A Novel Self-Organizing PID Approach for Controlling Mobile Robot Locomotion

Xiaowei Gu[1,2], *Member, IEEE*, Muhammad Aurangzeb Khan[1,2], Plamen Angelov[1,2,*], *Fellow, IEEE*, Bikash Tiwary[1], Elnaz Shafipour Yourdshah[1,2] and Zhao-Xu Yang[3]
[1]*School of Computing and Communications,Lancaster University, UK*
[2]*Lancaster Intelligent, Robotic and Autonomous Systems Centre (LIRA), Lancaster University, UK*
[3]*State Key Laboratory for Strength and Vibration of Mechanical Structures,*
Shaanxi Key Laboratory of Environment and Control for Flight Vehicle,
School of Aerospace, Xi'an Jiaotong University, Xi'an, People's Republic of China
* Email: p.angelov@lancaster.ac.uk

***Abstract*—A novel self-organizing fuzzy proportional–integral–derivative (SOF-PID) control system is proposed in this paper. The proposed system consists of a pair of control and reference models, both of which are implemented by a first-order autonomous learning multiple model (ALMMo) neuro-fuzzy system. The SOF-PID controller self-organizes and self-updates the structures and meta-parameters of both the control and reference models during the control process "on the fly". This gives the SOF-PID control system the capability of quickly adapting to entirely new operating environments without a full re-training. Moreover, the SOF-PID control system is free from user- and problem-specific parameters, and the uniform stability of the SOF-PID control system is theoretically guaranteed. Simulations and real-world experiments with mobile robots demonstrate the effectiveness and validity of the proposed SOF-PID control system.**

***Index Terms*—ALMMo neuro fuzzy system, PID controller, self-organizing system**

I. Introduction

Since being firstly introduced eighty years ago, proportional–integral–derivative (PID) controllers have been extensively used in industry and defense [1]–[3] for automation and process control thanks to their merits of, low costs, inexpensive maintenance, simplicity in structure and capability for accurate control. More importantly, the control parameters of the PID controllers have clear physical meaning [4], they are explainable and interpretable by human experts. The fundamentals of conventional PID controllers were summarized in [5].

However, the well-known drawbacks of the PID controllers are: 1) there are some strong recommendations and algorithm regarding PID tuning such as linear stability techniques [6], [7]; 2) the performance is fragile for nonlinear, complex and vague systems that have no precise mathematical models [8]; 3) the performance is fragile for new data patterns. The empirical approaches for parameter-tuning usually can ensure a satisfactory performance of the PID controller, but it requires heavy involvements of human expertise and a good understanding of the problem [6]. To address the second drawback, various types of modifications have been made, self-tuning adaptive PID controllers [9], [10] and fuzzy PID controllers [8], [11]–[16] have been developed and applied widely.

The architecture of adaptive controllers are pre-fixed during the design stage, and the self-tuning schemes concerns adjusting control parameters only [3], [11], [12], [17], [18]. Meanwhile, the vast majority of existing PID control models lack self-organizing structure that can be adaptive to entirely new environments. Therefore, in many real-world applications, e.g., autonomous driving and mobile robots, the performance of a well-tuned PID controller can significantly deteriorate when new situations appear. In other words, the adaptive (fuzzy) PID controllers are capable of dealing with the shifts in the data pattern, but are unable to handling the drifts [19]. More recently, self-evolving fuzzy rule-based controller [21], [39] was introduced, which is able to self-update the system architecture during the control process. In this paper, we propose to improve the evolving mechanisms of these self-evolving algorithms [22].

In this paper, we propose a self-organizing fuzzy PID (SOF-PID) control system. The SOF-PID control system is composed of two parts: i) reference model and ii) control model. Both models are implemented as autonomous learning multi-model (ALMMo) neuro-fuzzy systems [22]. A control system consisting of two models was firstly introduced in [23], where two neural networks are involved serving as the respective feedforward and feedback controllers. This type of structure inherits the advantages of both, feedforward and feedback controllers, and, generally, results in a better tracking performance [24] A number of control systems with similar architectures were introduced afterwards. [25] presents a generic algorithm-based feedforward-feedback multiple-input–multiple-output (MIMO) control system pressurized water reactor power plant. In [26], a feedforward-feedback fuzzy adaptive controller consisting of a pair of a self-adjusting feedforward fuzzy model and a pre-fixed error-feedback model is designed for multiple-input–multiple-

The first three authors contribute equally. This work is funded by Innovate UK under the project "Collaborative Technology Hardened for Underwater and Littoral Hazardous Environments" (Ref. 104061). * Corresponding Author: P. Angelov

output uncertain nonlinear systems. A temperature controller for of functionally graded plates is proposed in [27], which combines a feedforward inverse control model with a proportional–derivative (PD) controller for disturbance/noise attenuation. A robust evolving cloud-based controller is introduced in [28]. This approach is composed of an AnYa type fuzzy rule-based system [29], [30], which is used for heat-exchanger plant control, and a reference model, which is for producing the desired trajectory. Other successful implementation for real-world problems of feedforward-feedback control systems include, but not limited to sludge process pilot plant [31]; atomic force microscopes [32]; solid oxide fuel cell system [33]; piezoelectric-driven mechanism [24], etc. Nonetheless, the system structures of most existing approaches are designed by human experts based on prior knowledge of the problems. The control systems can be partially adjusted only, while the rest parts have been pre-fixed. Although, a properly designed control system usually demonstrates excellent performance under the experimental scenarios, the system structure and parameters can be less meaningful, and the performance will deteriorate when the operating environment is changed significantly.

ALMMo neuro-fuzzy system [22] is a new type of first-order multi-model system based on AnYa type (neuro-)fuzzy systems [29], [30]. Comparing with other more widely used types of (neuro-)fuzzy systems, e.g. Zadeh-Mamdani type [34], Takagi-Sugeno type [35], AnYa type models simplify the design process by reducing it to the choice of prototypes only, which are the most representative data samples and can be identified in a fully data-driven, nonparametric manner. This simplification significantly reduces the involvement of human expertise and also lifts the requirement of prior knowledge of the problems for system identification. The system structure of the ALMMo system is built upon non-parametric data clouds identified from streaming data in an autonomous, self-organizing and transparent manner without imposing any data generation models pre-defined in advance. The meta-parameters of the system are updated accordingly during the identification process without relying on any user- and problem-specific parameters. ALMMo system is a generic approach for nonlinear, nonstationary problem approximation, and it has demonstrated its strong performance on various benchmark problems [22] and successfully applied on real-world problems including stock market index prediction and foreign currency exchange rate prediction [36], [37].

Employing the ALMMo neuro-fuzzy system as the learning engine, both, the control and reference models of the SOF-PID control system are able to learn from data streams efficiently and effectively, and self-organize the rule-based structure and control parameters “on the fly”. In other words, the SOF-PID control system is capable of “learning as you control”. Moreover, the SOF-PID control system can adapt to an entirely new operating environment effectively without the requirement of full retraining.

The remainder of this paper is organized as follows. Section II presents the general architecture of the SOF-PID control system. The evolving mechanism of the proposed system is given in section III, and the computtaional complexity analysis is given in section IV. Section V demonstrates the effectiveness and validity of the SOF-PID control system through numerical experiments with mobile robots in simulated and real-world environments, and Section VI concludes the paper.

II. The Proposed SOC-PID Controller

A. General Architecture

The proposed SOF-PID control system works as an inverse plant approximation model [39], and the architecture is given in Fig. 1.

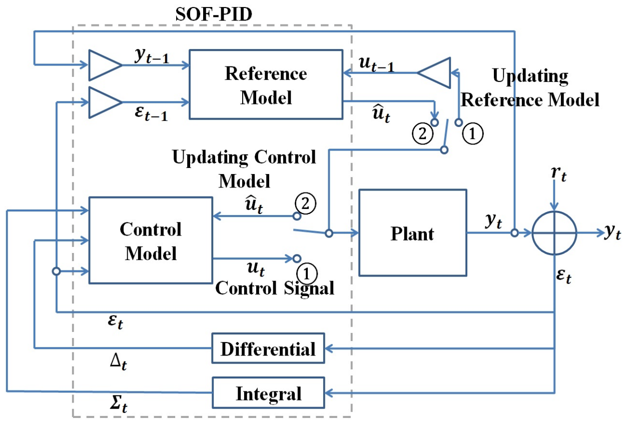

Fig. 1: Architecture of the SOF-PID control system.

As it is given by Fig. 1, SOF-PID control system consists of a control model and a reference model. The control model is for controlling the plant, and the reference model provides the desired output for the control model. Each models has an IF…THEN rule-based structure and is implemented using the first-order autonomous learning multi-model (ALMMo) neuro-fuzzy system [22] as the “learning engine”. The proposed SOF-PID control system also offers the possibility of implementing several subsets of PID-based controllers, e.g., P, PI, PD, etc. In this paper, we consider the general implementation of PID-based controllers by taking all three (proportional, integral, derivative) components as the control model inputs.

For the control model, the ALMMo system will self-organise during the process, a set of IF…THEN rules from the controller inputs $\boldsymbol{x}_t = [\epsilon_t, \Sigma_t, \Delta_t]^T$, namely, the tracking error, ϵ_t (the difference between the plant output, y_t and the desired trajectory, r_t), the discrete derivative and integral of the tracking error, Δ_t and Σ_t:

$$\epsilon_t = r_t - y_t; \quad \Delta_t = \epsilon_t - \epsilon_{t-1}; \quad \Sigma_t = \sum_{k=0}^{t-1} \epsilon_k; \qquad (1)$$

where t denotes the current control step; $t-1$ and t stand for the consecutive control steps.

Each IF…THEN rule is formulated in the following form as given below, and thus, it can be viewed as a PID controller in its THEN part, but with the parameters learned from the historical data and zone of influence limited and determined by its IF part:

$$\mathbf{R}_i^c: \ IF\ (\boldsymbol{x}_t \sim \boldsymbol{p}_{i,t-1})\ THEN\ (\boldsymbol{u}_{i,t} = \boldsymbol{a}_{i,t-1}^T \bar{\boldsymbol{x}}_t) \tag{2}$$

where $\sim$ denotes the similarity or a fuzzy degree of satisfaction/membership [30]; $\bar{\boldsymbol{x}}_t = [\boldsymbol{x}_t^T, 1]^T$; $\boldsymbol{p}_{i,t-1}$ is the prototype of the ith IF...THEN rule at the current step, which is identified and updated from the historical control inputs $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,..., $\boldsymbol{x}_{t-1}$ during the control process; $\boldsymbol{a}_{i,t-1} = [P_{i,t-1}, I_{i,t-1}, D_{i,t-1}, R_{i,t-1}]^T$; $P_{i,t-1}$, $I_{i,t-1}$ and $D_{i,t-1}$ are the controller gains; $R_{i,t-1}$ is the compensation of the operating point; $u_{i,t-1} = P_{i,t-1}\epsilon_t + I_{i,t-1}\Sigma_t + D_{i,t-1})\Delta_t + R_{i,t-1}$ denotes the output of this IF...THEN rule. Therefore, each IF...THEN rule in the rule base of the control model can be viewed as a PID controller (with the structure given by Fig. reffig12), and the overall control signal produced by the control model is formulated by equation (3) as a fuzzily weighted linear combination of outputs of the identified IF...THEN rules [22]:

$$u_t = f_c(\boldsymbol{x}_t) = \sum_{i=1}^{M_c} \lambda_{i,t} u_{i,t}. \tag{3}$$

where $f_c(\cdot)$ is the mathematical model between the inputs and output of the control model, which itself is a non-linear function approximated by the ALMMo neuro-fuzzy system ; M_c is the number of IF...THEN rules in the rule base of the control model; $\lambda_{i,t}$ denotes the firing strength of the ith IF...THEN rule, which is calculated as relative data density determined by the following expression [22]:

$$\lambda_{i,t} = \frac{\gamma_{i,t}(\boldsymbol{x}_t)}{\sum_{k=1}^{M_c} \gamma_{k,t}(\boldsymbol{x}_t)}; \tag{4}$$

where $\gamma_{k,t}$ denotes the local data density calculated based on the data cloud, $\mathbf{C}_k^c$ associated with the prototype of the kth IF...THEN rule [22]:

$$\gamma_{k,t}(\boldsymbol{x}_t) = \frac{1}{1 + \frac{S_{k,t-1}^2 ||\boldsymbol{x}_t - \boldsymbol{p}_{k,t-1}||^2}{(S_{k,t-1}+1)(S_{k,t-1}\chi_{k,t-1} + ||\boldsymbol{x}_t||^2) - ||\boldsymbol{x}_t - S_{k,t-1}\boldsymbol{p}_{k,t-1}||^2}}; \tag{5}$$

where $\mathbf{C}_k^c$ is composed of the historical controller inputs associated with prototype $\boldsymbol{p}_{k,t-1}$; $\boldsymbol{p}_{k,t-1}$ is the support of $\mathbf{C}_k^c$, namely, the number of members; $\chi_{k,t-1}$ is the average scalar product, which is calculated by $\chi_{k,t-1} = \frac{1}{S_{k,t-1}} \sum_{\boldsymbol{x} \in \mathbf{C}_k^c} ||\boldsymbol{x}||^2$; $||\boldsymbol{x}||$ denotes the norm of $\boldsymbol{x}$ and can be calculated by $||\boldsymbol{x}|| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$.

The reference model shares the same operating mechanism as the control except for the difference in the IF...THEN rule bases. The IF...THEN rules identified by the reference model are formulated in the following form due to the different input signal:

$$\mathbf{R}_i^r: \ IF\ (\boldsymbol{z}_t \sim \boldsymbol{q}_{i,t-1})\ \ THEN\ (\hat{u}_{i,t} = \boldsymbol{b}_{i,t-1}^T \bar{\boldsymbol{z}}_t) \tag{6}$$

where $\boldsymbol{z}_t = [\epsilon_(t-1), y_(t-1)]^T$; $\bar{\boldsymbol{z}}_t = [\boldsymbol{z}_t^T, 1]^T$; $\boldsymbol{q}_{i,t-1}$ is the prototype of the ith IF...THEN rule of the reference model, and it is identified and updated from the historical control inputs $\boldsymbol{z}_1$, $\boldsymbol{z}_2$,..., $\boldsymbol{z}_{t-1}$ during the control process; $\boldsymbol{b}_{i,t-1} = [Q_{i,t-1}, Y_{i,t-1}, W_{i,t-1}]^T$; $Q_{i,t-1}$,, $Y_{i,t-1}$ are the controller gains; $W_{i,t-1}$ is the compensation; $\hat{u}_{i,t} = Q_{i,t-1}\epsilon_{t-1} + Y_{i,t-1}y_{t-1} + W_{i,t-1}$. The mathematical model between the inputs and output of the reference model is denoted as: $\hat{u}_{i,t} = f_r(\boldsymbol{z}_t)$.

In the following subsection, the operating mechanism of the SOF-PID control system is described in detail.

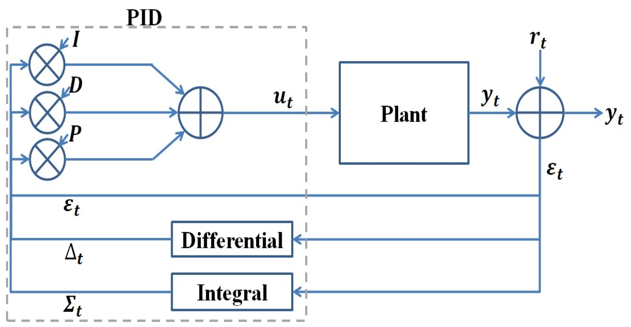

Fig. 2: Architecture of a PID controller.

B. Operating Mechanism

The SOF-PID control system requires to be initialized because the reference model of the proposed system is designed to provide the desired output for the control model. Therefore, in the first N control steps, the SOF-PID control system uses a PID controller with the same architecture as depicted in Fig. 2 to control the plant, meanwhile it keeps collecting the data for initialization. In this paper, we use the PID controller because of its simplicity and ease of implementation. Nonetheless, one can also initialize the system by legacy data or using controllers of other types, i.e. fuzzy controllers, self-evolving controllers for the starting period.

Stage 0. For the first N control steps, the proposed system uses a PID controller to control the plant; meanwhile, keeps collecting the data for initialization.

Stage 1. At the control step $t = N$, both the control and reference models are initialized based on the historical data, and then, it goes to **Stage 2**.

Stage 2. The proposed system reads the current tracking error ($t \leftarrow t+1$) and calculates the discrete derivative and integral of the tracking error, namely, ϵ_t, Σ_t and Δ_t using equation (1).

Stage 3. The reference model produces the desired output $\hat{u}_t$ based on the inputs: $\boldsymbol{z}_t = [\epsilon_{t-1}, y_{t-1}]^T$ and updates its structure and meta-parameters based on the control signal of the previous control step u_{t-1}.

Stage 4. The control model produces the control signal u_t based on the inputs: $\boldsymbol{x}_t = [\epsilon_t, \Sigma_t, \Delta_t]^T$ and updates its structure and meta-parameters based on the desired output, $\hat{u}_t$.

Stage 5. The proposed system goes back to **Stage 2** for the next time control step.

It has to be stressed that the amount of historical data for initializing the SOF-PID control system, namely, N is not a problem- and user-specific parameter, and it can be determined based on the user preference. Generally, the larger N is, the better and more stable the proposed system performs. Meanwhile, in the numerical experiment part (section VI), we will demonstrate that the SOF-PID control system surpasses the alternatives with a small value of N.

III. Self-Evolving Mechanism of SOF-PID Control System

In this section, the evolving mechanism of the proposed SOF-PID control system will be presented. As it has been stated in the previous sections, the SOF-PID control system is composed of a pair of control and reference models, and both models employ a first-order ALMMo system as the "learning engine". Both models self-organize and self-evolve their system structure and meta-parameters following the same algorithmic procedure, and, thus, we will focus on describing the evolving mechanism of the control model. The evolving mechanism of the reference model follows the same principles. It is worth to be noticed that despite the SOF-PID control system requires to be initialized by historical data, the control and reference models learn from data on a sample-by-sample basis.

The brief algorithmic procedure of the control model is given as follows [22].

Step 0. System Initialization: The global meta-parameters of the control model are initialized by the input at the first control step, $\boldsymbol{x}_t = [\epsilon_t, \Sigma_t, \Delta_t]^T$ $(t = 1)$:

$$\boldsymbol{\mu}_t \leftarrow \boldsymbol{x}_t; \quad X_t \leftarrow ||\boldsymbol{x}_t||^2; \quad M_c \leftarrow 1; \tag{7}$$

where $\boldsymbol{\mu}_t$ and X_t are the global mean and average scalar product, respectively.

The first data cloud of the control model, $\mathbf{C}^c_{M_c}$ is initialized by $\boldsymbol{x}_t$: $\mathbf{C}^c_{M_c} \leftarrow \{\boldsymbol{x}_t\}$. The meta-parameters of $\mathbf{C}$, which includes the prototype, $\boldsymbol{p}_{M_c,t}$, scalar product, $X_{M_c,t}$, support, $S_{M_c,t}$, accumulated firing strength, $\Lambda_{M_c,t}$, utility, $\eta_{M_c,t}$, and the control step when $\mathbf{C}^c_{M_c}$ is initialized, I_{M_c}, are given as:

$$\begin{aligned} &\boldsymbol{p}_{M_c,t} \leftarrow \boldsymbol{x}_t; \quad \chi_{M_c,t} \leftarrow ||\boldsymbol{x}_t||^2; \quad S_{M_c,t} \leftarrow 1; \\ &\Lambda_{M_c,t} \leftarrow 1; \quad \eta_{M_c,t} \leftarrow 1; \quad I_{M_c} \leftarrow 1; \end{aligned} \tag{8}$$

The first IF...THEN rule in the rule base of the control model is initialized as:

$$\mathbf{R}^c_{M_c}: \ IF\ (\boldsymbol{x}_t \sim \boldsymbol{p}_{M_c,t})\ THEN\ (\boldsymbol{u}_{M_c,t} = \boldsymbol{a}^T_{M_c,t-1}\bar{\boldsymbol{x}}_t) \tag{9}$$

and the consequent parameters of the IF...THEN rule are initialized as:

$$\boldsymbol{a}_{M_c,t} \leftarrow \mathbf{0}_{(L+1)\times 1}; \quad \boldsymbol{\Theta}_{M_c,t} \leftarrow \Omega_o \mathbf{I}_{(L+1)\times(L+1)}; \tag{10}$$

where L is the number of inputs, $L = 3$ for the control model and $L = 2$ for the reference model; $\mathbf{0}_{(L+1)\times 1}$ is a $(L+1)\times 1$ dimensional vector; $\boldsymbol{\Theta}_{M_c,t}$ is the co-variance matrix of the IF...THEN rule $\mathbf{R}^c_{M_c}$; $\mathbf{I}_{(L+1)\times(L+1)}$ is a $(L+1)\times(L+1)$ identity matrix; Ω_o is a user-controlled parameter initializing the co-variance matrix and $\Omega_o = 10$, which follows the same setting as [22].

Step 1. Control Signal Generation: For the next control step $(t \leftarrow t+1)$, the new input $\boldsymbol{x}_t = [\epsilon_t, \Sigma_t, \Delta_t]^T$ is available. Each IF...THEN rule within the control model produces the firing strength $\lambda_{i,t}$ $(i = 1, 2, \ldots, M_c)$ using equation (4). Then, the control signal is generated by: $u_t = f_c(\boldsymbol{x}_t)$.

Step 2. Global Meta-Parameter Updating: The global mean and average scalar product are updated by $\boldsymbol{x}_t$ using the following equations, respectively:

$$\boldsymbol{\mu}_t \leftarrow \frac{t-1}{t}\boldsymbol{\mu}_{t-1} + \frac{1}{t}\boldsymbol{x}_t; \quad X_t \leftarrow \frac{t-1}{t}X^2_{t-1} + \frac{1}{t}||\boldsymbol{x}_t||^2; \tag{11}$$

The data densities at $\boldsymbol{x}_t$ and the identified prototypes $\boldsymbol{p}_{i,t-1}$ $(i = 1, 2, \ldots, M_c)$ are, then, calculated by:

$$\gamma_t(\boldsymbol{w}) = \frac{1}{1 + \frac{||\boldsymbol{w} - \boldsymbol{\mu}_t||^2}{X_t - ||\boldsymbol{\mu}_t||^2}}; \tag{12}$$

where $\boldsymbol{w} = \boldsymbol{x}_t, \boldsymbol{p}_{1,t-1}, \boldsymbol{p}_{2,t-1}, \ldots, \boldsymbol{p}_{M_c,t-1}$.

Step 3. System Structure Updating: To update the system structure, **Condition 1** is firstly checked:

$$\begin{aligned} \textbf{Condition 1: } & IF\ (\gamma_t(\boldsymbol{x}_t) > \max_{i=1,2,\ldots,M_c}(\gamma_t(\boldsymbol{p}_{i,t-1}))) \\ & OR\ (\gamma_t(\boldsymbol{x}_t) < \min_{i=1,2,\ldots,M_c}(\gamma_t(\boldsymbol{p}_{i,t-1}))) \\ & THEN\ (\boldsymbol{x}_t\ is\ a\ new\ prototype) \end{aligned} \tag{13}$$

If **Condition 1** is met, **Condition 2** is, then, checked to determining whether the new data cloud that is associated with $\boldsymbol{x}_t$ is overlapping with any of the previously identified data clouds:

$$\begin{aligned} \textbf{Condition 2: } & IF\ (\gamma_t(\boldsymbol{x}_t) > 0.8) \\ & THEN\ (\boldsymbol{x}_t\ is\ very\ close\ to\ \boldsymbol{p}_{i,t-1}) \end{aligned} \tag{14}$$

where $\gamma_t(\boldsymbol{x}_t)$i s calculated by equation (5). If both **Conditions 1** and 2 are satisfied, the nearest data cloud $\mathbf{C}^c_{n*}$ to $\boldsymbol{x}_t$ is found out by equation (15):

$$n* = \underset{i=1,2,\ldots,M_c}{\text{argmax}}\ (||\boldsymbol{x}_t - \boldsymbol{p}_{i,t-1}||); \tag{15}$$

and the new data cloud is merged with $\mathbf{C}^c_{n*}$ $(\mathbf{C}^c_{n*} \leftarrow \mathbf{C}^c_{n*} + \boldsymbol{x}_t)$:

$$\begin{aligned} &S_{n*,t} \leftarrow \frac{\lceil S_{n*,t-1} + 1 \rceil}{2}; \ \boldsymbol{p}_{n*,t} \leftarrow \frac{\boldsymbol{p}_{n*,t-1} + \boldsymbol{x}_t}{2}; \\ &\chi_{n*,t} \leftarrow \frac{\chi_{n*,t-1} + ||\boldsymbol{x}_t||^2}{2}; \end{aligned} \tag{16}$$

where $\lceil \cdot \rceil$ denotes the operation of rolling up to the nearest integer.

If **Condition 1** is met, and **Condition 2** is unsatisfied, the new data cloud with $\boldsymbol{x}_t$ as the prototype is added to the system structure $(M_c \leftarrow M_c + 1)$: $\mathbf{C}^c_{M_c} \leftarrow \{\boldsymbol{x}_t\}$ with the meta-parameters set by equation (8).

The IF...THEN rule $\mathbf{R}^c_{M_c}$ corresponds to $\mathbf{C}^c_{M_c}$ is initialized in the same form as equation (9) with the consequent parameters set as:

$$\boldsymbol{a}_{M_c,t-1} \leftarrow \frac{1}{M_c-1}\sum_{j=1}^{M_c-1}\boldsymbol{a}_{j,t-1}; \;\; \boldsymbol{\Theta}_{M_c,t} \leftarrow \Omega_o \mathbf{I}_{(L+1)\times(L+1)}. \tag{17}$$

Otherwise, if both **Conditions 1** and **2** are not met, $\boldsymbol{x}_t$ is assigned to the nearest data cloud $\mathbf{C}^c_{n*}$ with the meta-parameters updated as:

$$\begin{aligned} S_{n*,t} &\leftarrow S_{n*,t-1} + 1; \\ \boldsymbol{p}_{n*,t} &\leftarrow \frac{S_{n*,t-1}}{S_{n*,t}}\boldsymbol{p}_{n*,t-1} + \frac{1}{S_{n*,t}}\boldsymbol{x}_t; \\ \chi_{n*,t} &\leftarrow \frac{S_{n*,t-1}}{S_{n*,t}}\chi_{n*,t-1} + \frac{1}{S_{n*,t}}||\boldsymbol{x}_t||^2. \end{aligned} \tag{18}$$

For each data cloud $\mathbf{C}^c_{n*}$ that does not receive new member at the current control step, its meta-parameters are set as:

$$S_{i,t} \leftarrow S_{i,t-1}; \;\; \boldsymbol{p}_{n*,t} \leftarrow \boldsymbol{p}_{n*,t-1}; \;\; \chi_{n*,t} \leftarrow \chi_{n*,t-1}. \tag{19}$$

Step 4. IF...THEN Rule Base Quality Monitoring: The firing strength of each IF...THEN rule, $\lambda_{i,t}$ is calculated by equation (4) and the accumulated firing strength is updated ($i = 1, 2, \ldots, M_c$):

$$\Lambda_{i,t} \leftarrow \Lambda_{i,t-1} + \lambda_{i,t}. \tag{20}$$

The utility of each rule is calculated by:

$$\eta_{i,t} \leftarrow \frac{1}{t - I_i}\Lambda_{i,t}. \tag{21}$$

Based on the updated utility, **Condition 3** is used for removing the data cloud and fuzzy IF...THEN rule that contributes little to the overall control signal [30], [38]:

$$\begin{aligned} &\textbf{Condition 3: } IF\ (\eta_{i,t} > \eta_o) \\ &\qquad THEN\ (\mathbf{R}^c_{n*}\ and\ \mathbf{C}^c_{n*}\ are\ removed) \end{aligned} \tag{22}$$

where η_o is another user-controlled parameter for monitoring the quality of IF...THEN rules, and $\eta_o = 0.1$ following [22]. If $\mathbf{R}^c_{n*}$ and $\mathbf{C}^c_j$ meet **Condition 3**, they are removed from the system structure and $M_c \leftarrow M_c - 1$.

Step 5. Consequent Part Updating: After the system structure updating, the consequent parameters of the IF...THEN rules in the rule base are updated by the FWRLS method as ($i = 1, 2, \ldots, M_c$) [39]:

$$\boldsymbol{\Theta}_{i,t} \leftarrow \boldsymbol{\Theta}_{i,t-1} - \frac{\lambda_{i,t}\boldsymbol{\Theta}_{i,t-1}\bar{\boldsymbol{x}}_t\bar{\boldsymbol{x}}_t^T\boldsymbol{\Theta}_{i,t-1}}{1 + \lambda_{i,t}\bar{\boldsymbol{x}}_t^T\boldsymbol{\Theta}_{i,t-1}\bar{\boldsymbol{x}}_t}. \tag{23}$$

$$\boldsymbol{a}_{i,t} \leftarrow \boldsymbol{a}_{i,t-1} + \lambda_{i,t}\boldsymbol{\Theta}_{i,t}\bar{\boldsymbol{x}}_t(\hat{u}_t - \boldsymbol{a}^T_{i,t-1}\bar{\boldsymbol{x}}_t) \tag{24}$$

Step 6. Rule Base Updating: All the IF...THEN rules in the rule base are updated with the new premise and consequent parameters ($i = 1, 2, \ldots, M_c$):

$$\mathbf{R}^c_i : \ IF\ (\boldsymbol{x}_t \sim \boldsymbol{p}_{i,t})\ THEN\ (\boldsymbol{u}_{i,t} = \boldsymbol{a}^T_{i,t-1}\bar{\boldsymbol{x}}_t) \tag{25}$$

After this, the system goes back to **Step 1** for the next control step.

The reference model can be learned using the same principles as described in this section. It needs to be noticed that the inputs of the reference model are $z_t = [\epsilon_{t-1}, y_{t-1})]^T$ and the desired output of the reference model is the output of the control model, u_t. Detailed mathematical proof for the uniform stability of the ALMMo system can be found in [40].

IV. Experimental Study

The performance of the proposed self-organizing fuzzy PID (SOF-PID) control system has been evaluated through several experiments. The experiments presented in this paper are organized as follows. Firstly, experiments in a simulated environment created in Gazebo (see Fig. 3) were performed to verify the validity and effectiveness of the proposed SOF-PID system. Then, the performance of the proposed controller was evaluated in various real-world environments using a Pioneer 3DX Mobile Robot (see Fig. 4).

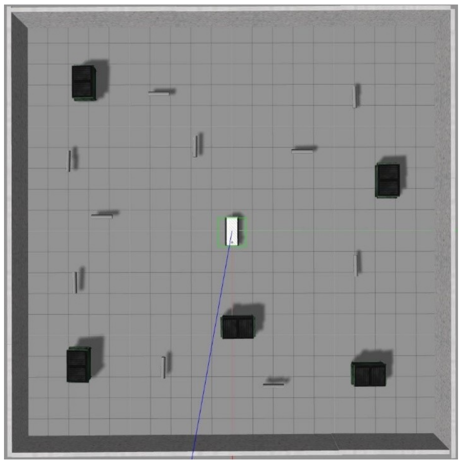
Fig. 3: Simulated Gazebo environment with the mobile robot in the middle.

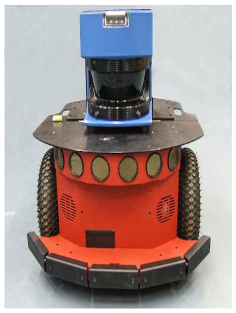
Fig. 4: Pioneer 3DX robot with a laser finder on top and 16 sonar sensors; 8 at front and 8 at rear.

A. Computational Complexity Analysis

In this section, the computational complexity of the proposed self-organizing fuzzy PID (SOF-PID) control system is analyzed. As the proposed control system is composed of a pair of control and reference models and the two components are, practically, the same except for the differences in terms of model inputs and output, we only analyze the computational complexity of the control model for clarity. It has to be stressed that the same conclusion applies to the reference model as well.

At the initial stage, the proposed approach requires to be primed by a PID controller until sufficient historical data has been collected. The computational complexity of the SOF-PID system at this stage is the same as a PID controller, namely, $O(LN)$.

Then, both control and reference models of SOF-PID are initialized by historical data, and continue to learn and produce control/reference signal at each control step. However, as the computational complexity of the control model is dynamically changing all the time, we assume that the analysis is conducted at thetth time instance.

For the control model, the computational complexity of step 0 is negligible since this step concerns mainly parameter-initialization and will be performed once only. For step 1, the computational complexity of calculating the local data density-based firing strength is $O(LM_c)$ thanks to the recursive calculation form of equation (5) and the complexity for calculating the control signal is also $O(LM_c)$.

Steps 2 and 3 concern mainly the system structure and meta-parameter updating. The complexity for calculating the data density in step 2 is $O(L(M_c + 1))$. The computational complexity for updating meta-parameters globally and locally per rule is $O(L)$. The complexity for adding new rules to the control model is negligible. Therefore, the computational complexity of steps 2 and 3 is $O(L(M_c + 1))$.

Step 4 is mostly for calculating the local data density, and thus, the complexity is $O(LM_c)$. Updating the consequent parameters of IF…THEN rules consumes significantly more computational resources because the co-variance matrix needs to be updated at each control step and, thus, the computational complexity of step 5 is $O(L^2M_c)$. Step 6 updates the IF…THEN rules in the rule base, and its computational complexity is negligible as well.

Therefore, the overall computational complexity of the control model of the SOF-PID is $O(L^2M_c)$ for each control step. The computational complexity of the reference model can be derived following the same principles.

B. Experimental Settings

In order to evaluate the performance of the proposed SOF-PID system, control of the forward motion of the mobile robot was implemented. The architecture of the overall forward move framework with a controller as a dotted block is shown in Fig. 5.

The simulated mobile robot (shown in the middle of Fig. 3) in the Gazebo environment is equipped with a 360^o scanning LIDAR sensor. The proposed SOF-PID system takes as the plant output, y the distance from the robot to the nearest detected object measured by the LIDAR senor. The Pioneer 3 robot is equipped with two types of input sensors that are a laser range finder and a set of 16-sonar sensors targeting at different angles, see Fig. 4. Similarly, the system takes as the plant output, y the distance between the robot and the nearest object in front measured by the two frontal sonars. The minimum distance is set to $r = 1$ meter and $r = 0.5$ meter for the two cases, respectively. In the experiments performed in this paper, we use the same setting for the SOF-PID control system unless specifically declared otherwise. The number of time instances, N to initialize the SOF-PID using a PID prime controller is set as: $N = 10$. The PID prime controller is defined as follows:

$$u = P\epsilon + I\Sigma + D\Delta \tag{26}$$

where $P = 0.25$, $D = 0.1$ and $I = 0$.

In order to justify the validity and effectiveness of the SOF-PID controller in various environments the experiments involve the commonly used PID [5] and Takagi-Sugeno (TS) fuzzy controllers [35] as comparative approaches following the same experimental setting. For a fair comparison, the PID controller used for comparison is the same as the PID prime controller. The TS fuzzy controller is defined as follows:

$$IF\ (\epsilon\ is\ Very\ Low)\ THEN(u = 0.25\epsilon + 0.001\Delta) \tag{27a}$$

$$\begin{aligned} &IF\ (\epsilon\ is\ Low)\ AND\ (\Delta\ is\ High)\\ &THEN(u = 0.5\epsilon + 0.002\Delta)\end{aligned} \tag{27b}$$

$$\begin{aligned} &IF\ (\epsilon\ is\ Medium)\ AND\ (\Delta\ is\ High)\\ &THEN(u = 0.5\epsilon + 0.002\Delta)\end{aligned} \tag{27c}$$

$$\begin{aligned} &IF\ (\epsilon\ is\ Medium)\ AND\ (\Delta\ is\ Low)\\ &THEN(u = \epsilon + 0.03\Delta)\end{aligned} \tag{27d}$$

$$IF\ (\epsilon\ is\ High)\ THEN(u = 2\epsilon + 0.02\Delta) \tag{27e}$$

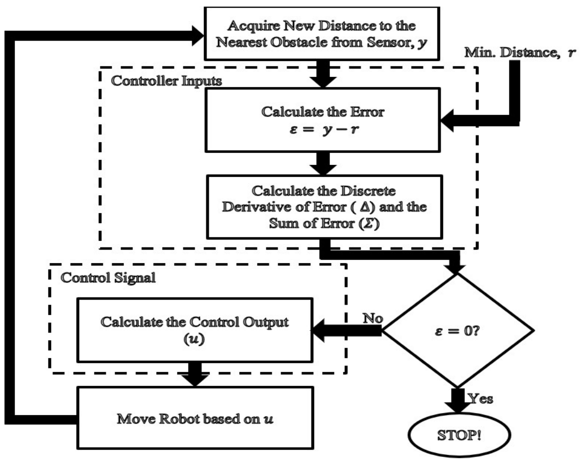
Acquire New Distance to the Nearest Obstacle from Sensor, y
Min. Distance, r
Controller Inputs
Calculate the Error $\varepsilon = y - r$
Calculate the Discrete Derivative of Error (Δ) and the Sum of Error (Σ)
Control Signal
Calculate the Control Output (u)
No
$\varepsilon = 0$?
Yes
Move Robot based on u
STOP!

Fig. 5: Block diagram of the SOF-PID controller for forward move.

The membership functions of the antecedent parts of the fuzzy rules are of triangular type and visualized in Fig. 6.

It is worth to be noticed that due to the specific requirements of experimental scenarios, the mobile robot must not go across the target in case of potential damages it may cause, both PID prime controller and TS fuzzy controller do not involve the integral of tracking error as input to prevent the mobile robot from moving fast when it is close to the target. The mobile robot will stop immediately if it goes beyond the target. All the reported experimental results are the average of 10 Monte-Carlo experiments unless specially declared otherwise.

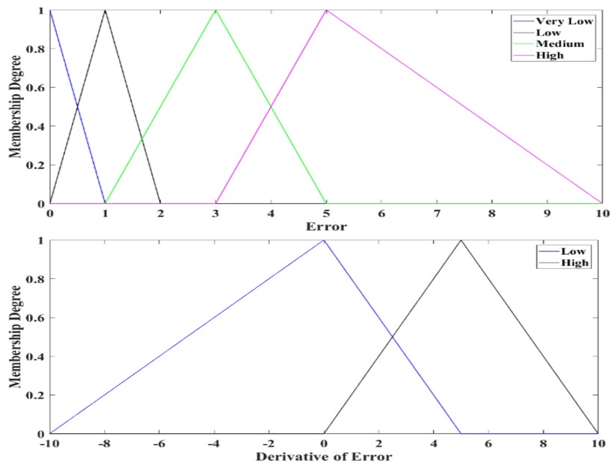

Fig. 6: Membership functions of the antecedent part.

C. Experimental Results

In this subsection, the experiments are presented to verify the proposed concept and the general principles.

Firstly, we perform experiments in the simulated Gazebo environment with the mobile robot to evaluate the influence of the different values of$N = \{5, 10, 15, 20\}$, namely, the number of steps used to initialize the controller, on the performance of the proposed SOF-PID controller. The correspondingly error and the plant output curves with different experimental settings are shown in Figs. 7 and 8, respectively. The results of the PID controller (with same setting as the PID prime controller used by SOF-PID) are also presented as the baseline.

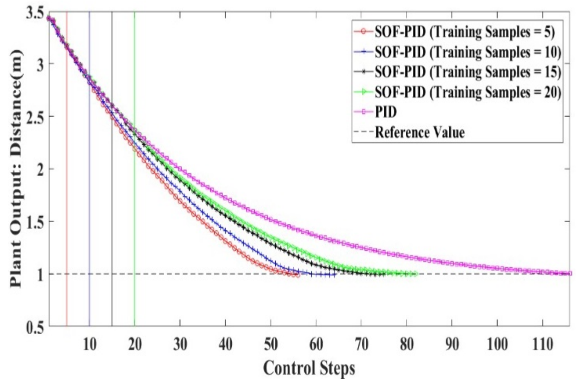

Fig. 7: The influence of different N on the SOF-PID control system performance in terms of plant output.

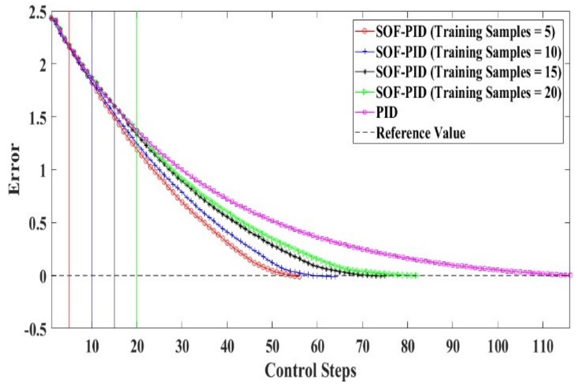

Fig. 8: The influence of different N on the SOF-PID control system performance in terms of tracking error.

As we can see from both figures, the proposed SOF-PID requires a smaller number of steps to converge than a PID controller in all the cases. One may also notice that SOF-PID takes more control steps before the mobile robot reaches the target when we increase the prime-training control steps. This is due to the fact that the SOF-PID is essentially trying to approximate the control error converging process of the priming PID controller. As the integral of tracking error is not used as the input of the PID prime controller, the converging speed consistently decreases as the tracking error decreases. As a result, increasing the prime-training control steps will decrease the convergence speed of SOF-PID. Nonetheless, it has to be stressed that the SOF-PID control system is not mimicking the behavior of the PID prime controller, but is trying to learn the relationship between the controller inputs and desired controller output from the historical data collected by the PID prime controller and further to self-adjust its control gains to adapt to the changing environment. The curves in Figs. 7 and 8 further suggest that the newly proposed SOF-PID controller can adapt to the environment even if it is trained with only five samples.

To further evaluate the performance of the proposed SOF-PID control system, four real-world experiments (see Figs. 9(a)-(d)) are performed. Details of the four experiment scenarios are given as follows:

Scenario 1 (Fig. 9(a)): The robot goes directly from one side of the room to a grey cupboard at the other side. The distance between the target (grey cupboard) and the starting position of the robot is 3.17 meters. There is one soft bump in the middle of the route.

Scenario 2 (Fig. 9(b)): This scenario is the same as the first one except that there are two soft bumps on the route.

Scenario 3 (Fig. 9(c)): In this scenario, the robot starts on the grass and goes straight to the target. The surface is changed to a brick road half way through. The distance between the starting position and the target is 4.07 meters.

Scenario 4 (Fig. 9(d)): In this scenario, the robot goes straight and stops before the wall. The robot, firstly, goes downwards on a slope, and then climbs another slope to reach

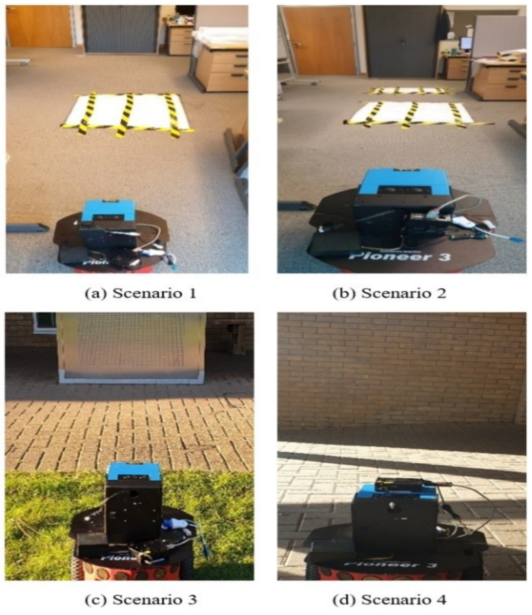

Fig. 9: Four different experiment scenarios.

the target. The distance between the starting position and the wall is 5 meters.

The ground surface of each experimental scenario is composed of either, patches with different levels of friction or slopes with different gradients. The mobile robot will go through a path with different surface conditions before reaching the target, which allow us to evaluate the adaptability of a controller to a changing environment.

The experimental results obtained with the SOF-PID system are depicted in Figs. 10(a)-10(d), and the performance is also compared with the performance of the PID and TS fuzzy controllers in terms of number of control steps for the robot to reach the target, the controller error and the plant output. From Fig. 10 one can see that the SOF-PID control system performs better than the other controllers in all the scenarios. It takes much less control steps for the control error to converge to zero compared with the two alternatives.

In order to illustrate the transparency and human-interpretability of the proposed SOF-PID controllers, we also illustrate the evolution of the number of fuzzy IF…THEN rules in both, the control model and the reference model during a particular experiment conducted on scenario 1 in Fig. 11. It has to be noticed that Fig. 11 starts from the $11th$ control step because the first 10 steps are performed with the PID controller used for priming. The IF…THEN rules of both models at the end of the process are also presented in Tables I and II, respectively.

V. Conclusion

In this paper, a novel SOF-PID control system is proposed, which consists of a pair of control and reference models. The control model and reference model are both implemented by first-order ALMMo neuro-fuzzy systems. Compared with the alternative controllers, the SOF-PID control system is capable of self-organizing and self-evolving its system structures and meta-parameters during the control process "on the fly", which enables the proposed system to adapt to new environments autonomously without a full re-training. We also mathematically prove the stability of the proposed system. The SOF-PID control system is a generic approach and offers different implementation possibilities. It effectively deals with nonlinear processes and requires no prior knowledge of the dynamics of the process and plant. It only requires a very short priming of 5-10 steps. Simulations on Gazebo platform and real-world experiments using Pioneer robot demonstrate that the proposed SOF-PID control system is able to provide stable control performance even in changing environments and its performance surpasses the alternative controllers.

It has to be stressed that the main purpose of this paper is to demonstrate the general concept and principles of SOF-PID, and provides primary experiments to prove the effectiveness and validity. As future works, we will further investigate the performance of the proposed SOF-PID control system in various types of challenging environments and compared with other advanced control approaches. We will also implement it for controlling other types of mobile robot movements, i.e. rotating, grabbing, as well as for other industrial processes, i.e. temperature and pressure control.

References

[1] M. Singh, A. Agrawal, S. Ralhan, and R. Jhapte, "Comparative performance analysis of adaptive tuned PID controller for multi-machine power system network," in Computational Intelligence in Data Mining, H. Behera, J. Nayak, B. Naik, and A. Abraham, Eds. Springer, Singapore, 2019, pp. 819–829.

[2] A. Marin Canoa Marin, J. A. Hernandez Riveroj Hernand, and J. A. Jimenez Builesj Jimenez, "Tuning multivariable optimal PID controller for a continuous stirred tank reactor using an evolutionary algorithm," IEEE Lat. Am. Trans., vol. 16, no. 2, pp. 422–427, 2018.

[3] H. Gonzalez, C. Arizmendi, J. Garcia, A. Anguo, and C. Herrera, "Design and experimental validation of adaptive fuzzy PID controller for a three degrees of freedom helicopter," in IEEE International Conference on Fuzzy Systems, 2018, pp. 1–6.

[4] S. Bennett, "Development of the PID Controller," IEEE Control Syst., vol. 13, no. 6, pp. 58–62, 1993.

[5] K. J. A strom and T. H'agglund, PID controllers: theory, design, and tuning. Research Triangle Park, NC: Instrument society of America, 1995.

[6] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," trans. ASME, vol. 64, no. 11, pp. 759–769, 1942.

[7] G. J. Silva, A. Datta, and S. P. Bhattacharyya, "On the stability and controller robustness of some popular PID tuning rules," IEEE Trans. Automat. Contr., vol. 48, no. 9, pp. 1638–1641, 2003.

[8] K. S. Tang, K. F. Man, G. Chen, and S. Kwong, "An optimal fuzzy PID controller," IEEE Trans. Ind. Electron., vol. 48, no. 4, pp. 757–765, 2001.

[9] K. J. Astrom, C. C. Hang, P. Persson, and W. K. Ho, "Towards intelligent PID control," Automatica, vol. 28, no. 1, pp. 1–9, 1992.

[10] C. Rosales, C. M. Soria, and F. G. Rossomando, "Identification and adaptive PID Control of a hexacopter UAV based on neural networks," Int. J. Adapt. Control Signal Process., vol. 0, no. 0, pp. 74–91, 2019.

[11] K. Lapa and K. Cpalka, "Flexible fuzzy PID controller (FFPIDC) and a nature-inspired method for its construction," IEEE Trans. Ind. Informatics, vol. 14, no. 3, pp. 1078–1088, 2018.

[12] S. Dettori, V. Iannino, V. Colla, and A. Signorini, "An adaptive Fuzzy logic-based approach to PID control of steam turbines in solar applications," Appl. Energy, vol. 227, pp. 655–664, 2018.

[13] S. Shao, "Fuzzy self-organizing controller and its application for dynamic processes," Fuzzy Sets Syst., vol. 26, no. 2, pp. 151–164, 1988.

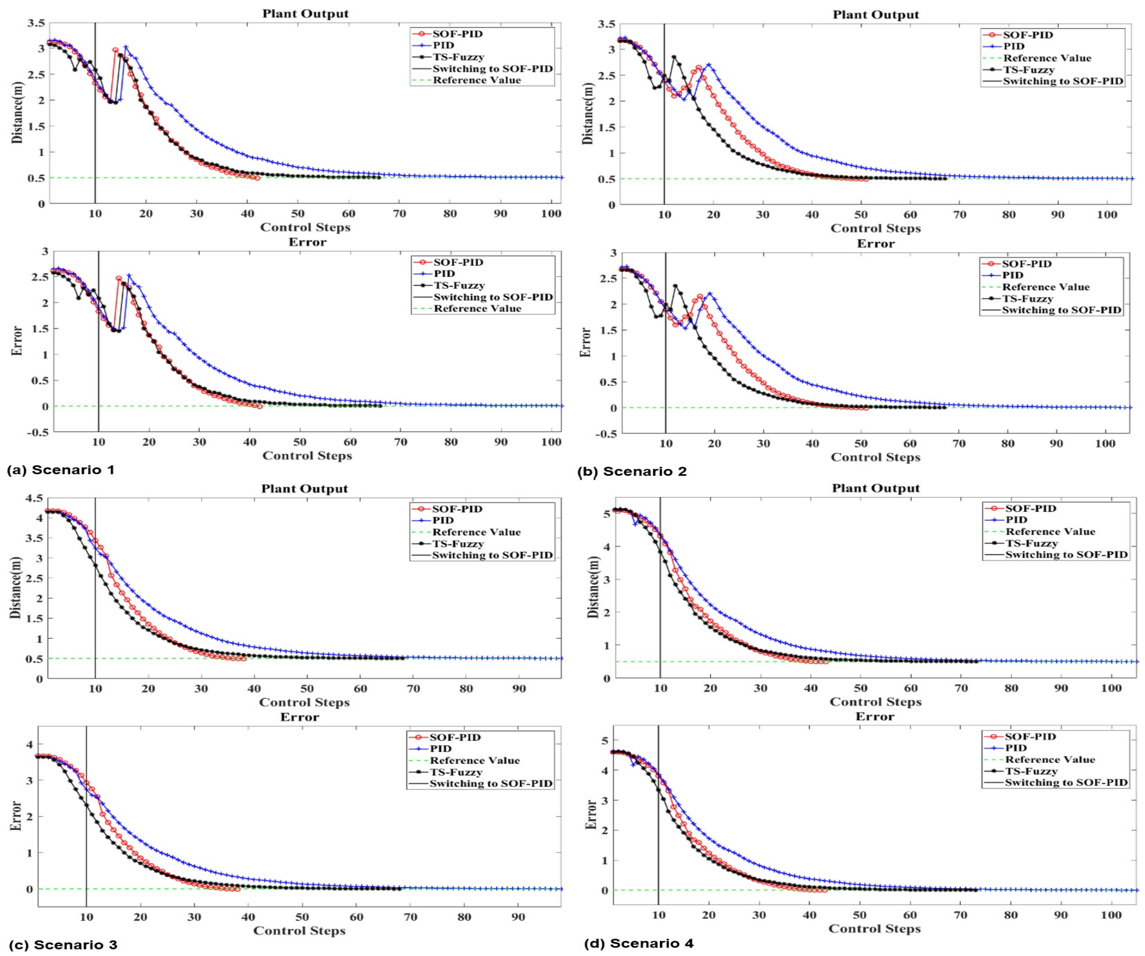

Fig. 10: Performance comparison in different experiment scenarios.

TABLE I: THE IDENTIFIED IF...THEN RULES OF THE CONTROL MODEL DURING THE CONTROL PROCESS

#	**IF...THEN Rule**
1	$IF\ (\boldsymbol{x} \sim [2.6500, 4.3841, 0.8815]^T)\ THEN\ (u = 0.2171\epsilon - 0.0017\Sigma + 0.0076\Delta + 0.0761)$
2	$IF\ (\boldsymbol{x} \sim [2.6300, 5.5792, -0.0062]^T)\ THEN\ (u = 0.2156\epsilon - 0.0011\Sigma + 0.0042\Delta + 0.0651)$
3	$IF\ (\boldsymbol{x} \sim [2.5700, 6.3352, -0.1987]^T)\ THEN\ (u = 0.2139\epsilon - 0.0007\Sigma + 0.0025\Delta + 0.0580)$
4	$IF\ (\boldsymbol{x} \sim [2.5700, 7.1291, 0.0000]^T)\ THEN\ (u = 0.2139\epsilon - 0.0006\Sigma + 0.0056\Delta + 0.0577)$
5	$IF\ (\boldsymbol{x} \sim [2.4700, 7.8732, -0.3320]^T)\ THEN\ (u = 0.2098\epsilon - 0.0001\Sigma + 0.0004\Delta + 0.0494)$
6	$IF\ (\boldsymbol{x} \sim [2.3600, 8.5840, -0.3652]^T)\ THEN\ (u = 0.2085\epsilon - 0.0000\Sigma + 0.0028\Delta + 0.0488)$
7	$IF\ (\boldsymbol{x} \sim [2.2200, 9.2528, -0.4647]^T)\ THEN\ (u = 0.2074\epsilon - 0.0001\Sigma + 0.0042\Delta + 0.0469)$
8	$IF\ (\boldsymbol{x} \sim [1.3400, 18.1528, -0.2247]^T)\ THEN\ (u = 0.2148\epsilon - 0.0006\Sigma + 0.0027\Delta + 0.0585)$
9	$IF\ (\boldsymbol{x} \sim [1.1600, 18.6895, -0.3815]^T)\ THEN\ (u = 0.2149\epsilon - 0.0006\Sigma + 0.0027\Delta + 0.0586)$
10	$IF\ (\boldsymbol{x} \sim [0.3395, 21.2997, -0.1423]^T)\ THEN\ (u = 0.2149\epsilon - 0.0007\Sigma + 0.0026\Delta + 0.0587)$

[14] M. M. Ferdaus, S. G. Anavatti, M. A. Garratt, and M. Pratama, "Development of c-means clustering based adaptive fuzzy controller for a flapping wing micro air vehicle," J. Artif. Intell. Soft Comput. Res., vol. 9, no. 2, pp. 99–109, 2018.

[15] P. Angelov and N. Kasabov, "Evolving computational intelligence systems," in International Workshop on Genetic Fuzzy Systems, 2005, pp. 76–82.

[16] P. Angelov and N. Kasabov, "Evolving intelligent systems, eIS," IEEE SMC eNewsLetter, vol. 15, pp. 1–13, 2006.

[17] K. Eltag, M. S. Aslam, and R. Ullah, "Dynamic stability enhancement using fuzzy PID control technology for power system," Int. J. Control. Autom. Syst., vol. 17, no. 1, pp. 234–242, 2019.

[18] S. Huang and J. Lee, "A stable self-organizing fuzzy controller for robotic motion control," IEEE Trans. Ind. Electron., vol. 47, no. 2, pp. 421–428, 2000.

[19] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," Appl. Soft Comput., vol. 11, no. 2, pp. 2057–2068, 2011.

[20] P. Angelov, "A fuzzy controller with evolving structure," Inf. Sci. (Ny)., vol. 161, no. 1–2, pp. 21–35, 2004.

[21] B. Costa, I. Skrjanc, S. Blazic, and P. Angelov, "A practical implementation of self-evolving cloud-based control of a pilot plant," in IEEE International Conference on Cybernetics, 2013, pp. 7–12.

[22] P. P. Angelov, X. Gu, and J. C. Principe, "Autonomous learning multi-model systems from data streams," IEEE Trans. Fuzzy Syst., vol. 26, no. 4, pp. 2213–2224, 2018.

TABLE II: THE IDENTIFIED IF...THEN RULES OF THE CONTROL MODEL DURING THE CONTROL PROCESS

#	**IF...THEN Rule**
1	$IF\ (\boldsymbol{z} \sim [2.6500, 3.1500]^T)\ THEN\ (\hat{u} = 0.1176\epsilon + 0.1096y - 0.0160)$
2	$IF\ (\boldsymbol{z} \sim [2.6300, 3.1300]^T)\ THEN\ (\hat{u} = 0.1154\epsilon + 0.1082y - 0.0144)$
3	$IF\ (\boldsymbol{z} \sim [2.4700, 2.9700]^T)\ THEN\ (\hat{u} = 0.1076\epsilon + 0.1047y - 0.0059)$
4	$IF\ (\boldsymbol{z} \sim [0.2900, 0.7900]^T)\ THEN\ (\hat{u} = 0.1103\epsilon + 0.1060y - 0.0089)$
5	$IF\ (\boldsymbol{z} \sim [0.2500, 0.7500]^T)\ THEN\ (\hat{u} = 0.1109\epsilon + 0.1064y - 0.0089)$
6	$IF\ (\boldsymbol{z} \sim [0.2200, 0.7200]^T)\ THEN\ (\hat{u} = 0.1118\epsilon + 0.1070y - 0.0095)$
7	$IF\ (\boldsymbol{z} \sim [0.1800, 0.6800]^T)\ THEN\ (\hat{u} = 0.1117\epsilon + 0.1070y - 0.0094)$
8	$IF\ (\boldsymbol{z} \sim [0.1500, 0.6500]^T)\ THEN\ (\hat{u} = 0.1119\epsilon + 0.1072y - 0.0095)$
9	$IF\ (\boldsymbol{z} \sim [0.1100, 0.6100]^T)\ THEN\ (\hat{u} = 0.1122\epsilon + 0.1074y - 0.0096)$
10	$IF\ (\boldsymbol{z} \sim [0.0420, 0.5420]^T)\ THEN\ (\hat{u} = 0.1121\epsilon + 0.1074y - 0.0095)$

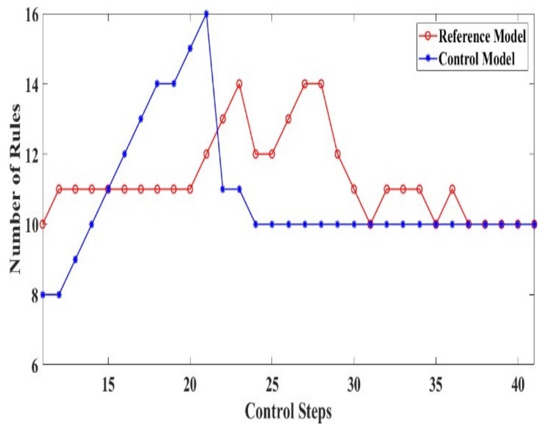

Fig. 11: The evolution of the number of rules in the control and reference models during one of the experiments.

[23] D. Psaltis and A. Sideris, "A multilayered neural network controller," IEEE Control Syst. Mag., vol. 8, no. 2, pp. 17–21, 1988.

[24] Y. Fan and U.X. Tan, "Design of a feedforward-feedback controller for a piezoelectric-driven mechanism to achieve high-frequency non-periodic motion tracking," IEEE/ASME Trans. Mechatronics, vol. 24, no. 2, pp. 1–1, 2019.

[25] Y. Zhao, R. M. Edwards, and K. Y. Lee, "Hybrid feedforward and feedback controller design for nuclear steam generators over wide range operation using genetic algorithm," IEEE Trans. Energy Convers., vol. 12, no. 1, pp. 100–105, 1997.

[26] C. S. Chiu, "Mixed feedforward/feedback based adaptive fuzzy control for a class of MIMO nonlinear systems," IEEE Trans. Fuzzy Syst., vol. 14, no. 6, pp. 716–727, 2006.

[27] M. R. Golbahar Haghighi, M. Eghtesad, D. S. Necsulescu, and P. Malekzadeh, "Temperature control of functionally graded plates using a feedforward-feedback controller based on the inverse solution and proportional-derivative controller," Energy Convers. Manag., vol. 51, no. 1, pp. 140–146, 2010.

[28] G. Andonovski, P. Angelov, S. Blazic, and I. Skrjanc, "A practical implementation of robust evolving cloud-based controller with normalized data space for heat-exchanger plant," Appl. Soft Comput. J., vol. 48, pp. 29–38, 2016.

[29] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," Int. J. Gen. Syst., vol. 41, no. 2, pp. 163–185, 2012.

[30] P. Angelov, Autonomous learning systems: from data streams to knowledge in real time. John Wiley & Sons, Ltd., 2012.

[31] D. Vrečko et al., "Improvement of ammonia removal in activated sludge process with feedforward-feedback aeration controllers," Water Sci. Technol., vol. 53, no. 4–5, pp. 125–132, 2006.

[32] L. Y. Pao, J. A. Butterworth, and D. Y. Abramovitch, "Combined feedforwardfeedback control of atomic force microscopes," in American Control Conference, 2007, pp. 3509–3515.

[33] D. Vrečko et al.,"Feedforward-feedback control of a solid oxide fuel cell power system," Int. J. Hydrogen Energy, vol. 43, no. 12, pp. 6352–6363, 2018.

[34] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," Int. J. Man. Mach. Stud., vol. 7, no. 1, pp. 1–13, 1975.

[35] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. Syst. Man. Cybern., vol. 15, no. 1, pp. 116–132, 1985.

[36] X. Gu and P. Angelov, "Self-boosting first-order autonomous learning neuro-fuzzy systems," Appl. Soft Comput., vol. 77, pp. 118–134, 2019.

[37] Y. Yong, Y. Lee, X. Gu, P. Angelov, D. C. Ling Ngo, and E. S. Yourdshahi, "Foreign currency exchange rate prediction using neuro-fuzzy systems," Procedia Comput. Sci., vol. 144, pp. 232–238, 2018.

[38] P. Angelov, "Evolving takagi-sugeno fuzzy systems from streaming data (eTS+)," in Evolving intelligent systems: methodology and applications, John Wiley & Sons., 2010.

[39] P. Angelov, J. Victor, A. Dourado, and D. Filev, "On-line evolution of Takagi-Sugeno fuzzy models," IFAC Proc., vol. 37, no. 16, pp. 67–72, 2004.

[40] P. Angelov and X. Gu, Empirical approach to machine learning. Springer International Publishing, 2018.

[41] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," IEEE Trans. Neural Networks, vol. 3, no. 5, pp. 807–814, 1992.

[42] N. J. Higham, Accuracy and stability of numerical algorithms, 2nd ed. Philadelphia, PA, USA: Siam, 2002.