International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I9P101

Volume 73 Issue 9, 1-9, September 2025

Original Article

Optimization of Fuzzy Logic Controllers for Autonomous Robot Navigation Using Similarity-Based Rule Simplification

Aggrey Shitsukane[1], Calvins Otieno[2], James Obuhuma[3], Lawrence Mukhongo[4], Gideon Wandabwa[5]

[1,2,3]*Department of Computer Science, Maseno University, Kisumu, Kenya.*
[4]*Department of Electrical Engineering, Technical University of Mombasa, Mombasa, Kenya.*
[5]*Training Department, Kenya School of Government, Mombasa, Kenya.*

[1]*Corresponding Author : kashux1976@gmail.com*

Received: 12 July 2025 Revised: 16 August 2025 Accepted: 02 September 2025 Published: 22 September 2025

Abstract *- Autonomous mobile robots have transformed industries by enabling operations without human intervention. Fuzzy logic controllers (FLCs) are widely used in robots for path planning due to their capability to handle uncertainties. However, large and complex fuzzy rule sets often result in computational inefficiencies, leading to increased navigational time. This study proposes a similarity-based rule reduction method to optimize fuzzy inference systems. The approach leverages a similarity threshold algorithm to identify and merge redundant rules using a similarity index. The hypothesis suggests that simplifying the rule set will either sustain or improve the robot's task completion time. The study is validated through simulations of an FLC-based path planning nonholonomic wheeled mobile robot navigating in a static, unknown environment. The robot detects obstacles, localizes itself, and maps its surroundings in real-time to achieve effective navigation. The primary inputs are obstacle distances, and the output determines wheel velocity. MATLAB and CoppeliaSim are used to optimize and evaluate the robot's performance, with traversal time as the primary metric. The results show that cutting down the number of fuzzy logic rules shortened the robot's traversal time and boosted its processing speed and overall performance. By simplifying the rule base by nearly half (48.15%), the robot was able to complete its navigation tasks more quickly and efficiently. This demonstrates a practical way to tackle the long-standing issue of fuzzy rule complexity, making decision-making systems more adaptive and responsive. These results are particularly significant for real-time applications where speed and dependability are crucial, like robotics, driverless cars, and predictive maintenance.*

Keywords *- Fuzzy logic controller, Autonomous robot, Rule set reduction, Similarity measure.*

1. Introduction

The rapid introduction of autonomous systems in manufacturing, transportation, and healthcare is creating a need for intelligent frameworks that are computationally efficient and flexible [1]. From autonomous vehicles to automation in the factories, as well as smart IoT devices, these systems need to decide in real-time under complex and only partially known situations. Their effectiveness is based on fast interpretation of sensing information and its correct transformation to actions [2]. However, such approaches tend to result in large rule sets for his fuzzy logic controller, for example, which can often decrease performance as the rule base grows larger [3].

Fuzzy logic has been a very effective tool for autonomous decision-making due to its capability in handling uncertainty, imprecision, and nonlinearity [3]. By converting input conditions into output actions, fuzzy systems provide a strong framework for reasoning that binary logic is unable to handle. However, when they are expanded, they usually have redundant or overlapping rules, which hinder processing, increase memory usage, and reduce the system's interpretive power [7]. This is why conventional fuzzy logic controllers with extremely large rule bases typically are unrealistic in real-time robotic control [4]. To address this issue, researchers have researched rule reduction techniques that aim to minimize the rule base, which facilitates computation more efficiently without compromising decision quality.

Numerous studies have been part of the effort. For instance, Chen and Linkens [10] gave a data-driven rule reduction, whereas Wu and Mendel [5] tested similarity measures for type-2 fuzzy sets. Similarly, Batti et al. [7] introduced a neuro-fuzzy hybrid model for robot pathfinding. Although these methods effectively reduce rule redundancy, they share a common drawback: they rely on fixed thresholds

or manual adjustments. As a result, they are limited in dynamic environments, where both sensor inputs and environmental conditions evolve continuously.

Similarity-based rule reduction has emerged as a promising technique to address this issue by merging highly similar fuzzy sets, thereby cutting down the number of rules while retaining decision-making ability [5]. Prior studies have shown that this approach can significantly shrink rule bases, but existing implementations are generally static and lack adaptability [6][7]. In real-world autonomous direction, this inflexibility shortens their applicability as the similarity of rules may vary with the usage context.

The present work addresses the deficiency by optimizing fuzzy logic controllers for autonomous robot navigation, emphasising minimizing rule redundancy and computational wastage. In particular, it suggests reducing the number of rules based on similarity, with the added advantage of an adaptive similarity threshold. As a result, the robot can experience effective and efficient navigation in real time, and the system can adapt to changing environments.

This study makes three main contributions:

1. Creating a rule reduction technique that incorporates adaptive thresholding and similarity metrics.
2. Tests of autonomous navigation in a robot simulation environment are used for experimental validation.
3. Proof that the technique improves the viability of fuzzy logic systems for real-time robotics and dramatically reduces the rule base without sacrificing accuracy or responsiveness.

2. Fuzzy Logic System's Rule Reduction Technique

The approach used to apply similarity-based rule reduction in fuzzy logic systems is explained in this section. It walks through the key steps: defining fuzzy sets, calculating similarity, applying an adaptive threshold optimization, and finally reducing the rule base.

2.1. Initial Rule Base Construction

The first step in the methodology involves constructing the original rule base for the fuzzy logic system. The inference engine is partitioned into four segments [8]. The fuzzifier handles the task of gauging input variables (input signals), conducting scale mapping, and carrying out fuzzification. It entails the conversion of calculated signals (crisp values) into fuzzy values, which are also referred to as "linguistic variables". Membership Functions (MFs) are utilized for this transformation. The membership function, ranging from 0 to 1, represents the extent to which something belongs to a particular fuzzy set. If it is absolutely certain that the quantity belongs to the fuzzy set, its value is 1; conversely, if it is certain that it does not belong to the set, its value is 0, which later undergoes rule reduction. The process includes defining fuzzy input and output variables, designing fuzzy sets, and constructing rules based on expert knowledge or data-driven modelling. Identify the input and output variables for the fuzzy system. Figure 1 shows the Mamdani fuzzy logic architecture adopted [9].

Three inputs were used with linguistic terms "Small (S), Medium (M), and Big (B)." The input variables (Right sensor (dr), Front sensor (df)and Left sensor (dl)) measuring the distance of the obstacle to the robot, two outputs (left and right wheel) with linguistic terms (Low(L), Moderate (M), High(H)) covering the range between 0 and 60 cm/s. using triangular membership functions for different linguistic terms with a maximum of 27 fuzzy rules, i.e., input linguistic terms raised to the power of variables ($Number\ of\ rules = 3^3 = 27$) Rules were formulated for practical and control purposes using the knowledge and skills of an expert car driver. Initial

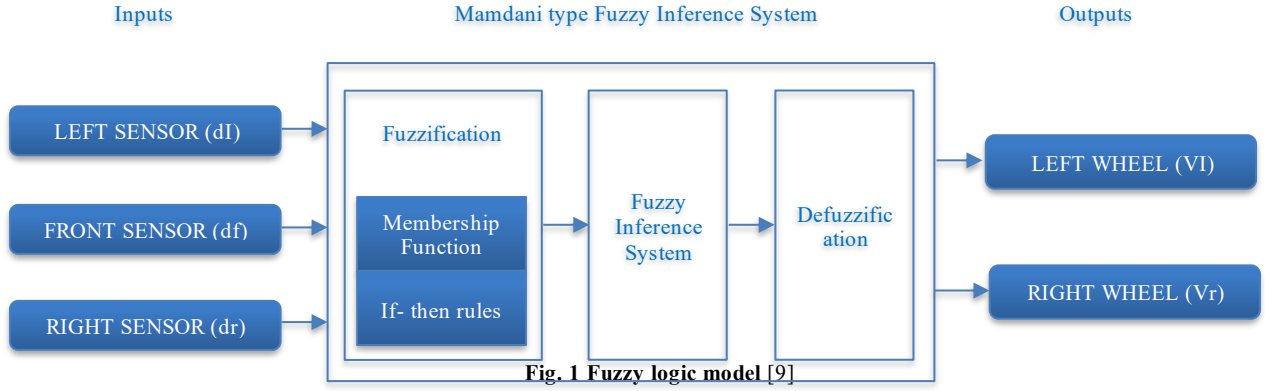

Fig. 1 Fuzzy logic model [9]

Fuzzy Logic Rules for the robot [10][11]

1. If dl is S and df is S and dr is S, then Vl is L and Vr is L.
2. If dl is S and df is S and dr is M, then Vl is L and Vr is M.
3. If dl is S and df is S and dr is B, then Vl is M and Vr is H.
4. If dl is S and df is M and dr is S, then Vl is L and Vr is M.
5. If dl is S and df is M and dr is M, then Vl is M and Vr is M.
6. If dl is S and df is M and dr is B, then Vl is M and Vr is H.
7. If dl is S and df is B and dr is S, then Vl is L and Vr is H.
8. If dl is S and df is B and dr is M, then Vl is M and Vr is H.
9. If dl is S and df is B and dr is B, then Vl is H and Vr is H.

10. If dl is M and df is S and dr is S, then Vl is M and Vr is L.
11. If dl is M and df is S and dr is M, then Vl is L and Vr is M.
12. If dl is M and df is S and dr is B, then Vl is M and Vr is H.
13. If dl is M and df is M and dr is S, then Vl is M and Vr is M.
14. If dl is M and df is M and dr is M, then Vl is M and Vr is M.
15. If dl is M and df is M and dr is B, then Vl is M and Vr is H.
16. If dl is M and df is B and dr is S, then Vl is M and Vr is H.
17. If dl is M and df is B and dr is M, then Vl is H and Vr is H.
18. If dl is M and df is B and dr is B, then Vl is H and Vr is H.
19. If dl is B and df is S and dr is S, then Vl is H and Vr is L.
20. If dl is B and df is S and dr is M, then Vl is M and Vr is M.
21. If dl is B and df is S and dr is B, then Vl is H and Vr is H.
22. If dl is B and df is M and dr is S, then Vl is H and Vr is M.
23. If dl is B and df is M and dr is M, then Vl is H and Vr is M.
24. If dl is B and df is M and dr is B, then Vl is H and Vr is H.
25. If dl is B and df is B and dr is S, then Vl is H and Vr is H.
26. If dl is B and df is B and dr is M, then Vl is H and Vr is H.
27. If dl is B and df is B and dr is B, then Vl is H and Vr is H.

3. Similarity Assessment and Rule Reduction

Rules reduction was implemented using similarity measures to identify and consolidate redundant or highly similar rules [12]. The similarity assessment was carried out between fuzzy sets of the same variable across different rules. A similarity measure was selected to quantify the similarity between the fuzzy sets. This measure was then used to evaluate the closeness of two fuzzy sets based on their membership functions. For each pair of fuzzy sets, a similarity score was calculated. For example, if two fuzzy sets within the variable "dl" had a high similarity score, it indicated that they represented nearly identical concepts and could potentially be merged.

An initial similarity threshold was set at 0.7 to determine if two fuzzy sets were similar enough to be merged. This threshold was later optimized, but an initial value was essential to begin the process. Pairs of fuzzy sets with similarity scores exceeding the threshold were identified and merged. Merging was achieved by creating a new fuzzy set that encompassed the two original sets, thereby reducing redundancy within the rule base. The rule base was then updated to reflect the merged sets. For instance, if the fuzzy sets "Small" and "Medium" for the variable "dl" were merged, all rules referencing "Small" or "Medium" for "dl" were updated to reference the merged set, decreasing the number of unique fuzzy sets. After merging fuzzy sets, the rule base was examined for duplicate or redundant rules (i.e., rules with identical antecedent conditions). These redundant rules were consolidated by averaging or combining the output actions, resulting in a more concise rule base

3.1. Adaptive Threshold Optimization

An adaptive threshold optimization mechanism was introduced, which dynamically adjusted the similarity threshold based on feedback and performance metrics. This adaptive process ensured the fuzzy system continuously optimized itself for efficiency without sacrificing accuracy. Key performance metrics were identified to evaluate the system's effectiveness after rule reduction. These metrics were the processing time and rule count. They were used to assess the impact of rule reduction on the fuzzy system's performance.

The model updated the threshold by learning from previous threshold adjustments and their impact on performance. After each rule reduction iteration, performance metrics were re-evaluated. If accuracy fell below a certain threshold, the similarity threshold was lowered to prevent excessive rule merging. Conversely, if performance improved without a significant accuracy loss, the similarity threshold was raised to allow further reduction. This adaptive feedback loop continued until an optimal balance between rule count and decision accuracy was achieved, as shown in Figure 2, the flow chart for Adaptive Similarity-Based Rule Reduction.

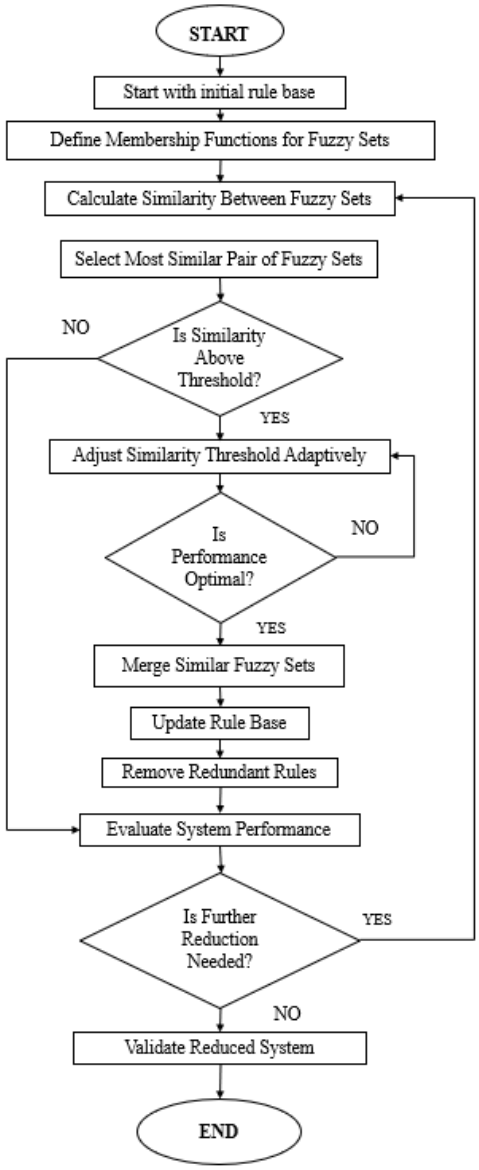

Fig. 2 Flowchart for Adaptive Similarity-Based Rule Reduction

3.2. Algorithm for Similarity-Based Fuzzy Rule Reduction

An Inputs:

1. Initial rule set $R = \{r_1, r_2, \ldots, r_n\}$
2. Threshold (T): A similarity threshold for merging rules
3. Distance measures (dl, df, dr) fuzzy variables for the distance to the left front and right.
4. Velocity outputs (Vl, Vr), fuzzy output variables for left and right velocity.

The output of the algorithm is the reduced rule set $R_{reduced}$.

1. Initialize.

- Set $R_{reduced} = R$, the initial rule set.
- Define a similarity function $S(r_i, r_j)$ that measures the similarity between the two rules. This function compares the input and output conditions of both rules.

2. Compute Similarity

For each pair of rules $(r_i\ r_j)$ in $R_{reduced}$, Calculate the similarity $S(r_i, r_j)$ based on Input similarity, compare the fuzzy condition for dl, df and dr in both rules and output similarity, and compare the fuzzy output values Vl and Vr in both rules.

If $S(r_i, r_j) \geq T\ consider\ r_i\ and\ r_j$ as similar and suitable for merging.

3. Merge Rules

For each pair $(r_i\ r_j)$ with $S(r_i, r_j) \geq T$, Create a merged rule r_{ij} with generalized input conditions, use combined fuzzy sets for dl, df and dr where the rules overlap. Unified output conditions set Vl and Vr as values that represent a general outcome for the merged rule.

Replace $r_i\ and\ r_j\ with\ r_{ij}\ in\ R_{reduced}$

4. Remove Redundant Rules

After merging, check for redundant rules where input and output conditions are identical and remove duplicate rules to avoid redundancy.

5. Check for Universal set Similarity

Remove fuzzy sets that have generalized conditions so broad that they become indistinguishable from a universal (e.g., all input conditions are “medium” or “big with high output velocities)

6. Iterate Until Stable

Repeat steps 2-5 until no further merging is possible (ie $R_{reduced}$ remain stable) or the rule set reaches the desired size.

7. Output the Reduced Rule Set

Return $R_{reduced}$ as the final minimized set of fuzzy rules, this algorithm allows the system to retain functionality while reducing complexity.

Similarity function can be mathematically defined as

$$S(r_i, r_j) = \frac{1}{3} \sum_{k \in \{dl, df, fr\}} \delta\left(I_k(r_i), I_k(r_j)\right) + \frac{1}{2} \sum_{l \in \{Vl, Vr\}} \delta\left(O_l(r_i), O_l(r_j)\right)$$

$S(r_i, r_j)$ gives a numeric value between 0 and 1 that measures the similarity between two fuzzy rules $r_i\ and\ r_j$. If $S(r_i, r_j) \geq T$, where T is a threshold, the two rules are considered similar enough to be merged.

The input similarity contribution is $\frac{1}{3}\sum_{k \in \{dl, df, fr\}} \delta\left(I_k(r_i), I_k(r_j)\right)$ and the output similarity contribution is $\frac{1}{2}\sum_{l \in \{Vl, Vr\}} \delta\left(O_l(r_i), O_l(r_j)\right)$

Where $\delta = 1$, if they are similar or $\delta = 0$, otherwise.

Python implementation of a similarity-based rule reduction algorithm. The code assumes that each rule is represented as a dictionary containing the fuzzy sets for the inputs and the outputs.

```
import itertools
initial_rules = [ ]
def calculate_similarity(rule1, rule2):
matches = sum(rule1[key] == rule2[key] for key in ['dl', 'df', 'dr', 'Vl', 'Vr'])
    return matches / 5.0  # similarity as fraction of total conditions
def merge_rules(rule1, rule2):
  merged_rule = {}
  for key in ['dl', 'df', 'dr', 'Vl', 'Vr']:
    if rule1[key] == rule2[key]:
      merged_rule[key] = rule1[key]
    else:
      merged_rule[key] = f"General({rule1[key]}/{rule2[key]})"
  return merged_rule

def reduce_rules(rules, threshold=0.7):
  reduced_rules = rules[:]
  merged_indices = set()
  for (i, j) in itertools.combinations(range(len(rules)), 2):
    if i in merged_indices or j in merged_indices:
      continue
      similarity = calculate_similarity(reduced_rules[i], reduced_rules[j])
      if similarity >= threshold:
      merged_rule = merge_rules(reduced_rules[i], reduced_rules[j])
```

```
            reduced_rules[i] = merged_rule  # Replace first rule with merged rule
            merged_indices.add(j)  # Mark the second rule as merged
        reduced_rules = [rule for idx, rule in enumerate(reduced_rules) if idx not in merged_indices]
        return reduced_rules
    threshold = 0.7  # Define a threshold for similarity (tune this as needed)
    reduced_rules = reduce_rules(initial_rules, threshold)
    print("Reduced Rule Set:")
    for i, rule in enumerate(reduced_rules):
        print(f"Rule {i+1}: {rule}")
    performance.
```

3.3. Similarity-Based Reduced Rules Results

The reduction of fuzzy logic rules from 27 to 18, then 14, represents a significant step towards optimizing the efficiency and interpretability of the system. By eliminating redundant rules while preserving the system's core functionality, this process streamlines decision-making and reduces computational complexity. The 3 rule sets were validated through a simulated environment.

Reduced Fuzzy Logic Rules (18 Rules):

1. If dl is S and df is S and dr is S, then Vl is L and Vr is L.
2. If dl is S and df is S and dr is M, then Vl is L and Vr is M.
3. If dl is S and df is M and dr is S, then Vl is L and Vr is M.
4. If dl is S and df is M and dr is B, then Vl is M and Vr is H.
5. If dl is S and df is B and dr is S, then Vl is L and Vr is H.
6. If dl is S and df is B and dr is M, then Vl is H and Vr is H.
7. If dl is M and df is S and dr is S, then Vl is M and Vr is L.
8. If dl is M and df is S and dr is M, then Vl is L and Vr is M.
9. If dl is M and df is S and dr is B, then Vl is M and Vr is H.
10. If dl is M and df is M and dr is S, then Vl is M and Vr is M.
11. If dl is M and df is M and dr is M, then Vl is M and Vr is H.
12. If dl is M and df is B and dr is S, then Vl is M and Vr is H.
13. If dl is M and df is B and dr is B, then Vl is H and Vr is H.
14. If dl is B and df is S and dr is S, then Vl is H and Vr is L.
15. If dl is B and df is S and dr is M, then Vl is M and Vr is M.
16. If dl is B and df is M and dr is S, then Vl is H and Vr is M.
17. If dl is B and df is M and dr is B, then Vl is H and Vr is H.
18. If dl is B and df is B and dr is S, then Vl is H and Vr is H.

Further Reduced Fuzzy Logic Rules (14 Rules):

1. If dl is S and df is S and dr is S, then Vl is L and Vr is L.
2. If dl is S and df is S and dr is M, then Vl is L and Vr is M.
3. If dl is S and df is M and dr is S, then Vl is L and Vr is M.
4. If dl is S and df is M and dr is B, then Vl is M and Vr is H.
5. If dl is S and df is B and dr is M, then Vl is H and Vr is H.
6. If dl is M and df is S and dr is S, then Vl is M and Vr is L.
7. If dl is M and df is S and dr is M, then Vl is L and Vr is M.
8. If dl is M and df is M and dr is B, then Vl is M and Vr is M.
9. If dl is M and df is B and dr is M, then Vl is H and Vr is H.
10. If dl is B and df is S and dr is S, then Vl is H and Vr is L.
11. If dl is B and df is S and dr is M, then Vl is M and Vr is M.
12. If dl is B and df is M and dr is M, then Vl is H and Vr is M.
13. If dl is B and df is B and dr is S, then Vl is H and Vr is H.
14. If dl is B and df is B and dr is B, then Vl is H and Vr is H.

The bar chart in Figure 3 visually represents the progressive reduction in the number of fuzzy logic rules used in the controller design. The initial 27-rule configuration represents the full combinatorial coverage of a fuzzy system with 3 inputs and 3 linguistic terms each. To optimize computational performance and reduce complexity without sacrificing control quality, reduced rule bases were explored:

$$Percentage\ reduction = \frac{original - reduced}{original}\ x\ 100$$

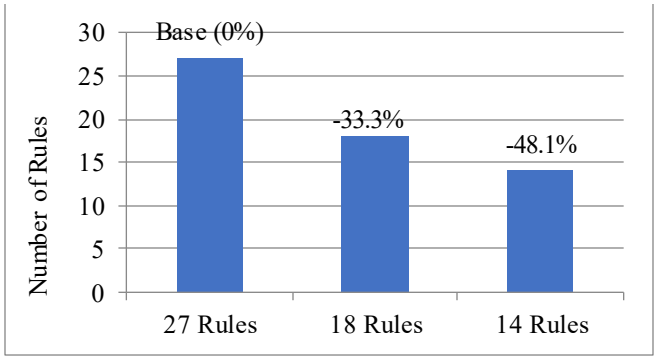

Fig. 3 Reduction in fuzzy logic rules

4. Simulation Environment and Fuzzy Logic Controller Design

Third, this study employed a simulation-based approach to evaluate the performance effects of fuzzy rule base reduction in a nonholonomic wheeled mobile robot navigating a static, unknown environment. The methodology integrates a simplified fuzzy logic control system based on triangular membership functions and minimal directional sensor input, implemented in a controlled virtual environment.

Tests were performed in a 5 m × 5 m maze-like simulated setting built with CoppeliaSim, a physics engine-based robotics simulator. The area of simulation was surrounded by static walls and held 12 cylindrical obstacles, which were deliberately positioned to create thin paths that mimic indoor navigation limitations. The robot was initialized to begin at the bottom-left, and its objective was to navigate to a pre-determined target without collision and with minimal travel time, which was the main performance metric.

The mobile robot was constructed on the Pioneer P3-DX differential-drive robot platform, which was selected for prior use in robotics research and kinematic appropriateness to autonomous navigation tasks. The proximity sensors were set to return distance readings in three directions: left (dl), front (df), and right (dr).

A Mamdani-type Fuzzy Logic Controller (FLC) developed in MATLAB controlled the robot's decision-making. The controller took the three sensor inputs and generated two outputs: left wheel Velocity (Vl) and right wheel Velocity (Vr). Triangular Membership Functions

(MFs) were used because they are easy and efficient to compute. Input variables were defined with three linguistic terms (Near, Medium, Far), and output variables were defined with three levels of velocity (Low, Medium, High). This configuration enabled the controller to convert obstacle proximity to instantaneous velocity updates to correct paths in real-time.

Simulations were executed on a 64-bit Windows 10 Pro system equipped with an Intel Core i7 2.4 GHz processor, 16 GB RAM, and a 512 GB SSD, using an HP EliteOne 1000 G2 workstation. MATLAB handled fuzzy inference and logic execution, while CoppeliaSim managed 3D physics, environmental feedback, and robot motion. Each experimental configuration was tested under standardized simulation conditions, with 10 independent replicates per rule base setting to ensure statistical robustness. The navigation task simulated an autonomous driving scenario where the robot responded to environmental obstacles in real time. Figure 4 illustrates the projected navigation behavior of the robot, highlighting key path transitions and orientation control during motion.

- If $V_l = V_r$ Then the robot moves straight.
- If $V_l < V_r$ Then the robot turns to the left side.
- If $V_l > V_r$ Then the robot turns to the right side.
- If $V_l = -V_r$ Then the robot rotates (spins) clockwise.
- If $-V_l = V_r$ Then the robot rotates (spins) anticlockwise.

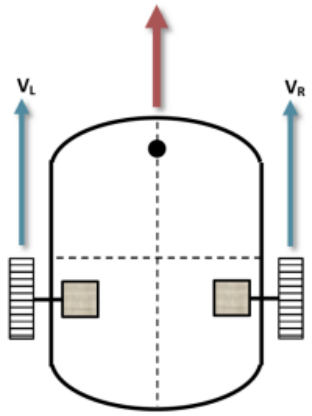

Fig. 4 ($V_l = V_r$) Robot moves in a straight line.

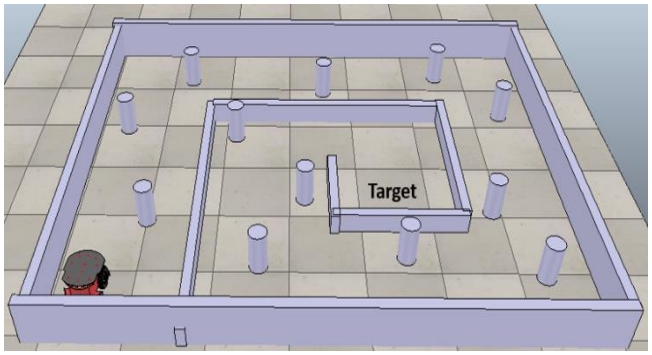

Fig. 5 Simulation field for the robot

As seen in Figure 5, the robot moves from a starting point to a target while gathering information on how long it takes to finish the task. The objective was to evaluate three rule sets for robot navigation and compare how well they performed in terms of how long it took a robot to finish a navigation task. Independent variables were the rule sets being tested: Rule Set A: Original 27 rules, Rule Set B: Reduced 18 rules and Rule Set C: reduced 14 rules. The dependent variable is the performance factor, which measures how long it takes the robot to finish the navigation task in seconds.

5. Results and Discussion

The Experiments were conducted with 10 runs for each rule set to ensure statistical robustness using the same navigation scenario for all runs to maintain consistency.

In the Testing Procedure, the Setup of the navigation scenario had a navigation course with obstacles and a fixed start and goal position. Measurements of time (in seconds) taken for the robot to navigate from start to finish in each trial, as shown in Table 1.

Table 1. Traversal time from 10 trials for each rule set

Trial	Rule set A (27 rules)	Rule set B (18rules)	Rule set C (14 rules)
1	181.5	180.4	178.7
2	180.9	179.3	179.1
3	182.1	180.1	178.9
4	181.0	179.8	178.8
5	181.8	179.5	179.2
6	180.7	179.0	179.0
7	182.3	180.2	178.5
8	181.4	180.0	178.6
9	181.2	179.7	179.4
10	182.0	179.6	178.3

Figure 6 shows a line chart showing how each rule set performed across the 10 trials. It can be clearly seen that Rule set A consistently has the highest values, Rule set B is generally lower than A but higher than C and Rule set C has the lowest values throughout.

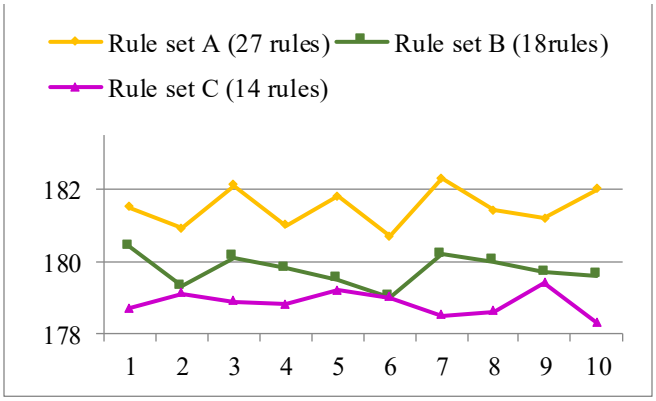

Fig. 6 Line chart showing rule set performance.

Figure 7 shows a box plot comparing the three rule sets. This gives a good view of the distribution, median, and variability of the measurements for each rule set. Rule set A has the highest median and a slightly wider spread, Rule set B has a lower median with a narrower range and Rule set C has the lowest values, and its spread is also fairly compact.

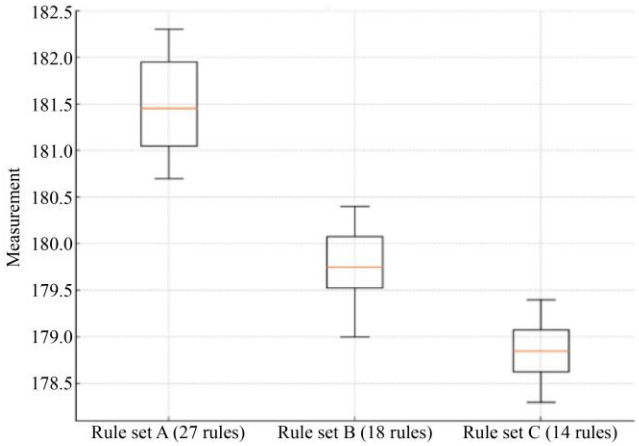

Fig. 7 Box plot comparison of rule sets

The Shapiro-Wilk test was applied to each group (Rule Sets A, B, and C). The null hypothesis of the Shapiro-Wilk test is that the data are drawn from a normally distributed population. A p-value above 0.05 indicates that we do not reject the null hypothesis, implemented via the swtest function available in the Machine Learning Toolbox in MATLAB.

The results of the Shapiro-Wilk test for the three rule sets were as follows: Rule Set A: Shapiro-Wilk Statistic = 0.97, p-value = 0.94, Rule Set B: Shapiro-Wilk Statistic = 0.98, p-value = 0.98 and Rule Set C: Shapiro-Wilk Statistic = 0.99, p-value = 0.99. The null hypothesis of the Shapiro-Wilk test asserts that the data follows a normal distribution. Since the p-values for all three rule sets are greater than the significance level of 0.05. Thus, the data for all three rule sets were considered to be normally distributed.

Levene's test was performed using MATLAB to assess the homogeneity of variance among the groups (Rule Set A, Rule Set B, and Rule Set C). The results of Levene's test were as follows: Statistic: 1.497 and p-value: 0.24. Since the p-value (0.24) exceeds the significance level of 0.05, we fail to reject the null hypothesis.

This indicates that the variances among Rule Set A, Rule Set B, and Rule Set C are not significantly different, thereby satisfying the assumption of homogeneity of variance. The independence of observations was typically ensured by the experimental design. In this case, each trial was conducted independently with the same navigation scenario and consistent conditions. Thus, the independence assumption is satisfied.

5.1. ANOVA Test

A test of variance analysis was conducted. The purpose of this statistical technique is to ascertain whether the means of three or more independent groups differ in any way that is statistically significant.

Null Hypothesis (H_0)

There is no significant difference in the mean traversal times among the three rule sets.

$$(\mu_1 = \mu_2 = \mu_3)$$

Alternative Hypothesis (H_1)

At least one rule set's mean traversal time is significantly different from the others.

Calculated Group Means

- Rule Set A: $\mu_1 \approx 181.49$ seconds
- Rule Set B: $\mu_2 \approx 179.76$ seconds
- Rule Set C: $\mu_3 \approx 178.85$ seconds

The Degrees of Freedom (df) were calculated for each component of the total variation in ANOVA, Number of groups (k) = 3. Total number of observations (N) = 30 (10 per group)

Between-Group Degrees of Freedom:

$$df_{between} = k - 1 = 3 - 1 = 2$$

Within-Group Degrees of Freedom:

$$df_{within} = N - k = 30 - 3 = 27$$

To partition the total variation in the data into Between-Group Variation (SSB), Within-Group Variation (SSW), and Total Variation (SST), the following formulas were used:

$$SSB = \sum_{i=1}^{k} n_i \ (\bar{X}_i - \bar{X})^2 = 32.97$$

Where:

k; Number of groups.

n_i : Number of observations in group i

$\bar{X}_i$: mean of group i

$\bar{X}$: Overall mean of all observations.

Within group variation (SSW)

$$SSW = \sum_{i=1}^{k} \sum_{j=1}^{n_i} n_i \ (\bar{X}_{ij} - \bar{X}_i)^2 = 8.38$$

Where

$\bar{X}_{ij}$: observation j in group i

Total variation (SST)

$$SST = SSB + SSW$$

Calculated values for SSB, SSW, and SST using MATLAB resulted as:

Between-Group Variation (SSB): 35.97
Within-Group Variation (SSW): 8.38
Total Variation (SST): 44.35

These results indicate the majority of the variation in the data is due to differences between the group means (SSB), with a smaller portion attributable to individual differences within the groups (SSW). Mean Squares (MS) was calculated by dividing each Sum of Squares (SS) by its corresponding degrees of freedom (df):

Between-group mean square (MSB)

$$MSB = \frac{SSB}{df_{between}} = \frac{35.97}{2} = 17.985$$

Within-group mean square (MSW)

$$MSW = \frac{SSW}{df_{within}} = \frac{8.38}{27} = 0.3104$$

The F-statistic in ANOVA was calculated to determine if the variability between group means was significantly larger than the variability within groups.

$$F_{calculated} = \frac{MSB}{MSW} = \frac{17.985}{0.310} = 58.01$$

To determine if the differences between group means were statistically significant. Compare the calculated F-statistic to a critical F-value obtained from an F-distribution table. MATLAB's "finv" function was used to find the critical F-value at a significance level α=0.05 and degrees of freedom between and within. Results gave an F-statistic of 58.01, and the critical F-value (calculated from F(2, 27, 0.05)) in MATLAB gave 3.354.After performing the ANOVA analysis and comparing the F-statistic with the critical F-value. The decision rule is if $F_{statistic} > F_{critical}$, reject the null hypothesis and if $F_{statistic} \leq F_{critical}$ do not reject.

Since $F_{statistic}$= 58.01 is much greater than the critical F-value = 3.354, we rejected the null hypothesis. This means there are statistical differences between the means of the three rule sets (A, B and C)

The shows that the variation between the group means is significantly larger than the variation within the groups. Therefore, at least one of the rule sets has a mean that differs significantly from the others.

Tukey's HSD test using the "multcompare" function in MATLAB was utilized after rejecting the null hypothesis in ANOVA to determine which specific group means are significantly different. The q-value in the Tukey HSD test comes from the Studentized Range Distribution Table (also known as the Tukey Table). It depends on:

1. Number of groups (k): In this case, 3 rule sets (A, B, and C).
2. Degrees of freedom for the error term (df within groups) is calculated as df = N-k, where:
   - N = total number of observations (10 trials × 3 groups = 30)
   - k = number of groups (3)
   - So, df = 30 - 3 = 27
3. Significance level (α) = Typically 0.05
   To find q, in the Tukey Table for k = 3 groups and df = 27 at α = 0.05.

From statistical reference tables, the q-value for k = 3, df = 27, and α = 0.05 is approximately 3.506.

$$HSD = q\, x \sqrt{\frac{MSE}{n}}$$

Where

- q is the critical value from the Tukey table (depends on the number of groups and degrees of freedom).
- MSE is the Mean Squared Error from ANOVA.
- n is the number of observations per group.

$$MSE = \frac{SSW}{N-k} = \frac{8.383}{30-3} = \frac{8.383}{27} = 0.3104$$

$$HSD = 3.506\, x \sqrt{\frac{0.3104}{10}} = 0.6176$$

Comparing mean differences with HSD

181.49 – 179.76 = 1.73 (significant since 1.73 > 0.6176
181.49 – 178.85 = 2.64 (significant since 2.64 > 0.6176
179.76 – 178.85 = 0.91 (significant since 0.91 > 0.6176

The Tukey HSD test shows that all three rule sets (A, B, and C) have statistically significant differences. The largest difference is between Rule Set A and Rule Set C (|181.49 - 178.85| = 2.64, which is much greater than the HSD value of 0.6176). Rule Set A is the most distinct, as it has the highest mean and is significantly different from both B and C.

The mean values of the three rule sets:

- Rule Set A: 181.49 (highest)
- Rule Set B: 179.76
- Rule Set C: 178.85 (lowest)

Rule Set A (27 rules) exhibited the highest mean performance (181.49) and was significantly different from both Rule Set B and Rule Set C. Rule Set C is the most

distinct and best-performing rule set as it takes the shortest time to complete a task, making it the optimal choice for applications requiring superior performance. The significant difference between Rule Set B and Rule Set C suggests that reducing the number of rules further increases performance. Future work may explore whether the reduction of rules below 14 would further improve performance or reach a plateau.

6. Conclusion

This study has demonstrated the effectiveness of a similarity-based rule reduction method in optimizing Fuzzy Logic Controllers (FLCs) for autonomous navigation. By reducing redundant rules through similarity assessment and an adaptive threshold optimization mechanism, the system significantly improved computational efficiency without compromising decision accuracy. Experimental results confirmed that a reduced rule set (from 27 to 14) led to faster traversal times and enhanced real-time processing, making the system more suitable for dynamic environments. Statistical analysis based on ANOVA and Tukey's HSD test confirmed the performance differences between the three rule sets, and Rule Set C (14 rules) emerged as the most efficient. The result reaffirms the significance of finding a balance between the rule complexity and efficiency for high-performance autonomous decision-making. The method has potential applications in real-world robotics, driverless vehicles, and predictive maintenance, where efficient and adaptive fuzzy logic-based systems are essential.

Future studies will seek further simplifications of rule complexity and examine the method's applicability for use in more complex, multi-variable contexts. The application of machine learning algorithms to automatically optimized rule determination could also make fuzzy logic-based decision systems more scalable.

References

[1] Anushka Biswas, and Hwang-Cheng Wang, "Autonomous Vehicles Enabled by the Integration of IoT, Edge Intelligence, 5G, and Blockchain," *Sensors*, vol. 23, no. 4, pp. 1-60, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mohsen Soori et al., "Intelligent Robotic Systems in Industry 4.0: A Review," *Journal of Advanced Manufacturing Science and Technology*, vol. 4, no. 3, pp. 1-29, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Hooi Hung Tang, and Nur Syazreen Ahmad, "Fuzzy Logic Approach for Controlling Uncertain and Nonlinear Systems: A Comprehensive Review of Applications and Advances," *Systems Science & Control Engineering*, vol. 12, no. 1, pp. 1-34, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Le Truong Giang et al., "Adaptive Spatial Complex Fuzzy Inference Systems With Complex Fuzzy Measures," *IEEE Access*, vol. 11, pp. 39333-39350, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Dongrui Wu, and Jerry M. Mendel, "Similarity Measures for Closed General Type-2 Fuzzy Sets: Overview, Comparisons, and a Geometric Approach," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 3, pp. 515-526, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Dimas Wibisono Prakoso, Asad Abdi, and Chintan Amrit, "Short Text Similarity Measurement Methods: A Review," *Soft Computing*, vol. 25, no. 6, pp. 4699-4723, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Habiba Batti, Chiraz Ben Jabeur, and Hassene Seddik, "Autonomous Smart Robot for Path Predicting and Finding in Maze Based on Fuzzy and Neuro-Fuzzy Approaches," *Asian Journal of Control*, vol. 23, no. 1, pp. 3-12, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] M. Khairudin et al., "The Mobile Robot Control in Obstacle Avoidance Using Fuzzy Logic Controller," *Indonesian Journal of Science and Technology*, vol. 5, no. 3, pp. 334-351, 2020. [Google Scholar] [Publisher Link]

[9] Aggrey Shitsukane et al., "Fuzzy Logic Sensor Fusion For Obstacle Avoidance Mobile Robot," *IST-Africa Week Conference (IST-Africa)*, Gaborone, Botswana, pp. 1-8, 2018. [Google Scholar] [Publisher Link]

[10] Chian-Song Chiu, Teng-Shung Chiang, and Yu-Ting Ye, "Fuzzy Obstacle Avoidance Control of a Two-Wheeled Mobile Robot," *International Automatic Control Conference*, Yilan, Taiwan, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[11] Habiba Batti, Chiraz Ben Jabeur, and Hassene Seddik, "Mobile Robot Obstacle Avoidance in labyrinth Environment Using Fuzzy Logic Approach," *International Conference on Control, Automation and Diagnosis*, Grenoble, France, pp. 1-5, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Min-You Chen, and D.A. Linkens, "Rule-Base Self-Generation and Simplification for Data-Driven Fuzzy Models," *Fuzzy Sets and Systems*, vol. 142, no. 2, pp. 243-265, 2004. [CrossRef] [Google Scholar] [Publisher Link]