Proceedings of the 21st International Conference on Global Research and Education (Inter-Academia 2024)

Mobile Robot Navigation System Using Reinforcement Learning with Path Planning Algorithm

E.W. Andarge[a,*], A. Ordys[a] and Y.M. Abebe[b]

[a] *Faculty of Mechatronics, Institute of Micromechanics and Photonics, Warsaw University of Technology, św. A. Boboli 8, 02-525 Warsaw, Poland*
[b] *College of Engineering, Department of Electrical and Computer Engineering, Addis Ababa Science and Technology University, Akaki Kality Sub-City, Addis Ababa, Ethiopia*

Doi: 10.12693/APhysPolA.146.452 *e-mail: ephrem_worku.andarge.stud@pw.edu.pl

Navigation systems are critical for the practical implementation of autonomous vehicles. While classical and modern control methods have been explored, simultaneous localization and mapping have limitations due to their complexity and sensor requirements. As a result, there is growing interest in using artificial intelligence algorithms. This paper proposes to use a reinforcement learning algorithm agent as a local planner for obstacle avoidance, coupled with rapidly exploring random trees for global path planning. Unlike prior methods relying solely on reinforcement learning or point of interest for global path planning, this approach integrates reinforcement learning with random trees to ensure efficient navigation in complex environments. By mapping the environment and generating optimal global paths, divided into intermediate waypoints, the proposed method facilitates efficient navigation toward the goal. Experimental results demonstrate the effectiveness of the approach, particularly in navigating through intricate environments, offering promising advancements in autonomous navigation systems.

topics: simultaneous localization and mapping (SLAM), artificial intelligence (AI), reinforcement learning (RL), rapidly exploring random trees (RRT)

1. Introduction

Mobile robots, unlike stationary robots, offer versatility by autonomously navigating in various environments. *Autonomous mobile robots* (AMRs) operate without physical or electro-mechanical guidance, unlike guided robots that follow set routes. Robust navigation systems are crucial to applications in industry, healthcare, and households, with advancements such as Industry 4.0 boosting manufacturing flexibility and productivity [1]. The field includes technologies such as humanoid robots, unmanned rovers, and drones, all of which depend on advanced cognitive systems. Key trends include *artificial intelligence* (AI), autonomous driving, and human-robot interaction, with robots categorized by locomotion: stationary, land-based, air-based, and water-based. Reviews highlight land-based robots' terrain navigation and collaboration capabilities [2], while systematic reviews address modular systems, robustness, and communication challenges [3]. Recent advancements in *reinforcement learning* (RL) have also been significant and have notably enhanced mobile robot navigation and control. Work [4] employs *deep reinforcement learning* (DRL) for autonomous navigation and exploration in unknown environments, using *points of interest* (POI) for goal-driven exploration, which improves navigation efficiency and real-time mapping without pre-existing maps.

In [4], DRL guided a robot using POI, but struggled with explicit path optimization in complex environments. Our experiments presented in this work show that increased complexity or goals hidden behind walls often cause the robot to get stuck in local minima. Our work improves pathfinding by using waypoints, expanding POI, and preventing the robot from getting trapped behind obstacles. Environment complexity is measured by the number of obstacles and the goal's position, with hidden goals considered particularly challenging due to past difficulties in reaching them.

The upcoming chapters will cover: mathematical formulation (Sect. 2), simulation environment setup (Sect. 3), results and discussion (Sect. 4), and conclusion (Sect. 5).

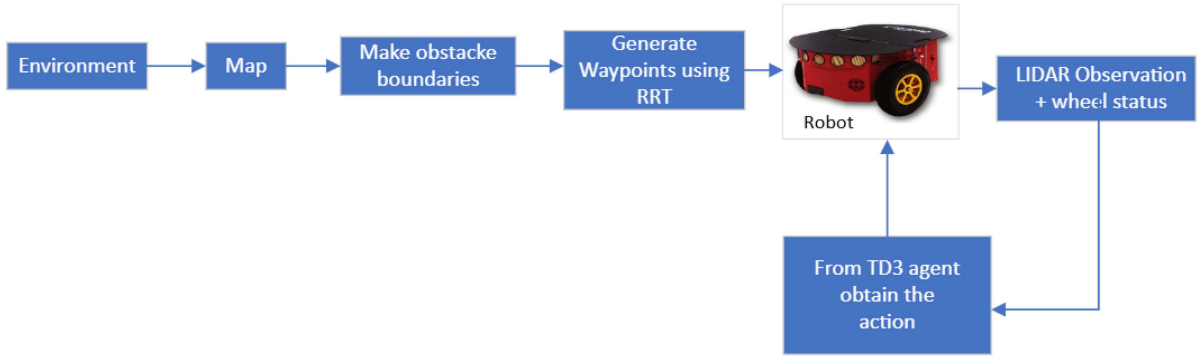

Fig. 1. Full pipeline of the algorithm for the mobile robot navigation.

2. Mathematical formulation

The proposed algorithm for navigating the mobile robot involves three key steps: pre-processing the environment map with the *rapidly exploring random tree* (RRT) algorithm to identify optimal waypoints, extracting points of interest (POIs) to guide the robot to the next optimal waypoint, and using a reinforcement learning algorithm to provide local guidance (see Fig. 1). The neural network receives waypoints in polar coordinates relative to the robot's location and heading, calculates actions based on sensor data, and navigates the robot towards the global goal.

2.1. Global navigation

For global navigation, the environment map is processed using the RRT path planning algorithm to find the optimal path from the start to the goal. The map uses black and white images, with black lines indicating walls and obstacles, and non-black areas as navigable (see Fig. 2). The algorithm generates the optimal path by refining edges and removing unnecessary nodes if needed. This preprocessing step is done at the beginning of navigation, and the resulting waypoints guide the robot's journey.

2.2. Points of interest (POI) navigation

After generating waypoints, the robot navigates to each of them using points of interest (POI) [4]. It avoids both dynamic and static obstacles, as well as dead ends, without prior knowledge of the environment. The robot identifies and stores POIs from its immediate surroundings using two methods for obtaining new POIs, detailed in [4].

If any of the POIs are found to be near an obstacle in subsequent steps, they are removed from memory.

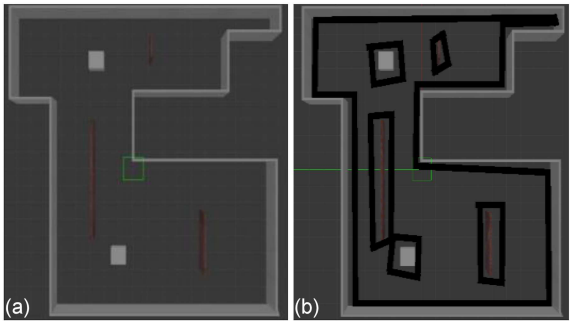

Fig. 2. Map (a) and map with boundary drawn on it (b).

The robot was equipped with two RpLidar laser sensors mounted at different heights, each with a maximum range of 10 m. Both sensors were carefully calibrated for their position and angle, with laser readings covering 180 degrees in front of the robot. The operation is as follows:

(i) POI is added if the difference in values between two sequential laser readings exceeds a threshold, indicating a navigable gap.

(ii) POI is placed if sequential laser readings return a non-numerical value, representing free space due to the laser sensors' maximum range.

POIs will not be obtained from laser readings in areas the robot has already visited. Furthermore, if a POI is chosen as a waypoint but cannot be reached within a certain number of steps, it is deleted, and a new waypoint is selected. As stipulated in [4], from available POI, the optimal waypoint at the time step is selected by using the *information-based distance limited exploration* (IDLE) evaluation method. The IDLE method evaluates the fitness of each candidate POI as

$$h\left(c_i\right) = \tanh\left(\frac{e^{d(p_t,c_i)^2}}{e^{(\frac{l_2}{l_2-l_1})^2}}\right) l_2 + d(c_i, g) + e^{I_{i,t}}. \quad (1)$$

The score h for each candidate POI, c, with index i is calculated as the sum of three components. The Euclidean distance component $d(p_t, c_i)$ between the robot's position p at time t and the candidate POI is represented as a hyperbolic tangent tanh function

$$\tanh\left(\frac{e^{d(p_t,c_i)^2}}{e^{\left(\frac{l_2}{l_2-l_1}\right)^2}}\right) l_2. \tag{2}$$

Here, e is Euler's number, and l_1 and l_2 are distance thresholds used to discount the score, determined based on the size of the DRL training environment area. The second component, $d(c_i, g)$, represents the Euclidean distance between the candidate points of interest and the global goal g. Finally, the map information score at time t is expressed as

$$e^{I_{i,t}}, \tag{3}$$

where I_i is calculated as

$$I_{i,t} = \frac{1}{k^2} \sum_{w=-\frac{k}{2}}^{\frac{k}{2}} \sum_{h=-\frac{k}{2}}^{\frac{k}{2}} C(x+w)(y+h). \tag{4}$$

In (4), k represents the size of the kernel used to calculate the information around the candidate points' coordinates x and y, while w and h denote the kernel's width and height, respectively. The POI with the lowest IDLE score from (1) is chosen as the optimal waypoint for local navigation.

2.3. Local navigation

The local planner algorithm navigates the robot using sensor input and a neural network trained with the DRL algorithm. Specifically, a *twin delayed deep deterministic* (TD3) policy gradient network trains the motion policy for continuous action spaces. The TD3 actor network takes input from laser readings within a 180-degree front range and the waypoint's polar coordinates. It features two fully connected layers with *rectified linear unit* (ReLU) activation, outputting two action parameters for linear and angular velocities, constrained within $(-1, 1)$ by a tanh function and scaled by the robot's maximum velocities. Backward movement is excluded, and the linear velocity is adjusted to remain positive

$$a = \left[v_{\max}\left(\frac{a_1+1}{2}\right), \omega_{max} a_2\right]. \tag{5}$$

The Q-value of the state-action pair Q_{sa} is evaluated by two critic networks with identical structures but delayed parameter updates to ensure divergence. Both networks take the state–action pair as input. The state is processed through a fully connected layer with ReLU activation, producing the output L_s. This output and the action are then fed into two *transformation fully connected* (TFC) layers, which are of the same size, before being combined, thus

$$L_c = L_s\, W_{\tau 1} + a\, W_{\tau_2} + b_{\tau_2}. \tag{6}$$

The *combined fully connected* (CFC) layer, L_c, uses weights W_{τ_1} and $W_{\tau 2}$ from layers τ_1 and τ_2, respectively, along with a bias $b_{\tau 2}$ from layer τ_2. ReLU activation is then applied to L_c, which connects to an output layer with one parameter representing the Q value. To prevent overestimation, the minimum value of Q from both critic networks is chosen as the final output.

In order to reward the agent, the following reward function employed [4]

$$r\left(s_t, r_t\right) = \begin{cases} r_g, & \text{if} D_t < \mu_D, \\ r_c, & \text{if collision}, \\ v - |\omega|, & \text{otherwise}. \end{cases} \tag{7}$$

The reward r of the state–action pair (S_t, a_t) at timestep t depends on three conditions. If the distance to the goal at the current timestep D_t is less than the threshold μ_D, a positive goal reward r_g is given. If a collision is detected, a negative collision reward r_c is applied. In the absence of these conditions, an immediate reward is based on the current linear velocity v and angular velocity ω. To guide the navigation policy towards the goal, a delayed attributed reward method is used as follows

$$r_{t-i} = r\left(s_{t-i}, a_{t-i}\right) + \frac{r_g}{i}, \quad \forall i = \{1, 2, 3, \ldots\}. \tag{8}$$

In this setup, n represents the number of preceding steps in which the rewards are adjusted. Consequently, the positive goal reward is not only assigned to the state–action pair at the goal but also distributed (diminishingly) over the preceding n steps. The network thus developed a local navigation policy that effectively reaches a local goal while avoiding obstacles using direct laser input data.

3. Result and discussion

The proposed algorithm has been tested in multiple environment setups (see Fig. 3).

3.1. System setup

Local navigation using DRL was trained on a computer with an NVIDIA GTX 4050 GPU, 16 GB RAM, and an Intel Core i5 CPU. The TD3 network trained in Gazebo with ROS commands for 800 episodes over 8 h, ending each episode upon reaching a goal, collision, or after 500 steps. The robot's max speeds were 0.5 m/s and 1 rad/s, with delayed rewards updating over the last 10 steps and parameter updates every 2 episodes. Training took place in a 10×10 m^2 environment with random obstacles and noise for generalization. A Pioneer P3-DX robot with RpLidar sensors was used, with waypoints/goals considered reached within 1 m. Although smaller than real industrial settings, this

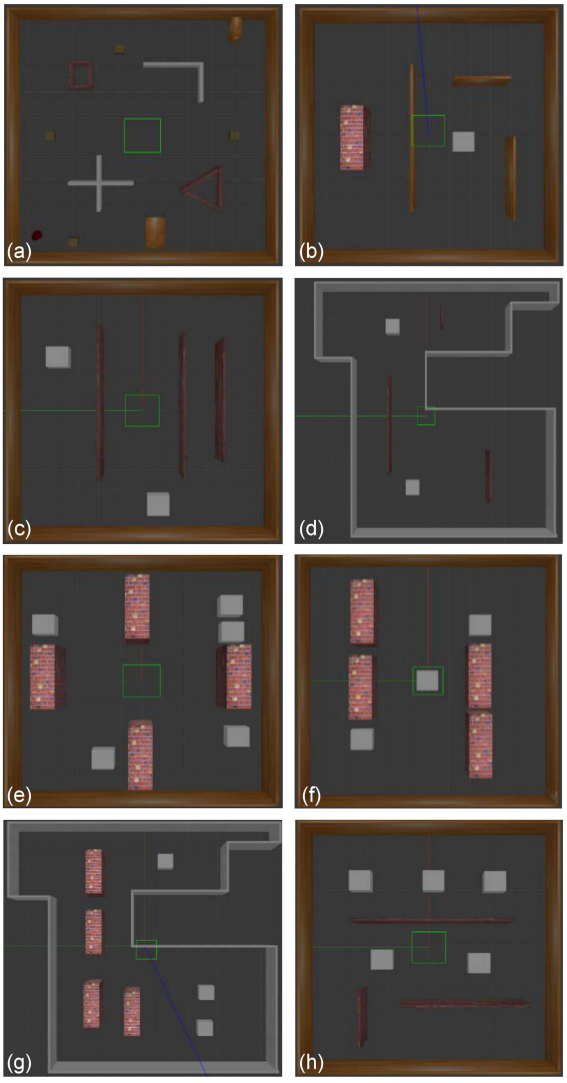

Fig. 3. Environments for simulating and testing the algorithm with different structure and obstacle setup.

experiment demonstrates the feasibility of the approach and can be extended to industrial differential drive robots.

3.2. Quantitative analysis

The algorithm was tested in various environments to evaluate its performance in guiding a robot to its goal. Each setup recorded the robot's success, comparing the proposed algorithm with the GDEA algorithm. The results in Table I show that while the GDEA algorithm performs well in simple environments, it struggles in complex environments, often getting stuck behind obstacles and incorrectly indicating goal achievement due to local minima. In contrast, the proposed algorithm excels in complex environments by using waypoints, ensuring consistent success.

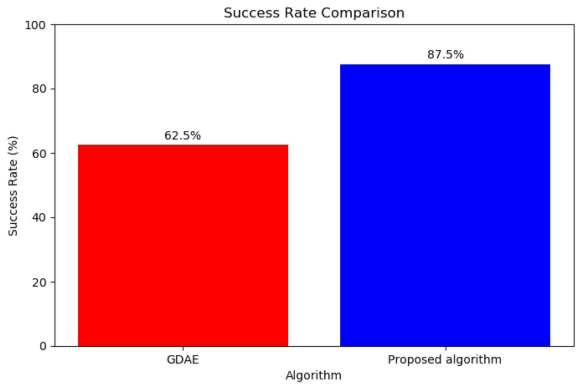

Fig. 4. Success rate comparison between the proposed algorithm and GDEA.

Table I compares performance across eight environments (env-1 to env-8) by measuring a time [s] and distance [m] to reach the goal. The proposed algorithm performs consistently well, taking 31 s and covering 9 m in env-1, and 17.31 s and 7.4 m in env-8, showing its robustness. In contrast, the GDEA algorithm performs well in simpler environments like env-1 (15 ss, 6.4 m) and env-4 (35 s, 13.67 m), but fails in more complex ones (e.g., env-2, env-3, env-5). For example, in env-7, GDEA took 96.76 s compared to 60.63 s for the proposed algorithm, which highlights its efficiency.

Figure 4 shows the success rates of the algorithms. Here, GDEA has a 62.5% success rate, indicating limitations in complex scenarios, while the proposed algorithm achieves an 87.5% success rate. This highlights its robustness, reliability, and effectiveness in navigating diverse environments, avoiding pitfalls like local minima that affect GDEA.

3.3. Qualitative analysis

3.3.1. Operate in the complex environment

The proposed algorithm navigates the robot effectively even in complex environments, while the GDEA algorithm struggles as complexity increases, often causing the robot to get stuck in local minima, mistaking them for the goal. Unlike GDEA, the proposed algorithm avoids obstacles, reliably guiding the robot to its target, showing superior robustness and performance in diverse scenarios.

3.3.2. Robustness of the proposed algorithm

As the environment complexity increases, the need for highly accurate algorithms grows, often leading to higher computational costs. The proposed algorithm improves navigation accuracy and robustness by dividing the complex environment

TABLE I

Quantitative comparison between the proposed algorithm and GDEA.

		env-1	env-2	env-3	env-4	env-5	env-6	env-7	env-8
Proposed algorithm	time [s]	31	79	38.52	33.24	57.0	23.36	60.63	17.31
	distance [m]	9	29	12.7	12.48	17.4	8.7	13	7.4
GDEA-algorithm	time [s]	15	failed	failed	35	failed	12.4	96.76	11.9
	distance [m]	9.5	failed	failed	13.67	failed	6.9	19	4.3

map into sub-optimal waypoints that guide the robot step-by-step to the destination. This approach breaks down complex tasks into manageable segments, ensuring reliable navigation without excessive computational demands, and preventing the robot from getting stuck in local minima or making significant errors.

3.3.3. Predictable navigation

The proposed algorithm improves navigation by preprocessing the environment map to select an optimal, predefined path, enhancing accuracy and reliability. In contrast, the GDEA algorithm relies on real-time decisions without a predefined path, which can be less reliable in complex environments and prone to local minima. By using waypoints and a planned route, the proposed algorithm ensures more consistent and successful navigation across diverse environments.

4. Conclusions

In conclusion, this paper presents a novel approach to autonomous vehicle navigation by integrating reinforcement learning (RL) with the rapidly exploring random tree (RRT) global path planning algorithm. Unlike previous methods, such as the one proposed by [4], which relied on points of interest (POI) for global navigation and lacked explicit optimization for pathfinding, the proposed method generates an optimal global path from the initial point to the final goal. By utilizing RRT for global planning, the method provides a clear, optimized route and avoids issues like local minima that plagued earlier approaches. The method starts by analysing an image map of the environment to identify obstacles, using this information to generate waypoints along the optimal path. These waypoints guide the RL-based local planner to navigate effectively while avoiding obstacles. This approach not only ensures a more efficient navigation process but also enhances the robot's ability to reach its goal despite environmental complexities.

The disadvantages of the proposed approach, which will be subject of the future research, are mainly related to its behaviour in simpler environment. While the GDEA algorithm may perform well by adapting to immediate obstacles, the proposed algorithm might generate unnecessary waypoints, leading the robot to cover more distance and follow sub-optimal paths. This can be improved with more efficient path planning algorithms like RRT*. On the other hand, in complex scenarios, the proposed method prevents the robot from getting stuck in local minima. Integrating RRT for global planning and RL for local navigation increases the robot's success rate in reaching its goal. Another topic that requires further exploration is the challenges emerging in applying the method within the complexity of a real industrial environment with industrial robots.

References

[1] R.K. Jain, B.J. Saikia, N.P. Rai, P.P. Ray, in: *2020 11th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE 2020.

[2] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, J.O. Strandhagen, *Ann. Oper. Res.* **308**, 125 (2022).

[3] F. Rubio, F. Valero, C. Llopis-Albert, *Int. J. Adv. Robot. Syst.* **16**, 1729881419839596 (2019).

[4] R. Cimurs, I.H. Suh, J.H. Lee, *IEEE Robot. Autom. Lett.* **7**, 730 (2022).