An Evolutionary General Type-2 Fuzzy Neural Network applied to Trajectory Planning in Remotely Operated Underwater Vehicles

Adrian Rubio-Solis
Laboratory of Submarine Robotics
CIDESI, Queretaro, Mexico
email: adrian.rubio@cidesi.edu.mx

Tomas Salgado-Jimenez
Laboratory of Submarine Robotics
CIDESI, Queretaro, Mexico
email: tsalgado@cidesi.edu.mx

Luis Govinda Garcia-Valdovinos
Laboratory of Submarine Robotics
CIDESI, Queretaro, Mexico
email: ggarcia@cidesi.edu.mx

Luciano Nava-Balanzar
Laboratory of Submarine Robotics
CIDESI, Queretaro, Mexico
email: inava@cidesi.edu.mx

Uriel Martinez-Hernandez
EEE Department
University Of Bath, Bath UK
email: u.martinez@bath.ac.uk

***Abstract*—In this paper, an evolutionary General Type-2 Radial Basis Function Neural Network (GT2-RBFNN) for trajectory planning in Remotely Operated underwater Vehicles (ROVs) is suggested. The GT2-RBFNN is used as a data-driven learning system to orient the current position of an ROV in underwater environments. To determine the parameters of GT2-RBFNN, Galactic Swarm Optimisation (GSO) was implemented. A BlueROV2 and a squared water container of $2.5m \times 2.5m \times 3.5m$ were employed to run all experiments. To control the ROV position, a sensory system that consists of a compass, a micro data sonar, a ping sonar and a pressure sensor was integrated. First, a Proportional Derivative fuzzy controller was implemented to control the depth and yaw positions of the ROV. Secondly, the GT2-RBFNN was applied to discriminate between two different types of contours, i.e. corners and walls in order to follow an obstacle-free trajectory. To compare the efficiency of the GT2-RBFNN, a number of learning techniques that are based on Extreme Learning Machine (ELM) and evolutionary optimisation were implemented. Based on our results, a high trade-off between model simplicity and low computational burden are provided by the GT2-RBFNN.**

***Index Terms*—Neural Networks, GT2 Fuzzy Logic, evolutionary computing, Remotely Operated Underwater Vehicles.**

I. Introduction

General Type-2 Fuzzy Logic Systems (GT2 FLSs) are credited to outperform their Type-1 (T1) and Interval Type-2 (IT2) counterparts in a number of different applications [1–8]. This is mainly due to their ability to better deal with uncertainties inherent in real world problems. Compared to T1 and IT2 FSs, in terms of system's design, GT2 FSs are characterised by a footprint of Uncertainty (FOU) that provides an extra dimension giving more degrees of freedom. In underwater applications, uncertainties may result specially from sensor's measurements, uncertainties in control actions, linguistic uncertainties due to system's design and uncertainties that are present in training data [3, 7, 9]. Such uncertainties may affect not only the performance of a GT2 FLS, but also the correct definition of the appropriate Membership Functions (MFs) and the parameter's identification of each antecedent and consequent. Within this context, different learning techniques have been suggested [5]. In particular Gradient-Descent-based (GD) approaches [5] and Evolutionary Optimisation (EO) [7] are the most popular for the design of FLSs of high order that use Karnik-Mendel algorithms (KM). Opposite to GD approaches, EO does not require the iterative sorting process that is usually carried out when computing each partial derivative. Moreover, the application of EO favours the opimisation search to avoid getting trapped in local minima.

This paper reports the use of Galactic Swarm Optimisation (GSO) to design a General Type-2 RBF Neural Network (GT2-RBFNN) that is based on the functional equivalence between the RBF Neural Network (RBFNN) and General Type-2 Fuzzy Logic Systems (GT2 FLSs) with a special application to trajectory planning in Remotely Operated Underwater Vehicles (ROVs). The computation complexity of GT2 FLSs usually makes them difficult to be deployed into real applications. In this sense, two versions of the GT2-RBFNN are implemented using GSO as a training optimisation are suggested, i.e. a GT2-RBFNN with a Karnik-Mendel type reduction layer, and a GT2-RBFNN with a Nie-Tan direct defuzzification layer. To compare the effectiveness with respect to other learning techniques such as Covariance Matrix Adaptation Evolutionary Strategies CMA-ES, Particle Swarm Optimisation (PSO), and GD applied to a GT2-RBFNN, as well as neural structures based on Extreme Learning Machine (ELM).

The rest of this paper is organised as follows: Section II provides an overview of GT2 FLS theory, GT2-RBFNN and GSO, while in section III the proposed methods are presented. Section IV presents experiments and results, and conclusions and future work are drawn in section V.

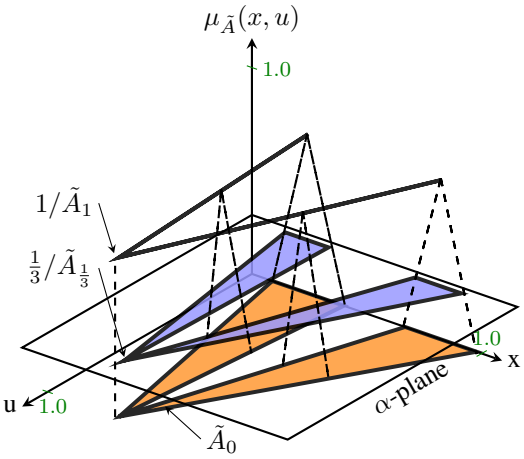


Fig. 1: Some $\alpha-$planes raised to level α for a GT2 FS.

II. Background Theory

This section provides a brief review of General Type-2 Fuzzy Sets (GT2 FSs) and theory of α-plane representation as well as the GT2-RBFNN and GSO methods are described.

A. Definition of a General Type-2 Fuzzy Set

A General Type-2 Fuzzy Set (GT2 FS) denoted by $\tilde{A}$ (also called T2 FS) is characterised by a bivariate MF $\mu_{\tilde{A}}(x,u) \subseteq [0,1]$ on the Cartesian product $\mu_{\tilde{A}} : X \times [0,1]$, where the primary variable is $x \in X$. And the $y-axis$ is called secondary variable or primary MF $u \in J_x \subseteq [0,1]$ as illustrated in Fig. 1. Thus, $\tilde{A}$ is represented by:

$$\tilde{A} = \{(x,u), \mu_{\tilde{A}}(x,u) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \quad (1)$$

$\{\mu_{\tilde{A}}(u) | u \in U\}$ is a vertical slice of $\mu_{\tilde{A}}(x,u)$.

B. $\alpha-$plane Representation

An $\alpha-$plane of a GT2 FS $\tilde{A}$ is denoted by $\tilde{A}_\alpha$, is the union of the primary MFs of $\tilde{A}$ whose secondary grades are greater than or equal to α $(0 \leq \alpha \leq 1)$

$$\tilde{A}_\alpha = \{(x,u), \mu_{\tilde{A}}(x,u) \geq \alpha | x \in X, u \in [0,1]\} \quad (2)$$

where the lower and upper limits for $\tilde{A}_\alpha$ are defined as [10] $[a_{\tilde{\alpha}}, b_{\tilde{\alpha}}] = [LMF(\tilde{A}_\alpha), UMF(\tilde{A}_\alpha)]$. That means when $\tilde{A}_\alpha$ is raised to level α, it is a plane at that level that can be obtained by connecting all the corresponding $\alpha-cuts$ of the associated vertical slices of the secondary MFs of $x \in X$ [1]. where the horizontal-slice representation of a GT2 FS $\tilde{A}$ is:

$$\tilde{A} = \sup_{\alpha \in [0,1]} \alpha / \left[\int_{x \in X} [a_\alpha(x), b_\alpha(x)]/x \right] = \bigcup_{\alpha \in [0,1]} \alpha / \tilde{A}_\alpha \quad (3)$$

C. General Type-2 Radial Basis Function Neural Network

According to [5, 11, 12], a Radial Basis Function Neural Network (RBFNN) can be viewed as Fuzzy Logic System (FLS) under some mild conditions. This functional equivalence has been extended in order to design higher order FLSs based on the model of the RBFNN. An RBFNN can be regarded as an FLSs whose main inference engine is

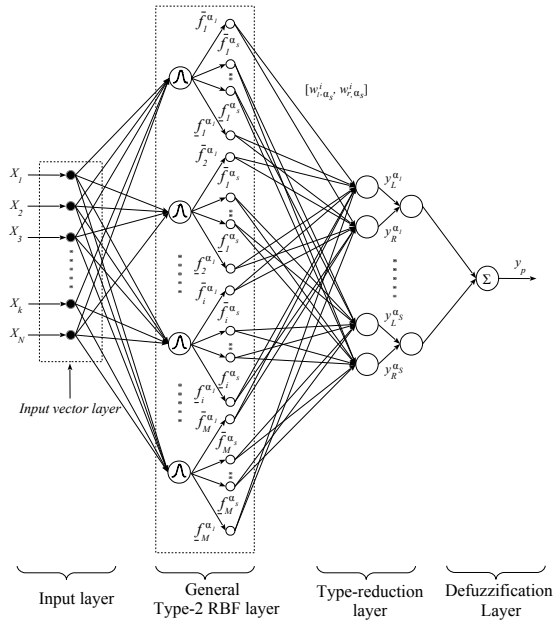


Fig. 2: General Type-2 Fuzzy Neural Network (GT2-RBFNN).

interpreted as an adaptive filter [1]. It resembles an additive weighted combination of the MFs of the fired-rule output sets in the hidden layer of the RBFNN (See Fig. 2) [1]. Each hidden receptive unit in the RBFNN is functionally equivalence to a fuzzy rule R^i described by a multivariable MF $\mu_{R^i}(\vec{x}_p, y_p) = \mu_{R^i}[x_1, \ldots, x_n, y]$ of Gaussian type, whose input vector $\vec{x}_p \in X_1 \times \ldots X_n$ and the implication engine can be defined as:

$$\mu_{R^i}(\vec{x}_p, y) = \mu_{A^i \to G^i} = \left[T^n_{k_1} \mu_{F^i_k}(x_k) \star \mu_{G^i(y)} \right] \quad (4)$$

Where $\star$ is the minimum $t-norm$ that represents the shortest Euclidean distance of the input vector $\vec{x}_p$. And the ith receptive unit is represented as a fuzzy rule in the form:

$$R^i : \text{IF } x_1 \text{ is } F^i_1 \text{ and} \ldots \text{IF } x_k \text{ is } F^i_k \text{ and} \ldots \text{IF } x_n \text{ is } F^i_n \text{ THEN } y \text{ is } G^i;\ i = 1, \ldots, M \quad (5)$$

The firing strength f_i of each receptive unit is defined as:

$$\mu_{A^i \to G^i}(\vec{x}_p, y) = fi \left(\exp \left[- \frac{\sum_{k=1}^n (x_k - m_{ki})^2}{\sigma_i^2} \right] \right) \quad (6)$$

The GT2-RBFNN is a Fuzzy Neural Network of general type-2 (GT2-RBFNN) with a Mamdani inference engine an a Karnik-Mendel type-reduction layer, where all the FSs are of GT2. The adopted neural structure for the GT2-RBFNN is a GT2 FLSs with an uncertain width $\sigma_i = [\sigma_i^1, \sigma_i^2]$ and a fixed mean m_k^i where the input layer is a singleton fuzzification with a secondary MF that is convex, a Center-Of-Sets (COS) type reduction that uses the Karnik-Mendel algorithm and an average of end-points defuzzification (AED). A horizontal-slice representation for the GT2-RBFNN is used. In order

to avoid additional parameters and as the secondary MFs are vertical slices, we choose a isosceles isosceles triangle function where its base is equal to $\bar{f}_i^0 - \underline{f}_i^0$ and its Apex location is

$$\text{Apex}(\vec{x}_p) = \underline{f}_i^0(\vec{x}_p) + w[\bar{f}_i^0(\vec{x}_p) - \underline{f}_i^0(\vec{x}_p)] \tag{7}$$

we choose a value for $w = 1/2$.

D. GT2-RBFNN Input Layer

The adopted GT2-RBFNN is a Multi-Input-Single-Output FLS, in which the input data is a multidimensional crisp vector represented by $\vec{x}_p = [x_1, \ldots, x_n] \in R^n$ where only the current state is fed into the layer and then forwarded to next layer.

E. General Type-2 RBF Layer

Singleton fuzzification is employed, i.e. for each value x_k only a T1 vertical slice for an antecedent GT2 FS $\tilde{F}_k^i$ is activated. Based on [13], for each fuzzy rule and input $\vec{x}_p$ in the GT2-RBFNN (Mamdani type), only one firing interval $F_i^{\alpha_s}$ is activated for level α_s in the **GT2 RBF layer** as follows (See Fig. 2) - $F_i^{\alpha_s} := [\underline{f}_i^{\alpha_s}(\vec{x}_p), \bar{f}_i^{\alpha_s}(\vec{x}_p)]$

$$F_i^{\alpha_s} := \begin{cases} \underline{f}_i^{\alpha_s}(\vec{x}_p) = \exp\left[-\sum_{k=1}^{n}\left(\frac{x_k - m_k^i}{\sigma_i^2}\right)^2\right]_{\alpha_s} \\ \bar{f}_i^{\alpha_s}(\vec{x}_p) = \exp\left[-\sum_{k=1}^{n}\left(\frac{x_k - m_k^i}{\sigma_i^1}\right)^2\right]_{\alpha_s} \end{cases} \tag{8}$$

Note the term α is not a variable, but a subscript $'s'$ to denote in which level the information in the GT2 RBFNN is being processed [1].

F. Type-reduction Layer

In the type reduction layer, we use a Center Of Sets Type Reduction (COS TR). This layer performs a mathematical operation that maps a GT2 FS into a T1 FS. Due to the adaptability of the GT2 RBFNN, the centroid of each consequent at the α_s-plane can be defined as:$C_{\tilde{G}_{\alpha_s}^i} = \alpha_s/[w_{l,\alpha_s}^i, w_{r,\alpha_s}^i]$. According to [13], for a Mamdani GT2-RBFNN $[w_{l,\alpha_s}^i, w_{r,\alpha_s}^i]$ is an Interval Weighted Average (IWA) that is used along with the firing interval $F_i^{\alpha_s}$ to compute the reduced set $[y_l^{\alpha_s}(\vec{x}_p), y_r^{\alpha_s}(\vec{x}_p)]$ for α_s-level as:

$$y_l^{\alpha_s} = \frac{\sum_{i=1}^{L_{\alpha_s}} w_{l,\alpha_s}^i \bar{f}_i^{\alpha_s} + \sum_{i=L_{\alpha_s}+1}^{M} w_{l,\alpha_s}^i \underline{f}_i^{\alpha_s}}{\sum_{i=1}^{L_{\alpha_s}} \bar{f}_i^{\alpha_s} + \sum_{i=L_{\alpha_s}+1}^{M} \underline{f}_i^{\alpha_s}} \tag{9}$$

$$y_r^{\alpha_s} = \frac{\sum_{i=1}^{R_{\alpha_s}} w_{r,\alpha_s}^i \underline{f}_i^{\alpha_s} + \sum_{i=R_{\alpha_s}+1}^{M} w_{r,\alpha_s}^i \bar{f}_i^{\alpha_s}}{\sum_{i=1}^{R_{\alpha_s}} \underline{f}_i^{\alpha_s} + \sum_{i=R_{\alpha_s}+1}^{M} \bar{f}_i^{\alpha_s}} \tag{10}$$

where $Y_{COS,\alpha_s} = 1/[y_l^{\alpha_s}(\vec{x}_p), y_r^{\alpha_s}(\vec{x}_p)]$.

G. Defuzzification Layer

This layer performs the defuzzification that consists of a process of aggregation of all horizontal slices. This work uses the Average of End-Points Defuzzification (AEPD) [3]

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s [(y_l^{\alpha_s}(\vec{x}_p) + y_r^{\alpha_s}(\vec{x}_p))/2] \Big/ \sum_{s=1}^{S} \alpha_s \tag{11}$$

H. Simplified GT2-RBFNN based on Nie-Tan Algorithm

To avoid the iterative nature that frequently results from the number of permutations that are needed to calculate the reduced set with KM algorithms, simplified structures with direct-defuzzification have been proposed [5]. Usually, the term direct-defuzzification or closed-form type reduction is used indistinctly to refer to the mapping that goes from GT2 FS to a crisp number (type-0). Because of its simplicity and accuracy with respect to KM algorithms, in this work, a GT2 RBFNN with a Nie-Tan closed-form (NT) as output layer is suggested as a comparison method to the GT2-RBFNN using KM. Such method uses the vertical representation of the Footprint of Uncertainty (FOU) [14] before the process of dedifuzzification to finally compute the centroid of the IT2 FS. The NT layer can be considered a zero order Taylor series approximation of Karnik-Mendel+dedifuzzification methods. It has been proved the Nie-Tan operator is equivalent to an exhaustive and accurate type-reduction for both discrete and continuous IT2 FSs [14]. Although there has been improvements on the Nie-Tan operator, in this paper, the centroid y_{NT,α_s} at each $\alpha-$level is:

$$y_{NT,\alpha_s} = \frac{\sum_{i=1}^{M} w_i^{\alpha_s}\left(\underline{f}_i^{\alpha_s} + \bar{f}_i^{\alpha_s}\right)}{\sum_{i=1}^{M} \underline{f}_i^{\alpha_s} + \sum_{i=1}^{M} \bar{f}_i^{\alpha_s}} \tag{12}$$

For each input vector $\vec{x}_p$, the GT2RBFNN output with a NT method is:

$$y_p(\vec{x}_p) = \sum_{s=1}^{S} \alpha_s y_{NT,\alpha_s} \Big/ \sum_{s=1}^{S} \alpha_s \tag{13}$$

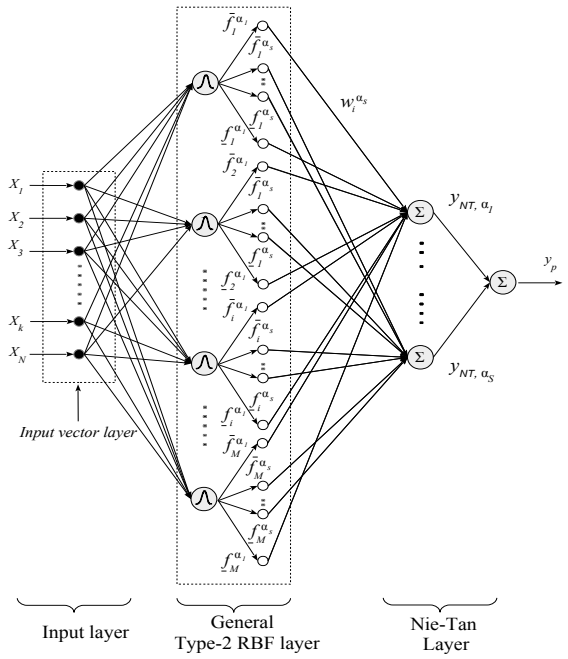


Fig. 3: Nie-Tan GT2 RBFNN.

Algorithm 1: PSEUDOCODE FOR THE EVOLUTION OF AN GT2-RBFNN USING A GALATIC SWARM OPTIMISATION (GSO)

Input: Input Training Data $(\mathbf{x}_p, t_p)$
Output: Optimal m_s^k, σ_k^1, σ_k^2 and $w_{l,\alpha_s}^i, w_{r,\alpha_s}^i$

1 **function** Galactic Swarm Optimisation (GSO)
2 **Level 1** Initialisation: $\mathbf{x}_{ij}, \mathbf{v}_{ij}$, $\mathbf{g}_{best,i}$, $\mathbf{p}_{best,ij}$
3 **Level 2** Initialisation: $\mathbf{v}_i$, $\mathbf{g}_{best}$, $\mathbf{p}_{best,i}$
4 Calculate $J_t(\mathbf{x}_{ij}),\ j = 1, \ldots, N_p$
5 Initialise the particle's best position $p_{best,j} \leftarrow \hat{x}_j$
6 **while** $t \leq\ Ep_{MAX}$ **do**
7 Begin PSO: **Level 1**
8 **for** $i \leftarrow 1$ *to* S_p **do**
9 **for** $l_1 \leftarrow 0$ *to* L_1 **do**
10 **for** $j \leftarrow 1$ *to* N_p **do**
11 Update $\mathbf{v}_{ij}$ and $\hat{\mathbf{x}}_{ij}$ (Eq. 19,20)
12 Calculate $J_t(\hat{\mathbf{x}}_j)$ of solutions $\hat{\mathbf{x}}_{ij}$
13 Select the best GT2 parameters
14 **if** $J_t(x_p) < \boldsymbol{p}_{best,ij}$ **then**
15 Update $\mathbf{p}_{best,ij} \leftarrow \mathbf{x}_i$
16 **if** $\boldsymbol{p}_{best,ij} < \boldsymbol{g}_{best,i}$ **then**
17 Update $\mathbf{g}_{best,i} \leftarrow \mathbf{p}_{best,ij}$
18 Begin PSO: **Level 2**
19 Initialise Swarm $\mathbf{y}_i = \mathbf{g}_{best,i},\ i = 1, \ldots, S_p$
20 **for** $l_2 \leftarrow 0$ *to* L_2 **do**
21 **for** $i \leftarrow 1$ *to* Np **do**
22 Update star's velocity
23 $\mathbf{v}_i = w_2\mathbf{v}_i + c_3 r_3 \left(\mathbf{p}_{best,i} - \mathbf{y}_i\right) + c_4 r_4 \left(\mathbf{g}_{best} - \mathbf{y}_i\right)$
24 Update particle's position $\mathbf{y}_i \leftarrow \mathbf{y}_i + \mathbf{v}_i$
25 Calculate fitness $J_t(\hat{\mathbf{x}}_j)$ of each optimal solution $\hat{\mathbf{x}}_{ij}$
26 **if** $J_t(\hat{\boldsymbol{x}}_p) < \boldsymbol{p}_{best,i}$ **then**
27 Update $\mathbf{p}_{best,i} \leftarrow \mathbf{y}_i$
28 **if** $\boldsymbol{p}_{best,i} < \boldsymbol{g}_{best}$ **then**
29 Update $\mathbf{g}_{best} \leftarrow \mathbf{p}_{best,i}$
30 $t = t + 1$
31 **return** $(m_s^k, \sigma_i^1, \sigma_i^2, w_{l,\alpha_s}^i, w_{r,\alpha_s}^i)_{best}$

I. Galactic Swarm Optimisation

Galactic Swarm Optimisation (GSO) is a new meta heuristic technique that mimics the movement of stars, galaxies and super galaxies [15]. GSO involves two independent levels of execution, where first multiple $'S_p'$ sub-populations of $'N_p'$ solutions (stars) are randomly created to explore search space efficiently. Such populations are small galaxies (also called subswarms) that interact themselves while updating their current position $\hat{x}_{ij}$ and minimising their potential energy [15]. Secondly, a super swarm of stars is recruited from the best-found solutions $\mathbf{g}_{best,i}$ at each ith sub-population. At each level, the main search engine is based on the original work of Particle Swarm Optimisation (PSO) [15]. That is, at first level every jth star in the ith sub-population moves with a specific velocity v_{ij} while keeping track of its best position $\mathbf{p}_{best,ij}$. Thus, at each time step, each particle changes its velocity and position (direction) $\mathbf{x}_{ij}$ towards the best location $\mathbf{g}_{best,i}$. The associated acceleration of each star is weighted by a random term. Hence, the velocity and position is defined

$$\mathbf{v}_{ij} = w_d\mathbf{v}_{ij} + c_1 r_1 \left(\mathbf{p}_{best,ij} - \mathbf{x}_{ij}\right) + c_2 r_2 \left(\mathbf{g}_{best,i} - \mathbf{x}_{ij}\right) \quad (14)$$

$$\hat{x}_{ij} = \hat{x}_{ij} + v_{ij}; j = 1, \ldots, N_p \quad (15)$$

where $c_1, c_2 > 0$ are acceleration constants; r_p and r_g are random numbers between 0 and 1. The term w_d is used for adaptation purposes as an inertial weight. Such parameter is decreased gradually as the number of generation for the PSO increases according to the rate:

$$w_d(t) = (w_{max} - w_{init})/Max_{iter} \quad (16)$$

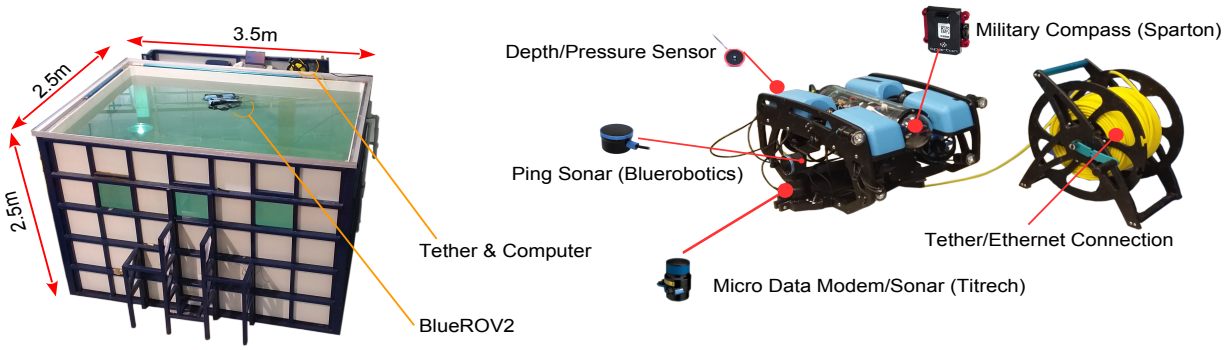
in which, w_{init} and w_{max} are the initial and final inertial weights respectively. Finally, PSO is applied again to find the set of the best global optimum using the set of $g_{best,ij}$.

III. METHODS

A. Evolution of the GT2-RBFNN using GSO

The evolution of the GT2-RBFNN with a Gaussian MF having a fixed mean m_{si} and a variable standard deviation $[\sigma_i^1, \sigma_2^2]$ and KM type-reduction, whose learning methodology is based on the GSO is described in the **Algorithm 1**. A Multiple-Input-Single-Output (MISO) structure for the GT2-RBFNN is selected. According to Algorithm 1, given a predefined number of fuzzy rules, the GSO starts from randomly selecting the values of each GT2 antecedent whose particle's codification $\hat{x}_j \sim U(l_j, u_j)$ is described in Eq. (21) (line 2), where l_j and u_j are the lower and upper dimension limits respectively. The fitness of each candidate model $J_t(\mathbf{x}_p)$ is estimated using the Root-Mean-Squared-Error (line 4, Eq. 20) in which N_p is the number of particles. At each iteration, two optimisation levels are performed (line 7-29). At first level (7-17), the best candidate (solution, $\mathbf{g}_{best,i}$) of each sub-population is found using the fitness function $J_t(\mathbf{x}_p)$, which is used to create a super swarm (line 13-17). At second level (line 18-29), the best solution $\mathbf{g}_{best}$ in the super swarm is used as the final parameters $m_s^k,\ \sigma_k^1, \sigma_k^2$ of each antecedent as well as the optimal value for the consequent vector $w_{l,\alpha_s}^i, w_{r,\alpha_s}^i$ defined in the output layer of the GT2-RBFNN.

$$\hat{x}_j = \Bigg(\underbrace{\overbrace{m_{k1}, \sigma_1^1, \sigma_1^2}^{\text{GT2 antecedent 1}}, \ldots, \overbrace{m_{kM}, \sigma_M^1, \sigma_M^2}^{\text{GT2 antecedent n}}}_{\text{Rule 1}}, \ \ldots, \underbrace{\overbrace{m_{k1}, \sigma_M^1, \sigma_M^2}^{\text{GT2 antecedent 1}}, \ldots, \overbrace{m_{kM}, \sigma_M^1, \sigma_M^2}^{\text{GT2 antecedent n}}}_{\text{Rule M}}, \underbrace{w_{l,1}^i, w_{r,1}^i, \ldots, w_{l,\alpha_S}^i, w_{r,\alpha_S}^i}_{\text{Consequent weights}} \Bigg) \quad (17)$$



(a) Water container used for experiments

(b) Remotely Operated Underwater Vehicle (BlueROV2) and its sensory system

Fig. 4: (a) Water container (open-tank) and computer station used for experiments, and (b) BlueROV2 equipped with a sensory system with four sensors.

$$J_t(\mathbf{x}_p) = \left(\frac{1}{P}\sum_{p=1}^{P}(y_p - t_p)^2\right)^{1/2} \tag{18}$$

Model accuracy of the GT2-RBFNN is calculated as follows:

$$Model\ Accuracy(\%) = \frac{TN + TP}{TP + TN + FP + FN} \tag{19}$$

Where TP and TN are the true positive and true negative classification respectively. FP and FN are false positive and false negative classification correspondingly. in which N_p is the number of particles and E_{pMAX} is used to denote the max number of evolutionary epochs.

B. Robotic Platform

In this research work, a Remotely Operated Underwater Vehicle (ROV) with a six-thruster vectored configuration from Bluerobotics (See Fig. 4) was used for all experiments. The ROV has an open-source electronics whose sensory system (See Fig. 5) was integrated at the Laboratory of Submarine Robotics, (LSR, CIDESI). As detailed in Fig. 4, such system consists of a pressure sensor that is able to measure up to 30 Bar (300m depth) with a depth resolution of 2mm (Bluerobotics), 2) a ping sonar which is an open-source sensor able to measure distances up to 30 meters with a 30 degree beam width, b) micro data sonar Titrech with a range resolution of 7.5 mm, a beam width of 3°, and a variable scanned sector and d) the Sparton compass that is a micro-sized and light weight attitude heading sensor with a Static Heading Accuracy of $0.2°$ RMS and full $360°$ rollover capability. As detailed in Fig. 4, the micron data sonar is used as a dynamic echo-sounder whose scanned sector is defined by a sample window of five beams separated at $8°$ one to the other. A water container of $2.5 \times 2.5 \times 3.5$ metres in height, width and length respectively, was used to carried the experiments with salty water whose

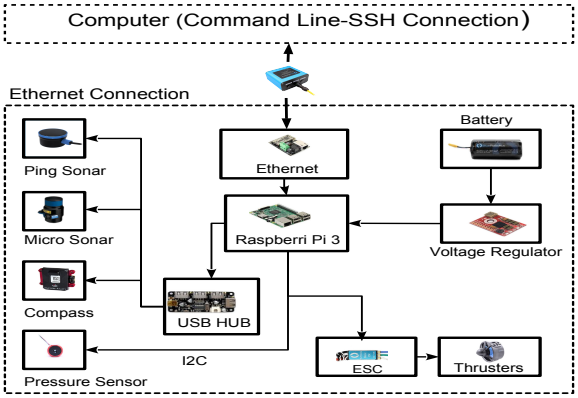
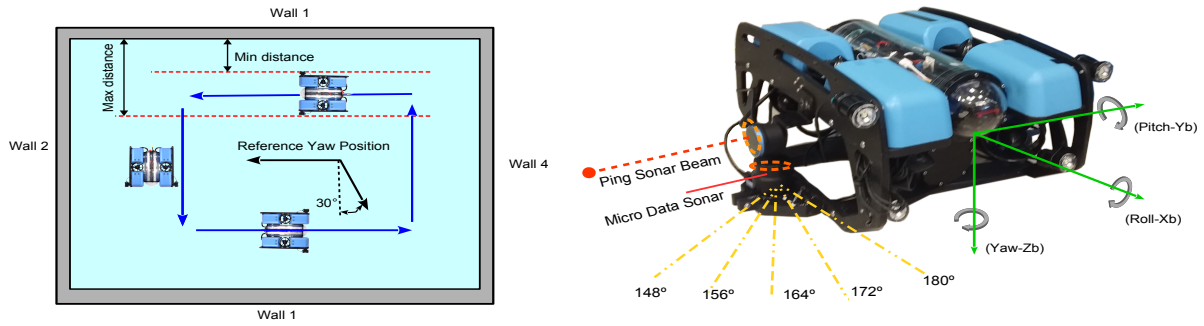


Fig. 5: System's configuration used by the BlueROV2.

density is about $1028kg/m^3$. As indicated in Fig. 5, the main computer in the ROV is a Raspberri Pi3, in which ROS ubiquity (algorithms coded in Python, C++ and Matlab) was installed in order to implement all machine learning algorithms and controllers. A line SSH connection between the Raspberri Pi3 and an Ubuntu computer was used to monitor sensor values and define parameters of each algorithm.

C. Proposed Methodology for Trajectory Planning based on an Evolutionary GT2-RBFNN

GT2 Fuzzy Logic has demonstrated to be an efficient concept to dealing with different types of uncertainties. In particular, in real modelling and prediction problems [5, 11] GT2 FLSs quantify uncertainty not only from imprecise boundaries in Fuzzy Sets (FSs) but also as ambiguity due to the variation in the output system. In this work, the implementation of a GT2 neural network (GT2-RBFNN) is implemented in the ROV to discriminate (binary contour classification) between walls and corners when moving forward in the water container



(a) Top view of the UAV BlueROV2 trayectory

(b) Site view of the UAV BlueROV2

Fig. 6: (a) Top view of one circuit (blue line), where the North reference position is aligned to 30° and (b) Site view of the BlueROV2 using a ping sonar (1 reading beam) and a micro data sonar with 5 reading beams.

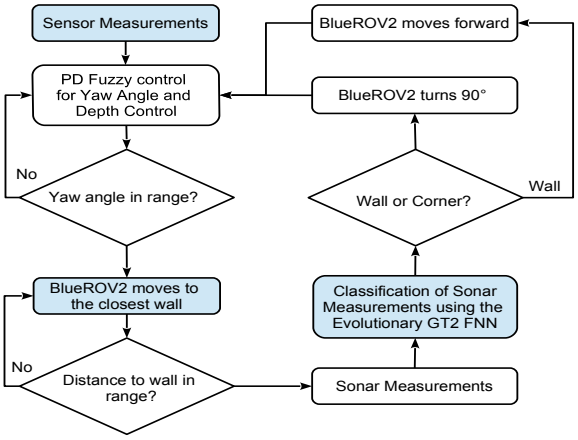


Fig. 7: Flow diagram of the processes implemented to the trajectory planning of the ROV to complete one circuit.

(tank) as it is illustrated in Fig. 4. The idea behind the proposed methodology is to integrate a number of control algorithms that guide the to complete a circuit (as shown in Fig. 6(a)) that consists of the trajectory passing by all walls (from 1 to 4 or from 4 to 1) either at clockwise or counterclockwise direction, Fig. 6). As illustrated in Fig. 7 (flow diagram), a compass and a pressure sensor are used to estimate the yaw angle (See Fig. 4(b)- axis Zb) and the ROV's depth (axis Zb) respectively. The ping sonar and the micro data sonar are utilised to estimate the distance between the ROV and the closest wall. While the ping sonar estimates a straight distance, as described in Fig. 4, the micro data sonar obtains five different distances (estimates) in a predefined scanned sector, where the value of the reading 180° is aligned to the ROV front. In Fig. 8, sample data obtained by the micro data sonar that corresponds to one circuit of the ROV in the container and from three different angles, namely; $[148^\circ, 164^\circ, 179^\circ]$ is presented. Based on Fig. 6, the reading frequency of the ROV sensory system is about 0.02Hz. The initial target of the ROV's trajectory is to achieve a predefined heading angle that is within the error range between $[-5^\circ, +5^\circ]$ making the heading of the vehicle parallel to the closest wall. To control the orientation and depth of the ROV (yaw angle) a PD fuzzy controller of type-1 (Mamdani) with three and five fuzzy rules were implemented respectively [16, 17]. Since a dynamic model of the BlueROV2 is not available, the dynamic properties of the closed-loop structure have to be derived intuitively and experimentally. Once the ROV heading is within a predefined yaw angle range, the ROV employs the ping sonar readings to move to a position that is within a predefined distance range [min distance, max distance] parallel to the closest wall as described in Fig. 6(a). At this position the ROV moves forward while scanning five different distances to the closest walls at the angles $[148^\circ, 156^\circ, 164^\circ, 172^\circ, 179^\circ]$. Such data is used by the evolutionary GT2-RBFNN to discriminate between a wall and a corner. If a corner is reached by the ROV, then the robot rotates 90° counterclockwise.

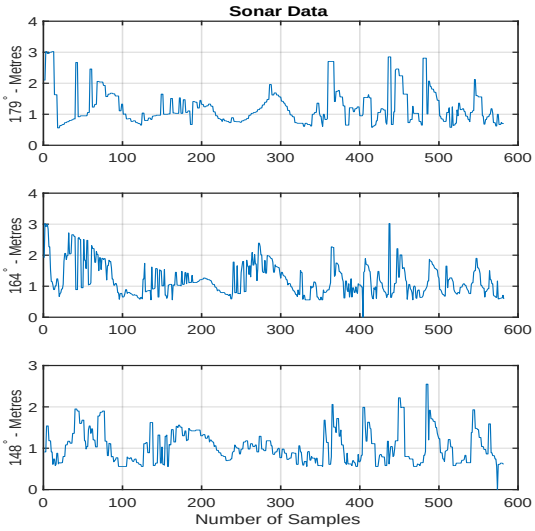


Fig. 8: Sample of Sonar measurements used as training data.

IV. Experimental Results

In this section, the efficiency of the proposed framework is compared to other existing machine learning techniques. Training data Sonar from five random experiments was collected using the micro data sonar resulting in a collection 586 records (See Fig. 8), where each input vector consists of five attributes, each one corresponding to a distance measured at angles $\{180°, 172°, 164° 156°, 148°\}$ (See Fig. 6(b)). The sonar data set is an imbalanced binary classification problem where label 1 is used to denote the presence of a corner, and 0 to denote a wall (438 records out of 582). For cross validation purposes, a number of ten off-line random experiments were implemented. From this, the sonar data set was divided into two subsets, i.e. 80% for training and 20% for testing. For real experiments, in order to make a decision about the type of contour, the average classification value of three consecutive values of y_p, $(y_p t + y_p^{t-1} + y_p^{t-2})/3 > \tau$ are used, where τ is a predefined threshold. The experimental setup for the GSO consists of a maximum number of 200 evolutionary epochs, optimal population size was found to be 40. The GT2-RBFNN consists of 5 fuzzy rules with three slices ($S = 3$), $\tau = 0.7$, and a $[min, max]$ value for $\sigma_k^1 = [0.3, 3.0]$, $\sigma_k^2 = [0.3, 2.0]$, and the limits for $m_s^k = [-1, 1]$. $w_{r,\alpha}^i = w_{l,\alpha}^i = [-5, 5]$. As indicated in Table I, to evaluate the GT2-RBFNN accuracy, four different training configurations are suggested, i.e. GT2-RBFNN GSO + KM = GSO used to train a GT2-RBFNN with a KM type-reduction layer, GT2-RBFNN + GSO + NT, GT2-RBFNN PSO + KM and GT2-RBFNN + GD + KM. In Fig. 9, their corresponding evolution performance is illustrated. In Table I, the average performance of ten random experiments using seven different machine learning algorithms is described. ELM, FELM and ML-ELM are neural structures based on Extreme Learning Machine (ELM), Fuzzy ELM (FELM) and Multilayer ELM (ML-ELM) respectively. From Table I, Column time(s) indicates the average training time required by each algorithm. According to our cross-validation results, it is clear that ML-EL with two layers of AutoEncoders (AEs) and one classification layer provides the highest trade-off between accuracy and computational burden. The number of Hidden Units (column No. of HU) is selected as $[200, 70]$, the first and second AEs has 200 and 100 units respectively, while a No. of HU for the classification layer is 70.

TABLE I: AVERAGE PERFORMANCE FOR CONTOUR CLASSIFICATION.

	Model	Training		Testing	
		Mean (%)	Time(s)	Mean (%)	No. of HU
GT2-RBFNN	GSO + KM	94.30	31.01	90.29	7
	CMA-ES + KM	92.95	28.56	91.22	7
	GSO + NT	92.08	29.57	89.37	7
	PSO + KM	92.74	24.33	89.01	7
	BEP + KM	93.22	22.14	86.12	7
	ELM	94.36	1.30	93.98	90
	FELM	95.77	3.21	94.19	90
	ML-ELM	98.61	1.23	95.01	[200, 70]

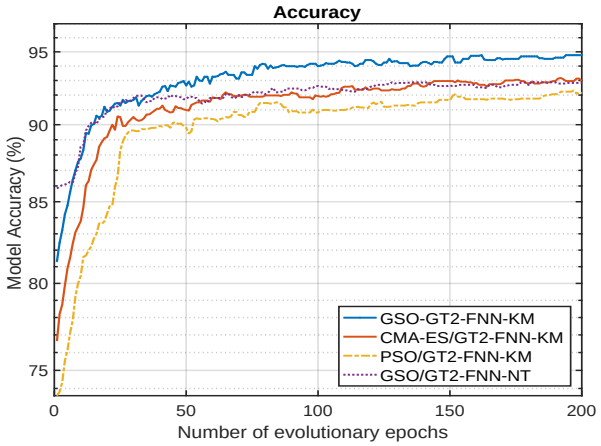


Fig. 9: Average training accuracy for the GT2-RBFNN using GSO, PSO and CMA-ESs.

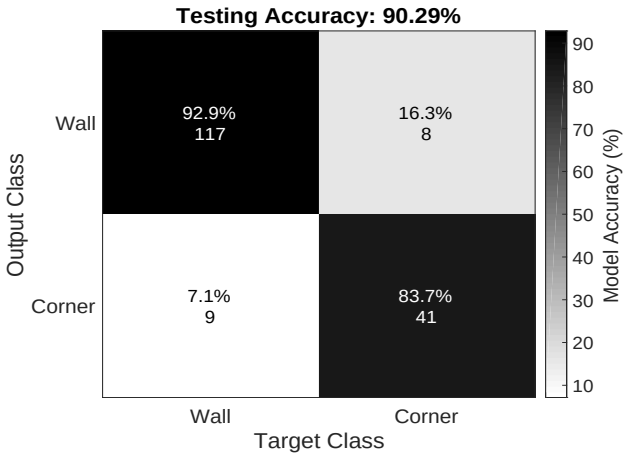


Fig. 10: Average testing accuracy.

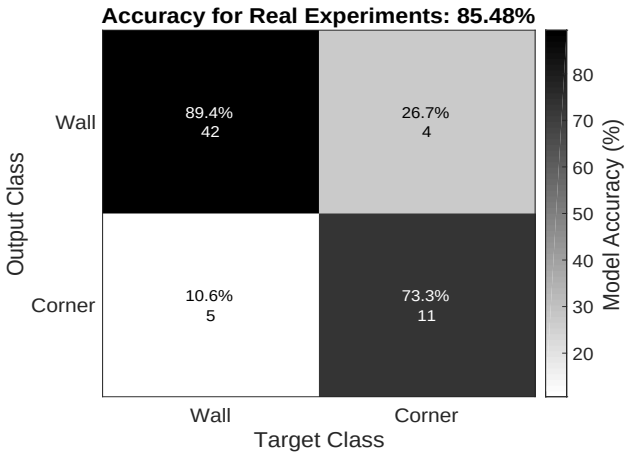


Fig. 11: Average testing accuracy for real experiments.

For practical purposes, those models trained with an evolutionary optimisation and BEP represent a higher trade-off between model accuracy and model simplicity. From this, in

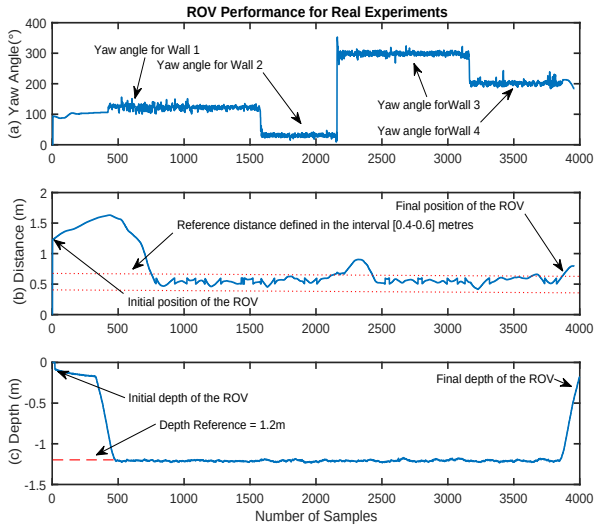


Fig. 12: ROV's position for a random experiment (One circuit).

particular, an evolutionary GT2-RBFNN provides a higher accuracy with better generalisation properties. In Fig. 10, a confusion matrix that corresponds to average testing performance of the evolutionary GSO is presented. To evaluate the performance of the ROV, five experiments were carried out using the trained neural structure of the GT2-RBFNN for a depth of $1.2m$, in order to complete a counterclockwise circuit delineated by the point sequence wall(1)-to-wall(2)-to-wall(3)-to-wall(4) as described in Fig. 6. The average classification of five experiments provided by the GT2-RBFNN to recognise contours (accuracy: 85.48%) is presented in Fig. 11. As can be noted from Fig. 11, the ability of the GT2-RBFNN to properly suffers a small decrease when discriminating between walls and corners. In Fig. 12, the performance achieved by the yaw and depth fuzzy controllers and the distance that the ROV kept between its current position and the closest wall throughout a random circuit is presented respectively. As can be observed from Fig. 12, the reference distance width established for these experiments was proposed with limits $[0.4 - 0.6]m$.

V. Conclusions and Future Work

In this paper, an evolutionary General Type-2 Fuzzy Neural Network (GT2-RBFNN) that is based on the functional equivalence between RBFNNs and GT2 FLS for trajectory planning in Remotely Operated underwater Vehicles (ROVs) is suggested. The proposed GT2-RBFNN is used as a data-driven learning system to orient the current position of ROVs in underwater environments. The parameter identification of the GT2-RBFNN is determined by using Galactic Swarm Optimisation (GSO). To compare the performance of the GT2-RBFNN, a number of different techniques based on evolutionary computing and Extreme Learning Machine (ELM) were suggested. A squared water container and an ROV BlueROV2 were employed to run all experiments. To control the ROV's position (depth, yaw and motion), a sensory system that includes a micro data sonar, a pressure sensor, a ping sonar and a compass was integrated. Based on our results, the GT2-RBFNN offers a robust trade-off between model accuracy and model simplicity in underwater environments.

Future work includes underwater environments whose contour is unknown as well as the implementation of online learning methods that allows the GT2-RBFNN updates its parameters in the presence of new evidence.

References

[1] J. M. Mendel, "General type-2 fuzzy logic systems made simple: a tutorial," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 5, pp. 1162–1182, 2013.

[2] C. Gonzalez, P. Melin, and O. Castillo, "Edge detection method based on general type-2 fuzzy logic applied to color images," *Information*, vol. 8, no. 3, p. 104, 2017.

[3] C. Wagner and H. Hagras, "Toward general type-2 fuzzy logic systems based on zslices," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 4, pp. 637–660, 2010.

[4] J. Andreu-Perez, F. Cao, H. Hagras, and G.-Z. Yang, "A self-adaptive online brain–machine interface of a humanoid robot through a general type-2 fuzzy inference system," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 101–116, 2016.

[5] A. Rubio-Solis, P. Melin, U. Martinez-Hernandez, and G. Panoutsos, "General type-2 radial basis function neural network: A data-driven fuzzy model," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 2, pp. 333–347, 2018.

[6] A. S. Handayani, N. L. Husni, S. Nurmaini, and I. Yani, "Application of type-1 and type-2 fuzzy logic controller for the real swarm robot." *International Journal of Online Engineering*, vol. 15, no. 6, 2019.

[7] M. Almaraashi, R. John, A. Hopgood, and S. Ahmadi, "Learning of interval and general type-2 fuzzy logic systems using simulated annealing: Theory and practice," *Information Sciences*, vol. 360, pp. 21–42, 2016.

[8] F. Ali, E. K. Kim, and Y.-G. Kim, "Type-2 fuzzy ontology-based semantic knowledge for collision avoidance of autonomous underwater vehicles," *Information Sciences*, vol. 295, pp. 441–464, 2015.

[9] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 178, no. 9, pp. 2224–2236, 2008.

[10] A. R. Solis and G. Panoutsos, "Granular computing neural-fuzzy modelling: A neutrosophic approach," *Applied Soft Computing*, vol. 13, no. 9, pp. 4010–4021, 2013.

[11] A. Rubio-Solis and G. Panoutsos, "Interval type-2 radial basis function neural network: a modeling framework," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 457–473, 2014.

[12] A. Rubio-Solis, U. Martinez-Hernandez, and G. Panoutsos, "Evolutionary extreme learning machine for the interval type-2 radial basis function neural network: A fuzzy modelling approach," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–8.

[13] J. M. Mendel, "Comparing the performance potentials of interval and general type-2 rule-based fuzzy systems in terms of sculpting the state space," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 1, pp. 58–71, 2018.

[14] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2012.

[15] V. Muthiah-Nakarajan and M. M. Noel, "Galactic swarm optimization: A new global optimization metaheuristic inspired by galactic motion," *Applied Soft Computing*, vol. 38, pp. 771–787, 2016.

[16] I. Akkizidis, G. Roberts, P. Ridao, and J. Batlle, "Designing a fuzzy-like pd controller for an underwater robot," *Control Engineering Practice*, vol. 11, no. 4, pp. 471–480, 2003.

[17] P. Londhe, B. Patre, L. M. Waghmare, and M. Santhakumar, "Robust proportional derivative (pd)-like fuzzy control designs for diving and steering planes control of an autonomous underwater vehicle," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 3, pp. 2509–2522, 2017.