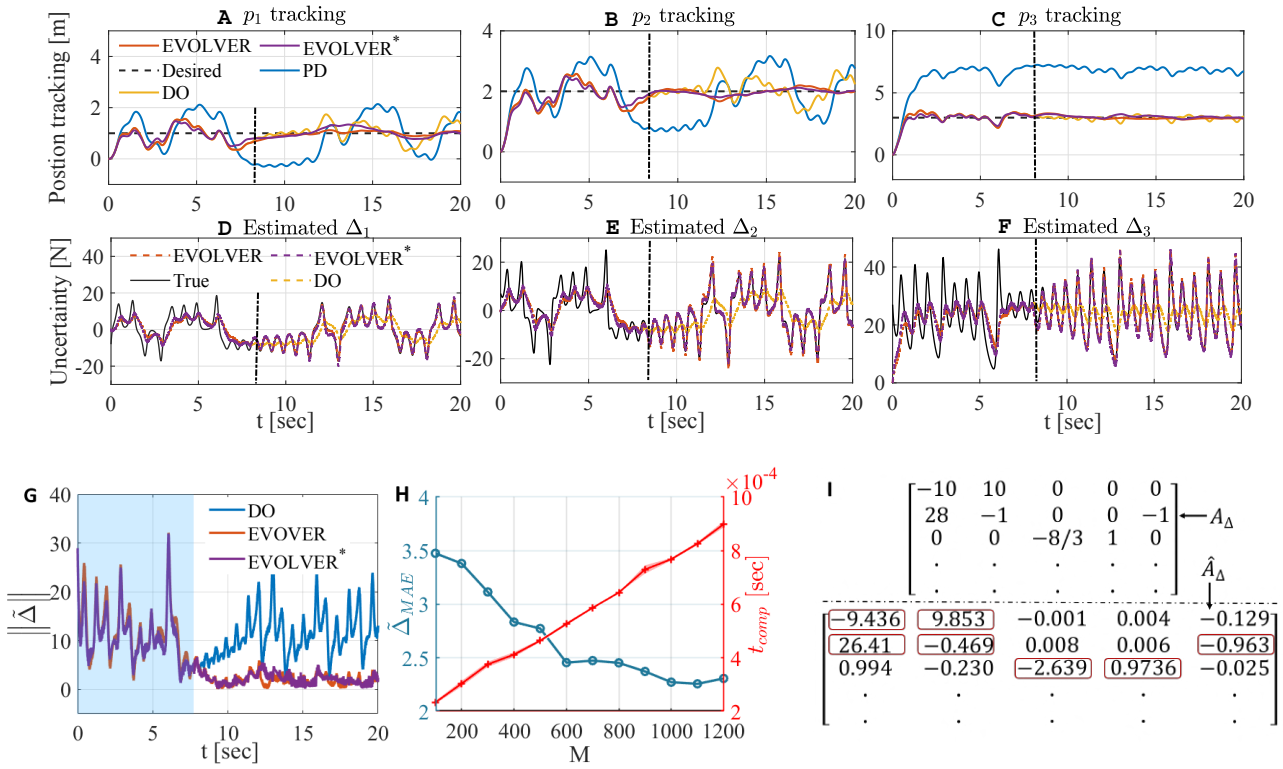

Fig. 5. Simulation results in the *Lorentz* example. (**A**-**C**) The $\boldsymbol{p}$ tracking result. (**D**-**F**) The $\boldsymbol{\Delta}$ estimation result. The transient and steady state phases are separated by a black dotted line at $t = 8s$, i.e., the moment that the learned uncertainty model is first employed. The simulated results of four different schemes are shown in (**A**-**F**): the PD controller, the normal disturbance observer-based method [20], and the EVOLVER with manual and automatic constructed lifting functions, abbreviated to PD, DO, EVOLVER, and EVOLVER*, respectively for simplicity. (**G**) The uncertainty estimation error $\left\|\tilde{\boldsymbol{\Delta}}\right\|$. (**H**) Computation time t_{comp} and uncertainty estimation error $\tilde{\Delta}_{MAE}$ under different sample number M. (**I**) The final estimated $\hat{\boldsymbol{A}}_\Delta$ and true value $\boldsymbol{A}_\Delta$.

The objective is to design $\boldsymbol{u}$ so as to ensure that $\boldsymbol{p}$ and $\boldsymbol{v}$ track their desired states $\boldsymbol{p_d}$ and $\boldsymbol{v_d}$, respectively. The baseline controller adopts the proportional-derivative (PD) control, and the estimated uncertainty $\hat{\boldsymbol{\Delta}}$ by our EVOLVER is compensated via feedforward. The controller is designed as

$$\boldsymbol{u} = \boldsymbol{K}_p \boldsymbol{e}_p + \boldsymbol{K}_v \boldsymbol{e}_v - \hat{\boldsymbol{\Delta}}, \tag{26}$$

with positive definite gain matrices $\boldsymbol{K}_p \in \mathbb{R}^{3\times3}$ and $\boldsymbol{K}_v \in \mathbb{R}^{3\times3}$, and tracking errors $\boldsymbol{e}_p = \boldsymbol{p_d} - \boldsymbol{p}$ and $\boldsymbol{e}_v = \boldsymbol{v_d} - \boldsymbol{v}$.

2) Implementation Details: Apart from our proposed EVOLVER being applied to estimate $\boldsymbol{\Delta}$, the normal disturbance observer-based method of [20] is chosen for comparison. For the sake of fairness, the gains $\boldsymbol{L}$ of both observers are set as the same. In addition, the measured $\boldsymbol{v}$ is corrupted by noise $\boldsymbol{\varepsilon}_v \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}^2\right)$. The desired position and velocity are set as $[1,2,3]^\top$ and $[0,0,0]^\top$, respectively. The initial point of the *Lorenz* uncertainty is set as $[-8,7,27]^T$. All relevant simulation parameters are summarized in Table. I. The involved scheme runs on a laptop with Intel(R) Core(TM) i7-8650U CPU.

The lifting function is chosen as $\boldsymbol{\Phi}(\boldsymbol{\Delta}) = [\Delta_1, \Delta_2, \Delta_3, \Delta_1\Delta_2, \Delta_1\Delta_3]^T$ according to the structure of the *Lorenz* system (25). Moreover, the case when the structure of *Lorenz* system (25) is unknown in advance is also considered. In this case, the lifting function is constructed by Hermite polynomial and Kronecker product automatically. The fundamental variables are chosen as $\{\Delta_1, \Delta_2, \Delta_3\}$. By following the guideline of [52], the lifting function is constructed as $kron\left(kron\left(\boldsymbol{H}\left(\Delta_1\right), \boldsymbol{H}\left(\Delta_2\right)\right), \boldsymbol{H}\left(\Delta_3\right)\right)$, with zero- and first-order Hermite polynomials $\boldsymbol{H}\left(\cdot\right) = \left[H_0\left(\cdot\right), H_1\left(\cdot\right)\right]^T$ and Kronecker product $kron\left(\cdot,\cdot\right)$. Different from [52] employing probabilist's Hermit polynomial, the physicist's Hermit polynomial is adopted here.

3) Simulation Results: Fig. 5 shows the simulation results. From Fig. 5. **A** to **C**, it can be seen that the tracking performance of the normal disturbance observer-based method and EVOLVER is superior to the PD controlled one in transient phase ($t < 8s$), with less overshoot. In the event of p_3 tracking with huge initial uncertainty, the normal disturbance observer-based method and EVOLVER method can rapidly suppress the sudden uncertainty. However, the normal uncertainty observer can only capture the trend of *Lorentz* uncertainty roughly, as shown in Fig. 5. **D** to **G**. In the steady state phase ($t > 8s$), after the learned uncertainty model is employed, the EVOLVER can accurately predict the small and rapid variations of the *Lorentz* uncertainty, with either manually or automatically chosen lifting functions. The final learned $\boldsymbol{A}_\Delta$ (denoted as $\hat{\boldsymbol{A}}_\Delta$) and true $\boldsymbol{A}_\Delta$ in (13) are shown in Fig. 5**I**, and it can be seen that those important non-zero elements are well captured.

Fig. 5**H** further presents computation time t_{comp} and un-

TABLE I
EXPERIMENTAL PARAMETERS

	Parameters
Section IV-A	$\boldsymbol{K}_p = 6.25\boldsymbol{I}_{3\times3}, \boldsymbol{K}_v = 5\boldsymbol{I}_{3\times3}, \boldsymbol{L} = -3\boldsymbol{I}_{3\times3},$ $t_s = 0.01\ s, M = 700, T_1 = 8\ s, T_2 = 1\ s,$ $\boldsymbol{\sigma} = 0.1, k_1 = 60, k_2 = 450, k_3 = 4.182.$
Section IV-B	$\boldsymbol{L} = -5\boldsymbol{I}_{3\times3}, t_s = 0.001\ s, M = 100,$ $\beta = 0.2, \alpha_1 = 10, \alpha_2 = 0, n_{\mathcal{G}} = 20,$ $T_1 = 0.05\ s, T_2 = 0\ s, \boldsymbol{D} = diag\,(0.1, 0.1, 0.096)\,.$
Section IV-C	$\boldsymbol{L} = -0.4\boldsymbol{I}_{3\times3}, t_s = 0.01\ s, M = 300,$ $T_1 = 3\ s, T_2 = 0\ s, \boldsymbol{D} = diag\,(0.7, 0.31, 0)\,,$ $n_{\Delta} = 2, k_1 = 16.97, k_2 = 36, k_3 = 1.48.$
Section IV-D	$\boldsymbol{K}_p = 8\boldsymbol{I}_{3\times3}, \boldsymbol{K}_i = 2\boldsymbol{I}_{3\times3}, \boldsymbol{L} = -3\boldsymbol{I}_{3\times3},$ $t_s = 0.08\ s, M = 200, T_1 = 1.6\ s, T_2 = 0\ s,$ $\beta = 0.2, \alpha_1 = 10, \alpha_2 = 0, n_{\mathcal{G}} = 10, n_{\Delta} = 5,$ $k_1 = diag\,(1.34, 1.34, 0.60)\,,$ $k_2 = diag\,(0.23, 0.23, 0.05)\,,$ $k_3 = diag\,(9.35, 9.35, 20.91)\,.$

[1] Guidelines for adjusting these parameters are provided at https://sites.google.com/view/buaa-evolver.

certainty estimation error under different sample numbers M. t_{comp} refers to the actual time of running a simulation step. For each case, five tests are performed. The mean and standard deviations of the five computation times are shown in solid red line and red shade, respectively. $\tilde{\Delta}_{MAE}$ denotes the MAE of uncertainty estimation error $\tilde{\Delta}$ of 10 s after the learned model is employed. It is obvious that with M increasing, the computation time gets longer, but the estimation accuracy is higher. The computation time of the proposed EVOLVER can fully meet the requirements for online control.

B. Trajectory Prediction of An Irregular Object

EVOLVER is applied to predict the future trajectory of an *uneven* free-flying object, to demonstrate its performance when encountering the uncertainty of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}\,(\boldsymbol{\Delta}, \boldsymbol{x})$.

1) Problem Formulation: The dynamics of an *irregular* object is formalized as

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{a}, \\ \boldsymbol{a} = \boldsymbol{g} + \boldsymbol{\Delta}, \end{cases} \tag{27}$$

with position $\boldsymbol{p} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$, acceleration $\boldsymbol{a} \in \mathbb{R}^3$, acceleration of gravity $\boldsymbol{g} = [0, 0, g]^T \in \mathbb{R}^3$, and uncertainty $\boldsymbol{\Delta} \in \mathbb{R}^3$. The uncertainty $\boldsymbol{\Delta} \in \mathbb{R}^3$ results from the sophisticated aerodynamic drag due to the *irregular* shape. We use the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ to denote the attitude of the object. The drag $\boldsymbol{\Delta}$ of the *irregular* object depends on the velocity and attitude of the object [34], [35]. Hence, the uncertainty in this case is modeled as the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}\,(\boldsymbol{\Delta}, \boldsymbol{x})$, where $\boldsymbol{x}$ contains the translational and rotational states of the object.

If the current acceleration $\boldsymbol{a}_0$ is known, the position after Δt can be predicted according to current states, i.e.,

$$\boldsymbol{p}\,(t_0 + \Delta t) = \boldsymbol{p}_0 + \boldsymbol{v}_0\Delta t + \frac{1}{2}\boldsymbol{a}_0\Delta t^2, \tag{28}$$

where t_0, $\boldsymbol{p}_0$, and $\boldsymbol{v}_0$ denote the current time, position, and velocity, respectively. A more rigorous prediction strategy is provided in Section VI-B4. The position $\boldsymbol{p}_0$ and velocity $\boldsymbol{v}_0$ can be accurately measured by the motion capture system. The objective of the prediction study is to accurately obtain current $\boldsymbol{a}_0$ by *model-based* or *data-driven-based* methods. Our EVOLVER provides an online learning way to the estimation of $\boldsymbol{\Delta}$ and the improvement of the current $\boldsymbol{a}_0$ estimate.

2) Implementation Details: An *irregular* empty bottle, as shown in Fig. 1**A**, is selected in this experiment. Several advanced methods are taken as comparisons, including drag coefficient fitting method [59]–[61] and machine learning-based method [62], [63]. 30 real free-flying trajectories are recorded by the motion capture system. 21 of these trajectories are taken as the training sets for offline comparison methods excluding the proposed one and the rest trajectories are used as the testing set for all of the chosen methods. The real acceleration $\boldsymbol{a}$ in the offline training set is obtained by the LS polynomial fit. Technical details for all implemented methods, including the proposed one, are listed as follows.

- EVOLVER: $\boldsymbol{\Delta}$ is estimated by Algorithm 1 as the object is thrown out. The lifting functions $\boldsymbol{\Phi}\,(\boldsymbol{\Delta})$ are chosen as $[\Delta_1, \Delta_2, \Delta_3]^T$, while the lifting functions $\boldsymbol{\Psi}\,(\boldsymbol{\Delta}, \boldsymbol{z})$ are set as same as the input variables of followed machine learning-based method, to make a fair comparison. Due to the short duration (around 1 s) of the whole mission, the uncertainty begins to be learned before enough data has been collected. After the learning dataset $\mathcal{D}_2$ has collected more than $\frac{M}{2}$ samples (by setting $T_1 = \frac{M}{2}t_s$), $\boldsymbol{A}$ begins to be learned. Once the learning dataset $\mathcal{D}_2$ has collected more than M samples, Algorithm 2 is enabled to construct the richness dataset. The related parameters are summarized in Table. I.
- Drag model-based method [59]–[61]: $\boldsymbol{\Delta}$ is modeled as the quadratic drag which is proportional to the square of $\boldsymbol{v}$, i.e., $\boldsymbol{a} = \boldsymbol{g} - \boldsymbol{D}\,\|\boldsymbol{v}\|\,\boldsymbol{v}$, where $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ is a symmetric positive matrix, denoting the drag coefficient. $\boldsymbol{D}$ is fitted by the LS algorithm with the training set, which is shown in Table. I.
- Machine learning-based method [62], [63]: The dynamics of the object is modeled as $\boldsymbol{a} = \boldsymbol{f}\,(\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{R})$ where the unknown nonlinear mapping $\boldsymbol{f}\,(\cdot)$ is learned by Support Vector Regression (SVR) [64]. To improve learning efficiency, different from [62] and [63], the variable set $\{\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{v}^2, \boldsymbol{R}\}$ is taken as training input, where the square of $\boldsymbol{v}$ is considered additionally. The linear kernel is adopted here.
- Others: A version only considering the gravity and a version enhanced with the normal disturbance observer [20] are conducted as the benchmarked comparisons.

3) Experimental Results: Fig. 6 shows the experimental results. Nine testing trajectories collected by the motion capture system are presented in Fig. 6**A**. For the convenience of data processing, the flight times for all trajectories are tailored to 1.057 s by setting start points. The prediction horizon is set as 0.5 s. The prediction errors are illustrated in Fig. 6**B**, where the colored shaded area represents the standard

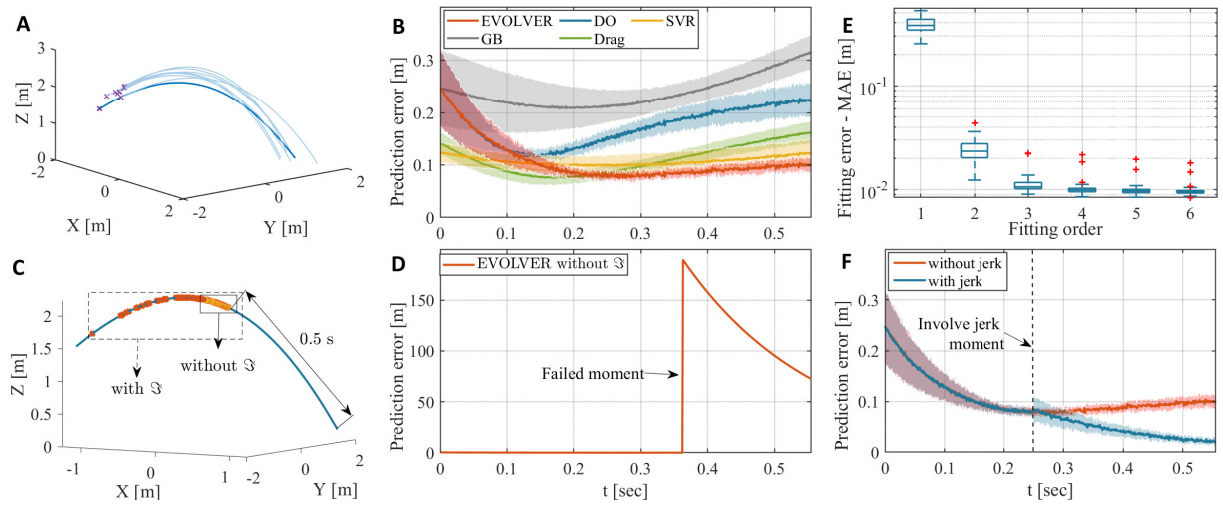

Fig. 6. Experimental results in the object trajectory prediction example. (**A**) All 9 testing trajectories. The darkened one is utilized in (**C**) and (**D**) below. (**B**) The prediction errors of object trajectory with 0.5 s prediction horizon under different methods: proposed EVOLVER-based, gravity-based, normal disturbance observer-based [20], drag model-based [59]–[61], and SVR-based [62], [63], abbreviated to EVOLVER, GB, DO, Drag, and SVR, respectively for simplicity. (**C**) The collected datasets in the final step with and without considering the information index $\Im$ are labeled on the test trajectory. (**D**) Failed convergence when $\Im$ is not considered. (**E**) Fitting errors of all recorded trajectories in different fitting orders. (**F**) The prediction errors with 0.5 s prediction horizon when the jerk information is involved.

deviations of the testing trajectories. It can be obtained that our EVOLVER method, as well as the drag model-based method and SVR-based method can achieve decent performance compared with the two other benchmarked methods (gravity-base and normal disturbance observer-based methods). The SVR-based method shows the most stable performance. With respect to EVOLVER, since the uncertainty model has not been learned before 0.05 s, its performance mainly depends on the initial uncertainty observer (6). After the learned uncertainty model is employed, the prediction error can converge to a level comparable to offline learning methods. Especially, the online learning regime in EVOLVER can quickly adapt to the changing environment, leading to slightly better performance than offline learning methods towards the end.

Furthermore, the efficiency of the designed dataset-construction method in Algorithm 2 is verified. The trajectory darkened in Fig. 6**A** is selected to illustrate the result. We implement EVOLVER to predict the object trajectory with and without Algorithm 2, respectively. Fig. 6**C** shows the benefits of our designed dataset-construction method. The constructed dataset in the final prediction step is labeled on the trajectory. When the information index $\Im$ is not considered, the samples in the training dataset mainly concentrate around the current state, resulting in ill-fitting and failed estimation convergence as highlighted in Fig. 6**D**. After the information index $\Im$ is involved, the samples cover richer states, which becomes sparse with the *timeliness* decreasing.

4) Further Improvement: Experimental results show that there is still an error of 0.1 m among the drag model-based, SVR-based, and EVOLVER methods. The reason can be attributed to simple Euler integration used in (28) at each prediction step, which assumes that the resultant force of the followed trajectory is the same as the one at the current prediction moment. The differential equation (27) does not have a closed-form solution in general as a result of the complex structure of $\boldsymbol{\Delta}$. Rigorous step-by-step forward integration would induce high computation cost [65]. To save the computational cost, the Euler integration (28) is used in most previous works [59], [61], [63], [66], [67]. In [65], a closed-form approximation of the flight trajectory is derived for a 2D moving regular object. Here we provide another prediction strategy for the *irregular* object by employing the inherent property of the proposed EVOLVER.

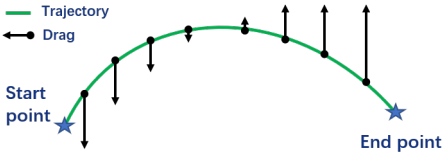

Fig. 7. Conceptual graph: The coarse variation of the aerodynamic drag along the height direction. The direction of the drag is opposite to the object velocity. The faster the speed, the greater the magnitude of the drag.

Fig. 7 illustrates the coarse variety of the aerodynamic drag along the height direction. The vertical motion is approximately a motion with a constant jerk. To examine this intuitive observation, polynomials of different degrees are used to fit all recorded collected trajectories. The fitting results are presented in Fig. 6**E**. The $3rd$-order fitting (i.e., acceleration varies uniformly) has a satisfactory performance with the lower order. Thus, the assumption of constant jerk motion is more reasonable than the assumption of constant acceleration motion in (28). Fortunately, the proposed EVOLVER provides $\dot{\boldsymbol{\Delta}}$ estimation in (17), i.e. the jerk of *irregular* object system (27). Denoting $\boldsymbol{j}_0$ as the current jerk, (28) is adjusted to

$$\boldsymbol{p}\left(t_0+\Delta t\right)=\boldsymbol{p}_0+\boldsymbol{v}_0\Delta t+\frac{1}{2}\boldsymbol{a}_0\Delta t^2+\frac{1}{6}\boldsymbol{j}_0\Delta t^3. \tag{29}$$

Fig. 6**F** shows the result that the prediction mechanism is switched from (28) to (29) at $t=0.25$ s. The prediction

error is found to have further decreased after involving the jerk information.

C. Application to A Quadrotor

In this experiment, EVOLVER is applied for an aggressively flying quadrotor under the wind disturbance, which belongs to the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta}, \boldsymbol{x}, \boldsymbol{l}_u)$. By employing the proposed EVOLVER, the quadrotor can well adapt to external wind disturbance, in comparison with several methods [20], [36].

1) Problem Formulation: The kinematics and dynamics of the translational loop are formalized as

$$\begin{cases} \dot{\boldsymbol{p}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{a}, \\ m\boldsymbol{a} = -f\boldsymbol{b}_3 + mg\boldsymbol{e}_3 + \boldsymbol{\Delta}, \end{cases} \tag{30}$$

with position $\boldsymbol{p} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$ (i.e., $\boldsymbol{x}$ in (1)), acceleration $\boldsymbol{a} \in \mathbb{R}^3$, acceleration of gravity $\boldsymbol{g} \in \mathbb{R}^3$, mass m, the total thrust magnitude $f \in \mathbb{R}$, inertial frame $\mathcal{I} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3]$, body frame $\mathcal{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3]$, and the wind disturbance $\boldsymbol{\Delta} \in \mathbb{R}^3$. The frame definition is further illustrated in Fig. 8**A**.

Different from the last object trajectory prediction example where $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta}, \boldsymbol{x})$, in the quadrotor case, $\boldsymbol{\Delta}$ additionally depends on external wind speed which is unknown for the quadrotor [34]–[36]. Thus, the uncertainty is considered as the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}(\boldsymbol{\Delta}, \boldsymbol{x}, \boldsymbol{l}_u)$ where $\boldsymbol{l}_u$ represents the relevant part of unknown wind speed.

2) Implementation Details: The control framework of the quadrotor is shown in Fig. 8**B**. The controller in our previous work [36] in the absence of the drag utilization and the WSO parts is employed as the baseline controller.

In indoor experiments, the quadrotor is commanded to follow a circle trajectory $\left[\begin{array}{ccc} r\sin(\frac{2\pi}{T}t) & r\cos(\frac{2\pi}{T}t) & 1 \end{array}\right]^\top m$, where the radius r and the period T are set as $1\ m$ and $2.1\ s$, respectively, i.e., $3\ m/s$ speed and $9\ m/s^2$ acceleration. Two $380\ W$ fans are placed at $[-2.5, 0.5, 1]\ m$ and $[-2.5, -0.5, 1]\ m$, along the direction of $\boldsymbol{e}_1$. Fig. 8**E** illustrates the indoor wind field distribution, which is obtained by spatial sampling through a digital anemometer AS8556. Notice that the sampling altitude is consistent with that of the quadrotor flight. The maximum wind speed is up to $6\ m/s$. In particular, a foam board ($0.125\ m \times\ 0.465\ m$) is attached to the back of the quadrotor (facing to the fans) to increase the drag surface. The wind disturbance is mainly in the direction of $\boldsymbol{e}_1$. Thus the uncertainty along $\boldsymbol{e}_1$, denoted as Δ_x, is studied. In outdoor experiments, the same flight mission is conducted within a natural wind field. The maximum wind speed is up to $5\ m/s$.

Apart from the EVOLVER, several methods (mainly focus on online paradigm) are implemented as comparisons: the baseline controller, the normal disturbance observer-based method [20], and the WSO-based method [36]. Technical details for all compared methods are listed as follows.

- EVOLVER: In the indoor experiment, the wind field produced by two fans is approximately constant. Thus the influence caused by $\boldsymbol{l}_u$ is disregarded. The lifting function $\mathbf{S}$ is chosen as $[\Delta_x, v_x, \theta, \|v_x\|v_x, \Delta_x\theta, qv_x]^T$, where v_x, θ, and q denote the velocity along $\boldsymbol{e}_1$, the pitch, and the attitude velocity along $\boldsymbol{b}_2$ of the quadrotor, respectively. In the outdoor experiment, considering the time-varying property of wind speed, according to Section IV-E more historical uncertainty states (n_Δ is set as 2) are added in the lifting function to attenuate the influence of $\boldsymbol{l}_u$. Algorithm 1 is implemented as the quadrotor begins its flight trajectory. The related parameters of the EVOLVER are summarized in Table. I. Since the diversity of the dataset can be guaranteed by setting T_1 greater than the trajectory period, we disenable the dataset construction process in this example.
- The baseline controller: The controller presented in Fig. 8**B** without the uncertainty observer is implemented. In the absence of the wind disturbance, the tracking error can reach $6\ cm$ (RMSE) at $4\ m/s$ and $13.23\ m/s^2$.
- The normal disturbance observer-based method [20]: With the assumption of $\dot{\boldsymbol{\Delta}} = 0$, the observer (19) is implemented to estimate the wind disturbance.
- The WSO-based method [36]: The wind disturbance is modeled as $\boldsymbol{F}_w = -\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}_a = -\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v} + \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}_w \in \mathbb{R}^3$ [34], where $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ represents drag coefficients, $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ denotes rotation matrix from body frame to inertial frame, $\boldsymbol{v}_a$ denotes the relative airspeed, and $\boldsymbol{v}_w$ denotes the wind speed. In [36] a drag utilization mathod is employed to utilize the part $-\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}$ and a wind speed observer (WSO) is designed to compensate the part $\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}_w$. The knowledge of $-\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T$ is utilized in the WSO, reducing the conservativeness of the normal disturbance observer-based method. By fitting prior data, the drag coefficient $\boldsymbol{D}$ can be obtained, which is shown in Table. I.

To make a fair comparison, the observer gains of all compared methods are set as the same. The positions of the quadrotor are measured by a motion capturing system (indoor) and a Real Time Kinematic (RTK) positioning system (outdoor), which are transmitted to the quadrotor through an ultra-wideband (UWB) transmission module at 50 Hz. The angular rate is measured by onboard IMU at 500 Hz. The whole control algorithm shown in Fig. 2 is implemented on an STM32F7 processor.

3) Experimental Results: Fig. 8. **C** to **G** show the indoor experimental results of the quadrotor. The trajectory tracking performances in the steady state of all implemented methods are presented in Fig. 8**C** and **E**. The transient performance of the EVOLVER is omitted which is similar to the normal disturbance observer-based one. It can be seen that the trajectory shifts a lot along the wind direction under the baseline controller. As the normal disturbance observer is enabled, the wind disturbance is partially attenuated. However, since the normal disturbance observer is only suitable for the slow variable disturbance, the wind disturbance cannot be accurately estimated when the attitude of the quadrotor changes quickly. After the knowledge of $-\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T$ is utilized, i.e., the WSO is activated, the trajectory tracking performance is improved. However, two drawbacks exist when employing the WSO. The first is the drag coefficient $\boldsymbol{D}$ needs to be obtained beforehand. The other is that the first-order disturbance model $\boldsymbol{F}_w$ is simplified. There may be other factors that affect the wind disturbance such as the velocity and the attitude velocity of the quadrotor. The EVOLVER addresses

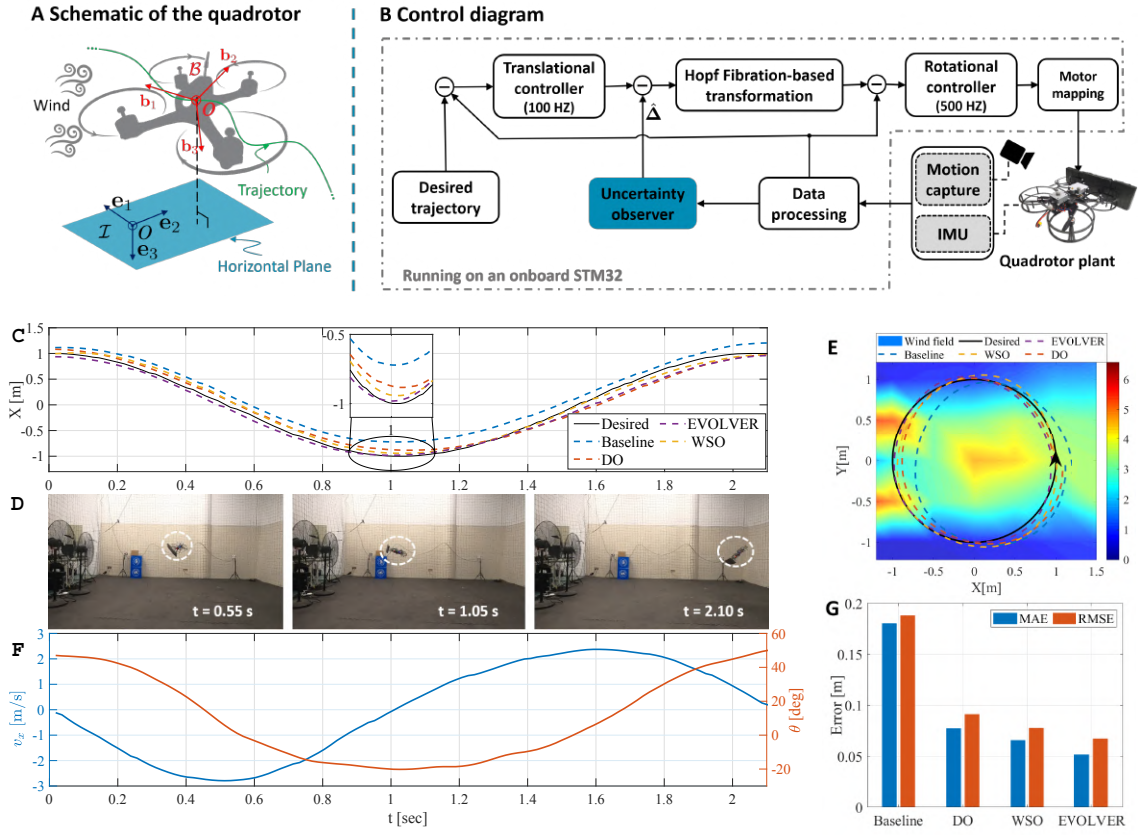

Fig. 8. Experimental results in the quadrotor example. (**A**) Schematic of the employed 6-inch quadrotor. (**B**) The control diagram. The baseline controller in our previous work [36] is employed, which consists of a PD translational controller with an acceleration feedforward, a Hopf fibration-based transformation [68], and a geometric rotational controller. The wind disturbance is estimated by the designed uncertainty observer and is subsequently compensated in the baseline controller. All control algorithms are implemented on an onboard low-computing-power processor. (**C**) The tracking performance in the direction of $\boldsymbol{e}_1$ of all compared methods: the proposed EVOLVER, the baseline controller, the normal disturbance observer-based method [20], and the WSO-based method [36], abbreviated to EVOLVER, baseline, DO, and WSO, respectively. (**D**) Three snapshots of the agile flight. (**E**) The 2-D tracking performance of all compared methods in the wind field. (**F**) The evolution of measured velocity and attitude along $\boldsymbol{e}_1$. The maximum attitude can reach $50°$. (**G**) The quantitatively compared result of all implemented methods over four laps.

the aforementioned issues. The flight trajectories and the preset circle path overlap more precisely as EVOLVER works. The quantitatively compared result is presented in Fig. 8**G**, which evaluates MAE and RMSE of all methods over four laps.

Three snapshots of the agile flight are presented in Fig. 8**D**. Moreover, Fig. 8**F** shows the evolution of the pitch θ and the velocity v_x of the quadrotor. In the flight test, the maximum pitch can reach $50°$. Fig. 9 shows the outdoor experimental results. The violin plot in Fig. 9**B** further demonstrates the effectiveness of the EVOLVER, compared with the normal disturbance observer-based method [20], with 39.5% improvement in MSE. The evolution of the pitch θ and the velocity v_x are depicted in Fig. 9**C**.

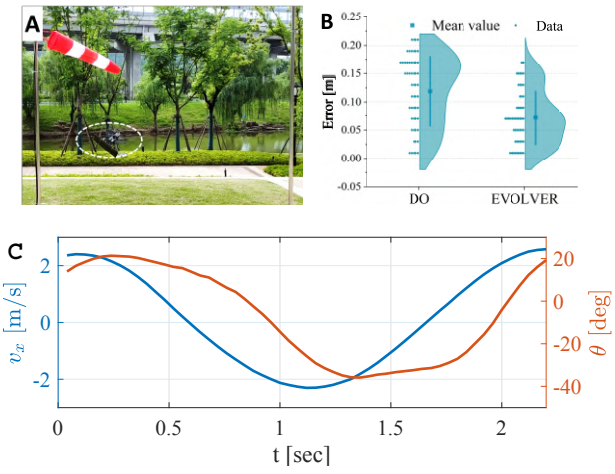

Fig. 9. Outdoor experimental result of the quadrotor example. (**A**) The outdoor experimental scenario. (**B**) The quantitative comparison of implemented methods (the proposed EVOLVER and the normal disturbance observer-based method [20], abbreviated to EVOLVER and DO, respectively). (**C**) The evolution of measured velocity and attitude along $\boldsymbol{e}_1$.

D. Application to A Manipulator

Finally, the end-effector trajectory tracking of the manipulator subject to unwanted base motion is considered. In this mission, the proposed EVOLVER is employed to estimate the uncertainty induced by the base motion, which belongs to the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}\left(\boldsymbol{\Delta}, \boldsymbol{x}, \boldsymbol{\ell}_k, \boldsymbol{\ell}_u\right)$.

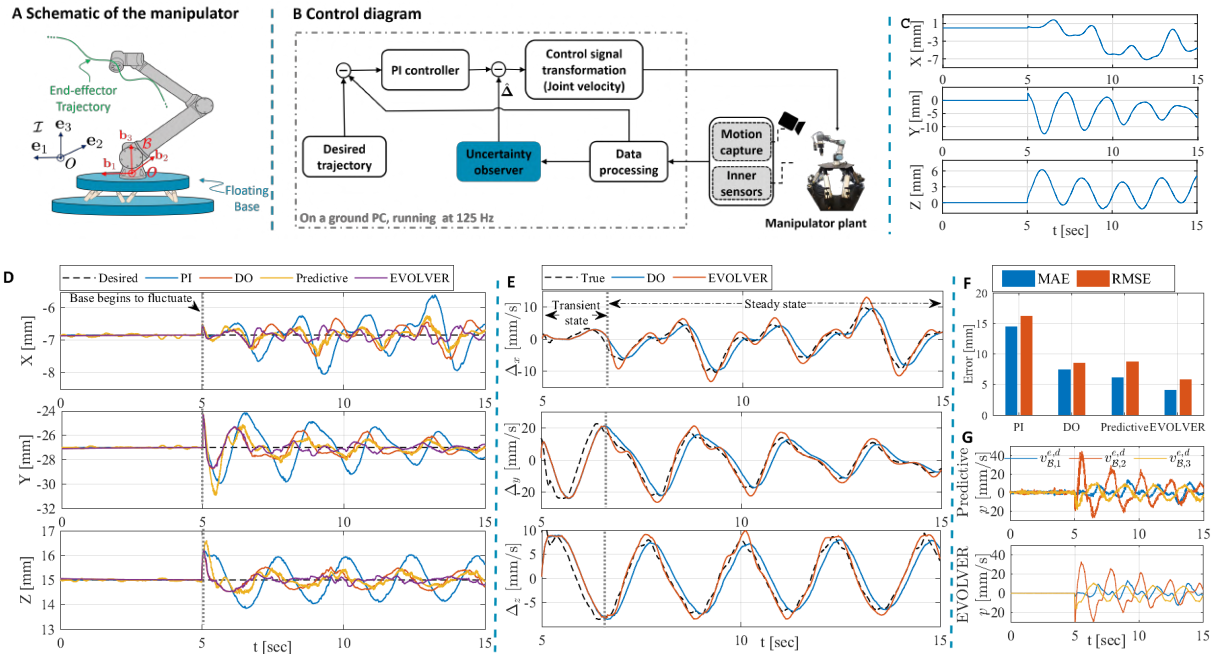

Fig. 10. Experimental results of *Scenario* #1 in the manipulator example. (**A**) Schematic of the employed UR-CB3 manipulator. (**B**) The control diagram. The PI baseline controller is employed. The position and attitude of the base are measured by the external motion capture system, and the position and attitude of the end-effector are obtained from the inner sensors. The desired joint velocity from the controller is sent to the manipulator finally. (**C**) The translational motion of a real AUV while under wave excitation in shallow water, which is extracted from Fig. 2 of [54]. (**D**) Tracking performance of the end-effector under different schemes: the proposed EVOLVER, the PI controller, the normal disturbance observer-based method [20], and the predictive control method [54], abbreviated to EVOLVER, PI, DO, and Predictive, respectively. (**E**) The estimation uncertainties of the normal disturbance observer-based method and the proposed EVOLVER. (**F**) The quantitatively compared result of all implemented methods. (**G**) The controller outputs $\boldsymbol{v}_{\mathcal{B}}^{e,d} = \left[v_{\mathcal{B},1}^{e,d}, v_{\mathcal{B},2}^{e,d}, v_{\mathcal{B},3}^{e,d}\right]^T$ of the predictive control method and the proposed EVOLVER.

1) Problem Formulation: As shown in Fig. 10**A**, the position of the end-effector $\boldsymbol{p}_{\mathcal{I}}^{e} \in \mathbb{R}^3$ expressed in the inertial frame $\mathcal{I}$ can be formalized as

$$\boldsymbol{p}_{\mathcal{I}}^{e} = \boldsymbol{p}_{\mathcal{I}}^{b} + \boldsymbol{R}\boldsymbol{p}_{\mathcal{B}}^{e}, \tag{31}$$

where $\boldsymbol{p}_{\mathcal{I}}^{b} \in \mathbb{R}^3$ denotes the position of the base expressed in $\mathcal{I}$, $\boldsymbol{p}_{\mathcal{B}}^{e} \in \mathbb{R}^3$ represents the position of the end-effector expressed in the base frame $\mathcal{B}$, and $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix from body/base frame to inertial frame.

The objective is to make sure that the position of the end-effector $\boldsymbol{p}_{\mathcal{I}}^{e}$ tracks a given desired trajectory $\boldsymbol{p}_{\mathcal{I}}^{e,d} \in \mathbb{R}^3$ by controlling the joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_q}$ for a n_q-degrees of freedom (DOF) manipulator. Thus the kinematics of the manipulator is needed. By differentiating (31), it can be obtained that

$$\boldsymbol{v}_{\mathcal{I}}^{e} = \boldsymbol{R}\boldsymbol{v}_{\mathcal{B}}^{e} + \boldsymbol{v}_{\mathcal{I}}^{b} + \boldsymbol{\omega}_{\mathcal{B}} \times \boldsymbol{R}\boldsymbol{p}_{\mathcal{B}}^{e}, \tag{32}$$

where $\boldsymbol{v}_{\mathcal{I}}^{e} \in \mathbb{R}^3$ denotes the velocity of the end-effector expressed in $\mathcal{I}$, $\boldsymbol{v}_{\mathcal{B}}^{e} \in \mathbb{R}^3$ is the velocity of the end-effector expressed in $\mathcal{B}$, $\boldsymbol{v}_{\mathcal{I}}^{b} \in \mathbb{R}^3$ represents the velocity of the base expressed in $\mathcal{I}$, and $\boldsymbol{\omega}_{\mathcal{B}} \in \mathbb{R}^3$ is the attitude velocity of the base expressed in $\mathcal{B}$.

Similar to [54], we assume the position and attitude of the base can be measured, whereas their derivatives cannot be obtained accurately. In other words $\boldsymbol{v}_{\mathcal{I}}^{b} + \boldsymbol{\omega}_{\mathcal{B}} \times \boldsymbol{R}\boldsymbol{p}_{\mathcal{B}}^{e}$ in (32) is unknown. Here $\boldsymbol{v}_{\mathcal{I}}^{b} + \boldsymbol{\omega}_{\mathcal{B}} \times \boldsymbol{R}\boldsymbol{p}_{\mathcal{B}}^{e}$ is treated as the uncertainty $\boldsymbol{\Delta}$, which belongs to the type of $\dot{\boldsymbol{\Delta}} = \boldsymbol{h}\left(\boldsymbol{\Delta}, \boldsymbol{x}, \boldsymbol{\ell}_k, \boldsymbol{\ell}_u\right)$. The EVOLVER is employed to learn it with $\boldsymbol{x} = \{\boldsymbol{p}_{\mathcal{B}}^{e}\}$, $\boldsymbol{\ell}_k = \{\boldsymbol{R}\}$, and $\boldsymbol{\ell}_u = \{\boldsymbol{v}_{\mathcal{I}}^{b}, \boldsymbol{\omega}_{\mathcal{B}}, \cdots\}$ in (2). Then (32) can be rewritten as

$$\boldsymbol{v}_{\mathcal{I}}^{e} = \boldsymbol{R}\boldsymbol{v}_{\mathcal{B}}^{e} + \boldsymbol{\Delta}. \tag{33}$$

2) Controller Design: The overall control framework for the manipulator is illustrated in Fig. 10**B**. Define the tracking error of the end-effector as $\boldsymbol{e}_p = \boldsymbol{p}_{\mathcal{I}}^{e,d} - \boldsymbol{p}_{\mathcal{I}}^{e}$. By recalling (33), the proportional-integral (PI) controller is designed as

$$\boldsymbol{v}_{\mathcal{B}}^{e,d} = \boldsymbol{R}^T\left(\boldsymbol{v}_{\mathcal{I}}^{e,d} + \boldsymbol{K}_p\boldsymbol{e}_p + \boldsymbol{K}_i\int \boldsymbol{e}_p\, dt - \hat{\boldsymbol{\Delta}}\right), \tag{34}$$

where $\boldsymbol{v}_{\mathcal{B}}^{e,d} \in \mathbb{R}^3$ and $\boldsymbol{v}_{\mathcal{I}}^{e,d} \in \mathbb{R}^3$ denote the desired velocities of the end-effector expressed in $\mathcal{B}$ and $\mathcal{I}$, respectively. $\boldsymbol{K}_p \in \mathbb{R}^{3\times3}$ and $\boldsymbol{K}_i \in \mathbb{R}^{3\times3}$ are the control gains of the PI controller.

In consequence, the desired joint velocity $\dot{\boldsymbol{q}}^d \in \mathbb{R}^{n_q}$ is rendered as

$$\dot{\boldsymbol{q}}^d = \boldsymbol{J}^{\dagger}\left[\boldsymbol{v}_{\mathcal{B}}^{e,d^T}, \boldsymbol{\omega}^{e,d^T}\right]^T, \tag{35}$$

where $\boldsymbol{J} \in \mathbb{R}^{n_q\times n_q}$ is the Jacobian matrix, and $\boldsymbol{\omega}^{e,d} \in \mathbb{R}^3$ denotes the desired attitude velocity of the end-effector.

In experiment, the control input $\boldsymbol{\omega}^{e,d}$ of the attitude loop in (35) is also designed as the PI controller [54]. The attitude anti-disturbance problem of the end-effector is not specifically considered to simplify the process. Our proposed EVOLVER can be easily transformed to the attitude loop by referring to the design process of the position loop.

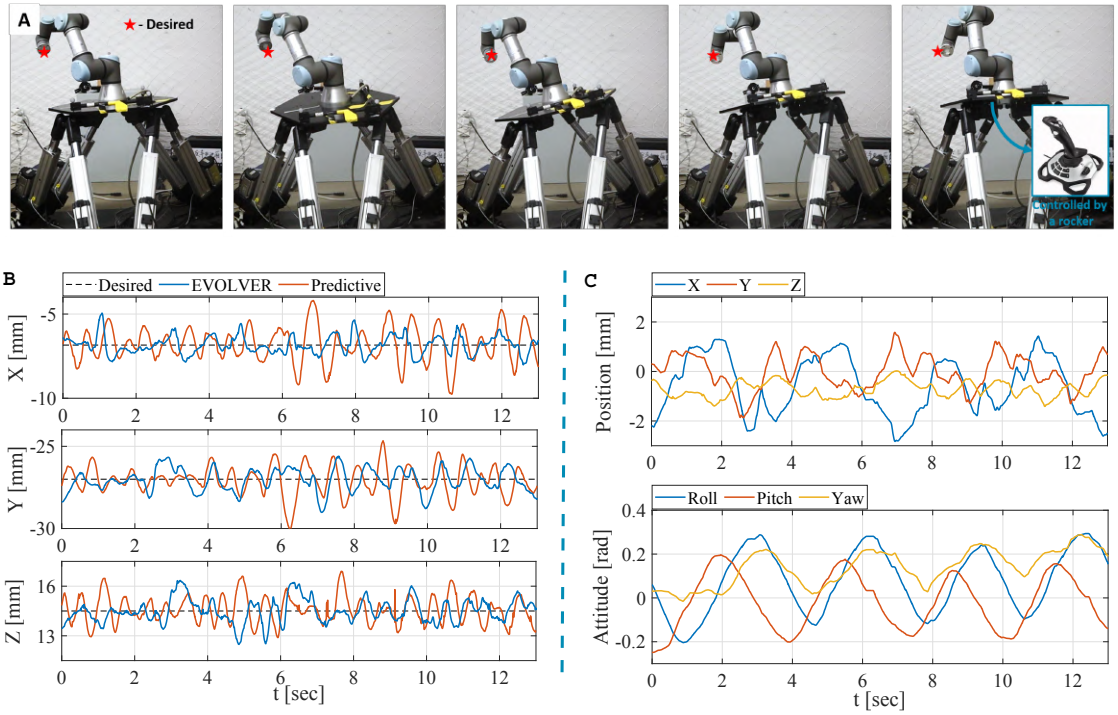

Fig. 11. Experimental results of *Scenario* #2 in the manipulator example. (**A**) Five snapshots of the operating manipulator on a parallel 6-DOF floating base controller by a 6-DOF rocker. (**B**) Steady state tracking performance of the end-effector under different schemes: the proposed EVOLVER and the predictive control method [54], abbreviated to EVOLVER and Predictive, respectively. (**C**) Real 6-DOF moving trajectory of the base.

3) Implementation Details: The UR-CB3 manipulator (Fig. 1) with 6-DOF (i.e., $n_d = 6$) is employed in this test. The end-effector is commanded to be fixed at a certain position with respect to the inertial frame $\mathcal{I}$. The joint velocity control model is employed. The states of the base and the end-effector are measured by the motion capture system and the inner sensors of the manipulator, respectively. All the involved schemes are implemented in Python running on a PC with Inter(R) Core(TM) i7-8700 CPU.

Several comparison methods are also implemented to demonstrate the advantages of the proposed EVOLVER: the PI controller, the normal disturbance observer-based method [20], and the predictive end-effector control method [54]. Technical details for the chosen methods are listed as follows.

- EVOLVER: According to the property of $\boldsymbol{\Delta}$ in (33), the lifting functions $\mathbf{S}$ are chosen as $\left[\boldsymbol{\Delta}^T, \boldsymbol{\Delta}_{-1}^T, \cdots, \boldsymbol{\Delta}_{-n_\Delta}^T, \boldsymbol{p}_{\mathcal{I}}^{b}\right]^T$. Algorithm 1 is implemented as the base begins to move. The related parameters of the EVOLVER are listed in Table. I. According to the quasi-periodic property of the autonomous underwater vehicle (AUV) motion (See Fig. 10**C**), the diversity of the dataset can be guaranteed by setting T_1 greater than the approximate period. Thus the dataset construction process in this example is disabled in real test. The dataset construction process is invoked when evaluating the computational cost.
- The PI controller: The controller (34) without considering uncertainty compensation (i.e., $\hat{\boldsymbol{\Delta}} = 0$) is implemented.
- The normal disturbance observer-based method [20]: With the assumption of $\dot{\boldsymbol{\Delta}} = 0$, the observer (19) is implemented to estimate uncertainty induced by the base movement. Then the estimated uncertainty is compensated in the PI baseline controller (34). The observer gain is set as same as the EVOLVER.
- The predictive control method [54]: The base movement is forecasted by linearly fitting the historical data, and the desired state of the end-effector with respect to the base frame is predicted. The end-effector tracking problem is finally formalized as a MPC framework. Consistent with our control scheme, the control signal to the manipulator is the joint velocity. The controller parameters are set as same as the unconstrained situation from [54]. For the parameters unmentioned in [54], the number of historical base data selected for fitting is set as $n_\Delta = 5$ in keeping with the EVOLVER, and the forecast horizon is set as 10.

Two scenarios are arranged to carry out the tests.

Scenario #1: In pursuit of authenticity, the base disturbance simulates a real autonomous AUV motion while under wave excitation in shallow water as shown in Fig. 10**C**, which is collected from [54]. The base disturbance is introduced at $t = 5\ s$. In pursuit of comparison fairness, the base motion needs to be accurately reproduced for all test methods. Whereas, the employed 6-DOF floating base can only be controlled by a rocker. In this scenario, we impose the base disturbance in a virtual way. Imagine the base positioning system is moving, i.e., the base is fixed with respect to the ground station and the inertial frame is commanded to follow the opposite trajectory of Fig. 10**C**. In such a way, the position of the end-effector

must follow the opposite trajectory of Fig. 10**C**, i.e., at rest with the moving inertial system.

Scenario #2: A more challenging condition is considered to examine the applicability of the proposed scheme. The manipulator is commanded to operate on a parallel 6-DOF floating base. A 6-DOF rocker is used to control the translational and rotational movements of the base, as shown in Fig. 11**A**. In the test, the EVOLVER method and the predictive control method [54] are implemented. The parallel 6-DOF floating base in our lab is controlled by the 6-DOF rocker manually. The perturbations are not guaranteed to be 100% reproducible. In the experiment, the perturbations with similar magnitude and frequency are tried to be applied for different methods. The attitude prediction strategy proposed in [54] is considered in the predictive control method. For the EVOLVER, the lifting functions are selected as in *Scenario* 1 (i.e., without considering the attitude information of the floating base), to illustrate the robustness of the proposed framework.

4) Experimental Results: The experimental results of the two scenarios are shown in Fig. 10 and Fig. 11, respectively.

Scenario #1: Fig. 10**D** presents the tracking performance of the end-effector under different control schemes and Fig. 10**F** shows the quantitative comparison results. In the presence of the same uncertainty, our proposed EVOLVER excels the PI control method, the normal disturbance observer-based method, and the predictive control method, with 71.8%, 45.8%, and 33.9% improvement in MSE, respectively.

The estimated uncertainties by the normal disturbance observer-based method and the EVOLVER are depicted in Fig. 10**E**. Notice that the truth value of the uncertainty is fitted offline using the polynomial fitting method. At the transient stage, the estimation performances of both methods are the same with a certain delay. After the learned uncertainty model is employed at $t = 6.6\ s$, the estimation delay is eliminated immediately, which confirms that the EVOLVER has successfully predicted the motion of the base. Moreover, the controller outputs of the predictive control method and the EVOLVER are provided in Fig. 10**G**. In comparison with the predictive control method, the EVOLVER can output smoother control signals to achieve better performance. The output of the predictive control is chattering, which is harmful to the actuators. Part of the reason can be attributed to the discontinuity of the predicted uncertainty, where the coefficients are discontinuously updated by linear LS.

Scenario #2: Fig. 1**C** and Fig. 11**A** present the long-exposure photo and five snapshots of the manipulator response with the EVOLVER method, respectively. Even though the base is moving randomly with 6-DOF (the real moving trajectory of the base is provided in Fig. 11**C**), the end-effector can still maintain at the desired position. The steady state performance is shown in Fig. 11**B**. The MSE and RMSE of the predictive control method are $0.013\ m$ and $0.015\ m$, respectively. By contrast, the MSE and RMSE of the EVOLVER are $0.009\ m$ and $0.010\ m$, which improve 30.8% and 33.3% as compared to the predictive control method, respectively.

Finally, Fig. 12 exhibits the computational cost ratio of various modules. Especially, the dataset construction process designed in Section V-A has an acceptable computational cost of 10.1%, which is suitable for the online situation.

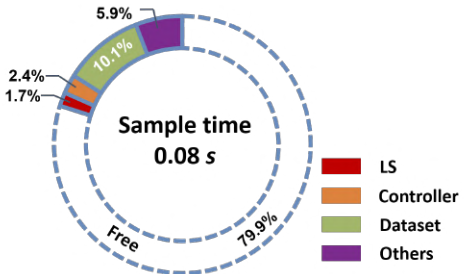

Fig. 12. Computational cost ratio of various modules. Controller: the PI baseline controller (34) and the observer (19). Dataset: the robust differentiator (22) and dataset construction flow stated in Algorithm 2. LS: the LS solving step (11) and the continuous step (14). Others: signal filtering and so on.

VII. Conclusion

In this article, we have made significant steps towards online learning and prediction of disturbances for various robotic systems. We achieve this by using the Koopman operator to model the unknown uncertainty and substituting the learned model into an evolutionary uncertainty observer. The presented scheme enables rapid transient reaction ability and high precision steady state tracking performance. The convergence guarantee can be preserved in several optimal conditions. To achieve these optimal requirements in real applications, practical solutions including training dataset construction, data noise handling, and lifting functions selection are provided. Finally, the proposed framework is demonstrated by sufficient simulation and experiments: stable control of a second-order *Newton* system with the chaotic *Lorenz* uncertainty, trajectory prediction of an irregular free-flying object under the aerodynamic drag, agile flight of the quadrotor under the wind disturbance, and end-effector control of the manipulator under unwanted base moving disturbance.

Appendix

A. The Reasonability and Limitation of Assumption 1

The reasonability and limitation of Assumption 1 are analyzed by combining the studied examples.

Reasonability. In the Lorentz example, from (25), it can be seen that the injected Lorentz uncertainty satisfies Assumption 1. In the irregular object and quadrotor examples, the uncertainties mainly result from the air drag, which can be roughly modeled as $\boldsymbol{F}_w = -\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}_a = -\boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v} + \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T\boldsymbol{v}_w \in \mathbb{R}^3$ [34]–[36], where $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ represents drag coefficients, $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ denotes rotation matrix from body frame to inertial frame, $\boldsymbol{v}_a$ denotes the relative airspeed, $\boldsymbol{v}$ denotes the velocity, and $\boldsymbol{v}_w$ denotes the wind speed. In the irregular object example ($\boldsymbol{v}_w = 0$), the system states of the object are bounded in reality. Assumption 1 is always held in this case.

In the quadrotor example ($\boldsymbol{v}_w \neq 0$), the air drag depends on the translational and rotational states, and the external wind speed. The boundness of the rotational states can be guaranteed by a stable rotational controller (e.g., the geometric controller [23]). For the translational state, it can be initially

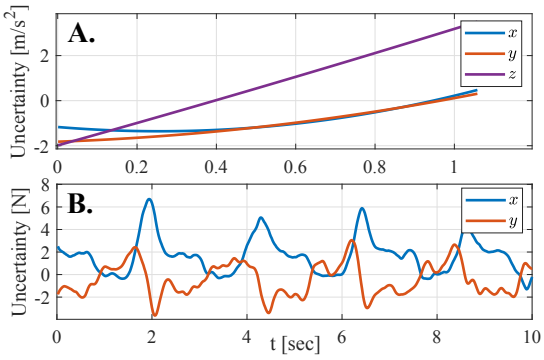

Fig. 13. The truth uncertainties in the object trajectory prediction and quadrotor examples under baseline control/prediction algorithms. Notice that the truth uncertainties are fitted offline by polynomial fitting. (**A**) The object trajectory prediction example. (**B**) The quadrotor example.

bounded by a baseline translational controller (e.g., the PD controller) despite of a large tracking error. After employing EVOLVER, with the decreasing of the uncertainty estimation error at each step, the change of translational state would not break through Assumption 1. Finally, we only need to hold the assumption that the variation of wind speed is bounded, which is acceptable in practice. With respect to the manipulator example, by following a similar analysis, Assumption 1 finally requires the variation of the base floating to be bounded.

The true uncertainties of experimental examples under baseline control/prediction algorithms are provided in Fig. 13**A**, Fig. 13**B**, and Fig. 10**E**, respectively, which can certify the reasonability of Assumption 1 from a practical aspect.

Limitation. Assumption 1 is not applicable to the disturbance like push or collision. The future work will pursue separating the part related to the system state from uncertainty based on *Chebyshev* polynomials [69], and reduce the conservativeness of Assumption 1.

B. Proof of Lemma 1

Proof. Differentiate $\tilde{\boldsymbol{\Delta}}$. From (6), it can be implied that

$$\begin{aligned}\dot{\tilde{\boldsymbol{\Delta}}} =&\dot{\boldsymbol{\Delta}} - \left(\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{p}}\left(\boldsymbol{x}\right)\right)\\ =&\dot{\boldsymbol{\Delta}} + \boldsymbol{L}\dot{\boldsymbol{x}} - \boldsymbol{L}\left(\boldsymbol{f}_x\left(\boldsymbol{x}\right) + \boldsymbol{f}_u\left(\boldsymbol{x}\right)\boldsymbol{u} + \hat{\boldsymbol{\Delta}}\right)\\ \overset{(a)}{=}&\dot{\boldsymbol{\Delta}} + \boldsymbol{L}\tilde{\boldsymbol{\Delta}},\end{aligned} \tag{36}$$

where (a) is obtained by using (1). Define the *Lyapunov* candidate $V_1 = \frac{1}{2}\tilde{\boldsymbol{\Delta}}^T\tilde{\boldsymbol{\Delta}}$. Differentiating V_1, one can obtain

$$\begin{aligned}\dot{V}_1 =&\tilde{\boldsymbol{\Delta}}^T\boldsymbol{L}\tilde{\boldsymbol{\Delta}} + \tilde{\boldsymbol{\Delta}}^T\dot{\boldsymbol{\Delta}}\\ \overset{(b)}{\leq}&\delta\tilde{\boldsymbol{\Delta}}^T\tilde{\boldsymbol{\Delta}} + \epsilon\left\|\dot{\boldsymbol{\Delta}}\right\|^2\\ \overset{(c)}{\leq}&\delta\tilde{\boldsymbol{\Delta}}^T\tilde{\boldsymbol{\Delta}} + \epsilon\bar{\Delta}^2,\end{aligned} \tag{37}$$

where ϵ is an arbitrarily positive number, $\delta = \lambda_M\left(\boldsymbol{L}\right) + \frac{1}{4\epsilon}$, (b) results from the conditions $\tilde{\boldsymbol{\Delta}}^T\boldsymbol{L}\tilde{\boldsymbol{\Delta}} \leq \lambda_M\left(\boldsymbol{L}\right)\tilde{\boldsymbol{\Delta}}^T\tilde{\boldsymbol{\Delta}}$ and $\tilde{\boldsymbol{\Delta}}^T\dot{\boldsymbol{\Delta}} \leq \frac{1}{4\epsilon}\tilde{\boldsymbol{\Delta}}^T\tilde{\boldsymbol{\Delta}} + \epsilon\left\|\dot{\boldsymbol{\Delta}}\right\|^2$ (*Young Inequality*), and (c) results from Assumption 1. From the definition of V_1, it can be further rendered that

$$\dot{V}_1 \leq 2\delta V_1 + \epsilon\bar{\Delta}^2. \tag{38}$$

Solving the first-order ordinary differential inequality, one can obtain that

$$0 \leq V_1 \leq \boldsymbol{e}^{2\delta t}\left[V_1\left(0\right) + \frac{\epsilon\bar{\Delta}^2}{2\delta}\right] - \frac{\epsilon\bar{\Delta}^2}{2\delta}. \tag{39}$$

Thus, if $\delta < 0$, i.e., $\lambda_M\left(\boldsymbol{L}\right) < -\frac{1}{4\epsilon}$, we have

$$0 \leq \left\|\tilde{\boldsymbol{\Delta}}\right\|^2 \leq 2\boldsymbol{e}^{-2|\delta|t}\left[V_1\left(0\right) + \frac{\epsilon\bar{\Delta}^2}{2\delta}\right] - \frac{\epsilon\bar{\Delta}^2}{\delta}, \tag{40}$$

resulting in $\lim_{t\to\infty}\left\|\tilde{\boldsymbol{\Delta}}\right\| \leq \sqrt{\frac{\epsilon\bar{\Delta}^2}{|\delta|}}$. In other words, the estimated error of model uncertainty can converge to the set $\sqrt{\frac{\epsilon\bar{\Delta}^2}{|\delta|}}$. ■

C. Proof of Lemma 3

Proof. The proof procedure is similar to Lemma 1. Combing (17) and (19), it can be implied that

$$0 \leq \left\|\tilde{\boldsymbol{\Delta}}\right\|^2 \leq 2\boldsymbol{e}^{-2|\delta|t}\left[V_1\left(0\right) + \frac{\epsilon\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{2\delta}\right] - \frac{\epsilon\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{\delta}. \tag{41}$$

If the conditions in Lemma 2 are satisfied and $\boldsymbol{\ell}_u$ is neglected, $\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}} = 0$ will be achieved. From (41), it is obvious that the estimated error of model uncertainty can converge to zero as $\delta < 0$, i.e., $\lambda_M\left(\boldsymbol{L}\right) < -\frac{1}{4\epsilon}$. Here, as $\epsilon \to \infty$, the required condition can be relaxed to $\boldsymbol{L}$ being negative definite. ■

D. Proof of Theorem 1

Proof. The whole procedure takes place over several discrete stages. The arbitrarily positive number ϵ is chosen as 1.
(i) $t \leq T_1$: From Lemma 1, it has been obtained

$$\left\|\tilde{\boldsymbol{\Delta}}(t)\right\|^2 \leq 2\boldsymbol{e}^{-2|\delta|t}\left[V_1\left(0\right) + \frac{\bar{\Delta}^2}{2\delta}\right] - \frac{\bar{\Delta}^2}{\delta}. \tag{42}$$

(ii) $T_1 < t \leq T_1 + T_2$: The learned uncertainty model is updated for the first time (i.e., $\boldsymbol{A}_1$ in *lines* 12 of Algorithm 1). Similar to (41), it can be rendered that

$$\begin{aligned}\left\|\tilde{\boldsymbol{\Delta}}(t)\right\|^2 \leq&2\boldsymbol{e}^{-2|\delta|t}\left[V_1\left(T_1\right) + \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{2\delta}\right] - \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{\delta}\\ =&2\boldsymbol{e}^{-2|\delta|t}\left[\frac{1}{2}\left\|\tilde{\boldsymbol{\Delta}}(T_1)\right\|^2 + \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{2\delta}\right] - \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{\delta}.\end{aligned} \tag{43}$$

(iii) $T_1 + T_2 < t \leq T_1 + 2T_2$: The learned uncertainty model is updated for the second time after a cycle T_2. We have

$$\begin{aligned}&\left\|\tilde{\boldsymbol{\Delta}}(t)\right\|^2\\ &\leq 2\boldsymbol{e}^{-2|\delta|t}\left[\frac{1}{2}\left\|\tilde{\boldsymbol{\Delta}}(T_1+T_2)\right\|^2 + \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{2\delta}\right] - \frac{\|\boldsymbol{C}\boldsymbol{\mathfrak{r}}_{\boldsymbol{3}}\|^2}{\delta}.\end{aligned} \tag{44}$$

(iv) $T_1 + jT_2 < t \leq T_1 + (j+1)T_2$: The learned uncertainty model is updated for the j-th time (i.e., $\boldsymbol{A}_j$ in $lines$ 16 of Algorithm 1). We have

$$\begin{aligned}&\left\|\tilde{\boldsymbol{\Delta}}(t)\right\|^2\\&\leq 2\boldsymbol{e}^{-2|\delta|t}\left[\frac{1}{2}\left\|\tilde{\boldsymbol{\Delta}}(T_1+jT_2)\right\|^2+\frac{\|\boldsymbol{C}\mathfrak{r}_{\boldsymbol{3}}\|^2}{2\delta}\right]-\frac{\|\boldsymbol{C}\mathfrak{r}_{\boldsymbol{3}}\|^2}{\delta}.\end{aligned} \tag{45}$$

The above procedure obeys two principles. One is that the initial estimation error in each period is the final estimation error of the last period. The other one is that the estimation error is asymptotically convergent during each period. Although the updates of the learned uncertainty model are discrete, the estimation error $\tilde{\boldsymbol{\Delta}}$ is continuous and asymptotically convergent. If the conditions in Lemma 2 are all satisfied and $\boldsymbol{\ell}_u$ is neglected, $\mathfrak{r}_{\boldsymbol{3}}$ will be zero. It is obvious that the uncertainty estimated error $\tilde{\boldsymbol{\Delta}}$ can converge to zero as $t \to \infty$.

Moreover, if $\boldsymbol{\ell}_u$ is considered, $\tilde{\boldsymbol{\Delta}}$ will converge to a bounded set $\sqrt{\frac{\|\boldsymbol{C}\mathfrak{r}_{\boldsymbol{3}}\|^2}{|\delta|}}$, which can be arbitrarily small by increasing absolute value of the eigenvalues of observer. ■

REFERENCES

[1] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Courier Corporation, 2014.

[2] I. S. Khalil, J. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.

[3] J.-M. Yang and J.-H. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 578–587, 1999.

[4] R. Sanner and J.-J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.

[5] L. Guo and W. Chen, "Disturbance attenuation and rejection for systems with nonlinearity via dobc approach," *International Journal of Robust and Nonlinear Control*, vol. 15, no. 3, pp. 109–125, 2005.

[6] J. Jia, K. Guo, X. Yu, L. Guo, and L. Xie, "Agile flight control under multiple disturbances for quadrotor: Algorithms and evaluation," *IEEE Transactions on Aerospace and Electronic Systems, early access*, 2022.

[7] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[8] P. Mitrano, D. McConachie, and D. Berenson, "Learning where to trust unreliable models in an unstructured world for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8170, 2021.

[9] H.-T. Nguyen, C. C. Cheah, and K.-A. Toh, "An analytic layer-wise deep learning framework with applications to robotics," *Automatica*, vol. 135, p. 110007, 2022.

[10] D. A. Markov, L. Petrucco, A. M. Kist, and R. Portugues, "A cerebellar internal model calibrates a feedback controller involved in sensorimotor control," *Nature Communications*, vol. 12, no. 1, pp. 1–21, 2021.

[11] E. Sariyildiz, R. Oboe, and K. Ohnishi, "Disturbance observer-based robust control and its applications: 35th anniversary overview," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2042–2053, 2020.

[12] C. P. Bechlioulis and G. A. Rovithakis, "Robust partial-state feedback prescribed performance control of cascade systems with unknown nonlinearities," *IEEE Transactions on Automatic Control*, vol. 56, no. 9, pp. 2224–2230, 2011.

[13] W. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methods—An overview," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1083–1095, 2016.

[14] X. Li, Q.-G. Wang, X. Li, K. K. Tan, and L. Xie, "Feedforward control with disturbance prediction for linear discrete-time systems," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 6, pp. 2340–2350, 2019.

[15] K. Ohishi, M. Nakao, K. Ohnishi, and K. Miyachi, "Microprocessor controlled DC motor for load-insensitive position servo system," *IEEE Transactions on Industrial Electronics*, vol. 34, no. 1, pp. 44–49, 1987.

[16] J. Han, "From PID to active disturbance rejection control," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 900–906, 2009.

[17] C. Johnson, "Further study of the linear regulator with disturbances–the case of vector disturbances satisfying a linear differential equation," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 222–228, 1970.

[18] H. Sira-Ramirez and M. A. Oliver-Salazar, "On the robust control of buck-converter DC-motor combinations," *IEEE Transactions on Power Electronics*, vol. 28, no. 8, pp. 3912–3922, 2013.

[19] L. B. Freidovich and H. K. Khalil, "Performance recovery of feedback-linearization-based designs," *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2324–2334, 2008.

[20] W. Chen, D. Ballance, P. Gawthrop, and J. O'Reilly, "A nonlinear disturbance observer for robotic manipulators," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 4, pp. 932–938, 2000.

[21] W. Chen, "Disturbance observer based control for nonlinear systems," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 4, pp. 706–710, 2004.

[22] K.-S. Kim, K.-H. Rew, and S. Kim, "Disturbance observer for estimating higher order disturbances in time series expansion," *IEEE Transactions on Automatic Control*, vol. 55, no. 8, pp. 1905–1911, 2010.

[23] M. Bisheban and T. Lee, "Geometric adaptive control with neural networks for a quadrotor in wind fields," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1533–1548, 2021.

[24] W. Chen and L. Guo, "Analysis of disturbance observer based control for nonlinear systems under disturbances with bounded variation," in *Proceedings of International Conference on Control*, 2004, pp. 1–5.

[25] Z. Gao, "Scaling and bandwidth-parameterization based controller tuning," in *Proceedings of the American Control Conference*, 2006, pp. 4989–4996.

[26] S. Singh, S. M. Richards, V. Sindhwani, J.-J. E. Slotine, and M. Pavone, "Learning stabilizable nonlinear dynamics with contraction-based regularization," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1123–1150, 2021.

[27] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[28] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[29] K. Karydis, I. Poulakakis, J. Sun, and H. G. Tanner, "Probabilistically valid stochastic extensions of deterministic models for systems with uncertainty," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1278–1295, 2015.

[30] V. R. Desaraju, A. E. Spitzer, C. O'Meadhra, L. Lieu, and N. Michael, "Leveraging experience for robust, adaptive nonlinear MPC on computationally constrained systems with time-varying state uncertainty," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1690–1712, 2018.

[31] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation*, 2019, pp. 6023–6029.

[32] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 4709–4715.

[33] M. Zheng, X. Lyu, X. Liang, and F. Zhang, "A generalized design method for learning-based disturbance observer," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 1, pp. 45–54, 2021.

[34] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel, "Nonlinear feedback control of quadrotors exploiting first-order drag effects," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8189–8195, 2017.

[35] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[36] J. Jia, K. Guo, X. Yu, W. Zhao, and L. Guo, "Accurate high-maneuvering trajectory tracking for quadrotors: A drag utilization method," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6966–6973, 2022.

[37] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, p. 315, 1931.

[38] A. Mesbahi, J. Bu, and M. Mesbahi, "Nonlinear observability via Koopman analysis: Characterizing the role of symmetry," *Automatica*, vol. 124, p. 109353, 2021.

[39] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *2016 IEEE 55th Conference on Decision and Control*, 2016, pp. 6492–6499.

[40] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using Koopman operator framework," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016.

[41] A. Broad, I. Abraham, T. Murphey, and B. Argall, "Data-driven Koopman operators for model-based shared control of human–machine systems," *The International Journal of Robotics Research*, vol. 39, no. 9, pp. 1178–1195, 2020.

[42] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing Koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[43] I. Mezić, "On applications of the spectral theory of the Koopman operator in dynamical systems and control theory," in *2015 54th IEEE Conference on Decision and Control*, 2015, pp. 7034–7041.

[44] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[45] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, 2016.

[46] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using Koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[47] G. Mamakoukas, M. L. Castaño, X. Tan, and T. D. Murphey, "Derivative-based Koopman operators for real-time control of robotic systems," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2173–2192, 2021.

[48] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using Koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2020.

[49] M. Haseli and J. Cortés, "Learning Koopman eigenfunctions and invariant subspaces from data: Symmetric subspace decomposition," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3442–3457, 2022.

[50] M. Korda and I. Mezić, "On convergence of extended dynamic mode decomposition to the Koopman operator," *Journal of Nonlinear Science*, vol. 28, no. 2, pp. 687–710, 2018.

[51] S. Klus, P. Koltai, and C. Schütte, "On the numerical approximation of the Perron-Frobenius and Koopman operator," *Journal of Computational Dynamics*, vol. 3, no. 1, pp. 51–79, 2016.

[52] L. Shi and K. Karydis, "ACD-EDMD: Analytical construction for dictionaries of lifting functions in Koopman operator-based nonlinear robotic systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 906–913, 2021.

[53] H. Shi and M. Q.-H. Meng, "Deep Koopman operator with control for nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7700–7707, 2022.

[54] J. Woolfrey, W. Lu, and D. Liu, "Predictive end-effector control of manipulators on moving platforms under disturbance," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2210–2217, 2021.

[55] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2013.

[56] R. Seeber, H. Haimovich, M. Horn, L. M. Fridman, and H. De Battista, "Robust exact differentiators with predefined convergence time," *Automatica*, vol. 134, p. 109858, 2021.

[57] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[58] E. Celeghini, M. Gadella, and M. A. del Olmo, "Hermite functions and fourier series," *Symmetry*, vol. 13, no. 5, p. 853, 2021.

[59] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1623–1629 vol.3.

[60] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Proceedings of 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5113–5120.

[61] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Proceedings of 2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 279–284.

[62] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, 2012.

[63] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[64] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[65] Y. Jia, M. Gardner, and X. Mu, "Batting an in-flight object to the target," *The International Journal of Robotics Research*, vol. 38, no. 4, pp. 451–485, 2019.

[66] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[67] H. Yu, D. Guo, H. Yin, A. Chen, K. Xu, Z. Chen, M. Wang, Q. Tan, Y. Wang, and R. Xiong, "Neural motion prediction for in-flight uneven object catching," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 4662–4669.

[68] M. Watterson and V. Kumar, "Control of quadrotors using the Hopf fibration on SO(3)," in *Proceedings of Robotics Research*, 2020, pp. 199–215.

[69] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, p. eabm6597, 2022.