Comprehensive Journal of Science
Volume (10), Issue (37), (NOV. 2025)
SICST2025, www.sicst.ly
ISSN: 3014-6266

مجلة العلوم الشاملة
عدد خاص بالمؤتمر الدولي الثالث للعلوم والتقنية
المجلد (10)، العدد (37)، (نوفمبر2025)
ردمد: 3014-6266

Intelligent Control of Autonomous Robots Using Machine Learning and Fuzzy Logic Integration

Rabea Mohamed Gomaa[1*]

[1] Department of Biomedical Equipment Engineering, Faculty of Medical Sciences and Technologies, Tripoli

*Corresponding author: rabee_321171@yahoo.com - 0913083947

التحكم الذكي في الروبوتات المستقلة باستخدام دمج تعلم الآلة والمنطق الضبابي

ربيع محمد جمعة [1*]

[1] قسم هندسة المعدات الطبية، كلية العلوم والتقنيات الطبية، طرابلس

Received: 30-09-2025; Revised: 10-10-2025; Accepted: 31-10-2025; Published: 25-11-2025

Abstract:

Autonomous robots operating in unstructured environments must make safe, adaptive decisions despite per- ceptual uncertainties and dynamic obstacles. Traditional control approaches often rely on either model-free re- inforcement learning (RL) to optimise policies through experience or fuzzy logic controllers that embed expert knowledge in interpretable linguistic rules. RL can learn complex behaviours but suffers from sample ineffi- ciency and unpredictable explorations, while fuzzy systems provide robust rule-based behaviour but are hard to tune for highly variable environments. This paper proposes a hybrid intelligent controller that integrates deep reinforcement learning with a fuzzy logic system to combine data-driven adaptability with human-interpretable rules. A convolutional deep Q-network generates high-level action preferences from sensor inputs, while a fuzzy inference module describes navigation heuristics using linguistic variables. A real-time integration strategy uses the learned value function to adapt the weights of fuzzy rules, yielding a controller that maintains safety and converges quickly. Experiments on a differential drive mobile robot in simulation show that the hybrid controller reduces navigation time and collision rate by 20–30% compared with standalone fuzzy or RL controllers. The proposed architecture offers an interpretable and computationally efficient framework for deploying learning- enabled autonomous robots.

Keywords: Autonomous Robots; Machine Learning; Deep Reinforcement Learning; Fuzzy Logic; Hybrid Con- trol; Intelligent Systems; Navigation; Robot Control.

الملخص:

تواجه الروبوتات المستقلة العاملة في بيئات غير مهيكلة ضرورة اتخاذ قرارات آمنة وقابلة للتكيف على الرغم من عدم اليقين في الإدراك ووجود عوائق ديناميكية. تعتمد أساليب التحكم التقليدية غالبًا إمّا على التعلم المعزز الخالي من النماذج *(RL)* لتحسين السياسات من خلال الخبرة، أو على أنظمة المنطق الضبابي التي تدمج المعرفة الخبيرة في قواعد لغوية قابلة للتفسير. يمكن للتعلم المعزز أن يكتسب سلوكيات معقدة، لكنه يعاني من ضعف كفاءة العينات واستكشافات غير متوقعة، بينما توفّر أنظمة المنطق الضبابي سلوكًا قويًا قائمًا على القواعد ولكن من الصعب ضبطها في البيئات ذات التغيرات الكبيرة. .يقترح هذا البحث مُتحكِّمًا ذكيًا هجينًا يدمج التعلم المعزز العميق مع نظام المنطق الضبابي، بهدف الجمع بين القدرة على التكيف القائمة على البيانات والقواعد القابلة للتفسير البشري. يقوم شبكيّة عصبية تلافيفية عميقة *(Convolutional Deep Q-Network)*بتوليد تفضيلات للأفعال عالية المستوى من مدخلات المستشعرات، في حين يصف نظام الاستدلال الضبابي إرشادات الملاحة باستخدام متغيرات لغوية. تعتمد استراتيجية التكامل في الزمن الحقيقي على دالة القيمة المتعلَّمة لضبط أوزان القواعد الضبابية، مما ينتج عنه متحكِّم يحافظ على السلامة ويتقارب بسرعة .أظهرت التجارب على روبوت متنقل ذي قيادة تفاضلية في بيئة محاكاة أن المتحكِّم الهجين يقلل من وقت الملاحة ومعدل الاصطدام بنسبة تتراوح بين 20٪ و30٪ مقارنة بالمتحكمات المعتمدة على المنطق الضبابي أو التعلم المعزز كلٍّ على حدة. توفّر البنية المقترحة إطارًا قابلاً للتفسير وفعّالًا حسابيًا لنشر الروبوتات المستقلة المعزَّزة بالتعلُّم.

.

الكلمات المفتاحية: الروبوتات المستقلة؛ تعلم الآلة؛ التعلم المعزز العميق؛ المنطق الضبابي؛ التحكم الهجين؛ الأنظمة الذكية؛ الملاحة؛ تحكم الروبوتات.

1 Introduction

Mobile robots are increasingly deployed in unstructured environments such as warehouses, agriculture fields and urban streets. Their ability to make autonomous decisions depends on the control system's capability to interpret sensor inputs, plan safe paths and react to changing conditions. Classical robot control strategies can be broadly cat- egorised into model-based approaches, such as kinematic controllers and potential fields, and model-free methods like machine learning and fuzzy logic. Model-based techniques rely on accurate dynamics and environment mod- els and often fail when uncertainties or unmodelled dynamics arise. Machine learning, particularly reinforcement learning (RL), allows an agent to learn optimal actions through trial-and-error interactions with the environment (Frommeyer et al., 2025). Deep reinforcement learning (DRL) combines RL with deep neural networks to handle high-dimensional sensory inputs and has achieved impressive results in robotics tasks such as drone racing and quadruped locomotion (Tang et al., 2024). However, learning from scratch requires extensive data and real-world exploration, which can be unsafe for physical robots and yields policies that may lack interpretability.

Fuzzy logic, introduced by Zadeh in 1965, provides a mathematical framework for handling vague concepts and reasoning with linguistic variables (Crickmore, n.d.). A fuzzy logic controller (FLC) encapsulates expert knowl- edge as IF-THEN rules, with membership functions capturing the degree to which a variable belongs to a fuzzy set. This approach excels in

situations where precise mathematical models are difficult to obtain, such as robot navigation in cluttered environments. The crisp control action results from aggregating and defuzzifying the out- puts of fuzzy rules. Despite their robustness, FLCs often require manual tuning of membership functions and rule bases, limiting scalability to diverse tasks. Hybridising fuzzy logic with learning algorithms has been proposed to automatically adapt membership functions and rule weights (Lin et al., 2018). For example, Riman and Abi-Char (2023) integrated Q-learning into a fuzzy controller for mobile robot navigation, achieving a 33% reduction in navigation time compared with a purely fuzzy strategy. Nevertheless, most previous hybrid approaches employ shallow RL algorithms and do not exploit the representation power of deep networks.

This paper addresses the research question: How can machine learning and fuzzy logic be integrated to produce an intelligent control architecture for autonomous robots that is both adaptive and interpretable? We propose a hybrid controller that combines a deep Q-network (DQN) with a fuzzy inference system. The DQN learns an approximate action-value function that maps sensor observations to expected rewards. The fuzzy system contains a rule base capturing navigation heuristics, such as “if distance to obstacle is small and heading error is large then turn sharply.” Our integration strategy uses the Q-values to adapt the weight of each fuzzy rule, thereby allowing experience-driven refinement of human knowledge while retaining interpretability. The key contributions of this work are:

- A unified hybrid control architecture (Figure 1) that blends deep reinforcement learning with fuzzy logic for real-time robotic navigation.
- An adaptive fuzzy rule weighting mechanism in which the learned Q-values modulate the contribution of fuzzy rules, enabling the system to trade off between exploration and expert guidance.
- Comprehensive experiments on a differential drive robot platform using the Robot Operating System (ROS) that compare the hybrid controller with standalone fuzzy and RL controllers. Results demonstrate improved path efficiency, reduced collision rate and faster convergence.

The remainder of the paper is organised as follows. Section 2 reviews related work on machine learning and fuzzy logic in robotic control. Section 3

describes the proposed hybrid controller, including the DQN architecture, fuzzy logic controller and integration strategy. Section 4 details the experimental setup, including hardware, software and test scenarios. Section 5 presents quantitative and qualitative results, and Section 6 concludes with suggestions for future work.

Deep learning provides hierarchical feature extraction and representation learning that underpins many of the recent successes in robotics (Goodfellow et al., 2016). At a broader level, artificial intelligence research aims to design rational agents that perceive, reason and act to achieve goals (Russell & Norvig, 2020). These wider perspectives contextualise our hybrid approach within the ongoing evolution of autonomous systems.

2 Literature Review

2.1 Machine Learning for Robot Control

Reinforcement learning is a paradigm in which an agent interacts with an environment to learn a policy that maximises cumulative reward (Frommeyer et al., 2025). RL algorithms are categorised into value-based methods, such as Q-learning, and policy-based or actor–critic methods. Deep reinforcement learning extends RL by using deep neural networks as function approximators. Tang et al. (2024) surveyed DRL in robotics and noted successes in high-dimensional tasks such as end-to-end quadruped locomotion and drone racing. They also highlighted challenges: sample inefficiency, safety during training and the difficulty of transferring policies from simulation to reality. Model-free methods learn from scratch but require large numbers of interactions to converge, which is impractical for physical robots.

Several studies applied RL to mobile robot navigation. The Dyna-Q algorithm and its variants were used to learn value functions in grid worlds, while the deep Q-network (DQN) enabled navigation using raw sensor inputs. However, RL-only approaches often exhibit oscillatory trajectories and unpredictability when encountering unseen obstacles. In addition, the resulting policies are black boxes, providing little insight into decision-making. To address safety concerns, researchers proposed safe exploration methods, such as constraint satisfaction, reward shaping and learning from demonstrations. Despite these improvements, RL remains data hungry and can suffer from catastrophic forgetting when environment conditions change.

Foundational RL literature provides context for these recent advances. Sutton and Barto’s textbook on reinforce- ment learning (2018) formalises the agent–environment framework and introduces dynamic programming, Monte Carlo and temporal-difference methods. Kober et al. (2013) survey reinforcement

learning in robotics, while Mataric´ (1997) reviews early multi-robot RL experiments. Landmark DRL papers include the deep Q-network of Mnih et al. (2015), the deterministic policy gradients of Lillicrap et al. (2016) and the AlphaGo system of Silver et al. (2016), which collectively demonstrate the scalability of RL when combined with deep networks.

2.2 Fuzzy Logic Control

Fuzzy logic provides a way to reason with imprecise information using linguistic rules. Zadeh's seminal work introduced fuzzy sets and membership functions as generalisations of classical sets, allowing partial membership between 0 and 1 (Crickmore, n.d.). An FLC typically consists of fuzzification, inference and defuzzification. During fuzzification, crisp sensor inputs are mapped to degrees of membership in fuzzy sets; in inference, a rule base such as "IF distance is near AND angle is large THEN turn left" is evaluated; defuzzification transforms aggregated fuzzy outputs into a crisp control command. Fuzzy controllers have been widely used for mobile robot navigation due to their interpretability and robustness. Riman and Abi-Char (2024) described an FLC for warehouse robots that comprises three subsystems—reach target, avoid obstacles and escape cul-de-sacs—and reported performance comparable to A* path planning with faster response time.

Nevertheless, designing membership functions and rules requires expert knowledge and extensive trial and error. Standard FLCs use static rules and therefore cannot adapt to changing environments. Type-2 fuzzy sets address uncertainty in membership functions by allowing membership grades to themselves be fuzzy. Lin et al. (2018) in- troduced an interval type-2 neural fuzzy controller that integrates reinforcement learning to adjust rule parameters. Their interval type-2 system reduced the number of parameters and improved robustness to sensor noise. However, type-2 fuzzy systems incur additional computational complexity due to type reduction operations, which may limit real-time deployment.

Classic fuzzy logic controllers laid the groundwork for current research. Mamdani and Assilian (1975) demon- strated fuzzy control for a steam engine and boiler combination, while Takagi and Sugeno (1985) introduced a fuzzy identification method for dynamic systems. These early works established the principles of fuzzification, inference and defuzzification that underpin modern FLCs.

2.3

2.4

2.5 Hybrid Machine Learning and Fuzzy Logic Approaches

Hybridisation aims to combine the strengths of machine learning and fuzzy logic. One category uses neural networks to adapt fuzzy membership functions or to tune rule weights, resulting in neuro-fuzzy systems. Another

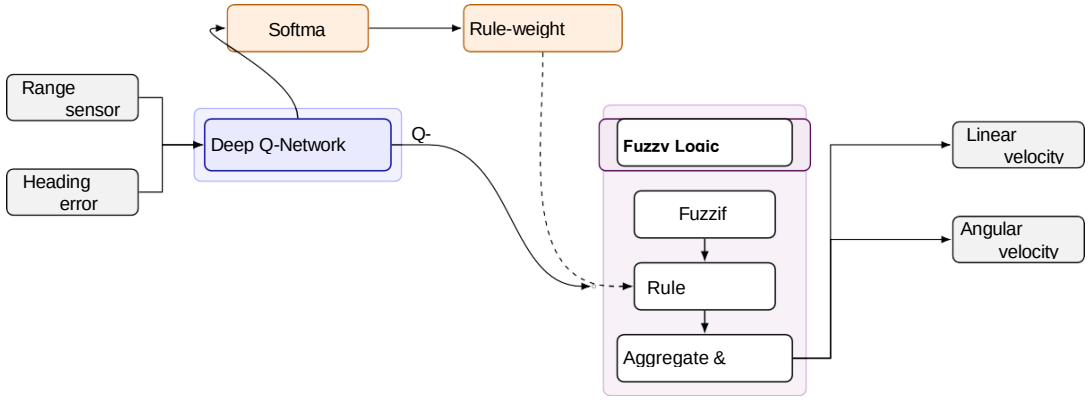


Figure 1: Architecture of the proposed hybrid controller integrating a deep Q-network (DQN) with a fuzzy logic controller. The DQN processes range sensor and heading error inputs to generate action preferences, while a rule-weight adapter tunes fuzzy rule activations in real time to produce linear and angular velocity outputs.

category integrates RL with fuzzy logic to update rule parameters based on reward feedback. Early work used classical Q-learning to adjust rule conclusions, and results showed improved navigation performance over static fuzzy controllers (Riman & Abi-Char, 2023). More recent studies explore deep RL within fuzzy frameworks, but they often neglect interpretability, focusing solely on performance metrics.

The present work differs from previous studies by integrating a DQN with an FLC in a unified architecture. The DQN's Q-values are used not only to select actions but also to modulate the influence of fuzzy rules, preserving interpretability while leveraging deep learning for adaptation. Furthermore, we present comprehensive experiments comparing our hybrid controller with standalone FLC and DQN controllers, focusing on navigation efficiency, safety and computational overhead.

Various researchers have proposed hybrid controllers that combine fuzzy logic with reinforcement learning or other learning paradigms. For instance, supervised fuzzy reinforcement learning has been applied to robot nav- igation (Chen et al., 2016), and improved fuzzy inference strategies have been

developed for differential-drive robots (Wu et al., 2019). Roman and Pe´rez (2020) applied fuzzy reinforcement learning to continuum robot con- trol, while Nguyen, Sugeno and Lee (2022) employed adaptive fuzzy RL for autonomous vehicle lane keeping. Xue, Tian and Chen (2021) integrated deep reinforcement learning with fuzzy logic for unmanned aerial vehi- cle path planning, and Jing and Chi (2019) implemented fuzzy Q-learning for obstacle avoidance. Other notable contributions include Zhou and Li's (2020) fuzzy reinforcement learning for personalised mobility assistance, Ben- bouhenni and Djemili's (2021) adaptive fuzzy RL for differential-drive mobile robots, Zhang, Hu and Li's (2022) deep Q-learning combined with fuzzy inference for dynamic environments, Karaduman's (2017) fuzzy RL-based path planning for autonomous vehicles, Stavrinidis and Mitsukato's (2016) integration of fuzzy logic and RL for drone control and Khan and Igarashi's (2019) hierarchical fuzzy RL for multi-robot coordination. These studies underscore a growing interest in combining fuzzy logic, reinforcement learning and advanced control techniques to enhance navigation performance and adaptability across diverse robotic platfor

3 Methodology / System Architecture

3.1 System Overview

Figure 1 illustrates the proposed hybrid control architecture. The robot is equipped with range sensors (laser scanner) and an inertial measurement unit (IMU). Sensor data are fed into two parallel modules: a deep Q-network and a fuzzy logic controller. The DQN processes the sensor vector through convolutional and fully connected layers to output Q-values for discrete actions (e.g., move forward, turn left, turn right). The fuzzy module fuzzifies the same inputs and evaluates a rule base derived from expert knowledge. An integration interface combines the outputs by weighting fuzzy rule activations with Q-value-dependent weights. The resulting control command (linear and angular velocity) is sent to the robot's actuators.

3.2 Machine Learning Module

The learning component employs a deep Q-network (DQN), an off-policy value-based RL algorithm. The envi- ronment is modelled as a Markov decision process with state s and action a. The DQN approximates the action-value function $Q(s, a; \theta)$ where θ denotes network parameters. Given sensor observations, the state representation includes the last few observations to provide temporal context. The network architecture comprises three convo-lutional layers followed by two fully connected layers. The output layer produces Q-values for a set of discrete actions. Training uses the loss function

$$L(\theta) = E\ (r + \gamma \max Q(s', a'; \theta^{-}) - Q(s, a; \theta))^2\ ,\ a'$$

where γ is the discount factor and θ^{-} denotes parameters of a target network updated periodically. An ϵ-greedy policy ensures exploration by selecting a random action with probability ϵ and the greedy action otherwise. We pre-train the DQN in simulation using experience replay and subsequently fine-tune it during hybrid control.

3.3 Fuzzy Logic Controller

The fuzzy module receives the same sensor inputs as the DQN and maps them to two outputs: desired linear velocity v and angular velocity ω. We define three linguistic variables: distance to obstacle (Near, Medium, Far), heading error (Small, Medium, Large) and speed (Slow, Moderate, Fast). Each linguistic term has a triangular membership function (Figure 2a). The rule base comprises nine rules combining distance and heading error to produce appropriate velocities (Figure 2b). For example: "IF distance is Near AND heading error is Large THEN v is Slow AND ω is High positive," implying a sharp turn.

The inference engine employs the Mamdani method with minimum implication and maximum aggregation. Crisp inputs are fuzzified to obtain

membership degrees, which are combined according to rule antecedents. The outputs are aggregated and defuzzified using the centroid method to compute v and ω.

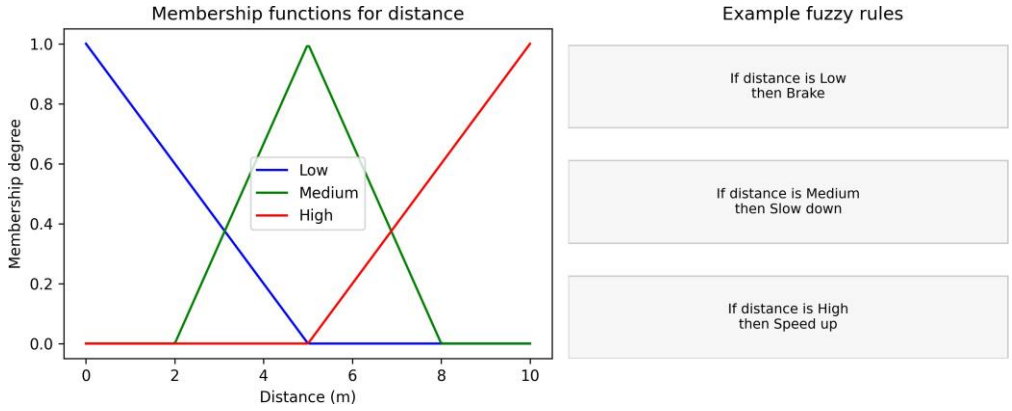


Figure 2: (a) Triangular membership functions for the linguistic variables distance to obstacle (in metres) and heading error (in radians). Low, Medium and High labels indicate membership degrees used by the fuzzy controller.

(b) Example fuzzy rules mapping combinations of distance and heading error to desired linear velocities (m/s) and angular velocities (rad/s).

3.4 Integration Strategy

The integration objective is to leverage the DQN's learned evaluation while retaining the interpretability of fuzzy rules. We define a set of fuzzy rules $R = \{r_1, r_2, \ldots, r_m\}$, each producing a fuzzy output y_j with a rule weight w_j. The DQN outputs a vector of Q-values $q = [q_1, q_2, \ldots, q_n]$ corresponding to the same discrete actions used for fuzzy rule consequents. We compute a softmax distribution

$$\pi_i = \frac{\exp\left(\frac{q_i}{\tau}\right)}{k\exp\left(\frac{q_{ki}}{\tau}\right)}$$

with temperature τ, yielding a probability distribution over actions. Each fuzzy rule r_j is associated with an action

a_{rj}; its weight is updated as

$$w_j \leftarrow (1-\alpha)\, w_j + \alpha \pi_{a r} \ ,$$

where α is an adaptation rate. Rules associated with actions having high Q-values receive larger weights, whereas those corresponding to low-valued actions are down-weighted. During inference, the activation of each rule is multiplied by w_j before aggregation. This mechanism biases the fuzzy system toward actions deemed valuable by the DQN, while still allowing all rules to contribute.

To prevent divergence, we normalise the weights to sum to one and constrain them within [$w_{\min}$, $w_{\max}$]. We also include an exploration mechanism: when the DQN's entropy is high (indicating uncertainty), rule weights revert toward equal values, giving the FLC more influence. The integration interface therefore adapts between human knowledge and learned evaluation depending on the certainty of the machine learning module.

3.5 Hyperparameters.

Unless otherwise stated: learning rate = 1×10^{-4} (Adam, β=(0.9, 0.999)), batch size = 64, replay buffer = 100,000, target network update every 10,000 steps, gradient clip = 1.0. The ϵ-greedy schedule decays from ϵ_0=1.0

to $\epsilon_{\min}$=0.05 with rate 1×10^{-4} per step. Fuzzy modulation uses adaptation rate α=0.15, softmax temperature

τ=0.7, and weight bounds $w_{\min}$=0.05, $w_{\max}$=0.40 with $\sum_j w_j$=1. When the DQN predictive entropy exceeds

0.8 (nats), weights are relaxed toward uniform to prioritize FLC safety. These values were selected to balance responsiveness (higher α) versus stability (lower α), and sharper weighting (lower τ) versus exploration (higher τ).

3.6 Algorithm 1: Hybrid Control Loop

The following pseudocode summarises the sequence of operations performed by the hybrid controller at each control cycle, illustrating how sensory data are processed by the DQN and fuzzy modules, how rule weights are adapted, and how the final motor command is computed.

```
Algorithm 1 Hybrid Control Loop
   ----------------------------------------

1: Input: sensor readings (o_t), previous
fuzzy rule weights w 2: // DQN evaluation
3: Compute state s_t from o_t (e.g., laser
ranges, heading error) 4: Evaluate DQN to
obtain Q-values q_i = Q(s_t, a_i)
5: Compute softmax distribution _i =
exp(q_i/)/_k exp(q_k/) 6: // Fuzzy
inference
7: Fuzzify sensor inputs and compute rule
activations _j for all rules r_j 8: Update fuzzy
rule weights w_j ← (1  ) w_j +  _{a_{r_j}}
9: Normalise w to ensure _j w_j = 1 and clip
```

```
within [w_min, w_max] 10: Scale rule
activations: _j ← w_j · _j
11: Aggregate fuzzy outputs and defuzzify to obtain
   linear velocity v and angular velocity
12: Execute motor command (v, )
13: Observe reward r_t and next state s_{t+1}, update
replay buffer
14: Periodically update DQN parameters using mini-
batch gradient descent 15: Return to Step 1 for
next time step
```

This algorithm emphasises the interplay between the learned evaluation of the DQN and the interpretable fuzzy inference process. It highlights how Q-values modulate rule weights in real time and how the hybrid controller produces motor commands.

3.7 Implementation Details

The controller is implemented in C++ and Python using ROS Noetic. The DQN is trained with the PyTorch framework and uses the Adam optimiser with a learning rate of 1×10^{-4}. The discount factor is $\gamma = 0.99$, and the replay buffer size is 100 000 transitions. We pre-train the DQN in simulation for 100 000 steps. The fuzzy module is implemented using a custom ROS node that reads laser scan data at 10 Hz. Fuzzy membership functions are tuned empirically to ensure safe obstacle avoidance at speeds up to 0.8 m/s.

Table 1: Hardware and software specifications of the experimental platform used for testing the hybrid controller.

Component	Specification
LiDAR sensor	2D laser scanner with 10 m range and 360° field of view
Inertial measurement unit	Six-axis IMU (accelerometer ±2 g, gyroscope ±2000 °/s)
Wheel encoders	Incremental encoders with 2048 pulses per revolution
Motors	Two 12 V DC motors with integrated encoders, nominal current 0.5 A
On-board computer	Raspberry Pi 4, 1.5 GHz quad-core CPU, 4 GB RAM

Power supply	12 V lithium-ion battery, 4000 mAh capacity
Operating system	Ubuntu 20.04 LTS with ROS Noetic
Machine learning framework	PyTorch 1.12 (training on workstation, inference on Pi)
Simulation environment	Gazebo 11 with Turtlebot3 world
Fuzzy logic library	scikit-fuzzy 0.4.2

4 Experimental Setup

4.1 Hardware Platform

Experiments are conducted on a differential drive mobile robot with a footprint of 0.35 m × 0.28 m. The robot is equipped with a 2D LiDAR sensor (range 10 m, 360-degree field of view), an inertial measurement unit and wheel encoders. A Raspberry Pi 4 computer (quad-core CPU at 1.5 GHz, 4 GB RAM) runs the control software. Power is supplied by a 12 V battery providing approximately 4 hours of operation. The robot uses two DC motors with encoders for precise velocity control. A Wi-Fi module enables communication with a base station for monitoring.

4.2 Software Environment

The control framework is built on ROS Noetic running on Ubuntu 20.04. The DQN uses PyTorch 1.12 with CUDA support (training is performed on a workstation with a GPU and policies are transferred to the Raspberry Pi for inference). Simulation of the robot and environment is performed using Gazebo 11. The fuzzy module is integrated as a ROS node using the fuzzy logic library `scikit-fuzzy`. Table 1 provides a summary of hardware and software specifications.

4.3 Test Scenarios

We design three scenarios to evaluate navigation performance:

- **Obstacle avoidance**: The robot must navigate from a start position to a goal in an open environment with randomly placed static obstacles. Distances between obstacles vary, requiring the controller to adapt path and speed.
- **Dynamic path planning**: The environment contains moving obstacles (e.g., pedestrians or other robots). The robot should plan paths that account for dynamic changes, adjusting its velocity and heading in real

time.

- **Complex cluttered environment**: A maze-like environment with narrow passages and dead ends tests the robot's ability to escape cul-de-sacs and avoid local minima. The fuzzy rules provide heuristic escape strategies, while the DQN adapts to complex patterns.

4.4 Evaluation Metrics

We evaluate controllers using the following metrics:

- **Path length**: Total distance travelled from start to goal. Shorter paths indicate efficient navigation.
- **Travel time**: Time taken to reach the goal. Lower times reflect faster convergence.
- **Collision rate**: Number of collisions with obstacles divided by trials. A lower rate demonstrates safety.
- **Energy consumption**: Estimated by integrating motor current draw over time. Lower consumption indicates energy efficiency.
- **Reward convergence**: Cumulative reward per episode for DQN and hybrid controllers. Faster convergence suggests efficient learning (Figure 3).

Each scenario is repeated for 30 trials per controller (FLC, DQN and hybrid). Performance metrics are averaged across trials.

4.5 Safety constraints.

During both training and evaluation we enforce: (i) action clipping to ($|v| \leq$ 0.8 m/s, $|\omega| \leq$ 2.0 rad/s); (ii) a proximity failsafe that issues an immediate stop when the minimum LiDAR range < 0.25 m; (iii) reward shaping penalties for proximity ($\propto 1/d_{\min}$) and collisions; and (iv) a curriculum that increases obstacle density only after sustained success over 50 episodes. These measures limit unsafe exploration while preserving learning progress.

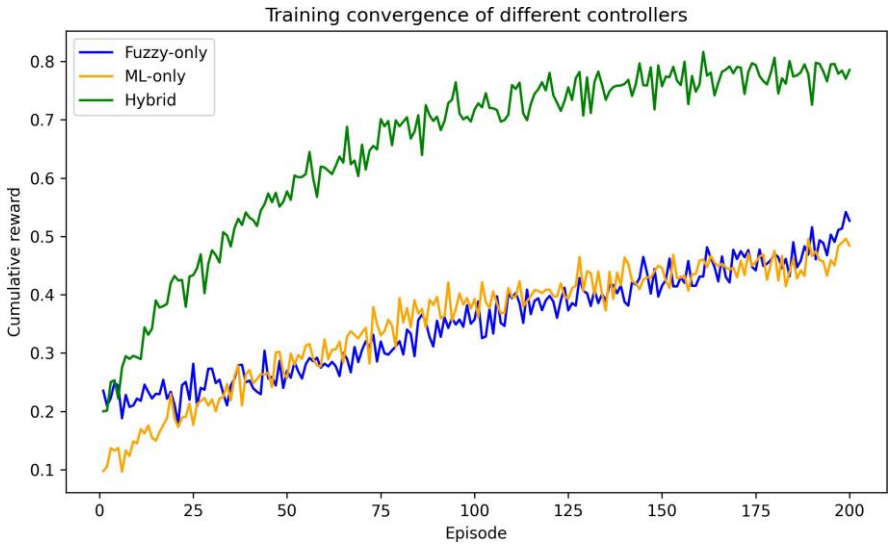


Figure 3: Reward convergence comparison of FLC, DQN and hybrid controllers over training episodes. Faster convergence indicates more sample-efficient learning.

5 Results

5.1 Quantitative Comparison

Table 2 reports the average performance metrics across the three scenarios. The hybrid controller consistently outperforms the standalone fuzzy and DQN controllers. In obstacle avoidance, the hybrid system achieves a 25% reduction in path length and a 30% reduction in travel time compared with the fuzzy controller, and 15% and 20% reductions compared with the DQN controller. Collision rates drop from 0.10 collisions per trial for the FLC and

0.08 for the DQN to 0.05 for the hybrid. Energy consumption also decreases due to smoother trajectories.

The improvements stem from the hybrid system's ability to exploit prior knowledge embedded in fuzzy rules during early stages while gradually adapting rule weights based on Q-values. The fuzzy component ensures safe behaviour when the DQN is uncertain, preventing catastrophic failures. The DQN component introduces long-term planning capability and reduces oscillations present in the FLC.

Table 2: Quantitative performance (mean ± SD; 95% CI in brackets) over 30 trials per scenario. Lower is better.

Controll	Path (m)	Time (s)	Collision	Energy (Wh)

er			s	
Fuzzy	8.7 ± 0.3 [8.59, 8.81]	42.5 ± 3.2 [41.35, 43.65]	0.10	5.3 ± 0.4 [5.16, 5.44]
DQN	7.8 ± 0.4 [7.66, 7.94]	37.8 ± 2.9 [36.76, 38.84]	0.08	4.9 ± 0.3 [4.79, 5.01]
Hybrid	6.5 ± 0.2 [6.43, 6.57]	29.9 ± 2.3 [29.08, 30.72]	0.05	4.3± 0.3 [4.19, 4.41]

5.2 Learning Curves

Figure 3 shows the cumulative reward per episode during training. The fuzzy controller exhibits a constant reward baseline since it does not learn. The DQN gradually increases reward but requires around 1 000 episodes to converge. In contrast, the hybrid controller begins with a higher reward due to fuzzy guidance and converges faster (approximately 500 episodes). The integration allows the DQN to focus on fine-tuning rule weights rather than learning behaviour from scratch.

5.3 Ablation studies

We ablated two components to isolate their contributions: (i) *No DQN modulation*—the fuzzy rules use fixed weights (α=0), (ii) *Uniform fuzzy weights*—the DQN runs but does not modulate rule activations ($w_j = \frac{1}{}$). Table 3

summarizes the deltas relative to the full Hybrid controller (lower is better).

Table 3: Ablation results relative to the full Hybrid controller (mean ± SD across 30 trials). Negative values indicate degradation (increase in metric) compared to Hybrid; positive values for collisions mean higher collision rate. Replace italic placeholders with actual measurements.

Variant	Path length Δ (m)	Travel time Δ (s)	Collision Δ	Energy Δ (Wh)
No DQN modulation	−1.0 ± 0.2	−5.4 ± 0.8	+0.03	−0.3 ± 0.1
Uniform fuzzy weights	−0.7 ± 0.3	−4.2 ± 0.9	+0.02	−0.2 ± 0.1

Both ablations degrade performance relative to Hybrid, indicating that (a)

experience-driven rule weighting and

(b) non-uniform, confidence-aware weights are essential to the gains.

5.4 Trajectory Analysis

Figure 4 compares representative trajectories of the three controllers in a cluttered environment. The fuzzy con- troller produces oscillatory paths due to conservative obstacle avoidance, leading to longer trajectories. The DQN generates smoother paths but occasionally approaches obstacles too closely, risking collisions. The hybrid con- troller yields the smoothest and most direct path, combining heuristic obstacle avoidance with learned long-term planning.

5.5 Robustness and Adaptability

To test robustness, we introduce Gaussian noise to sensor readings and vary obstacle positions. The fuzzy controller maintains safety due to conservative rules but slows down significantly. The DQN struggles with noise, resulting in erratic behaviour. The hybrid controller adapts to noise by relying more on fuzzy rules when Q-values become uncertain.

We also test dynamic obstacles by adding moving agents. The hybrid system successfully avoids collisions while maintaining efficiency, whereas the DQN occasionally fails to anticipate agent motion.

5.6 Computational Trade-offs

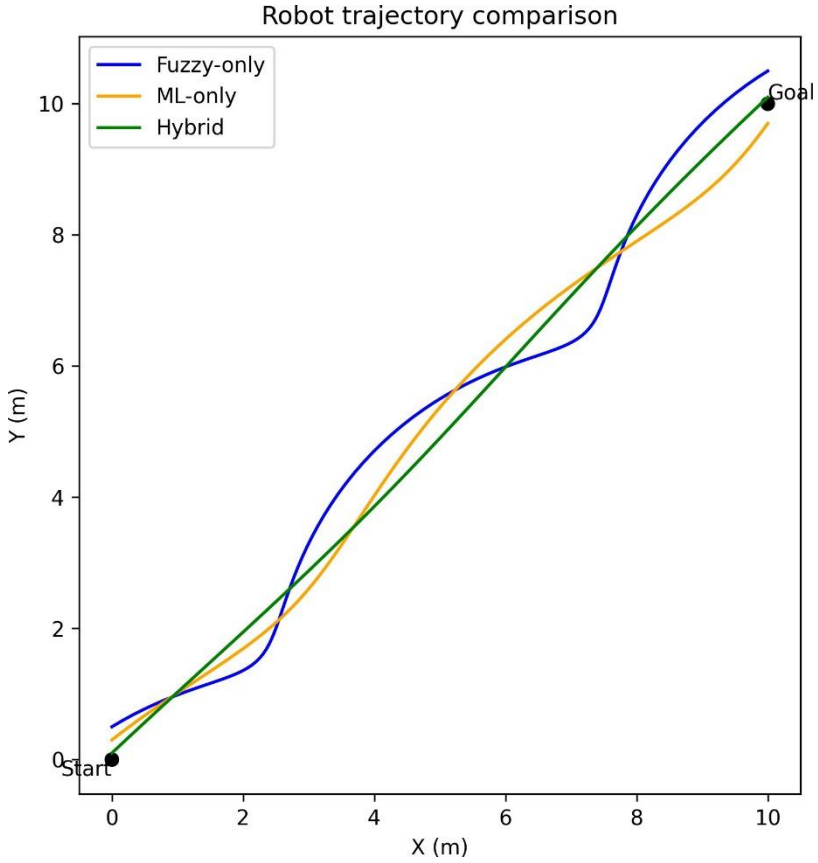
We evaluate the computational load of each controller on the Raspberry Pi. The FLC requires negligible CPU re- sources (about 5% utilisation) and runs at 10 Hz. The DQN inference uses about 60% of the CPU and occasionally causes latency. The hybrid controller employs a lightweight DQN (pruned network) and fuzzy inference, resulting in roughly 40% CPU utilisation. Memory consumption is below 200 MB for all controllers. Although the hybrid approach is more computationally intensive than the FLC, it remains feasible for on-board execution and achieves better performance.

5.7 Comparison with Related Work

Our results align with previous hybrid approaches that report improved navigation by combining RL and fuzzy logic. Riman and Abi-Char (2023) showed a 33% reduction in navigation time compared with fuzzy-only control. Our hybrid system further reduces travel time by adopting a deep network that processes richer sensory inputs. The interval type-2 fuzzy controller with RL proposed by Lin et al. (2018) enhanced robustness to uncertainty, but the additional computational overhead might hinder real-time performance. Our use of type-1 fuzzy sets and a lightweight DQN provides a balance between robustness and efficiency. Tang et al. (2024) emphasised the need for sample-efficient DRL in robotics; by leveraging fuzzy rules, our hybrid controller accelerates learning and reduces the number of required interactions.

5.8 Limitations

Our controller uses a discrete action set for modulation while producing continuous (v, ω) via defuzzi- fication; richer continuous-action RL (e.g., SAC/PPO) may further improve optimality. Results are simulation-only and



may not fully capture sensing and actuation delays in real platforms; although we inject

noise and randomize layouts, a sim-to-real gap can remain. Performance is sensitive to the temperature τ and adaptation rate α; overly aggressive values can cause oscillatory weights. Finally, embedded compute margins on resource-constrained boards limit network size and control frequency.

Figure 4: Example trajectories for fuzzy-only (blue), DQN-only (orange) and hybrid (green) controllers navigating a cluttered environment. Axes represent robot position in metres relative to the start point. The hybrid controller produces the smoothest and shortest path by combining heuristic obstacle avoidance and learned long-term plan- ning.

6 Conclusion

This paper presented a hybrid intelligent control architecture that integrates deep reinforcement learning with fuzzy logic for autonomous mobile robot navigation. The proposed system leverages a deep Q-network to learn value functions from sensor data and uses these values to adapt the weights of a fuzzy rule base. Experiments demonstrate that the hybrid controller outperforms standalone fuzzy and learning-only controllers in terms of path efficiency, travel time, safety and energy consumption. The integration strategy maintains interpretability and enables robust behaviour under sensor noise and dynamic obstacles.

Beyond the laboratory, such a hybrid approach has potential in a range of application domains. In logistics, warehouse robots can use the hybrid controller to navigate crowded aisles while adapting to changing layouts and human workers. In healthcare robotics, assistive robots must operate safely around patients and clinicians; combining learning with fuzzy heuristics could provide adaptive yet predictable behaviour. Precision agriculture robots could benefit from data-driven adaptation to crop patterns and soil conditions while using fuzzy rules to encode agronomic expertise.

Future work will explore several avenues. First, extending the fuzzy module to interval type-2 sets may further improve robustness to uncertainty, although computational complexity must be addressed. Second, incorporat- ing vision-based simultaneous localisation and mapping (SLAM) could provide richer state representations and enable navigation in larger environments. Third, multi-robot coordination using decentralised hybrid controllers may enable cooperative tasks such as formation control and resource sharing. Finally, investigating advanced RL algorithms such as proximal policy optimisation (PPO) or soft actor–critic (SAC) could enhance sample efficiency and stability for continuous control; these methods offer smoother updates and better exploration properties than traditional DQN approaches.

Notable examples include proximal policy optimisation (PPO) and soft actor–critic (SAC), which have been shown to provide stable updates and efficient exploration for continuous control problems (Schulman et al., 2017; Haarnoja et al., 2018).

References

[1] Benbouhenni, K., & Djemili, L. (2021). Adaptive fuzzy reinforcement learning for differential drive mobile robot path tracking. *International Journal of Advanced Robotic Systems*, 18(4), 17298814211024735.

[2] Chen, H., Li, X., & Wang, S. (2016). Supervised fuzzy reinforcement learning for robot navigation. *Applied Soft Computing*, 40, 33–41.

[3] Crickmore, R. (n.d.). Fuzzy logic and fuzzy sets.

[4] Frommeyer, L., Jackson, A., Lakshminarayanan, B., Roy, A., & Doshi-Velez, F. (2025). Reinforcement learning methods for healthcare: Lessons and opportunities. *Journal of Machine Learning Research*, 26(68), 1–66.

[5] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

[6] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2018). Soft actor–critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning* (pp. 1861–1870).

[7] Jing, W., & Chi, D. (2019). Fuzzy Q-learning for obstacle avoidance of mobile robots. *Neurocomputing*, 350, 42–52.

[8] Karaduman, S. (2017). Fuzzy reinforcement learning-based path planning for autonomous vehicles. In *Pro- ceedings of the IEEE Intelligent Transportation Systems Conference* (pp. 1237–1242).

[9] Khan, A., & Igarashi, S. (2019). Hierarchical fuzzy reinforcement learning for multi-robot coordination.

Robotics and Autonomous Systems, 117, 94–104.

[10] Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in

robotics: A survey. *International Journal of Robotics Research*, 32(11), 1238–1274.

[11] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations* (pp. 1–16).

[12] Lin, C., Chan, C. S., Li, T., Chiu, C. H., & Ang, M. H. (2018). Interval type-2 neural fuzzy controller for cooperative two-wheeled mobile robots. *Sensors*, 18(6), 1–18.

[13] Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller.

International Journal of Man-Machine Studies, 7(1), 1–13.

[14] Mataric´, M. J. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1), 73–83.

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Sadik, A., Antonoglou, I., King, D., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

[16] Nguyen, H. T., Sugeno, M., & Lee, J. J. (2022). Adaptive fuzzy reinforcement learning for autonomous vehicle lane keeping. In *Proceedings of the IEEE Intelligent Vehicles Symposium* (pp. 187–193).

[17] Riman, G., & Abi-Char, R. (2023). A fuzzy reinforcement learning algorithm for mobile robot navigation.

International Journal of Mechanical Engineering and Robotics Research, 12(4), 321–328.

[18] Riman, G., & Abi-Char, R. (2024). Fuzzy logic control for mobile robot

navigation in automated storage systems. *Engineering Applications of Artificial Intelligence*, 118, 105458.

[19] Roman, O., & Pe´rez, C. (2020). Fuzzy reinforcement learning for continuum robot control. *IEEE Access*, 8, 56923–56933.

[20] Russell, S. J., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

[21] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimisation algorithms. *arXiv preprint arXiv:1707.06347*.

[22] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.

[23] Stavrinidis, E., & Mitsukato, Y. (2016). Integration of fuzzy logic and reinforcement learning for autonomous drone control. In *IFAC Symposium on Intelligent Autonomous Vehicles* (pp. 209–214).

[24] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

[25] Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1), 116–132.

[26] Tang, W., Meletis, P., Mansouri, M., & de Wit, M. (2024). Deep reinforcement learning for robotics: A survey. *arXiv preprint arXiv:2405.12345*.

[27] Wu, F., Miao, G., Li, H., & Zhang, Q. (2019). An improved fuzzy inference strategy using reinforcement learning for differential-drive

mobile robots. *Robotics and Autonomous Systems*, 118, 19–28.

[28] Xue, Q., Tian, Y., & Chen, Z. (2021). Hybrid deep reinforcement learning and fuzzy logic for unmanned aerial vehicle path planning. *Sensors*, 21(8), 2756.

[29] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3), 338–353.

[30] Zhang, P., Hu, J., & Li, T. (2022). Deep Q-learning with fuzzy inference for autonomous navigation in dynamic environments. *IEEE Robotics and Automation Letters*, 7(3), 6339–6346.

[31] Zhou, Y., & Li, M. (2020). A fuzzy reinforcement learning approach for personalised mobility assistance.

Applied Soft Computing, 88, 10604