Implementation of Neuro-like Network Defuzzifier for Mobile Platform Motion Control System

Vasyl Teslyuk [1], Ivan Tsmots [1], Roman Tkachenko [1], Pavlo Tkachenko [2], Vasyl Rabyk [3], Yurii Opotyak [1] and Oleksandr Oliinyk [1]

[1] *Lviv Polytechnic National University, 12 Bandery St, Lviv, 79013, Ukraine*
[2] *Department of Information Technologies, IT STEP University, 83a Zamarstynivska St, Lviv 79090, Ukraine*
[3] *Ivan Franko National University of Lviv, 1 Universytetska St, Lviv, 79000, Ukraine*

Abstract

Modern mobile platforms use artificial intelligence technologies for movement control. A feature of the mobile platform control system is the ability to adapt to changing conditions of the external environment, which is advisable to implement using fuzzy logic. However, in systems with fuzzy logic, there is difficulty in choosing a defuzzification algorithm. This paper proposes a suitable method and structure of a neuro-like network defuzzifier for a mobile platform motion control system. For a wheeled mobile platform, the main stages of fuzzy motion control are described, the structure of the block for forming fuzzy conclusions is developed, and the corresponding methods of training and functioning of the neural network defuzzifier are described. The method of defuzzification using a neural network was developed, and the structure of the fuzzy logic software controller and defuzzifier was determined. The obtained duration of the defuzzification operation in the software implementation for computer and microcomputer platforms allows us to talk about ensuring the control process in real time.

Keywords

wheeled mobile platform, fuzzy control system, neuro-like network defuzzification.

1. Introduction

Today, classical methods of mobile platform (MP) control in a stationary environment are known, which are based on the use of dynamic and kinematic characteristics of the movement of the MP. However, such methods are ineffective when it is necessary to operate the MP in uncertain conditions using inaccurate or incomplete information about the external environment, which is unstructured and dynamic. Fuzzy logic (FL) methods are used to control the movement of vehicles in such conditions, in which the state of the external environment and the values of vehicle movement parameters are described by fuzzy values.

FL methods are used in cases where it is difficult or impossible to describe the control system using traditional mathematical methods. The main advantages of using FL methods to control the movement of vehicles are:

- increasing the stability of management compared to other methods;
- the possibility of obtaining a result even when using ambiguously specified input values;
- reducing the time of development of management tools;
- implementation of management in conditions of uncertainty in the presence of only qualitative information;
- low sensitivity to changes in the parameters of the control object.

COLINS-2024: 8th International Conference on Computational Linguistics and Intelligent Systems, April 12–13, 2024, Lviv, Ukraine

vasyl.m.teslyuk@lpnu.ua (V. Teslyuk); ivan.h.tsmots@lpnu.ua (I. Tsmots); roman.o.tkachenko@lpnu.ua (R. Tkachenko); pavlo.tkachenko@itstep.org (P. Tkachenko); vasyl.rabyk@lnu.edu.ua (V. Rabyk); yurii.v.opotiak@lpnu.ua (Yu. Opotyak); oleksandr.o.oliinyk@lpnu.ua (O. Oliinyk)

0000-0002-5974-9310 (V. Teslyuk); 0000-0002-4033-8618 (I. Tsmots); 0000-0002-9802-6799 (R. Tkachenko); 0000-0003-1858-9063 (P. Tkachenko); 0000-0003-2655-0812 (V. Rabyk); 0000-0001-9889-4177 (Yu. Opotyak); 0009-0000-5093-7334 (O. Oliinyk)

The main disadvantages of the methods of controlling the movement of vehicles using FL are the following:

• the type and parameters of the membership functions describing the input and output variables of the system are chosen subjectively and may partially not correspond to reality;

• the difficulty of adapting the movement of vehicles to changes in the external environment due to the static nature of the rule base;

• the difficulty of choosing a defuzzification algorithm.

Therefore, an urgent task is to develop an approach to the implementation of the defuzzification process for use in control systems based on fuzzy logic. One of the ways to implement such tools is the use of neural or neuro-like networks (NLN).

2. Related Works

The development of means of mobile platform control with the help of computational intelligence is an important problem. In paper [1] for the given environment, a fuzzy control is designed based on the fuzzy controller and the respective fuzzy Control rules.

Paper [2] presents a robot platform used to demonstrate applications of fuzzy logic. The actions of the robot are based on a predefined set of fuzzy rules, which may be altered as preferred by the user. The design and implementation of a wheeled mobile robot using fuzzy logic principles with Mamdani's fuzzy inference system for obstacle avoidance capability are described in [3]. Robots that use type-2 fuzzy logic for controlling their behavior in detail are described in papers [4, 5].

But, in recent years, the neural network has been considered a part of the robot control system [6, 7]. Neural networks can learn and as a result, adapt to environmental changes. To develop intelligent control the application of neural networks in control systems is important thing. In the paper [8], a variety of schemes of Neural Networks applications, used in robotic areas are discussed.

In the paper [9] presented a method for designing robust adaptive control of strict-feedback systems based on neural network controller. In this case neural network is used to approximate the uncertainty functions.

Adaptive Neural Network Dynamic Surface Control algorithm presented in [10] to design a motion controller for a wheeled robot that must track the desired trajectory.

Analysis shown that a promising approach is to combine neural or neuro-like networks and fuzzy logic methods to achieve more robust control [11, 12] in all aspects. To solve the problem of robot path control, the intelligent fuzzy control algorithm was studied in [13]. Paper [14] described a method of a fuzzy controller design to control the mobile platform using fuzzy logic. Separate works are devoted to the construction of defuzzifiers, which is one of the important elements of a fuzzy system [15, 16].

The main problem here is choosing the appropriate method for calculating the defuzzified value [17]. In [18] to solve this problem it is proposed polar coordinates transformation of the periodic membership function and its defuzzification method on polar coordinates. In the article [19] authors propose a fast method for calculating the center of gravity defuzzification of polygonal interval type-2 fuzzy sets on discrete and continuous domains without the need of discretization. This method calculates the accurate center of gravity for trapezoidal and triangular interval type-2 fuzzy sets. Authors in [20] considered the possibility of defuzzifying multiple general type-1 fuzzy sets, given that their defuzzified values are bound by a single, known constraint. This stems from spatial data processing, where obtained number of fuzzy sets whose defuzzified values should sum up to a crisp and known value.

In the paper [21] proposed a general framework for automated reasoning with rules containing Vertical Slice based General Type-2 propositions in both the antecedent and the consequent. Proposed Vertical slice based defuzzification is time-efficient in comparison to its z-slice based counterpart as the former requires type-1 centroidal defuzzification only in μ-u plane, while the latter requires Karnik-Mendel defuzzfication over each z-slice.

In [22] it is shown that the expected value operator is a widely used defuzzifier on fuzzy numbers. Paper introduces the weighted expected value operators from fuzzy numbers, which allow weighting the α-cuts. It's shown that the weighted expected value operators are defuzzifiers and that they are additive and scale invariant.

3. Methods and Materials

3.1. Fuzzy Logic Methods for Controlling Mobile Platform Motion

Fuzzy logic methods for controlling the movement of MP operate with fuzzy values, which include fuzzy sets, fuzzy variables, and linguistic variables. A fuzzy set A in some space X is given by a set of pairs

$$A = \{(x, \mu_A(x)); x \in X\}, \quad (1)$$

where $\mu_A(x)$ is the membership function, which shows the degree of belonging of each element to the fuzzy set . The maximum value of the membership function is one, which means that the element belongs to a fuzzy set.

A fuzzy variable is given by a set of $< \alpha, X, A >$, where is the name of the fuzzy variable; X – the scope of its definition; A is a fuzzy set containing the possible values that the fuzzy variable α can take.

A linguistic variable is given by a set of $< \beta, T, X, G, M >$, where β is the name of the linguistic variable; T is the set of values of a linguistic variable (term-set); X – the area of definition of fuzzy variables; G is a syntactic procedure that allows you to generate new terms (meanings) using T elements; M is a semantic procedure that allows you to match a new meaning of a linguistic variable obtained with the help of a procedure G with the corresponding fuzzy set.

The structure of MP motion control systems using FL is shown in Figure 1.

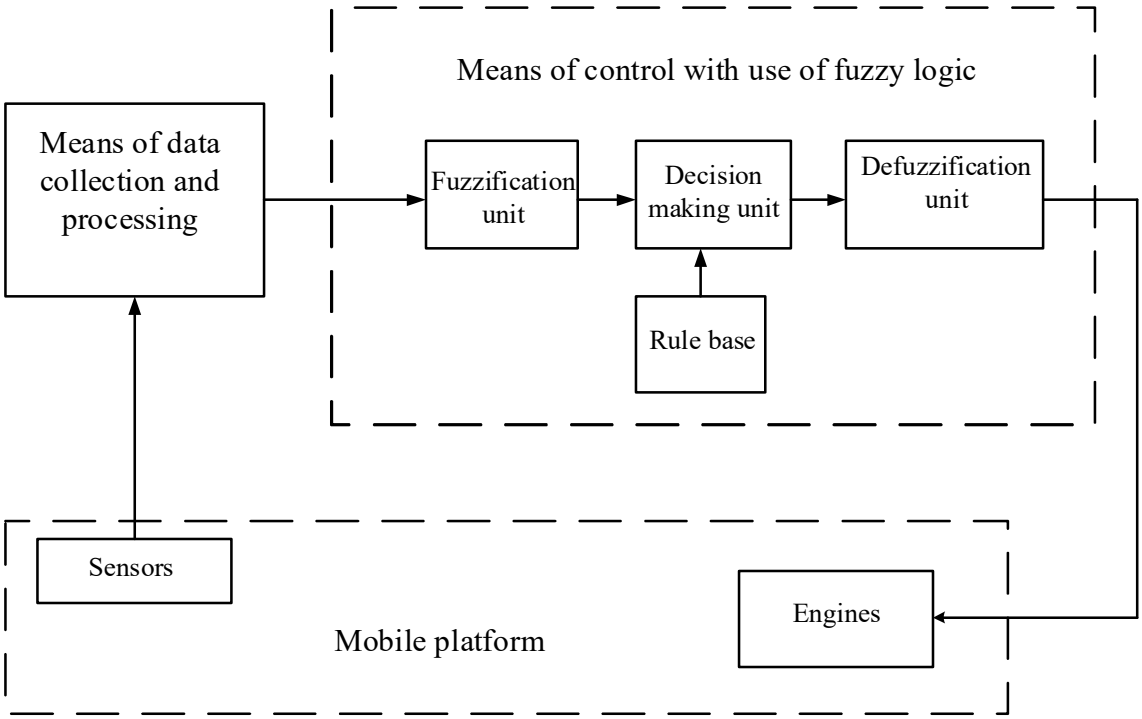

Figure 1: Structure of the MP Motion Control System Using Fuzzy Logic

The main components of the MP control system using fuzzy logic are distance sensors, a fuzzification unit, a decision-making unit, a rule base, and a defuzzification unit. Changes in the speed and trajectory of the MP are implemented by means of control using fuzzy logic based on information from onboard navigation sensors. The main stages of the formation of signals for

controlling the movement of MP using fuzzy logic are fuzzification, decision-making, and defuzzification. Let's consider each stage of the operation of the MP motion control tools using FL in more detail.

3.2. The main stages of fuzzy control of the movement of the MP

Fuzzification stage. At the fuzzification stage, the input data $x = (x_1, x_2, \ldots, x_n)$ from the navigation range sensors is converted from concrete precise values to fuzzy values. Such a transformation is carried out by comparing the specific value of the input value and the value of the membership function $\mu\ \ (x)$ of the corresponding term of the input linguistic variable. The peculiarity of the fuzzification stage is the formation of membership functions and the determination of the number of terms of the linguistic variable.

The formation of membership functions can be carried out by a direct or indirect method. With the direct method, the rules for determining the values of membership functions are formed directly by the expert himself. For example, membership functions can be defined by a table, a graph, or a formula.

The disadvantages of using this method include the subjectivity of the formed meaning. In indirect methods, the values of the membership function are chosen in such a way as to satisfy the predetermined conditions, in the event that the measurement process cannot be carried out.

Expert information is only initial information for further processing.

The group of these methods includes the construction of membership functions on the basis of paired comparisons, with the use of statistical data, with the use of L-R functions on the basis of rank estimates, etc.

The determination of the position of the terms for each linguistic variable was carried out empirically, taking into account the requirements for completeness, consistency and continuity. To control the motion of the MP, the triangular shape of the membership function is chosen, which is given by the following equation:

$$f(x,a,b,c) = \begin{cases} 0, in\ case\ x \le a \\ \frac{x-a}{b-a}, in\ case\ a \le x \le b \\ \frac{c-x}{c-b}, in\ case\ b \le x \le c \\ 0, in\ case\ x \ge c \end{cases}, \quad (2)$$

where the variables *a* and c define the base of the triangle, and *b* defines its vertex. One of the advantages of the triangular shape of the membership function is its ease of implementation.

Graphical representations of the corresponding membership functions of the input linguistic variables and the output in the implementation of different modes of movement are given in the Figure 2**.**

In this figure: a) is the distance to obstacles from the 1st sensor; b) – the distance to obstacles from the 2nd sensor; c) – the distance to obstacles from the 3rd sensor; d) – the difference in angles between the direction of movement of the MP and the direction of movement towards the target; e) is the angle of rotation of the MP.

The values of the input linguistic variables indicating the distance to obstacles to the right and left of the wheeled MP are in the range from 0 to 80 m; the value of the input linguistic variable, which denotes the distance to the obstacle directly in front of the MP, is in the range from 0 to 60 m; the value of the input linguistic variable denoting the difference in angles between the direction of movement of the MP and the direction of movement towards the target is in the range from minus 180 to 180 degrees and the value of the output linguistic variable denoting the angle of rotation of the MP is in the range from minus 90, which means that a turn to the right is made by 90 degrees to plus 90 degrees, which means that a turn to the left is made.

To control the movement of the MP, three terms are chosen for each linguistic variable. For the input linguistic variables that determine the distance to obstacles (*L1, L2, L3*), the set of values (terms) is given in the amount of three: T=*{"Small", "Average", "High"},* denoting respectively small, medium and large distances to the obstacle.

The input linguistic variable that determines the angle difference between the direction of movement of the MP and the direction of movement towards the target (*Qerr*) contains the terms T=*{"Negative", "Zero", "Positive"},* which denote, respectively, finding the target point to the right, directly in front and to the left of the MP.

The output linguistic variable determines the angle of rotation (*Q*) MP and is described by the terms: T=*{"Right", "Forward", "Left"},* which denote a turn to the right, a move straight, and a turn to the left, respectively.

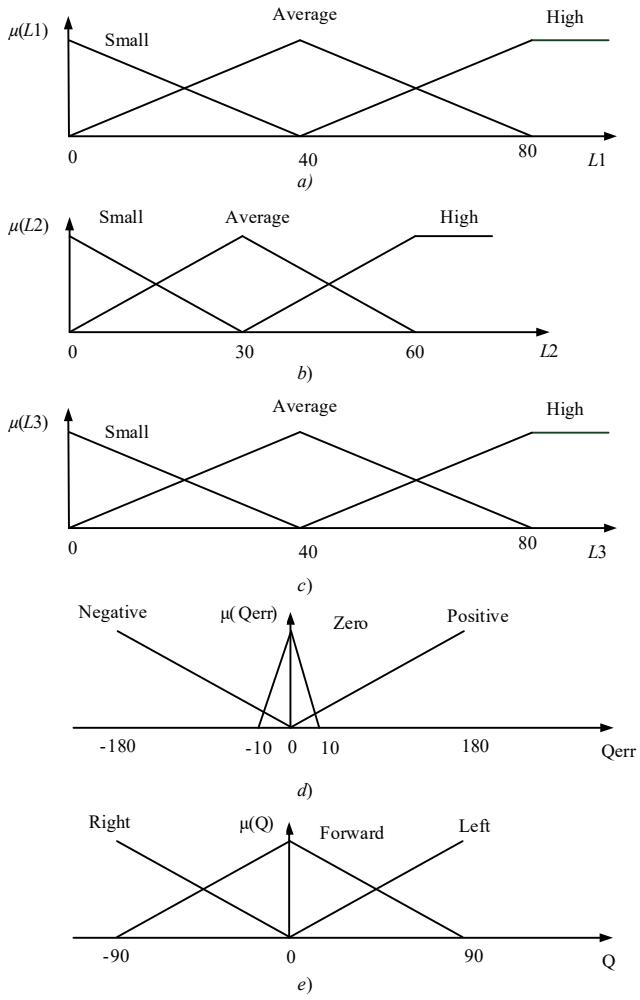

Figure 2: Graphical representation of the membership functions of the input variable and the output variable

The stage of decision-making is based on fuzzy rules. The specifics of the implementation of this stage are to ensure a sufficient number and consistency of fuzzy rules, which should adequately reflect the purpose and cover all possible cases of controlling the movement of the MP. For the decision-making block, the rules are written in the following form:

$$R_k(B): (X_1 is A_{1k} and \dots and X_n is A_{nk}) or \dots or (X_1 is A_{1m} and \dots and X_n is A_{nm}), \quad (3)$$

where X_i are the input variables, R_k the rule, A_{ik}, A_{im}, B_k is the terms of the linguistic variables.

When constructing a base of fuzzy rules, it is first necessary to isolate certain terms in each initial linguistic variable and define their membership functions.

The next step for each term of the original linguistic variable is to define a rule using a conjunction (the "AND" operation) and a disjunction (the "OR" operation). For the original linguistic variable "angle of rotation", the values, i.e. its terms, can be: *T = {"right", "forward", "left"}.* For example, for the term "*right*", a fuzzy rule is written:

Rule 1 ("*right*"): ("*distance to the obstacle of the first sensor*" is "*small*" and "*distance to the obstacle of the second sensor*" is "*small*" and "*distance to the obstacle of the third sensor*" is "*high*") or... or ("*distance to the obstacle of the second sensor*" is "*average*" and...and "*distance to the obstacle of the third sensor*" is "*average*").

At the decision-making stage, the input fuzzy set is converted into the output fuzzy set using a fuzzy rulebase according to the following expression:

$$R_k = M_k \rightarrow N_k, \tag{4}$$

where R_k is the rule, M_k, N_k respectively, are the input and output fuzzy sets.

Defuzzification stage. At the stage of defuzzification, the fuzzy initial value N_k coming from the decision-making unit is converted into a precise output value y_k supplied to the MP engines.

The defuzzification stage is the last step in the process of fuzzy motion control of the MP, which translates fuzzy inferences into concrete values or actions.

There are many methods of defuzzification, the main of which are: center of gravity (Centroid); Largest Maximum; "Height" ratio; and Weighted Average.

The Center of Gravity (Centroid) method is one of the popular methods of defuzzification in fuzzy control. This method is used to convert fuzzy output values into a specific quantitative value. The basic idea is to find the center of gravity or the average value of an indeterminate mass representing a fuzzy output value.

The Largest Maximum method involves finding the highest value among all input values. This method selects the point with the maximum value on the curve of the fuzzy membership function and uses it as the result of defuzzification. This can be the maximum value or the average value after selecting the maximum point.

The "Height" method is used in blurred control systems to convert a fuzzy output into a corresponding specific output signal. Such a signal is calculated as the arithmetic mean of the weighting coefficients multiplied by the maximum value of the corresponding category. The weighting factor is defined as the ratio of the degree of belonging of the input value to a certain category to the sum of the degrees of belonging of all categories.

The Weighted Average method is based on the calculation of the average value of fuzzy conclusions, taking into account their weight.

However, the main disadvantages of those methods are that the type and parameters of the methods are chosen subjectively and may partially not correspond to reality, and the difficulty of adapting the movement of MP to changes in the environment due to the static nature of the chosen methods.

3.3. Structure of the block for the formation of fuzzy conclusions

The Fuzzy Inference Generation Unit (FIGU) is a logical inference system based on an algorithm for obtaining fuzzy inferences based on fuzzy premises using the concepts of fuzzy logic. The process of fuzzy inference combines all the basic concepts of fuzzy set theory: membership functions, linguistic variables, fuzzy logical operations, fuzzy implication techniques, and fuzzy composition.

The logic inference unit in a fuzzy control system plays an important role in the decision-making process based on fuzzy rules and inputs. The formation of fuzzy output is based on rules and is performed by performing the following stages: aggregation, activation, and accumulation. The structure of the FIGU is shown in Figure 3.

Let's take a closer look at the stages of forming fuzzy conclusions.

The aggregation stage consists of determining the degree of truth of the conditions of each of the rules of the fuzzy inference system. Depending on how the rules are defined in the rulebase, at this stage is the degree of truth of the conditions of the rulebase.

At the same time, depending on the form of representation of the rules, logical conjunctions or disjunctions are calculated, which are binary logical operations with a result in the form of a fuzzy statement, the truth of which is determined by formulas.

At the activation stage, the degree of truth of the conclusions for each rule from the rule base is found. For each rule, there is a specific weighting factor. If it is not specified explicitly, then its value is assumed to be equal to 1.

The degree of truth of the conclusions of each rule is determined by the algebraic product of the weighting factor by the degree of truth of the conditions of the rule base found at the stage of aggregation.

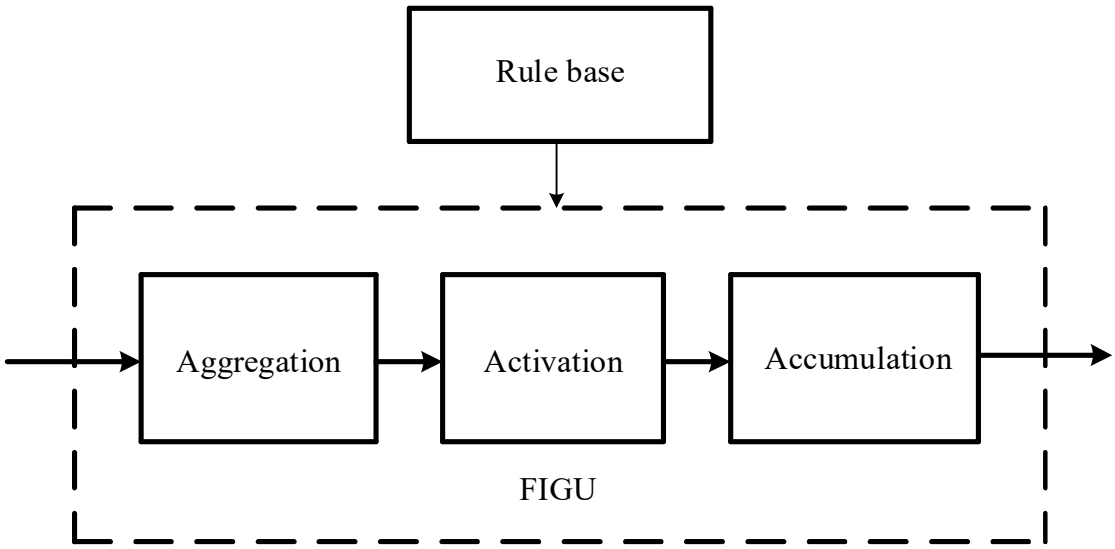

Figure 3: Structure of the Fuzzy Inference Generation Unit

The accumulation stage is reduced to finding the membership function for each of the initial linguistic variables of the set. The purpose of accumulation is to combine or accumulate all degrees of truth of inferences to obtain a membership function of each of the original variables.

The reason for the need to perform this step is that inferences that refer to the same original linguistic variable belong to different rules of the fuzzy inference system. The stage of accumulation is considered to be completed when for each of the initial linguistic variables the final membership functions of fuzzy sets of their values are determined.

3.4. Development of Methods for Training and Functioning of the Neural Network Defuzzifier

The defuzzification block provides a numerical value for each of the initial linguistic variables. The purpose of defuzzification is to use the results of the accumulation of all the initial linguistic variables to obtain the usual quantitative value of each of the output variables that will be used to control the movement of the MP.

Diagrams of the interaction of the FIGU with the Neural-Like Defuzzification Unit (NLDU) are shown in Figure 4, where *x1*,...*xn* – input control variables; *Fy1*,...*Fym* – the initial parameters of the block for generating fuzzy conclusions, the number of which corresponds to the number of terms of the initial linguistic variable; *y* is the resulting value of the control signal.

The peculiarity of defuzzification with the use of NLN is the replacement of the stages of accumulation and defuzzification at the calculation of the control signal *y*, which will reduce the computational complexity and increase the accuracy of obtaining the control signal for the MP.

To implement the defuzzifier, we choose an NLN based on a model of sequential geometric transformations [6].

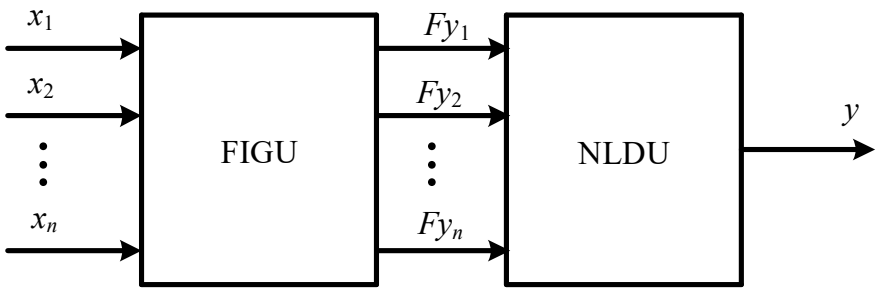

Figure 4: Scheme of interaction between FIGU and NLDU

The method of training a NLN with a teacher consists of the following steps:

Step 1. Formation of a matrix of input, output data *X* [N,n*+1*], which are used for training NLN. This matrix is formed from *N* training vectors of dimension *n+1.* View of the matrix of training vectors:

$$\begin{matrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} & \rightarrow & y_1 \\ . & . & . & . & . & . & . \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} & \rightarrow & y_i \\ . & . & . & . & . & . & . \\ x_{N1} & \dots & x_{Nj} & \dots & x_{Nn} & \rightarrow & y_N \end{matrix}, \quad (5)$$

Each training vector $x_{j1}, \dots, x_{jn} y_j$ is formed by inputs and outputs y_j corresponding to them.

Step 2. Normalization of training vectors. Since the input data for each of the training vectors is in the range [0,1], they do not need to be normalized. Let's normalize only the column of the original values $y_1, \dots, y_N$ according to the expression:

$$y_i^{(n)} = \frac{y_i - y_{min}}{ymin_{max}, i=1,\dots,N}, \quad (6)$$

where is $y_i^{(n)}$the *i-th* normalized value of the output column, y_{min}, y_{max} the minimum and maximum values of the output column $y_1, \dots, y_N$.

Step 3. Find the average value for each of the columns in the table.

$$\bar{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_{ji}, \ i = 1, \dots, n, \quad (7)$$

$$\overline{y} = \frac{1}{N}\sum_{j=1}^{N} y_j \quad (8)$$

Step 4. Input centering. From each element of the matrix $X(x_{ji}, y_j)$ we subtract the average value of $\bar{x}_i, \bar{y}$.

$$x_{ji}^c = x_{ji} - \bar{x}_i, \ j = 1, \dots, N; \ i = 1,\dots,n, \quad (9)$$

$$y_j^c = y_j - \bar{y}, \ j = 1, \dots, N$$

Step 5.1. Perform the first step of the data table transformations (*m=0).* Calculation of the norm of each of the rows of the matrix $X^C$and search among them for the row with the maximum value of the norm: $S\,max = (\sqrt{x_{L1}^2 + \dots + x_{Lj}^2 + \dots + x_{Ln}^2}$.

Let us denote this string by the index *L*. Then the elements of this row of the matrix $X^{c(m)}$ at the m-step of its transformations can be written as $x_{Li}^{c(m)}, \ i = 1, \dots, n$.

Step 5.2. Calculation of the vector of coefficients , $K_i^{(m)}$*i*=1, ..., *N*.

$$K_i^{(m)} = \frac{\sum_{j=1}^{n} x_{ij}^{C(m)} x_{Lj}^{C(m)}}{\sum_{j=1}^{n} \left(x_{Lj}^{C(m)}\right)^2},\ i = 1,\ \ldots,\ N, \tag{10}$$

Step 5.3. Calculation of the vector of coefficients , $M_i^{(m)}$ *i*=0, ..., *n-1*.

$$M_j^{(m)} = \frac{\sum_{i=1}^{N} x_{ij}^{C(m)} K_i^{(m)}}{\sum_{i=1}^{N} \left(K_i^{(m)}\right)^2}, j = 1, \ldots, n, \tag{11}$$

Step 5.4. Recalculation of coefficients, $K_i^{(m)}$ *i*=1, ..., *N*.

$$K_i^{(m)} = \frac{\sum_{j=1}^{n} x_{ij}^{C(m)} M_j^{(m)}}{\sum_{j=1}^{n} \left(M_j^{(m)}\right)^2}, i = 1, \ldots, N, \tag{12}$$

Step 5.5. Execution *Nrecal*=3..5 times of steps 5.3, 5.4 to recalculate the coefficients $M_j^{(m)}, K_i^{(m)}$.

Step 5.6. At the end of step 5.5 for the current *m-th* step of the matrix transformation $X^{C(m)}$, we get the vector of principal components $K_j^{(m)}$, which we remember. We also memorize in the matrix M the vector of coefficients ($M_i^{(m)}$ m = 1, ..., n-1; i=1, ...,n).

Step 5.7. Calculating coefficients

$$\alpha_j^{(m)} = \frac{\sum_{i=1}^{N} x_{ij}^{C(m)} K_i^{(m)}}{\sum_{i=1}^{N} \left(K_i^{(m)}\right)^2}, j = 1, \ldots, n, \tag{13}$$

$$\beta^{(m)} = \frac{\sum_{i=1}^{N} y_i^{C(m)} K_i^{(m)}}{\sum_{i=1}^{N} \left(K_i^{(m)}\right)^2}. \tag{14}$$

Step 5.8. Execution of the *m-th* step of the transformation of the training vector matrix:

$$x_{ij} = x_{ij} - \alpha_j^{(m)} K_i^{(m)},\ i = 1, \ldots, N; j = 1, \rightleftarrows \ldots, \rightleftarrows n, \tag{15}$$

$$y_i = y_i - \beta^{(m)} K_i^{(m)}, i = 1,\ \ldots, N. \tag{16}$$

The training outcomes in the *m-th* step: the values of the vectors $M_j^{(m)}$, $\alpha_j^{(m)}, j = 1, \ldots, n$, $\beta^{(m)}$, as well as the transformed matrix of training vectors.

Checking the condition *m<=(n-1)*. If the condition is met, then increase *m* by one and proceed to step 5.1. Otherwise, the end of the NLN training.

3.5. Development of a Defuzzification Method on a Neuro-like Network

The method of functioning of the neural network at the stage of defuzzification involves the following steps:

Step 1. A running input vector $x_{u1}, \ldots, x_{uj}, \ldots, x_{un}$ is received at the input of the NLN. For each of these vectors, we center it. To do this, subtract from the vector the vector of mean values $\bar{x}_j$ obtained during training NLN.

Step 2. For each current step *m=1, ..., n-1,* we calculate the coefficients ($K_u^{(m)} K_u^{(1)} K_u^{(2)} K_u^{(3)} K_u^{(4)}$) according to the expression (12), using $M_j^{(m)}$, obtained during the training of NLN. Also, for each step *m=1, ..., n-1,* we perform the transformation according to step 5.8 for the NLN training mode, but only on the input vectors. That is, at each step *m=1, ..., n*-1 we transform the input vectors as in the training mode, but we already use the values $M_j^{(m)} \alpha_j^{(m)}$ of the vectors obtained during the training of NLN.

Step 3. We calculate the desired (predicted) value at the NLN output according to the expression:

$$y_u = \sum_{m=1}^{n-1} \beta_m K_u^{(m)} + \bar{y}, \quad (17)$$

Step 4. We perform denormalization of the obtained value y_u at the output of NLN.

3.6. Development of the Fuzzy Logic Software Controller Structure

The fuzzy logic controller software is an important component of the MP motion control system. This controller uses fuzzy logic algorithms to make decisions about controlling the movement of the MP, taking into account various factors and constraints. When using MP, several complex tasks arise related to the problem of movement control and the organization of collective interaction.

Based on the developed blocks for fuzzification, the formation of fuzzy conclusions with the help of the base of rules, and the block of defuzzification based on NLN, a software controller of fuzzy logic is synthesized. The structure of the synthesized controller with neuro-like defuzzification is shown in Figure 5, where FU is the fuzzification unit, FIGU is the fuzzy inference generation unit, NLDU is the neuro-like defuzzification unit, $x_1, x_2,...,x_n$ – input data from navigation sensors, and y is the output value of the control signal.

A fuzzy logic software controller with neuro-like defuzzification provides the generation of MP motion control signals (speed, angle of rotation) at the output y. Input navigation data $x_1, x_2,...,x_n$ (distance to obstacles, distance to target, speed, direction of movement to the target) is used to generate signals for motion control MP y.

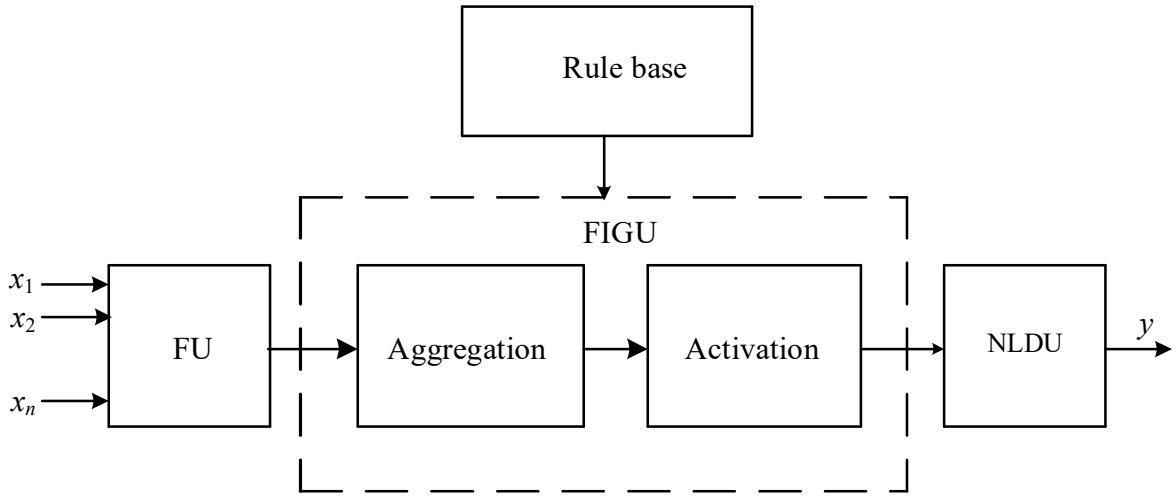

Figure 5: Structure of a Fuzzy Logic Software Controller with Neuro-Like Defuzzification

In the FU block, fuzzification is performed over the input navigation data, which is reduced to the transition from a clear value of the navigation parameter to the value of the membership function of the corresponding term of the input linguistic variable.

The peculiarity of such a transformation lies in the formation of membership functions and the determination of the number of terms of the linguistic variable. To control the movement of the MP, a triangular membership function is chosen, which in parametric form is a set of three points, forming a triangle. Each linguistic variable is defined by three terms.

The information from the output of the FU goes to the input of the FIGU, which combines membership functions, linguistic variables, fuzzy logic operations, methods of fuzzy implication, and fuzzy composition. The formation of fuzzy output is based on rules and is implemented by performing aggregation and activation. To control the movement of the MP, a database of rules

is used, in which the rules of fuzzy logic are presented in the form of a structured text and consist of two parts - conditions and conclusions.

The results of the FIGU operation are sent to the input of the NLDU, at the output of which we obtain the numerical value of the MP motion control signals.

4. Experiment

The implementation of a fuzzy logic controller with neuro-like defuzzification requires the formation of a matrix of training vectors to control the motion of the MP based on fuzzy logic.

Let's consider the control of the movement of the MP based on fuzzy logic. Let's introduce the input and output linguistic variables for MP control:

- Distance (D) – distance from MP to object (input variable);
- Angle (θ) – the angle between the object and the MP (input variable);
- Deviation (δ) – the deviation of MP relative to the input data D and θ (output variable).

The output variable (δ) has 5 terms: T(δ)={LT, AL, A, AR, AT}:

- LT – "on the left";
- AL – "slightly to the left";
- A – "forward";
- AR – "slightly to the right";
- RT – "on the right".

The membership function of the output variable (δ) is given in the form of a triangular shape, which is defined by the parameters *a*, *b*, *c*. Its value at point x is calculated by the expression:

$$MF_{\delta}(x) = \begin{cases} 1 - \frac{b-x}{b-a}; & a \le x \le b \\ 1 - \frac{x-b}{c-b}; & b \le x \le c \\ 0; & x \notin (a, c) \end{cases}, \quad (18)$$

A view of the membership function $MF_{\delta}(x)$ for the source variable (δ) is shown in Figure 6.

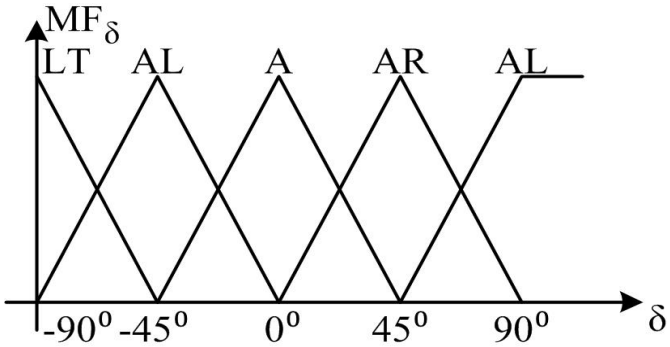

Figure 6: View of the Membership Function of the Source Variable δ

The neural network architecture for δ output is shown in Figure 7.

To form a table of training vectors for the output δ MP, we use the membership function of this output variable (Fig. 6). We will set the values of the output variable *y= δ,* and for each of them find the values MF_1, MF_2, MF_3, MF_4, MF_5. To do this, we use the expression (18).

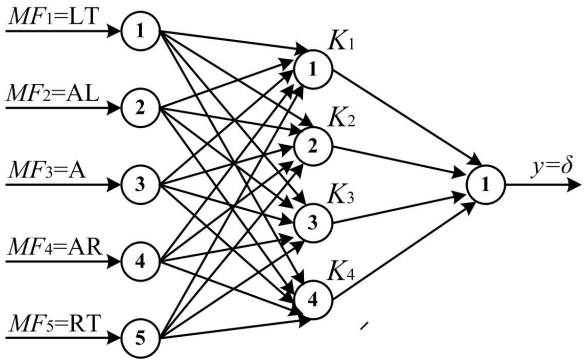

Figure 7: Neural Network Architecture for δ Output

The obtained values of the training vectors are given in Table 1. The number of training vectors for NLN is *N=12,* and the number of NLN inputs is *n=5.* The control of the movement of the MP is carried out using fuzzy logic methods, which are used in cases where there is high uncertainty and complexity in control. The stage of defuzzification in the control system based on FL methods implements an NLN, which allows to obtaining a predicted signal for controlling the movement of the MP. The first stage of the implementation of the defuzzification operation is the training of NLN. For this purpose, a Training_MGT program has been developed that prepares a table of training vectors.

Table 1
Table of training vectors for the output δ control of MP

Number	*MF1*	*MF2*	*MF2*	*MF2*	*MF2*	δ (deg)
1	0.533	0.467	0	0	0	-69
2	0.578	0.422	0	0	0	-71
3	0.378	0.622	0	0	0	-62
4	0	0.333	0.667	0	0	-15
5	0	0.289	0.711	0	0	-13
6	0	0	0.756	0.244	0	11
7	0	0	0.800	0.200	0	9
8	0	0	0.156	0.844	0	38
9	0	0	0	0.889	0.111	50
10	0	0	0	0.933	0.067	48
11	0	0	0	0.044	0.956	88
12	0	0	0	0	1.000	90

Training_MGT_FL program, using a table of training vectors, train an NLN: calculates vectors ($M_j^{(m)}$*j=0, ..., n*; *m=1, ...,n-1*) for a given network architecture and vectors $\alpha_j^{(m)}, \beta_j$. The obtained results are recorded in files and used when the NLN operates in the operating mode.

For the Training_MGT_FL program to work, you need to specify:

- *N* is the number of training vectors;
- *n* is the number of neurons in the input layer of the network;
- a table of training vectors.

In the *Training_MGT_FL program,* the input matrix X*[*N,n*+1]* is used to train NLN, which is read from a file. The training vector matrix for *N=12 n=5* is given below:

```
-- Number of training vectors       N_V  = 12 --
-- Number of inputs                 N_T  = 5 --
--- --- --- --- --- --- --- --- --- --- --- --- ---
 0.533   0.467   0.000   0.000   0.000   -69.000
 0.578   0.422   0.000   0.000   0.000   -71.000
 0.378   0.622   0.000   0.000   0.000   -62.000
```

0.000	0.333	0.667	0.000	0.000	-15.000
0.000	0.289	0.711	0.000	0.000	-13.000
0.000	0.000	0.756	0.244	0.000	11.000
0.000	0.000	0.800	0.200	0.000	9.000
0.000	0.000	0.156	0.844	0.000	38.000
0.000	0.000	0.000	0.889	0.111	50.000
0.000	0.000	0.000	0.933	0.067	48.000
0.000	0.000	0.000	0.044	0.956	88.000
0.000	0.000	0.000	0.000	1.000	90.000

The last column of the matrix represents the output variable (δ is the MP deviation in degrees). From this matrix, we can see that the columns of the input variables are normalized and only the output variable requires normalization. Matrix of training vectors after normalization:

0.533	0.467	0.000	0.000	0.000	0.012
0.578	0.422	0.000	0.000	0.000	0.000
0.378	0.622	0.000	0.000	0.000	0.056
0.000	0.333	0.667	0.000	0.000	0.348
0.000	0.289	0.711	0.000	0.000	0.360
0.000	0.000	0.756	0.244	0.000	0.509
0.000	0.000	0.800	0.200	0.000	0.497
0.000	0.000	0.156	0.844	0.000	0.677
0.000	0.000	0.000	0.889	0.111	0.752
0.000	0.000	0.000	0.933	0.067	0.739
0.000	0.000	0.000	0.044	0.956	0.988
0.000	0.000	0.000	0.000	1.000	1.000

In the next step, the input matrix is centered:

0.409259	0.288888	-0.257407	-0.262963	-0.177778	-0.482402
0.453703	0.244444	-0.257407	-0.262963	-0.177778	-0.494824
0.253703	0.444444	-0.257407	-0.262963	-0.177778	-0.438923
-0.124074	0.155555	0.409259	-0.262963	-0.177778	-0.146998
-0.124074	0.111110	0.453704	-0.262963	-0.177778	-0.134576
-0.124074	-0.177778	0.498148	-0.018519	-0.177778	0.014493
-0.124074	-0.177778	0.542593	-0.062963	-0.177778	0.002070
-0.124074	-0.177778	-0.101852	0.581481	-0.177778	0.182195
-0.124074	-0.177778	-0.257407	0.625925	-0.066667	0.256729
-0.124074	-0.177778	-0.257407	0.670370	-0.111112	0.244306
-0.124074	-0.177778	-0.257407	-0.218519	0.777777	0.492754
-0.124074	-0.177778	-0.257407	-0.262963	0.822222	0.505176

After NLN training, we get the following vectors:

-- Matrix Matr_M[i][j], i=1,..., N_T-1; j=1,..., N_T ---

-0.002847	-0.093250	-0.475787	-0.202573	0.774457
-0.202229	-0.286712	-0.182267	0.648755	0.022452
0.270880	0.209906	-0.375173	0.078160	-0.183773
-0.099577	0.101307	-0.015584	0.009192	0.004662

-- Matrix M_Alpha[i][j], i=1,..., N_T-1; j=1,..., N_T ---

0.005782	-0.087679	-0.491708	-0.194011	0.767617
-0.208524	-0.291590	-0.173548	0.646939	0.026723
0.270880	0.209906	-0.375173	0.078160	-0.183773
-0.099577	0.101307	-0.015584	0.009192	0.004662

-- Vector V_Beta[i], i=1,..., N_T-1 ---

0.396151	0.393827	-0.290977	0.032524

This concludes the NLN training. The resulting vectors are stored in files.

In the mode of functioning of a NLN, an input vector is supplied to its input, and at the output of this network, we receive a predicted control signal of the MP. For the input data that was used to train the NLN (data from the matrix of training vectors), we obtain the values at its output, which fully coincide with the tabular ones.

Let one of the vectors from the matrix of training vectors be fed to the input of the NLN:

0.000 0.000 0.800 0.200 0.000

After centering it, we get:

-- X_IN[i], i=1,..., N_T ---

-0.124074 -0.177778 0.542593 -0.062963 -0.177778

After completing steps 2, ..., 4 of paragraph 1.1.2, we receive a signal at the output of the NLN:

--------- Y_Out= 0.002070 ----------

The last steps are to decenter and denormalize it. Get:

--------- Y_Out= 9.0 ----------

The obtained value at the output of the NLN coincides with the value from the matrix of training vectors.

To assess the technical characteristics of the developed software, they were tested on various hardware platforms. The implementation of the defuzzification operation was tested using the personal computer in the Code::Blocks environment and a microcomputer of the OrangePi Zero type based on SoC H3 on the Linux OS.

Testing was performed for variants with different numbers of NLN defuzzification inputs. The network configurations and datasets used were the same in both cases The test results are presented in Table 2.

To obtain the duration of defuzzification operations, the time command is used, which takes the name of the program module as a parameter and determines the duration of its execution in the environment. The obtained data is converted to time (milliseconds).

Table 2
Duration of training and defuzzification operations based on neuro-like network

Hardware Platform	Training, msec	Defuzzification, msec
Personal Computer	76	17
Microcomputer	85	22

As the test results show, in both cases, the duration of the operations of both NLN training and defuzzification procedures for computer and microcomputer platforms are almost the same.

In addition, for software implementations of neural network defuzzification, the execution time in both cases is about 80 msec for NLN training and about 20 msec for the NLN-based defuzzification procedure. Such values can be considered sufficient for real-time management processes.

5. Results and Discussions

The development of fuzzy logic control, especially in the case of building rules and implementing the defuzzification process, is difficult, therefore it is proposed to implement them on a neural network approach.

Motion control of the mobile platform is carried out using fuzzy logic methods, which are used in cases where there is high uncertainty and complexity in control. The defuzzification stage in the control system based on fuzzy logic methods is implemented with the use of a neural network, which allows for obtaining a predicted motion control signal of the mobile platform.

To develop the algorithm, neural-like networks were used based on the paradigm of the model of successive geometric transformations [6], which provide fast non-iterative training and, as a

result, repeatability of results. In addition, these NLN have a simple, uncomplicated architecture, which ensures their implementation by hardware and software methods.

In the work, the software implementation of the defuzzifier is performed on the basis of the specified implementation of a NLN based on the paradigm of the model of successive geometric transformations.

The performed testing confirms the expediency of the specified implementation of the defuzzifier, and the reaction time (current value processing) is about 20 ms, which should be considered sufficient for real-time control of the mobile platform.

6. Conclusions

It is suggested that in the case of creating a control system for a mobile platform, it should be implemented using fuzzy logic. This approach ensures the operation of the MP in uncertain conditions using inaccurate or incomplete information about the external environment, and its state and value of the parameters of the MP movement are described by fuzzy values.

Fuzzy logic methods are advisable to use in case when it is difficult to describe the control system using traditional mathematical methods, which provides the possibility of obtaining a result even when using ambiguously specified input values.

However, one of the disadvantages of fuzzy control is the difficulty of choosing a defuzzification algorithm. The article considers and tests an approach to the implementation of the defuzzification process using neural networks.

The process of developing a neuro-like network-based fuzzy control system includes the following main stages: problem formulation when it is determined what the neuro-like network control system should solve.

Next, the system requirements are analyzed: data processing speed, number of sensors and actuators, energy efficiency, and others. Also, at this stage, the functions of the system and its capabilities are defined. Hardware design determines the type of microcontrollers, sensors, actuators, means of communication, and other elements. The development of the neuro-like network control algorithm takes into account the qualitative and quantitative parameters of the MP obtained as a result of the analysis of the initial data.

To develop the defuzzifier, a neuro-like network based on the paradigm of the model of successive geometric transformations and fuzzy logic was used. The developed software should provide interaction with the sensors and control devices of the MP, as well as perform processing and analysis of incoming data.

To implement this task, the main stages of fuzzy control of the movement of vehicles are described. The structure of the block for forming fuzzy conclusions and the corresponding methods of training and functioning of the neural network defuzzifier have been developed. The method of defuzzification using a neural network was developed, and the structure of the fuzzy logic controller software was determined.

The technical characteristics of the developed software based on various hardware platforms were evaluated. The implementation of the defuzzification operation was tested using the personal computer in the Code::Blocks environment and a microcomputer of the OrangePi Zero type based on SoC H3 in the Linux OS. In both cases, the duration of neural network training operations is almost the same. The execution time of defuzzification of software implementations for computer and microcomputer platforms is about 20 msec. Such values can be considered sufficient to implement control processes in real-time.

Acknowledgments

This work was performed within the R&D "Methods and means of neurofuzzy control of a group of mobile robotic platforms" carried out by Lviv Polytechnic National University and funded from the state budget of the Ministry of Education and Science of Ukraine for 2023-2024.

References

[1] S. Wang, Mobile Robot Path Planning based on Fuzzy Logic Algorithm in Dynamic Environment, in: International Conference on Artificial Intelligence in Everything (AIE), Lefkosa, Cyprus, 2022, pp. 106-110, doi: 10.1109/AIE57029.2022.00027.

[2] H. S. Selaka, K. A. T. S. Perera, M. A. W. T. Deepal, P. D. R. Sanjeewa, H. P. Chapa Sirithunge and A. G. B. P. Jayasekara, Fuzzy-Bot: A Food Serving Robot as a Teaching and Learning Platform for Fuzzy Logic, in: Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 2018, pp. 565-570, doi: 10.1109/MERCon.2018.8421898.

[3] A. Najmurrokhman, Kusnandar, U. Komarudin, Sunubroto, A. Sadiyoko and T. Y. Iskanto, Mamdani based Fuzzy Logic Controller for A Wheeled Mobile Robot with Obstacle Avoidance Capability, in: International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE), Bali, Indonesia, 2019, pp. 49-53, doi: 10.1109/MoRSE48060.2019.8998720.

[4] F. Cuevas, O. Castillo and P. Cortes-Antonio, Towards an Adaptive Control Strategy Based on Type-2 Fuzzy Logic for Autonomous Mobile Robots, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, 2019, pp. 1-6, doi: 10.1109/FUZZ-IEEE.2019.8858801.

[5] F. Wildani, R. Mardiati, E. Mulyana, A. E. Setiawan, R. R. Nurmalasari and N. Sartika, Fuzzy Logic Control for Semi-Autonomous Navigation Robot Using Integrated Remote Control, in: 8th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 2022, pp. 1-5, doi: 10.1109/ICWT55831.2022.9935458.

[6] I. Tsmots, V. Teslyuk, A. Łukaszewicz, Y. Lukashchuk, I. Kazymyra I., A. Holovatyy, Y. Opotiak, An approach to the implementation of a neural network for cryptographic protection of data transmission at UAV, Drones (Basel). – 2023. – Vol. 7, iss. 8, doi: 10.3390/drones7080507.

[7] Z. Wang, C. He and Z. Miao, Navigation control of mobile robot based on fuzzy neural network, in: 3rd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), Shenyang, China, 2023, pp. 98-102, doi: 10.1109/ACCTCS58815.2023.00070.

[8] X. Gao, Formation control, manipulator control and soft robot based on neural network control system, in: 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT), Changzhou, China, 2021, pp. 37-41, doi: 10.1109/ISRIMT53730.2021.9596861.

[9] T. -H. Nguyen, P. -X. Minh, H. -M. Son, N. -C. Dan and H. -G. Quyet, Robust adaptive control of robots using neural network and sliding mode control, in: International Conference on Control, Automation and Information Sciences (ICCAIS), Nha Trang, Vietnam, 2013, pp. 322-327, doi: 10.1109/ICCAIS.2013.6720576.

[10] D. -H. Thi Kim, T. -N. Manh, N. -P. Van Bach and T. -P. Duc, Trajectory Tracking Control for Omnidirectional Mobile Robots Using Direct Adaptive Neural Network Dynamic Surface Controller, in: First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 2019, pp. 127-130, doi: 10.1109/ICA-SYMP.2019.8646146.

[11] V. Teslyuk, I. Tsmots, N. Kryvinska, T. Teslyuk, Y. Opotyak, M. Seneta, R. Sydorenko, Neuro-controller implementation for the embedded control system for mini-greenhouse, PeerJ Computer Science 9:e1680, doi: 10.7717/peerj-cs.1680

[12] J. Jeon, M. Jeong and H. Myung, Fuzzy Logic and Neural Network-Based Intelligent Control System for Quadruped Robot on Extreme Terrain, in: 23rd International Conference on Control, Automation and Systems (ICCAS), Yeosu, Korea, Republic of, 2023, pp. 1885-1889, doi: 10.23919/ICCAS59377.2023.10317046.

[13] Z. Yu, Research on Intelligent Fuzzy Control Algorithm for Moving Path of Handling Robot, in: Proceedings of the International Conference on Robots & Intelligent System (ICRIS), Haikou, China, 2019, pp. 50-54, doi: 10.1109/ICRIS.2019.00022.

[14] I. Tsmots, V. Teslyuk, Y. Opotyak, V. Rabyk. Intelligent motion control system for the mobile robotic platform, in: CEUR Workshop Proceedings. – 2023. – Vol. 3403 : Computational linguistics and intelligent systems 2023 : proceedings of the 7th International conference on

computational linguistics and intelligent systems. Vol. III: Intelligent systems workshop. Kharkiv, Ukraine, April 20-21, 2023. – P. 555–569, https://ceur-ws.org/Vol-3403/paper42.pdf
[15] T. Mitsuishi, Uncertain Defuzzified Value of Periodic Membership Function, in: International Electrical Engineering Congress (iEECON), Krabi, Thailand, 2018, pp. 1-4, doi: 10.1109/IEECON.2018.8712319.
[16] A. Pourabdollah, Fuzzy Number Value or Defuzzified Value; Which One Does It Better?, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 2020, pp. 1-6, doi: 10.1109/FUZZ48607.2020.9177533.
[17] A. K. Mallick and A. Das, An Analytical Survey of Defuzzification Techniques, in: IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON), Kuala Lumpur, Malaysia, 2021, pp. 1-6, doi: 10.1109/GUCON50781.2021.9573993.
[18] T. Mitsuishi, Uncertain Defuzzified Value of Periodic Membership Function, in: International Electrical Engineering Congress (iEECON), Krabi, Thailand, 2018, pp. 1-4, doi: 10.1109/IEECON.2018.8712319.
[19] M. Naimi, H. Tahayori and A. Sadeghian, A Fast and Accurate Method for Calculating the Center of Gravity of Polygonal Interval Type-2 Fuzzy Sets, in: IEEE Transactions on Fuzzy Systems, vol. 29, no. 6, pp. 1472-1483, June 2021, doi: 10.1109/TFUZZ.2020.2979133.
[20] J. Verstraete, W. Radziszewska, K. Kaczmarek-Majer and S. Bykuć, Combined defuzzification under shared constraint, in: IEEE Transactions on Fuzzy Systems, doi: 10.1109/TFUZZ.2024.3367008.
[21] L. Ghosh, A. Konar and A. K. Nagar, Vertical Slice Based General Type-2 Fuzzy Reasoning and Defuzzification for Control Applications, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Padua, Italy, 2022, pp. 1-8, doi: 10.1109/FUZZ-IEEE55066.2022.9882577.
[22] Q. -S. Mao, Weighted expected value operators on fuzzy numbers, in: International Seminar on Computer Science and Engineering Technology (SCSET), Indianapolis, IN, USA, 2022, pp. 393-397, doi: 10.1109/SCSET55041.2022.00095.