IUST
Iran University of Science and Technology

Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Evolutionary Fuzzy Force Control for Effective Mobile Robot Navigation

Majid Golkhatab *, Aref Shahmansoorian *(C.A) and Mohsen Davoudi *

Abstract: This paper presents a novel hybrid navigation approach for autonomous mobile robots in obstacle-rich environments. The method integrates artificial potential fields for obstacle avoidance with fuzzy logic for path planning, which is optimized by a genetic algorithm to enhance adaptability and robustness to sensor uncertainties. Experimental results demonstrate significant improvements over traditional artificial potential field methods and are validated through real-time implementation on a ROS-based mobile robot.

Keywords: Autonomous Mobile Robot, Fuzzy Logic System Optimization, Hybrid Approach, Path Planning

1 Introduction

DRIVEN by advancements in robotics and intelligent algorithms in recent years, path planning has emerged as a critical area of research within the field of autonomous robot control technology. Path planning can be categorized into two main problem types: local path planning and global path planning. Local path planning addresses scenarios where environmental information is partially unknown or incomplete. Conversely, global path planning assumes complete knowledge of the operating environment. Path planning methodologies can be broadly categorized into traditional and reactive approaches.

Traditional methods: These methods typically focus on finding optimal paths based on pre-defined criteria. Examples include cell decomposition [1-4], roadmap [5-8], and artificial potential field (APF) approaches [9-12].

Reactive methods: These methods are well-suited for environments with frequent changes and can leverage real-time data for decision-making. Examples include genetic algorithms (GA) [13-16], fuzzy logic [9, 17-20], neural networks [21-26], firefly algorithms [27, 28], ant colony optimization [29-32], cuckoo search [33-35], and others [36].

The APF method is a popular approach among traditional methods. Compared to other traditional path planning techniques, APF offers advantages in terms of its user-friendliness, low computational burden, and straightforward implementation. The APF method suffers from a well-documented limitation: convergence to local minima. Local minima traps may lead to situations where the target becomes unreachable. This occurs when the resultant force acting on the agent cancels out (zero resultant force). The complex geometries and relative positions of obstacles within the environment significantly contribute to this issue. Numerous researchers have addressed the challenge of path planning for autonomous mobile agents. Notably, Jia et al.[37] modified the repulsive potential function by discretizing obstacle outlines to enable more nuanced navigation; however, this approach may not fully resolve local minima in highly cluttered or complex settings. Additionally, Li et al. [38] developed an improved APF-based regression search method tailored for fully known environments, limiting its applicability to scenarios with incomplete or changing environmental data. Orozco-Rosas et al. [39] integrated membrane computing with APF, using GA to optimize parameters and generate feasible paths, but this method may incur high computational costs, making it less suitable for real-time applications. Rizqi et al. [40] proposed a potential function-based approach for quadrotors, incorporating

Iranian Journal of Electrical & Electronic Engineering, 2025.
Paper first received 02 Sep. 2024 and accepted 10 Mar. 2025.
* The authors are with the Department of Electrical Engineering is part of the Faculty of Technical and Engineering at Imam Khomeini International University. Qazvin, Iran.
E-mail: ma.golkhatab@edu.ikiu.ac.ir, shahmansoorian@ ikiu.ac.ir, davoudi@ ikiu.ac.ir.
Corresponding Author: A. Shahmansoorian.

wall-following behavior to escape local minima; while effective for quadrotors, this strategy may not suit other robots.

Building on these efforts, this paper introduces a novel hybrid approach that addresses the limitations of traditional APF methods by integrating fuzzy logic control—a reactive technique—into the APF framework. Unlike prior studies, our method uses fuzzy logic to dynamically adjust the robot's heading angle, enhancing its ability to escape local minima traps and navigate complex environments more effectively. Additionally, we employ genetic algorithms (GA) as a machine learning technique to optimize the fuzzy logic system's parameters, improving its adaptability to diverse environments and robustness against imprecise sensor data. This combination of fuzzy logic and GA within the APF framework distinguishes our approach from existing methods, offering a balance of computational efficiency, real-time adaptability, and generalizability across robot types.

This work focuses on path planning for a single robot navigating toward a single target in an environment rich in obstacles. Our contributions are threefold:

- A hybrid path planning method that synergizes APF with fuzzy logic control to overcome local minima and improve navigation performance.
- Optimization of the fuzzy logic system using GA, enhancing its effectiveness and adaptability in varied scenarios.
- Validation through simulations and implementation on a ROS-based omnidirectional differential drive mobile robot, demonstrating superior performance compared to conventional APF algorithms.

The remainder of this paper is organized as follows: Section 2 introduces the basic concept of the APF method. Section 3 defines the Fuzzy-Based Force Control for Improved Performance in APF Path Planning and then presents the GA to optimize the FIS rule base and parameters. Section 4 presents simulation and implementation results, which demonstrate the effectiveness of the proposed algorithm. Finally, Section 5 summarizes this paper.

2 Artificial Potential Field

This section introduces the Artificial Potential Field (APF) method, as shown in Fig. 1. The APF method uses two main components: an attractive field and a repulsive field. The attractive field, originating from the target, pulls the object towards its goal. Conversely, repulsive fields from obstacles push the object away, preventing collisions. The object's overall force is the sum of these attractive and repulsive forces. This combined force guides the object's path. Following sections will explore how these attractive and repulsive fields are specifically created.

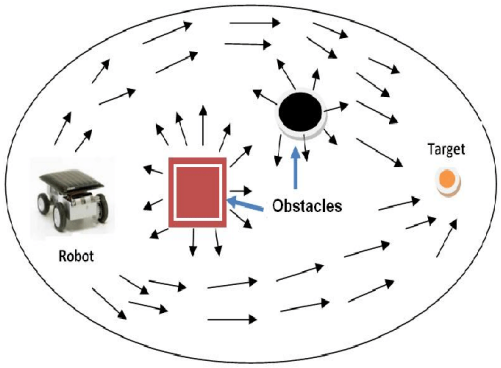

Fig 1. Attractive and repulsive force [41].

2.1 Attractive potential functions

For robot navigation, an attractive field guides the robot to its target. This field's strength increases with the distance between the robot and the target. Common attractive potential functions include conic and quadratic forms [42]. However, the conic function can cause abrupt changes in target positioning during calculations, requiring careful handling of the zero point. Therefore, the quadratic potential function is generally preferred, as detailed below [43] :

$$U_{\text{att}}(q) = \begin{cases} \frac{1}{2}\xi\sigma^2(q, q_{\text{target}}), \sigma(q, q_{\text{target}}) \le \sigma_0 \\ \xi\sigma(q, q_{\text{target}})\sigma_0 - \frac{1}{2}\xi\sigma_0^2, \sigma(q, q_{\text{target}}) > \sigma_0 \end{cases} \quad (1)$$

Where $U_{\text{att}}(q)$ is the numerical value of the attractive field; and ξ is the attractive gain constant, which is a positive parameter. $q_{\text{target}} = (x_{\text{target}}, y_{\text{target}})^T$ is the position of the target, $q = (x, y)^T$ is the current position of the robot, and $\sigma(q, q_{\text{target}}) = \|q - q_{\text{target}}\|_2$ represents the Euclidean distance between the robot and the target. We define σ_0 as distance threshold that determines the attractive force exerted by the target on the robot. This attractive force is directly related to the robot's position and is calculated as the negative gradient of the target's attractive potential function:

$$\vec{F}_{\text{att}}(q) = -\nabla U_{att}(q) = \begin{cases} -\xi\left(q - q_{\text{target}}\right), & \sigma\left(q, q_{\text{target}}\right) \le \sigma_0 \\ -\frac{\xi\sigma_0\left(q - q_{\text{target}}\right)}{\sigma\left(q, q_{\text{target}}\right)}, & \sigma\left(q, q_{\text{target}}\right) > \sigma_0 \end{cases} \quad (2)$$

To design the attractive field, a distance threshold, σ_0, is used to adjust the force on the robot depending on its distance to the target. If the robot is closer than σ_0, a stronger attractive force is applied, inversely proportional to the distance. If the distance, d, is greater than σ_0, the attractive force decreases with distance, following an inverse square law. This threshold helps prevent collisions caused by overly strong attractive forces at long distances. The following section introduces the repulsive potential function, a key component of the APF method that ensures obstacle avoidance while maintaining path stability.

2.2 Repulsive potential function

A repulsive force field helps guide a robot and prevent collisions. This field pushes the robot away from obstacles with a force that increases as the robot gets closer. It's important that this repulsive force only acts within a safe distance. Beyond a certain threshold, the force should be negligible, allowing the robot to move freely when it's sufficiently far from obstacles. Different control algorithms use different functions to calculate the repulsive force. An example function is given in [43].

$$U_{\text{repi}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_i(q, q_{\text{obst}})} - \frac{1}{\rho_0}\right)^2, & \rho_i(q, q_{obst}) \le \rho_0 \\ 0, & \rho_i(q, q_{obst}) > \rho_0 \end{cases} \quad (3)$$

Given an environment with n obstacles, the total repulsive potential field can be expressed mathematically as:

$$U_{rep}(q) = \sum_{i=1}^{n} U_{\text{repi}}(q) \quad (4)$$

Let represent the repulsive potential field generated by obstacle i. The parameter η denotes the repulsion gain constant, which is a strictly positive value. $q_{obst} = (x_{obst}, y_{obst})^T$ represents the position on the obstacle's surface closest to the robot. The current position of the robot is denoted by $q = (x, y)^T$. The distance between the robot and the closest point on obstacle i is given by $\rho_i(q, q_{obst}) = \|q - q_{obst}\|_2$. The obstacle's radius of influence is denoted by ρ_0. Therefore, the repulsive force exerted by obstacle i on the robot can be expressed as the negative gradient of the repulsive potential function:

$$\overrightarrow{F_{\text{repi}}(q)} = -\nabla U_{\text{repi}}(q) = \begin{cases} \eta\left(\frac{1}{\rho_i(q, q_{\text{obst}})} - \frac{1}{\rho_0}\right)\frac{1}{\rho_i^2(q, q_{\text{obst}})}\frac{q - q_{\text{obst}}}{\rho_i(q, q_{\text{obst}})}, & \rho_i(q, q_{\text{obst}}) \le \rho_0 \\ 0, & \rho_i(q, q_{\text{obst}}) > \rho_0 \end{cases} \quad (5)$$

The repulsive field defines a distance threshold ρ_0 between the robot and obstacles. As the robot approaches an obstacle, the repulsive force increases sharply according to the gradient of this threshold. When the robot's distance from the obstacle drops below ρ_0, the repulsive force strengthens significantly. Conversely, once the robot moves beyond ρ_0, the repulsive force diminishes, becoming negligible. The next section introduces the total potential function.

2.3 Total potential function

The total potential function, $U(q)$, was determined by summing the individually calculated attractive potential function, $U_{att}(q)$, and repulsive potential function, $U_{repi}(q)$.

$$U(q) = U_{att}(q) + \sum_{i=1}^{n} U_{repi}(q) \quad (6)$$

In consequence, the resultant force can be mathematically represented as:

$$\vec{F}(q) = \vec{F}_{att}(q) + \sum_{i=1}^{n} \vec{F}_{repi}(q) \quad (7)$$

In the next section, we discuss the limitations of the traditional APF method, highlighting its key shortcomings.

2.4 Identifying Limitations in the Traditional APF Method

The APF method is computationally efficient, responsive, and easy to implement, but it faces challenges in specific path-planning scenarios where targets become unreachable. A target is deemed unreachable when a robot cannot approach it and instead drifts farther away. This issue frequently occurs in environments with obstacles near the target. As the robot moves toward the goal, nearby obstacles amplify repulsive forces, destabilizing path planning.

This phenomenon arises from a diminishing $\vec{F}_{att}$ towards the target and a concurrently escalating $\vec{F}_{repi}$ due to obstacles. When $\vec{F}_{repi} > \vec{F}_{att}$, the resultant force compels the robot away from the target, rendering it inaccessible.

Local optimization and local stability occur when the combined attractive and repulsive forces acting on the robot equate to zero or are collinearly. Under these conditions, the robot becomes stationary or oscillates within a localized area, unable to escape this force equilibrium.

Two case studies are presented based on the configurations illustrated in Fig. 2 and Fig. 3.

Case 1: When the robot, target, and obstacle are aligned (Fig. 2), the robot is subjected to an attractive force, propelling it towards the target. As the robot approaches the obstacle, the attractive force diminishes while the repulsive force increases. If these forces equilibrate, the robot becomes stationary. Conversely, if the attractive force dominates, the robot advances until the repulsive force again counterbalances it, resulting in oscillatory motion near the obstacle.

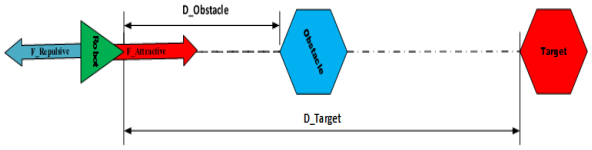

Fig 2. Forces acting on the robot in case 1

Case 2: In environments with multiple obstacles and non-aligned paths (Figure 3), the robot may move unpredictably. Moreover, when repulsive forces from obstacles balance the attractive force toward the target, the robot becomes stuck, unable to progress.

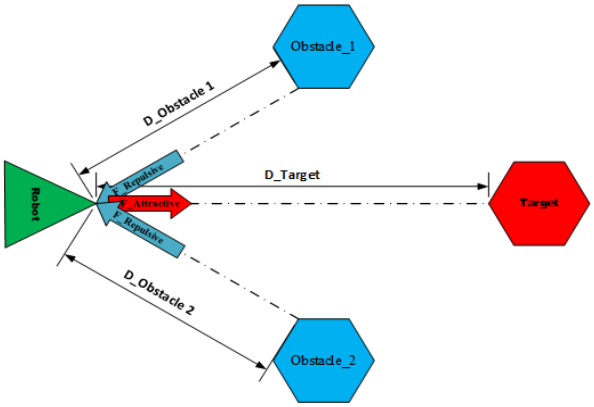

Fig 3. Forces acting on the robot in case 2

3 Fuzzy-Based Force Control for Improved Performance in APF Path Planning

Building upon the established objective of navigation, which is to reach a designated target while avoiding obstacles. We define two key vectors, as is typical in APF algorithms. In Fig. 4 the attractive force vector $\vec{F}_{att}$ is defined as a unit vector pointing directly from the robot's current position toward the target. Conversely, the repulsive force vector $\vec{F}_{repi}$ is a unit vector originating from the closest obstacle in the environment and pointing toward the robot.

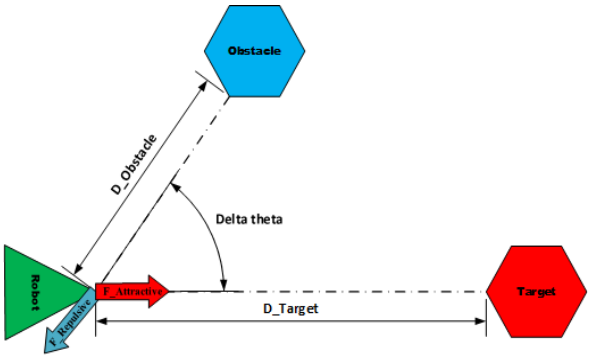

Fig 4. Force analysis

We propose Eq. (8) as an alternative to Eq. (1) and Eq. (2) for achieving a more smoothly and safely navigation towards the target. In Eq. (8), $\vec{F}_w$ represents the resultant force, θ denotes the direction of the resultant force vector, and w represents the weight of the force vector.

$$\vec{F}_w = w\vec{F}_{repi} + (1-w)\vec{F}_{att}\ , 0 < w < 1 \qquad (8)$$
$$\theta = \measuredangle F$$

Certain mobile robot locomotion models may be constrained to avoid sharp turns. To achieve this limitation, the maximum angular displacement per control cycle is restricted to $-\frac{\pi}{4} < \theta_{max} < \frac{\pi}{4}$ Consequently, the robot's heading direction at the next time step, denoted as $\theta_{r(k+1)}$, is calculated based on the current heading direction, $\theta_{r(k)}$, using the Eq. (9):

$$\theta_r(k+1) = \theta_r(k) + \begin{cases} min\left(\theta - \theta_r(k), \frac{\pi}{4}\right) & , \theta - \theta_r(k) > 0 \\ max\left(\theta - \theta_r(k), -\frac{\pi}{4}\right) & , \theta - \theta_r(k) < 0 \end{cases} \qquad (9)$$

$\vec{F}_w$ is the resultant force and the magnitude w of the force vector is determined using the Eq. (10):

$$w = F_w(\alpha, |\Delta\theta|) \qquad (10)$$

where:

$|\Delta\theta|$ (delta theta): absolute difference between the target and obstacle directions with respect to the robot (Fig. 4).

α (alpha): ratio of the robot-to-obstacle distance, $d_{obstacle}$ and the robot-to-target distance, d_{target}.

To successfully navigate, the function should prioritize obstacle avoidance by generating high output values when the following conditions are met:

Angular Difference is Minimal: The absolute value of the difference between the target and obstacle directions relative to the robot $|\Delta\theta|$ is low. This indicates that the target and obstacle lie in similar directions from the robot's perspective.

Obstacle Proximity: The distance between the robot and the obstacle $d_{obstacle}$ is lower than the distance between the robot and the target d_{target}. This signifies that the obstacle poses a more immediate threat.

Under these circumstances, high output values from the function will guide the robot's movement to prioritize obstacle avoidance. Conversely, the function should generate low output values when the aforementioned conditions are not met. This signifies that the robot should focus on reaching the target.

This study aims to develop a Fuzzy Inference System (FIS) capable of autonomously acquiring fuzzy rules and optimizing its internal parameters. This optimized FIS will then be employed to model the function responsible for collision-free robot navigation within a specified environment.

To model the function F_w in Eq. (10), a FIS is constructed as depicted in Fig. 5. In this instance, a Mamdani FIS is employed.

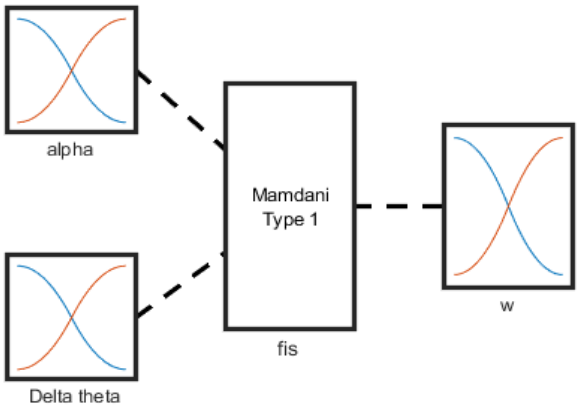

Fig 5. Mamdani Fuzzy Inference System (FIS) Architecture for determining the weighting factor $\vec{F}_w$. The FIS takes two inputs, Angular Difference ($\Delta\theta$) and Relative Distance Ratio (α), and outputs the weight w.

The employed fuzzy system utilizes two input variables. The first input variable, denoted by Δθ, represents the absolute difference between the target direction and the direction of the nearest obstacle relative to the robot's orientation. The second input variable α, is ratio of the robot-to-obstacle distance and the robot-to-target distance.

The range of the first input variable is set to $[0, \pi/2]$. This implies that the second fuzzy input, potentially representing the difference between target and obstacle directions, influences obstacle avoidance when this difference is less than or equal to $\pi/2$ radians.

Similarly, the range of the second input variable is defined as $[0, 2]$. This signifies that the fuzzy input, likely representing obstacle distance, contributes to obstacle avoidance behavior when the obstacle is located at a distance less than or equal to twice the target distance.

To minimize the number of fuzzy rules, a strategy employing two membership functions (MFs) per input variable is implemented. This approach directly reduces the number of combinations required compared to traditional methods. Furthermore, to accommodate input values beyond the designated ranges, a combination of Z-shaped and S-shaped curve MFs is utilized for providing smooth and continuous transition, which is essential for capturing the gradual changes in input data. These functions ensure appropriate membership value assignment even for outlying data points. The first input variable is assigned MFs, as detailed in Fig. 6.

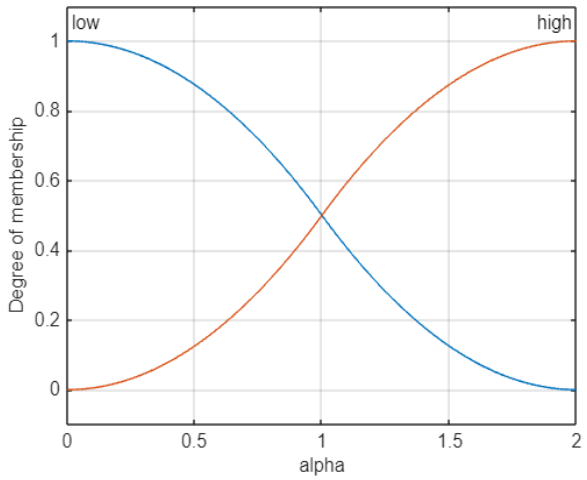

Fig 6. Membership Functions for the Input Variable $\Delta\theta$ defined using Z-shaped and S-shaped curves for smooth and continuous transition, respectively

And the second input variable is assigned membership functions, as detailed in Fig. 7.

To achieve finer granularity in the output values, two MFs can be added to the output layer. While additional MFs can be employed for even greater granularity, this approach necessitates the optimization of a larger set of tuning parameters.

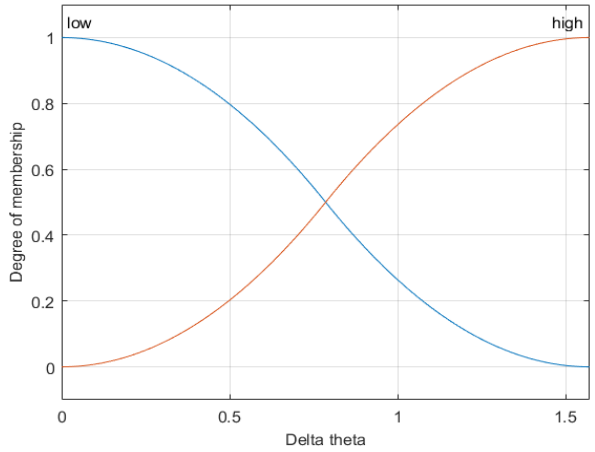

Fig 7. Membership Functions for the Input Variable α (Relative Distance Ratio). Two membership functions are used:" Close Obstacle" and" Far Obstacle," defined using Z-shaped and S-shaped curves for smooth and continuous transition, respectively.

Fig. 8 illustrates the addition of two MFs to the system's output. While including more MFs can refine the granularity of output values, it also leads to an increase in the number of parameters requiring tuning. Notably, the output MFs utilize Z-shaped and S-shaped curve to extend membership values beyond the defined input ranges. The tuning process subsequently optimizes the parameter values associated with these output MFs.

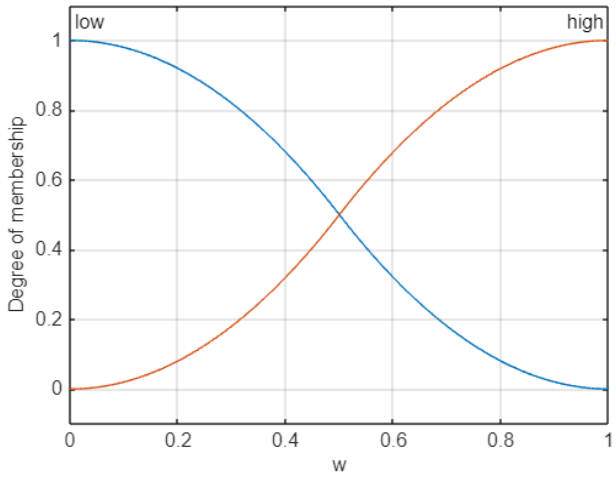

Fig 8. Output Membership Functions for the Weighting Factor w. Two membership functions are used:" Low Weight" and" High Weight," defined using Z-shaped and S-shaped curves for smooth and continuous transition, respectively.

3.1 FIS Rule base Learning and Parameter Optimization

A key challenge in deploying Fuzzy Inference Systems (FIS) is the design and tuning of their rule base and membership function parameters. Typically, this process relies on expert knowledge or extensive trial-and-error. However, for complex systems like robot navigation in dynamic environments, manually crafting optimal fuzzy rules and parameters can be exceedingly difficult and time-consuming. Furthermore, in many real-world applications, labeled training data for FIS design is often scarce or unavailable. To address these challenges and enable automated FIS design and optimization, we employ a Genetic Algorithm (GA)-based learning approach. In the absence of explicit input-output training data, our GA leverages a custom-designed reward function to evaluate and guide the evolution of candidate FIS configurations. This reward function acts as a surrogate for expert knowledge, encoding the desired robot navigation behavior and allowing the GA to autonomously discover effective FIS parameters. The general process of FIS parameter setup with reward function definition is illustrated in Fig. 9.

Our GA-based optimization process focuses on tuning the rule base and membership function parameters of both the input and output variables of the FIS.

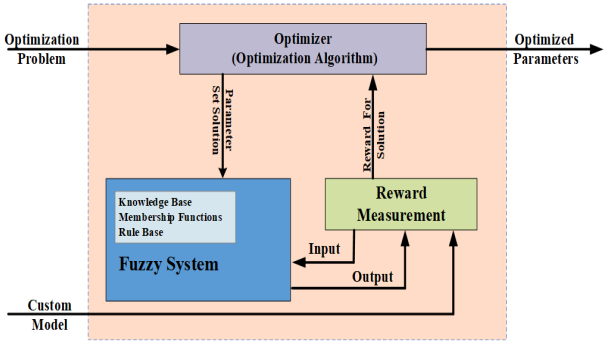

Fig 9. Genetic Algorithm-based Fuzzy Inference System (FIS) Parameter Optimization Process. The GA as optimizer iteratively generates and evaluates candidate FIS parameter sets using a custom-designed reward function based on simulated robot navigation performance. The best-performing FIS configurations are selected and evolved over generations to find an optimized FIS.

Specifically, the GA evolves the parameters that define the shapes and positions of the Z-shaped and S-shaped membership functions shown in Fig. 6, Fig. 7, and Fig. 8. The GA's primary task is to optimize the rule base and numerical parameters of the membership functions so that the fuzzy logic achieves robust robot navigation. The reward function, R, is crucial for guiding the GA's search. It is designed to quantitatively assess the navigation performance of a robot controlled by a given FIS configuration across multiple simulated training environments. These training environments are designed to represent a range of typical navigation scenarios, featuring different obstacle arrangements and target locations. The reward function aggregates the performance across these environments to provide a holistic evaluation of each candidate FIS. For each training environment i, the reward R_i is calculated based

on the robot's navigation outcome. We define a positive reward for successful navigation and a negative reward (penalty) for failures. Successful navigation is defined as reaching the target without colliding with any obstacle or going out of bounds. In this case, the reward is set to the total distance traveled by the robot. We aim to minimize path length while achieving successful navigation, thus rewarding efficiency. Failed navigation, on the other hand, occurs if the robot collides with an obstacle or fails to reach the target within a reasonable time frame or distance. In case of failure, a fixed penalty of −100 is assigned. This penalty is chosen to be significantly negative to discourage failure modes and prioritize successful navigation in the GA's optimization process. Mathematically, for each training environment i, the reward R_i is defined as:

The reward for the i-th environment, R_i, is defined as:

$$R_i = \begin{cases} travelledDistance_i, & if\ reachedTarget_i = true\ and\ notSafe_i = false, \\ -100, & otherwise. \end{cases} \quad (11)$$

The total reward R across all n environments is the sum of individual rewards:

$$R = \sum_{i=1}^{n} R_i. \quad (12)$$

The reward function can be expressed as:

$$R = \sum_{i=1}^{n} \left(travelledDistance_i \cdot \mathbb{I}_{reachedTarget_i \wedge \neg notSafe_i} - 100 \cdot \mathbb{I}_{\neg reachedTarget_i \vee notSafe_i} \right). \quad (13)$$

where $\mathbb{I}_{condition}$ is an indicator function that returns 1 if the condition is true and 0 otherwise, $\wedge$ denotes the logical and operator, and $\vee$ denotes the logical OR operator.

The indicator functions are defined as follows: $\mathbb{I}_{reachedTarget_i \wedge \neg notSafe_i}$ is 1 if the robot successfully reaches the target without collisions, and 0 otherwise; $\mathbb{I}_{\neg reachedTarget_i \vee notSafe_i}$ is 1 if the robot fails to reach the target or encounters a collision, and 0 otherwise. If the robot successfully navigates to the target, the reward is the traveled distance $travelled\ Distance_i$. If the robot fails (due to collision or not reaching the target), the reward is the fixed penalty 100. The total reward R is the sum of rewards across all environments.

The objective of the Genetic Algorithm is to maximize this total reward function R over generations by iteratively refining the membership function parameters of the FIS. Algorithm. 1 outlines the steps of our GA-based FIS tuning process.

Algorithm. 1 Genetic Algorithm for Tuning Fuzzy Inference System (FIS) Parameters for Robot Navigation.

Step 1: Define the Desired Navigation Behavior

- Specify the desired navigation behavior, here: Obstacle Avoidance with Goal Reaching:

Step 2: Design the Custom Reward Function

- Develop a reward function R that quantifies the robot's performance. The reward function can be expressed as Eq. 11:

$$R = \sum_{i=1}^{n} \left(\text{travelledDistance}_i \cdot \mathbb{I}_{\text{reachedTarget}_i \wedge \neg \text{notSafe}_i} - 100 \cdot \mathbb{I}_{\neg \text{reachedTarget}_i \vee \text{notSafe}_i} \right)$$

Step 3: Select and Configure the Optimization Algorithm

- Choose an optimization algorithm suitable for tuning the Fuzzy Inference System (FIS) parameters, such as: *Genetic Algorithm (GA)*.
- Define the parameters to be tuned, including: Membership function parameters (e.g., shapes, centers, widths). Rule weights (if applicable).
- Configure the optimization algorithm parameters (e.g., population size, number of iterations, mutation rate for GA).

Step 4: Evaluate Candidate FIS Configurations in Simulation

- Utilize a robot simulator to evaluate candidate FIS configurations efficiently and safely.

Step 5: Crossover

- Create new FIS configurations (offspring) by applying crossover operations to pairs of selected parent configurations. Crossover combines parameter sets from parents to explore new regions of the parameter space.

For each candidate FIS parameter set:

- Initialize the robot's state (position and orientation) in the simulation.
- Run the simulation under the control of the candidate FIS for a defined duration or until task completion.
- Collect relevant data during the simulation, such as: Trajectory, errors, control signals, and obstacle distances. Calculate the reward value R using the reward function based on the collected data.

Step 6: Iterate and Converge:

- The optimization algorithm iteratively refines

the FIS parameters based on the reward values obtained in Step 4. GA algorithm explores the parameter space and maximize the reward function R.

- Continue the iteration until: A satisfactory FIS performance is achieved (i.e., the reward function R is maximized). The maximum number of iterations is reached.

Step 7: Output

- Output the best-performing FIS configuration found during the GA optimization process, representing the tuned Fuzzy Logic Controller for robot navigation.

The FIS learning algorithm adjusts the rule parameters based on the reward function, resulting in enhanced generalization capabilities. A 1 × 5 fuzzy rule base is created, with each rule containing a textual description summarizing the antecedent and consequent, the IF part of the rule (a combination of fuzzy propositions using linguistic variables and fuzzy operators), the THEN part of the rule (specifying the resulting fuzzy output), and a numerical weight associated with the rule (often set to 1) that influences its overall contribution. Details of the fuzzy rule base are presented in Table. 1.

Table 1. Rules for the FIS

Rule Number	Description	Antecedent	Consequent	Weight
1	Low alpha and high delta theta implies low weight	alpha is LOW AND delta theta is HIGH	w is LOW (weight = 1)	1
2	Low alpha and low delta theta implies high weight	alpha is LOW AND delta theta is LOW	w is HIGH (weight = 1)	1
3	High delta theta implies low weight	delta theta is HIGH	w is LOW (weight = 1)	1
4	High alpha and high delta theta implies low weight	alpha is HIGH AND delta theta is HIGH	w is LOW (weight = 1)	1
5	Low alpha implies low weight	alpha is LOW	w is LOW (weight = 1)	1

3.2 Rules Reduction

To facilitate rule reduction and enhance the FIS's efficiency, we provide a detailed description of the FIS rules in Table 1, emphasizing their connection to the anticipated behaviors of F_w represented in Eq. (10). The FIS employs five initial rules to evaluate the significance of obstacles during robot navigation. We now analyze the relationships between these rules and their impact on the overall behavior to justify the rule reduction.

Rules 1 and 4: Rules 1 and 4 both contribute to assigning a low significance weight to obstacles when they are not directly in the robot's path towards the target, characterized by a high antecedent value for "obstacle location" ($\Delta\theta$ is HIGH). Specifically, Rule 1 considers scenarios where such off-path obstacles are also relatively close (low "obstacle distance" or α is LOW), while Rule 4 addresses cases where they are further away (high "obstacle distance" or α is HIGH). Since Rule 3 also assigns a low weight based solely on "obstacle location" being high (obstacle not in front), potentially encompassing the conditions of Rules 1 and 4, one might initially deem Rules 1 and 4 redundant. However, a more nuanced perspective acknowledges potential edge cases. For instance, in highly cluttered environments, closer off-path obstacles might arguably warrant slightly more consideration than distant ones, and Rules 1 and 4 could theoretically offer this finer granularity.

Despite these potential edge cases, we justify the removal of Rules 1 and 4 based on the following considerations: The primary factor determining obstacle weight should be whether the obstacle is in the robot's direct path $\Delta\theta$ being "HIGH" strongly indicates the obstacle is not an immediate threat to the robot's forward progress, making the influence of obstacle distance α secondary in these off-path scenarios. Furthermore, retaining Rules 1 and 4 introduces unnecessary complexity to the FIS without a demonstrably significant improvement in core navigation performance, especially considering our objectives of robust obstacle avoidance and escaping local minima. The simplified FIS with fewer rules is computationally more efficient and easier to interpret. Empirical validation, as presented in the Experimental Evaluation and Results section, supports the effectiveness of the reduced rule set, indicating that the simplification is justified in practice. Therefore, while acknowledging the subtle nuances Rules 1 and 4 could provide, their removal for simplification and efficiency is well-founded.

Rule 5: Rule 5 is designed to assign a low weight to obstacles that are in close proximity to the robot ("obstacle distance" or α is LOW). However, this rule presents a potential conflict with Rule 2, which assigns a high weight to nearby obstacles that are also positioned directly in the robot's path ("obstacle distance" is LOW AND "obstacle location" or $\Delta\theta$ is LOW). In edge cases where an obstacle is very close but perhaps not perfectly aligned with the robot's intended direction, both Rule 2 and Rule 5 could technically become active. While Rule 5 might subtly reduce the overall weight in such borderline situations, the "high weight" output from Rule 2, designed to trigger a strong avoidance response for immediate threats, would inherently dominate the final

FIS output. Therefore, the contribution of Rule 5 becomes inconsequential in practice. Consequently, Rule 5 can also be removed without significantly altering the overall behavior, as the dominant Rule 2 effectively addresses scenarios where nearby obstacles pose a direct threat. To achieve a more concise representation and improve computational efficiency, we eliminated these redundant rules. The modified FIS rules are detailed in Table 2.

Table 2. Modified rules for the FIS

Rule Number	Description	Antecedent	Consequent
1	Low alpha and low delta theta	alpha is low & delta theta is low	w is high (weight is 1)
2	High delta theta	delta theta is high	w is low (weight is 1)

The modified output surface of the FIS is depicted in Fig. 10. To further assess the practicality of our approach, the next section provides a comprehensive analysis of its computational complexity.

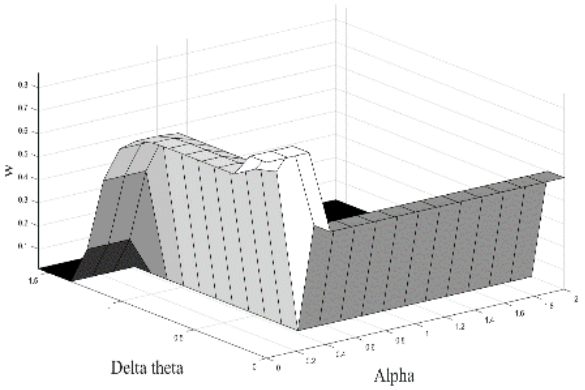

Fig 10. Control surface of modifies FIS

3.3 Computational Complexity Analysis

The practicality of a navigation algorithm for mobile robots is significantly influenced by its computational demands, particularly for real-time applications. To strengthen the argument for the feasibility of our proposed method, this section provides a detailed analysis of its computational complexity, differentiating between offline and online phases and comparing it with relevant alternative approaches.

Offline Phase: Genetic Algorithm Optimization Our framework employs a Genetic Algorithm (GA) to optimize the Fuzzy Logic Controller's (FLC's) membership functions and rule base. It is crucial to acknowledge that GA optimization is inherently computationally intensive. However, this computational burden is strategically confined to an offline design phase. The computational complexity of the GA optimization phase can be approximated as:

$$Complexity\_offline = O(P \,.\, G \,.\, C_f) \qquad (14)$$

where:

P is the population size of the GA, representing the number of candidate solutions evaluated in each generation. G is the number of generations, indicating the iterative process of evolution and refinement of solutions. C_f represents the computational cost of evaluating the fitness or reward function for each candidate solution. This cost primarily involves simulating the controller's performance within a representative environment. This is because the outcome of this phase is a pre-optimized FLC that is subsequently deployed in a real-time environment where online computational efficiency is paramount. The offline optimization allows for a more thorough exploration of the design space, leading to a potentially more robust and effective controller without burdening the robot's real-time processing capabilities.

Online Phase: Real-Time FLC Execution The online operation of the Fuzzy Logic Controller, which is executed in real-time on the mobile robot, is designed for computational efficiency. The online phase comprises a well-defined sequence of steps:

1. **Fuzzification**: This step involves converting crisp sensor inputs (e.g., distance measurements) into fuzzy linguistic variables based on predefined membership functions. The computational cost of fuzzification is typically linear with respect to the number of inputs and membership functions, which are usually small and fixed
2. **Rule Evaluation (Inference)**: The core of the FLC's online computation lies in rule evaluation. Given a finite and fixed set of fuzzy rules (in our case, a reduced set as shown in Table 2), the inference engine determines the degree to which each rule's antecedent is satisfied by the fuzzified inputs. With m representing the number of fuzzy rules, and assuming a fixed number of antecedents per rule, the complexity of rule evaluation is approximately proportional to the number of rules.
3. **Defuzzification**: The final step is defuzzification, where the fuzzy output values (representing control actions) are converted back into precise, crisp control commands that can be directly applied to the robot's actuators. Common defuzzification methods, such as the centroid method, have a computational cost that is also relatively low and constant per control cycle. Therefore, since the number of fuzzy rules ($m = 2$) in our design is deliberately kept small and fixed, the per-cycle computational complexity of the online FLC operation can be approximated as:

$$Complexity\_online = O(m) \qquad (15)$$

This linear complexity with respect to the (small and constant) number of rules signifies a very low per-cycle computational cost. This characteristic is highly

advantageous for real-time execution, especially on resource-constrained mobile robot platforms where processing power and energy are limited.

Comparison with Alternative Methods

To further contextualize the computational efficiency of our proposed method, it is beneficial to compare its online complexity with alternative navigation approaches, such as Artificial Potential Field (APF) variations:

- **Harmonic Artificial Potential Field (HAPF) Controller [45]**:
  When implemented using standard iterative solvers like Gauss-Seidel, HAPF exhibits a computational complexity that is often approximated as $O\ (n*\ I)$, where n is the number of discretized grid cells representing the environment, and I is the number of iterations required for the solver to converge to a stable potential field. While advanced methods like multigrid solvers can improve this complexity to $O(n)$ in many practical scenarios, the online computational cost remains dependent on the environment discretization (n), which can be substantial for high-resolution maps.
- **Pseudo-Bacterial Potential Field (PBPF) Controller [46]:**
  PBPF, as a bio-inspired approach, simulates a swarm of B pseudo bacteria (agents) to explore and navigate the potential field. For each pseudo bacterium, the algorithm typically computes the local potential (or gradient) across the environment. Assuming that each bacterium's evaluation over n potential components (e.g., obstacles, goal) is independent, the cost per iteration becomes $O(B\ .n)$. If the algorithm requires T iterations (or time steps) to converge toward a satisfactory navigation path, the overall worst-case computational complexity can be estimated as $O(T\ .B\ .n)$. This complexity is influenced by the number of simulated bacteria (B), the environment complexity (n), and the convergence time (T), potentially leading to higher online computational demands compared to our FLC-based approach.

Practical Implications and Advantages

The computational complexity analysis highlights several practical implications and advantages of our proposed hybrid navigation method:

- **Real-Time Efficiency**: The online evaluation of a fixed and small set of fuzzy rules in our FLC ensures quick and reliable performance, making it highly suitable for real-time control loops in mobile robotics. This efficiency is crucial for responsive navigation and obstacle avoidance, even on embedded platforms with limited computational capacity.
- **Suitability for Resource-Constrained Environments:**
  By strategically offloading the computationally intensive GA optimization to an offline design phase, our method becomes particularly well-suited for mobile robots operating in resource-constrained environments where processing power and energy efficiency are critical design considerations.
- **Robustness vs. Complexity Trade-off:** While alternative online-adaptive methods might offer flexibility in complex environments, their increased online computational overhead can introduce delays and uncertainties in control execution. Our approach prioritizes robustness in safety-critical tasks by maintaining a consistently low and predictable online computational demand, ensuring timely responses to sensor inputs and environmental changes.

In conclusion, the proposed method strategically decouples the computationally intensive GA optimization (performed offline) from the lightweight real-time fuzzy controller operation. This design choice enhances both the practical feasibility and the real-time response performance of the navigation system, making it a compelling solution for mobile robot navigation, especially when compared to alternative methods that may impose higher online computational requirements.

4 Experimental Evaluation and Results

To rigorously validate the performance of our proposed Evolutionary Fuzzy Force Control (EFFC) method, we conducted extensive experiments in both simulation and real-world environments. Robotics simulation software is an invaluable tool for algorithm development, enabling rapid prototyping, testing, and debugging of robot control strategies in a controlled virtual setting before physical deployment. For our simulation experiments, we utilized the Robot Operating System (ROS) framework in conjunction with the Gazebo robot simulator. The simulated robot platform was the widely used TurtleBot 3 Burger (TB3), chosen for its realistic dynamics and ROS compatibility.

For the Genetic Algorithm (GA) optimization of the Fuzzy Inference System (FIS), we configured a population size of 100 individuals to ensure sufficient exploration of the parameter search space. The GA was run for a maximum of 25 generations, a number empirically determined to allow for convergence while maintaining computational feasibility. To evaluate the fitness of each FIS configuration within the GA, we

employed the custom-designed reward function (described in Section 3.1) and simulated robot navigation in two distinct training environments. Each navigation task within the GA fitness evaluation consisted of 10 simulation iterations, with the robot commanded to move at a constant linear velocity of 0.1 $\frac{m}{s}$. This relatively low linear velocity was chosen to mimic typical indoor robot operation scenarios, such as navigation in office or residential environments, where safety and controlled movements are prioritized. The GA optimization process resulted in tuned membership functions for both input and output variables of the FIS. Fig. 10, Fig. 11, and Fig. 12 visually depict these optimized membership functions for the first input ($\Delta\theta$), the second input (α), and the output (w), respectively. These figures illustrate the refined fuzzy sets learned by the GA to effectively map input conditions to the desired weighting factor for obstacle avoidance.

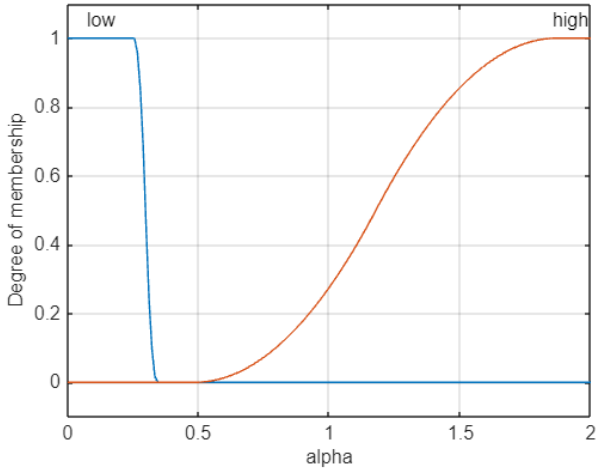

Fig 11. The first input membership functions after ending tuning process

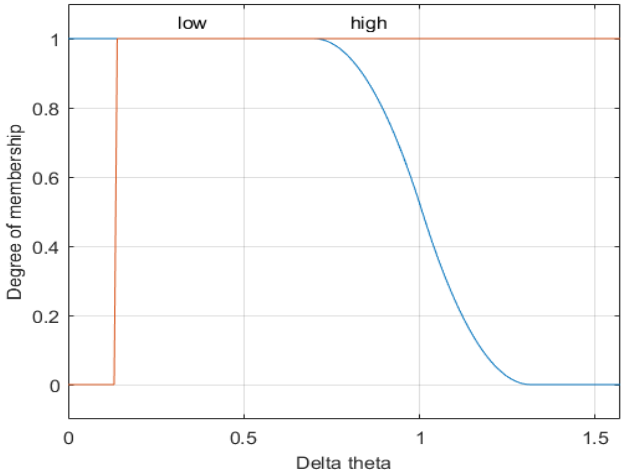

Fig 12. The second input membership functions after ending tuning process.

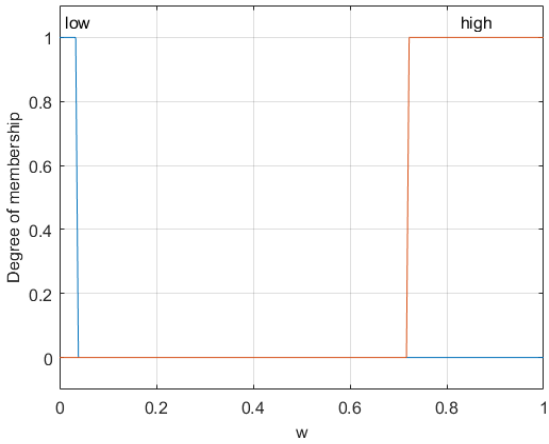

Fig 13. The output membership functions after ending tuning process.

The training environments were configured with specific initial positions for the obstacle, target, and robot. In all training environments, the robot's initial heading was set to π/4 radians and the target was located at (3, 3), as illustrated in Fig. 14. To evaluate the robot's navigation capabilities under varying conditions, two distinct training tasks were employed. These tasks differed solely in the placement of the obstacle, allowing the robot to learn navigation strategies for obstacle-avoidance scenario during its movement towards the target location.

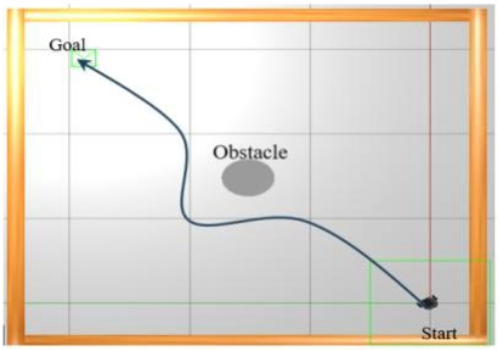

Fig 14. Spatial configuration of the robot, obstacle and target for training process

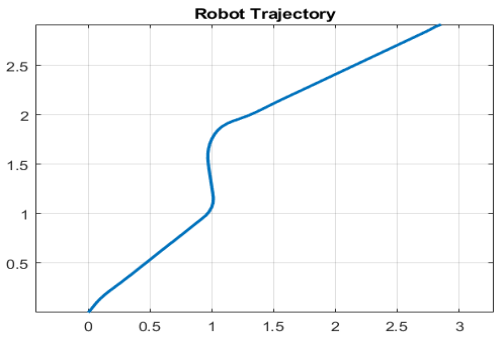

Fig 15. Robot's trajectory in the real-world application scenario from Fig. 17

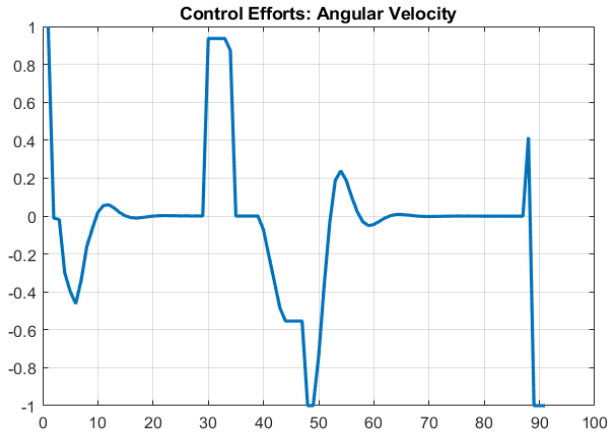

Fig 16. Control effort signal (angular velocity) in the real-world application scenario from Fig. 17

The performance of each FIS in the population is evaluated using a reward function that simulates robot navigation in training environments. Each navigation task is executed for 10 iterations with a constant linear velocity of 0.1 m/s for the TB3.

Following the completion of the training process, Fig. 14 depicts the robot's trajectory, while Fig. 16 illustrates the control effort signal (angular velocity) exerted.

To evaluate the proposed method under realistic conditions, we utilized a physical mobile robot, TB3. The TB3 is a commercially available, differential-drive mobile robot platform widely used in robotics research and education[44]. It is known for its modular design, affordability, and compatibility with the ROS. The TB3 comes in various configurations, with the most common being the Burger variant (featuring two omni-directional wheels for maneuverability in indoor environments). It is equipped with a suit of sensors for perception, including encoders for odometry, an inertial measurement unit (IMU) for orientation, and optionally integrated depth cameras or LiDAR for obstacle detection and mapping. The TB3's open-source software framework and extensive documentation resources make it a popular choice for developing and testing algorithms in robotics research areas such as navigation, Simultaneous Localization and Mapping (SLAM), and robot manipulation. To evaluate the proposed algorithm, we employed a real environment (Fig. 17) representative of typical office or home settings, reflecting real-world application scenarios. The robot's trajectory and corresponding control-effort signal are visualized in Fig. 18 and 19, respectively. A supplementary video illustrating the test procedure is available online at https://aparat.com/v/myvqzu0 .

Fig 17. Real mobile robot (TB3) and environment

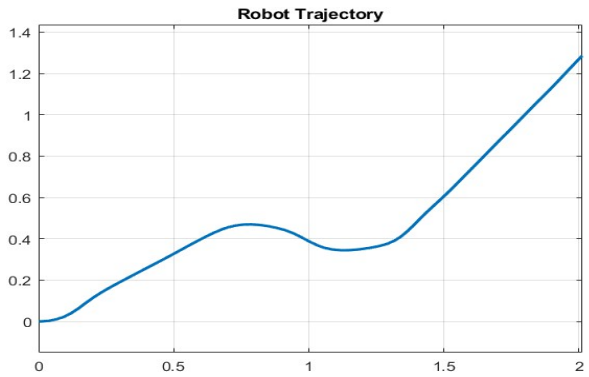

Fig 18. Robot's trajectory

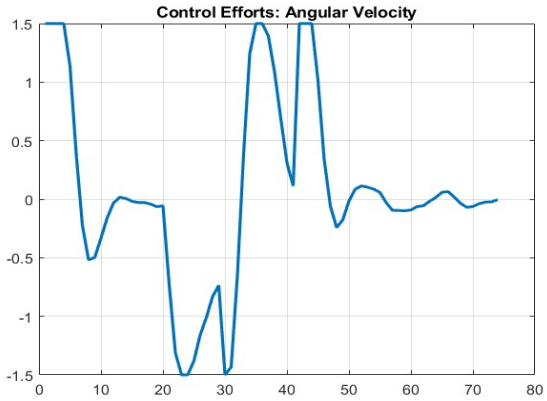

Fig 19. Control effort signal (angular velocity)

A comparative analysis of the proposed Evolutionary Fuzzy Force Control (EFFC) algorithm with the Harmonic Artificial Potential Field (HAPF) [45] and the Pseudo-Bacterial Potential Field (PBPF) [46] is presented in Table 3 for two environments in Fig. 20. The results demonstrate that proposed algorithm exhibits superior real-time performance and robust path planning capabilities. Ten independent runs were conducted for each method in both Environment (a) and Environment (b). The resulting data are visualized in Fig. 21 and 22, respectively.

(a) (b)

Fig 20. Two distinct environments for comparative analysis. (a) depicts a non-convex configuration, while (b) illustrates a more complex scenario

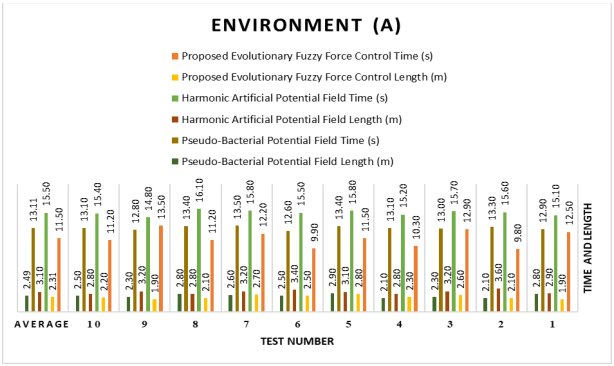

Fig 21. The results of ten independent runs were obtained for each method for environment (a) from Fig. 20

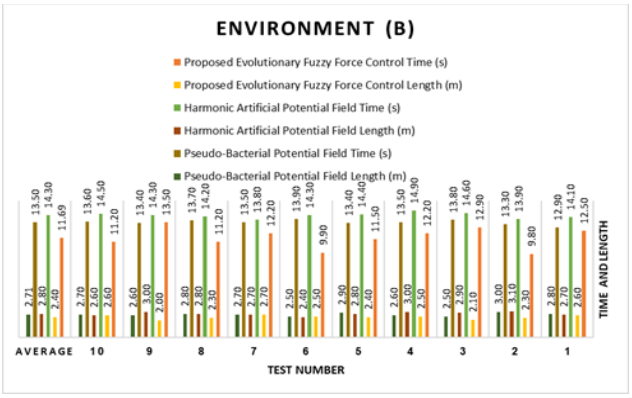

Fig 22. The results of ten independent runs were obtained for each method for environment (b) from Fig. 20

Table 3. The performance of various methods was evaluated over 10 independent runs.

Environments	Statistics	Proposed Evolutionary Fuzzy Force Control		Harmonic Artificial Potential Field		Pseudo-Bacterial Potential Field	
		Time (s)	Length (m)	Time (s)	Length (m)	Time (s)	Length (m)
(a)	Average	11.5	2.3	15.5	3.1	13.1	2.6
(b)	Average	12.2	2.4	14.3	2.8	13.5	2.7

5 Conclusion

In this work, we demonstrated a successful solution for mobile robot navigation in obstacle-filled environments. The proposed method combines the strengths of multiple techniques leverages the strengths of Artificial potential fields for real-time obstacle avoidance, Fuzzy logic control for efficient path planning, allowing for adaptation to imprecise sensor data and Machine learning (genetic algorithm) to optimize the fuzzy logic system for different environments.

Experimental results from a comparative analysis showed significant improvement over traditional potential field methods (HAPF and PBPF). Furthermore, real-time implementation on a mobile robot platform validated the effectiveness of the approach in practical scenarios. This proposed method offers a robust and adaptable navigation strategy for mobile robot platforms capable of implementing APF**,** paving the way for its wider application in various environments.

Future work could explore incorporating additional navigation subtasks (e.g., wall following) and adapting to other robot platforms for complex environments. Additionally, Tuning the FIS parameters also requires extensive implementation efforts. To expedite this process, incorporating human operator decision data as training data can be a promising direction for future research.

Conflict of Interest

The authors declare no conflict of interest.

Informed Consent Statement

Not applicable.

References

[1] Z. E. Kanoon, A. S. Al-Araji, and M. N. Abdullah, "Enhancement of Cell Decomposition Path-Planning Algorithm for Autonomous Mobile Robot Based on an Intelligent Hybrid Optimization Method," *International Journal of Intelligent Engineering & Systems,* vol. 15, no. 3, 2022.

[2] O. A. Salama, M. E. Eltaib, H. A. Mohamed, and O. Salah, "RCD: radial cell decomposition algorithm for mobile robot path planning," *IEEE Access,* vol. 9, pp. 149982-149992, 2021.

[3] G. E. Jan, C. Luo, C.-Y. Yang, and H.-C. Hsieh, "A survey of cell decomposition-based path planning," in *2023 4th International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, 2023: IEEE, pp. 83-87.

[4] S. K. Debnath *et al.*, "Different cell decomposition path planning methods for

unmanned air vehicles-A review," in *Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019: NUSYS'19*, 2021: Springer, pp. 99-111.

[5] J. D. a. Zhengtian Wu a, Baoping Jiang a, Hamid Reza Karimi b, "Robot path planning based on artificial potential field with deterministic annealing," *ISA Transactions,* 2023.

[6] G. Chen, N. Luo, D. Liu, Z. Zhao, and C. Liang, "Path planning for manipulators based on an improved probabilistic roadmap method," *Robotics and Computer-Integrated Manufacturing,* vol. 72, p. 102196, 2021.

[7] S. Kumar and A. Sikander, "A modified probabilistic roadmap algorithm for efficient mobile robot path planning," *Engineering Optimization,* vol. 55, no. 9, pp. 1616-1634, 2023.

[8] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, "HPPRM: hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots," *Ieee Access,* vol. 8, pp. 221743-221766, 2020.

[9] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications,* vol. 227, p. 120254, 2023.

[10] J. Luo, Z.-X. Wang, and K.-L. Pan, "Reliable path planning algorithm based on improved artificial potential field method," *IEEE Access,* vol. 10, pp. 108276-108284, 2022.

[11] R. Szczepanski, A. Bereit, and T. Tarczewski, "Efficient local path planning algorithm using artificial potential field supported by augmented reality," *Energies,* vol. 14, no. 20, p. 6642, 2021.

[12] R. Szczepanski, T. Tarczewski, and K. Erwinski, "Energy efficient local path planning algorithm based on predictive artificial potential field," *IEEE Access,* vol. 10, pp. 39729-39742, 2022.

[13] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of autonomous path planning algorithms for mobile robots," *Drones,* vol. 7, no. 3, p. 211, 2023.

[14] R. Sarkar, D. Barman, and N. Chowdhury, "Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets," *Journal of King Saud University-Computer and Information Sciences,* vol. 34, no. 7, pp. 4269-4283, 2022.

[15] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot," *IEEE Access,* vol. 8, pp. 121267-121284, 2020.

[16] K. Hao, J. Zhao, Z. Li, Y. Liu, and L. Zhao, "Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm," *Ocean Engineering,* vol. 263, p. 112421, 2022.

[17] C. Ntakolia, S. Moustakidis, and A. Siouras, "Autonomous path planning with obstacle avoidance for smart assistive systems," *Expert Systems with Applications,* vol. 213, p. 119049, 2023.

[18] J. Wang, Z. Xu, X. Zheng, and Z. Liu, "A fuzzy logic path planning algorithm based on geometric landmarks and kinetic constraints," *Information Technology and Control,* vol. 51, no. 3, pp. 499-514, 2022.

[19] Y. Lei, Y. Wang, S. Wu, X. Gu, and X. Qin, "A fuzzy logic-based adaptive dynamic window approach for path planning of automated driving mining truck," in *2021 IEEE International Conference on Mechatronics (ICM)*, 2021: IEEE, pp. 1-6.

[20] M. Yao, H. Deng, X. Feng, P. Li, Y. Li, and H. Liu, "Improved dynamic windows approach based on energy consumption management and fuzzy logic control for local path planning of mobile robots," *Computers & Industrial Engineering,* vol. 187, p. 109767, 2024.

[21] I. Sung, B. Choi, and P. Nielsen, "On the training of a neural network for online path planning with offline path planning algorithms," *International Journal of Information Management,* vol. 57, p. 102142, 2021.

[22] M. V. J. Muthugala, S. B. P. Samarakoon, and M. R. Elara, "Toward energy-efficient online Complete Coverage Path Planning of a ship hull maintenance robot based on Glasius Bio-inspired Neural Network," *Expert systems with applications,* vol. 187, p. 115940, 2022.

[23] Z. Zhao, S. Liu, J. Wei, and F. Qin, "Improved biological neural network approach for path planning of differential drive agricultural robots with arbitrary shape," *Computers and Electronics in Agriculture,* vol. 216, p. 108525, 2024.

[24] Y. Liu, Z. Zheng, F. Qin, X. Zhang, and H. Yao, "A residual convolutional neural network based approach for real-time path planning," *Knowledge-Based Systems,* vol. 242, p. 108400, 2022.

[25] M. Chen and D. Zhu, "Optimal time-consuming path planning for autonomous underwater vehicles based on a dynamic neural network model in ocean current environments," *IEEE*

Transactions on Vehicular Technology, vol. 69, no. 12, pp. 14401-14412, 2020.

[26] A. Abdi, M. H. Ranjbar, and J. H. Park, "Computer vision-based path planning for robot arms in three-dimensional workspaces using Q-learning and neural networks," *Sensors,* vol. 22, no. 5, p. 1697, 2022.

[27] F. Li, X. Fan, and Z. Hou, "A firefly algorithm with self-adaptive population size for global path planning of mobile robot," *IEEE Access,* vol. 8, pp. 168951-168964, 2020.

[28] G. Xu, T.-W. Zhang, Q. Lai, J. Pan, B. Fu, and X. Zhao, "A new path planning method of mobile robot based on adaptive dynamic firefly algorithm," *Modern Physics Letters B,* vol. 34, no. 29, p. 2050322, 2020.

[29] M. Morin, I. Abi-Zeid, and C.-G. Quimper, "Ant colony optimization for path planning in search and rescue operations," *European Journal of Operational Research,* vol. 305, no. 1, pp. 53-63, 2023.

[30] M. M. Gangadharan and A. Salgaonkar, "Ant colony optimization and firefly algorithms for robotic motion planning in dynamic environments," *Engineering Reports,* vol. 2, no. 3, p. e12132, 2020.

[31] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Computers & Industrial Engineering,* vol. 156, p. 107230, 2021.

[32] M. Wang, C. Zhu, F. Wang, T. Li, and X. Zhang, "Multi-factor of path planning based on an ant colony optimization algorithm," *Annals of GIS,* vol. 26, no. 2, pp. 101-112, 2020.

[33] K. Sharma, S. Singh, and R. Doriya, "Optimized cuckoo search algorithm using tournament selection function for robot path planning," *International Journal of Advanced Robotic Systems,* vol. 18, no. 3, p. 1729881421996136, 2021.

[34] W. Wang, Q. Tao, Y. Cao, X. Wang, and X. Zhang, "Robot time-optimal trajectory planning based on improved cuckoo search algorithm," *IEEE access,* vol. 8, pp. 86923-86933, 2020.

[35] H. Kundra, W. Khan, M. Malik, K. P. Rane, R. Neware, and V. Jain, "Quantum-inspired firefly algorithm integrated with cuckoo search for optimal path planning," *International Journal of Modern Physics C,* vol. 33, no. 02, p. 2250018, 2022.

[36] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics,* vol. 12, no. 4, p. 93, 2023.

[37] Q. Jia and X. Wang, "An improved potential field method for path planning," in *2010 Chinese control and decision conference*, 2010: IEEE, pp. 2265-2270.

[38] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," in *2012 IEEE International Conference on Mechatronics and Automation*, 2012: IEEE, pp. 1227-1232.

[39] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied soft computing,* vol. 77, pp. 236-251, 2019.

[40] A. A. A. Rizqi, A. I. Cahyadi, and T. B. Adji, "Path planning and formation control via potential function for uav quadrotor," in *2014 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, 2014: IEEE, pp. 165-170.

[41] F. Matoui, B. Boussaid, and M. N. Abdelkrim, "Local minimum solution for the potential field method in multiple robot motion planning task," *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA),* pp. 452-457, 2015.

[42] F. Janabi-Sharifi and D. Vinke, "Integration of the artificial potential field approach with simulated annealing for robot path planning," in *Proceedings of 8th IEEE international symposium on intelligent control*, 1993: IEEE, pp. 536-541.

[43] Y. Shin and E. Kim, "Hybrid path planning using positioning risk and artificial potential fields," *Aerospace Science and Technology,* vol. 112, p. 106640, 2021.

[44] R. Amsters and P. Slaets, "Turtlebot 3 as a robotics education platform," in *Robotics in Education: Current Research and Innovations 10*, 2020: Springer, pp. 170-181.

[45] H. J. S. Feder and J.-J. Slotine, "Real-time path planning using harmonic potentials in dynamic environments," in *Proceedings of International Conference on Robotics and Automation*, 1997, vol. 1: IEEE, pp. 874-881.

[46] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Pseudo-bacterial potential field based path planner for autonomous mobile robot navigation," *International Journal of Advanced Robotic Systems,* vol. 12, no. 7, p. 81, 2015.

Majid Golkhatab was born in 1979 in Tehran, Tehran province, Iran. He received B.S and M.S degrees in Electronics and Mechatronics, respectively. Currently, he is a Ph.D. candidate in Electrical. Engineering Control theory at Imam Khomeini International University (IKIU), Qazvin, Iran. His research interest includes Optimal Control, Machine Learning, Optimization, Robotics, Mechatronics, Fuzzy Logic and Neural Networks.

Aref Shahmansoorian received the B.S and M.S degrees on electrical engineering from Tehran University and received Ph.D. in Control theory from K. N. Toosi University of Technology (KNTU), Tehran, in 2005. Currently, he is with the group of electrical engineering, Imam Khomeini International University (IKIU), Qazvin, as an assistant professor. His research interest includes Nonlinear Control Systems, Optimal Control, Robust Control, Machine Learning and Multivariable Control.

Mohsen Davoudi received his Ph.D. in Electrical Engineering from Polytechnic University of Milan (Politecnico di Milano), Milan, Italy, in 2011. Currently he is an assistant professor at Imam Khomeini International University (IKIU), Qazvin, Iran. His research interest includes Fuzzy Logic, Neural Networks, Computational Intelligence, and Expert systems