Autonomous intelligent navigation in unknown environments*

Bohdan Zhurakovskiy[1,*,†], Sergii Otrokh[1,†], Anatoliy Makarenko[1,†], Oleksandr Pliushch[2,†], and Serhij Mamchenko[3,†]

[1] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37 Peremogy ave., 03056 Kyiv, Ukraine
[2] Taras Shevchenko National University of Kyiv, 60 Volodymyrska str., 01601 Kyiv, Ukraine
[3] National University of Life and Environmental Sciences of Ukraine, 15 Heroiv Oborony str., 03041 Kyiv, Ukraine

Abstract

The research is devoted to the problem of autonomous navigation of a wheeled mobile robot in an unknown and partially observed environment using deep learning methods with reinforcement. The relevance of the work is due to the growing role of mobile robots in industrial, logistics and service applications, where traditional algorithms for trajectory planning and obstacle avoidance show limitations in the absence of a ready-made map, noisy sensor data and variable obstacle configuration. In such conditions, intelligent approaches that are able to independently form movement strategies based on experience of interaction with the environment acquire special importance. A software complex for autonomous navigation of a wheeled mobile robot has been created, which provides the possibility of testing and comparing different control strategies in simulation, further integration of the navigation module into real robotic platforms, as well as the use of the obtained solutions in scientific research.

Keywords

autonomous navigation, mobile robot, reinforcement learning, TD3 algorithm, GRU recurrent network

1. Introduction

The development of autonomous intelligent systems is one of the key directions of modern science and technology. Mobile robots are increasingly used in transport, production, household and rescue spheres, where an important factor in successful functioning is their ability to navigate independently without operator intervention. In such conditions, navigation systems are required not only to be accurate, but also adaptable, resistant to noise and able to make decisions in real time.

With the increasing complexity of environments and the emergence of dynamic obstacles, classical navigation algorithms (A*, Dijkstra, DWA, SLAM) increasingly demonstrate their limitations. They do not have self-learning mechanisms and do not take into account the impact of previous actions on the current state of the system. This makes them unsuitable for use in real, unpredictable conditions.

The current scientific paradigm of robotic navigation is shifting towards intelligent methods, in particular Deep Reinforcement Learning (DRL). This approach allows for the creation of systems that are able to independently form behavioral policies, optimizing them through interaction with the environment. However, most classical DRL algorithms consider each state in isolation, which limits their ability to analyze temporal dependencies, an important factor for navigation tasks.

That is why the development of architectures capable of combining reinforcement learning with memory and attention mechanisms is relevant. The integration of recurrent neural networks allows you to store information about the sequence of previous states, and the attention mechanism — to highlight the most significant elements of this sequence. This allows the agent to make decisions, taking into account the context of past actions, which is especially important in conditions of partial observability of the environment [1].

* CPITS 2026: Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2026, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
zhurakovskybiyu@tk.kpi.ua (B. Zhurakovskiy); 2411197@ukr.net (S. Otrokh); makarenkoa@ukr.net (A. Makarenko); opliusch@yahoo.com (O. Pliushch); s.mamchenko@nubip.edu.ua (S. Mamchenko)
0000-0003-3990-5205 (B. Zhurakovskiy); 0000-0001-9008-0902 (S. Otrokh); 0000-0002-4081-328X (A. Makarenko); 0000-0001-5310-0660 (O. Pliushch); 0009-0006-8743-5606 (S. Mamchenko)

Thus, the development and research of an autonomous navigation system for a mobile robot based on the modified GRU-Attention TD3 algorithm is a scientifically and practically significant task. The proposed approach provides [2]:

- increased stability of the learning process;
- improved generalization of the agent's experience;
- greater resistance to noise in sensory data;
- more efficient decision-making in dynamic environments.

The results of such research can be used in the creation of next-generation robotic platforms, autonomous vehicles, drones, and service robots, as well as in the development of simulation systems for training agents.

Therefore, the relevance of the topic is determined by the need for high-level intelligent navigation models that can combine the advantages of deep learning, memory and attention mechanisms to ensure reliable movement of mobile robots in complex, unknown and dynamic environments.

The subject area of research covers the process of autonomous navigation of a mobile robot in a complex, partially unknown environment. A navigation system of this type should ensure movement to a given goal, avoiding collisions with obstacles and adapting to changing conditions in real time [3].

A mobile robot is an autonomous platform capable of moving using wheeled or tracked drives, equipped with sensor systems to perceive the environment. The main sources of data in navigation systems are laser rangefinder (LIDAR), odometry, and target position information. Based on this information, the state of the environment is formed, which is used by the agent to make decisions.

A feature of the system under study is its operation in a partially observed environment, where the robot does not have a complete map of the space, but perceives it only within the range of the sensors. This creates conditions of uncertainty, under which the efficiency of navigation depends on the system's ability to generalize previous experience and predict the consequences of actions.

The environment is simulated using the Gazebo simulation platform integrated with the Robot Operating System (ROS) [4], which provides interaction between the physical model of the robot, sensors, and software modules. In this study, the Pioneer 3-DX robot, which has two drive wheels, differential control, and a front laser sensor, is used as the base model. This allows for accurate reproduction of realistic movement conditions in two-dimensional space.

The robot state is described by a vector of observations, which includes distances to obstacles, orientation relative to the target, linear and angular velocity. The agent's actions are determined by two parameters, namely linear and angular velocity, which form a continuous action space. The goal of the system is to reach the target position with a minimum number of collisions and an optimal trajectory.

From the perspective of intelligent control, the navigation task is considered as a Markov Decision Process (MDP), in which the agent learns to maximize the total reward by sequentially choosing actions in a dynamic environment [5]. Within this subject area, deep reinforcement learning methods are investigated, which combine neural network structures for processing sensory data and mechanisms for optimizing the agent's behavior.

Thus, the features of the subject area are that the navigation system must:

- operate under conditions of partial observability and uncertainty;
- process data from noisy sensors in real time;
- learn optimal actions based on interaction with the environment;
- ensure smooth, safe and efficient movement to the target.

It is these characteristics that determine the complexity of the task and justify the need to use deep reinforcement learning methods to create an intelligent navigation system for a mobile robot.

2. Description of the subject area

1.1. Limitations of classical RL algorithms in navigation

Despite the significant potential of reinforcement learning, classical RL algorithms have a number of limitations, which complicates their direct application in autonomous robot navigation tasks. Basic methods, such as table approaches (Q-learning, SARSA) or discretized policies, work well in

environments with a small number of states and actions, but are not suitable for high-dimensional and continuous spaces typical of robotics [6].

In mobile robot navigation systems, the state of the environment is described by dozens or hundreds of sensor measurements (laser distances, depth, odometry, relative target position), and the action consists of continuous parameters — linear and angular velocity. Such dynamic and continuous spaces make classical RL methods ineffective or even unsuitable.

Furthermore, robot learning occurs in complex and partially observable environments (POMDP), where sensors provide only limited or noisy information. Basic RL algorithms do not take into account the temporal context, considering each state independently, which leads to the loss of important historical information about previous observations. As a result, the policy becomes unstable and error-prone in situations where short-term changes in sensor data do not reflect the full dynamics of the environment.

Another significant limitation of classical RL methods is their high sensitivity to noise, which is inevitably present in real robotic systems. Instability in sensor measurements can drastically reduce the quality of decision-making, especially when the agent does not have an internal smoothing mechanism or state history modeling [7].

Therefore, classical RL methods need to be expanded, to models that are capable of processing continuous actions, large-scale sensor data, and temporal dynamics. Such capabilities are provided by modern actor-critical algorithms, on the basis of which DDPG, TD3, and their modifications were developed.

1.2. Evolution of RL methods for continuous actions in robot navigation

In autonomous navigation tasks of mobile robots, control is carried out in a continuous space of action, where linear and angular velocity can take on an infinite number of values. This makes classical reinforcement learning methods, which rely on discrete actions or tabular Q-functions, unsuitable for real robotics scenarios. The advent of neural networks allowed us to move from discrete models to approaches that approximate policy and value functions in complex environments, which became the basis for the development of modern DRL algorithms. One of the key directions was the combination of gradient methods for policy optimization with policy quality assessment, which is how the Actor-Critic architecture emerged. In it, two components perform different but interrelated functions:

- actor a neural network that generates an action *at* based on the current state *st*;
- critic a neural network that evaluates the quality of the selected action by approximating the value function.

The critic returns a so-called TD-error, which acts as a correction signal for updating the actor's policy. This interaction provides more robust learning compared to pure policy-gradient methods and allows for efficient operation in high-dimensional environments with noisy or partially observed data [3].

The general structure of the interaction between the actor, the critic, and the environment is shown in Figure 1. It demonstrates a cyclic process: the environment returns state and reward, the critic evaluates the quality of the action, and the actor updates the policy taking into account the received evaluations. It is this architecture that has become the foundation of modern reinforcement learning algorithms for continuous control, including methods used in robotics [3].

Further development of deep Actor-Critic methods led to the emergence of algorithms capable of working with complex, dynamic, and incompletely observable environments. One of the first such solutions was Deep Deterministic Policy Gradient, which combined deterministic policy with convolutional neural networks. However, practical use of DDPG revealed a number of problems, namely high sensitivity to sensor noise, a tendency to overestimate Q-function values, and learning instability when interacting with the real environment [8].

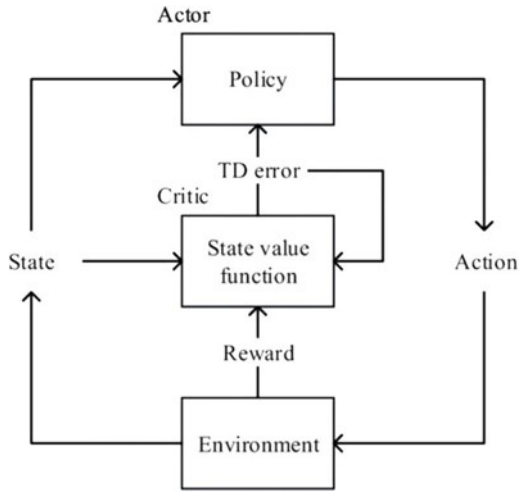

Figure 1: The Actor-Critic architecture scheme has been adjusted [3]

The need to overcome these limitations has led to the emergence of more robust methods, in particular Twin Delayed DDPG (TD3). This algorithm retains the general Actor-Critic structure, but supplements it with a number of modifications aimed at increasing the stability and reliability of learning:

- using two independent critics instead of one;
- smoothing the policy by adding noise to the target actions;
- deferred updating of the actor parameters.

The combination of these mechanisms makes TD3 one of the most effective algorithms for navigation tasks of autonomous mobile platforms, where decisions are made in real time and under conditions of incomplete information.

Thus, the development of reinforcement learning methods for continuous control has gone from classical tabular models to deep Actor-Critic algorithms, and their evolution naturally led to the creation of TD3 — one of the most stable approaches that meets the requirements of robotics applications. The following section 2.3 will consider in detail the structure, principle of operation and improvement of the TD3 algorithm, as well as its modifications designed to increase adaptability and the ability to work with time dependencies.

1.3. TD3 algorithm and its modifications

Modern intelligent reinforcement learning methods are actively used in mobile robot control tasks, but their effectiveness largely depends on the ability to work with continuous actions and high-dimensional states. Among such methods, actor-critical algorithms have become particularly widespread, which allow combining the advantages of political and value-based approaches to agent training. One of the most successful representatives of this group is the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, which provides high stability, noise resistance, and improved convergence compared to previous methods [9].

This section discusses the basic principles of TD3, as well as its modifications aimed at taking into account the temporal dynamics of the environment — in particular, GRU-TD3 and GRU-AttentionTD3. A comparison of these options allows us to assess the impact of short-term memory and the attention mechanism on the quality of mobile robot navigation.

2.3.1 Algorithm Twin Delayed Deep Deterministic Policy Gradient (TD3)

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm is one of the most effective deep reinforcement learning methods for continuous-space tasks, including mobile robot navigation [10]. TD3 was proposed as an improvement of the Deep Deterministic Policy Gradient (DDPG) algorithm, inheriting its actor-critical structure but supplementing it with mechanisms that significantly increase the stability and convergence speed of learning.

The main idea of TD3 is to separate roles between two components: an actor and a critic. The actor network forms an action based on the current state of the robot, while the critical networks evaluate the quality of this action by approximating the value function $Q(s,a)$. However, unlike

DDPG, TD3 uses two independent critics, which reduces the problem of systematic re-estimation of the Q-function characteristic of DDPG.

The second key improvement is Target Policy Smoothing. When calculating the target values, the actions generated by the target actor are supplemented with a small amount of noise. This makes the target policy less sensitive to local errors of the critic and contributes to a more stable learning process.

The third component is Delayed Policy Update. Unlike critics, the actor is not updated after every learning step, but with a lower frequency. This approach allows critics to form a more stable estimate of the value function, on the basis of which the actor adjusts its behavior.

Another important part of TD3 is the use of target networks (actor target and critic target) that slowly converge to the main networks using a soft-update mechanism. This reduces the variability of estimates, stabilizes gradients, and prevents policy divergence [11].

2.3.2 TD3 modifications: GRU-TD3 and Attention-TD3

Although the TD3 algorithm provides high learning stability and efficiency for tasks with continuous action space, its key limitation remains the lack of mechanisms for handling time dependencies. In mobile robot navigation, the current state of the environment is only a snapshot that does not contain information about the robot's previous positions, changing obstacles, or motion dynamics. Systems operating in partially observed or noisy environments require the ability to take into account the observation history to form a more accurate and stable control policy [12].

To address this shortcoming, the TD3 algorithm integrates a GRU (GatedRecurrentUnit) recurrent layer, which enables the model to process sequences of input states, rather than just their individual values. Thanks to this, GRU-TD3 is able to preserve short-term dependencies between states and build actions taking into account the context of the last steps of the trajectory. This allows the robot to respond more consistently, smooth out sensor noise, and compensate for information loss in cases of partial observability (POMDP). However, even recurrent networks cannot always adequately determine which moments of the sequence are most important for decision-making. In complex environments with numerous dynamic obstacles or sensor noise, the same weighting of all elements of the sequence can degrade the quality of state assessment. Therefore, the next step in development is to modify GRU-TD3 with the addition of an attention mechanism (AttentionMechanism), which learns to automatically highlight the most informative elements of the state history. The GRU-AttentionTD3 architecture combines the advantages of recurrent memory and selective attention focusing, which allows for effective ignoring of irrelevant or noisy data, reducing estimation errors, and increasing policy stability [13].

As a result, the modifications GRU-TD3 and GRU-AttentionTD3 extend the capabilities of the basic TD3 algorithm, making it significantly more suitable for autonomous navigation tasks in complex, unpredictable, partially observed, and dynamic environments. A generalized comparison of the characteristics of the three architectures is given in Table 1.

As can be seen from Table 1, each modification of the TD3 algorithm is aimed at overcoming key shortcomings of the basic architecture. The addition of a recurrent GRU layer allows the model to take into account temporal dynamics and process sequences of sensor observations, which significantly improves the quality of control in partially observed environments. Further expansion of the architecture due to the attention mechanism provides selective focusing on the most informative fragments of the state history, which reduces the influence of noise and increases the stability of the policy. Thus, GRU-Attention TD3 demonstrates the best efficiency among the three considered approaches and is the most promising method for building an intelligent navigation system for a mobile robot.

1.4. Features of navigation in a partially observed environment

The task of autonomous navigation of mobile robots in real conditions is much more complex than in classical simulated or laboratory scenarios. Unlike idealized models, the real environment is rarely completely known, static or deterministic [14].

The robot operates in conditions where information about the surrounding space is only partially available, often distorted by noise, and the space itself can change over time. All this

creates a complex combination of factors that limit the effectiveness of traditional machine learning and reinforcement learning algorithms. The features of partial observability, the presence of noise, the ambiguity of sensory data and the unpredictability of dynamic obstacles determine the need for more flexible approaches to modeling behavior. It is these aspects that primarily justify the use of modified TD3 variants with memory and attention mechanisms. The key properties of the environments of robotic systems that affect the choice of algorithms are discussed below.

Table 1. Order TD3, GRU-TD3 and GRU-Attention TD3

Characteristic	TD3	GRU-TD3	GRU-Attention TD3
Processing states	Current status only	Sequence of states	Sequence of states with key fragments highlighted
Memory availability	Missing	Yes (recurrent memory GRU)	Yes (GRU + attention mechanism)
Sensor noise immunity	Medium	Increased	High
Working in conditions POMDP	Limited	The best	The best
Taking into account time dynamics	None	There are	Available with selective focus
Stability of learning	High	Higher than in TD3	Highest among the three models
Computational complexity	Low	Medium	High
Suitability for real-world applications	For simple environments	For dynamic environments	For challenging and noisy environments
Expected navigation quality	Basic	Improved	The best

2.4.1 Partial observability and the need for modeling time dependencies

In real navigation, the robot does not have access to the full configuration of the environment. Its perception is limited to sensor data, such as:

- a laser rangefinder with a limited viewing angle;
- an odometer that accumulates error;
- a depth camera with a limited viewing range;
- locators and other sensor systems.

These signals form a partial and imperfect representation of the world. A significant part of the information remains hidden, and the current state is not a complete Markov representation [15]. In many situations, understanding what is happening in the current step requires knowledge of previous states. For example:

- the robot could have already registered an obstacle that temporarily disappears from view in the next step;
- the change in the configuration of the environment occurs gradually, and individual states do not carry enough information;
- sensory noise can mask important events, and only a sequence of observations allows them to be interpreted correctly.

In such conditions, the basic TD3 is insufficient, since it processes each state in isolation, without preserving the context of previous observations. This can lead to:

- loss of critical information;
- erroneous decisions in complex configurations;
- unstable behavior in confined spaces;
- chaotic trajectories during dynamic scene changes.

The GRU-TD3 modification, which supplements TD3 with a recurrent neural network, allows storing and processing a short-term history of states. Thus, the model begins to focus not only on the current measurement, but on generalized information from several previous steps, which significantly improves the quality of behavior in POMDP environments.

2.4.2 Noise, sensory ambiguity, and the importance of selective attention

Sensors of mobile robots are a source of ambiguous, often indirect signals. Noise can be systematic (caused by the properties of the device) or random (due to laser reflection, interference, air movement, etc.). In robotic systems, such noise can lead to sharp jumps in data or to the fact that the obstacle temporarily "disappears" or "appears" in the perception of the robot [16].

Classical DRL algorithms, including TD3, do not distinguish between significant and insignificant parts of the data — all measurements enter the network with the same weight. This makes the policy sensitive to random fluctuations and can make convergence difficult.

The Attention mechanism integrated into GRU-Attention TD3 performs the function of sorting information. In particular, it allows the model to:

- determine which states in the history are truly important;
- highlight trends and patterns from past observations;
- suppress the influence of noise data that does not carry useful information;
- focus computational resources on key aspects of environmental behavior.

As a result, the agent is able to not only memorize history, but also prioritize its elements, forming much more informative implicit representations. This enhances TD3's stability in complex navigation tasks and improves its generalization ability.

2.4.3. Conclusions and justification for the choice of models for comparative analysis

The combination of the factors described above — partial observability, data noise, and local measurement ambiguity — makes work in real robotic environments particularly challenging. The basic version of TD3, despite its significant advantages over DDPG (thanks to Twin Critics, Policy Smoothing, and Delayed Updates), is unable to fully compensate for these limitations.

That is why the following part of the work is devoted to a comprehensive experimental comparison of three approaches:

- TD3 main model, which acts as a baseline for evaluation;
- GRU-TD3 model, which overcomes the problem of partial observability through the use of temporal memory mechanisms;
- GRU-AttentionTD3 modification, which additionally applies selective attention to increase noise resistance and to better model dynamic changes in the environment. Such a comparison allows not only to assess the advantages of each architecture, but also to identify their limitations, establish the relationship between network structure and navigation quality, and justify the choice of the model that is best suited for autonomous operation in complex, dynamic, and stochastic conditions.

2. Implementing and training an agent in a simulation environment

3.1 General architecture of the software system

In the previous parts of the study, the requirements for the intelligent navigation system of a mobile robot were formulated and the choice of reinforcement learning methods and hardware and software for their implementation was justified. In this part, these theoretical provisions are transformed into specific software modules that interact with each other in a single simulation environment ROS-Gazebo [17].

A general idea of the relationships between system components is provided by the structural diagram of the architecture.

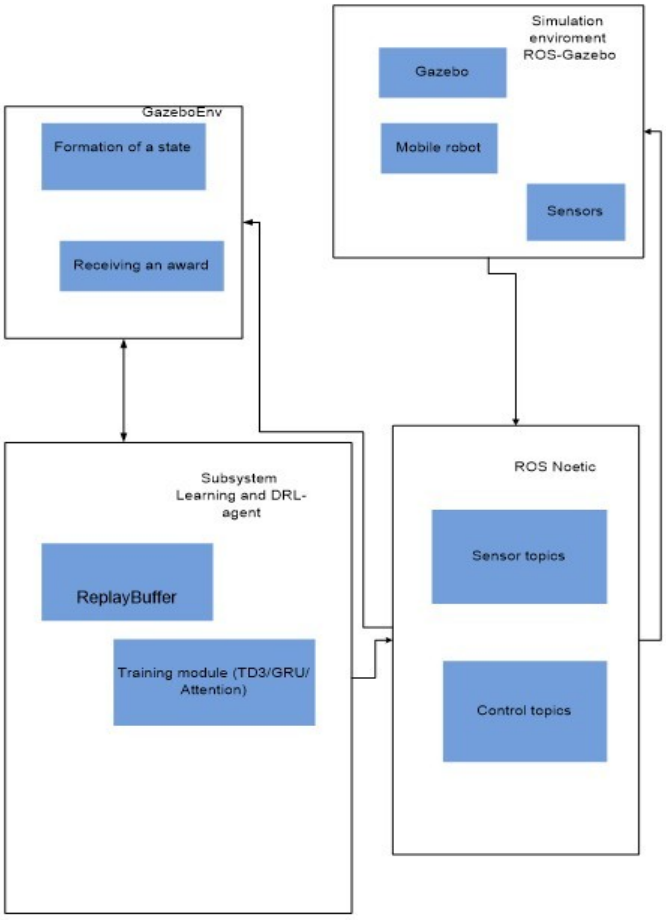

Figure 2: System block diagram

This diagram highlights four main subsystems that form a single intelligent circuit:

- ROS–Gazebo simulation environment;
- ROSNoetic data exchange platform;
- GazeboEnv environment module;
- training subsystem and DRL agent with experience buffer.

The "ROS-Gazebo Simulation Environment" block contains a model of the world with obstacles, a mobile robot and its sensors (laser rangefinder, LiDAR, odometry). The robot moves in the simulated space, and the sensors form a flow of measurements, which in the form of messages are sent to the ROS Noetic platform through the corresponding topics /scan, /velodyne_points, /odom.

The "ROS Noetic" block in the diagram reflects the role of ROS as the system's single communication environment. Through it, sensor data is transmitted to other nodes, and in the reverse direction, commands for controlling the robot's movement are received via the /r1/cmd_vel topic. This approach ensures a weak dependence between the simulation and the training subsystem and creates the prerequisites for further transfer of the algorithm to a real robot without significant code changes.

The next element of the architecture is the GazeboEnv environment module. This module implements the transformation of raw sensor measurements into a compact state vector (state), which includes aggregated laser scan values, distance and angle to the target, as well as current robot motion parameters. The same module calculates the reward value (reward), which takes into account the approach to the target, the presence or absence of collisions, and exceeding the limit episode duration. Thus, GazeboEnv is an intermediate link between physical simulation and a formalized representation of the environment in terms of a reinforcement learning problem [17].

The lower part of the structural diagram, in the block "Training subsystem and DRL agent", shows the operation of neural network models. The state vector from GazeboEnv is fed to the Actor network, which generates an action in the form of linear and angular velocity (v,w). This

action, on the one hand, is transmitted to ROS through the control topic /r1/cmd_vel and determines the movement of the robot in the Gazebo environment, and on the other hand, together with the corresponding state and reward, is recorded in the experience buffer (Replay Buffer) [18].

A separate block shows the learning module that implements the TD3 algorithm taking into account the recurrent GRU components and the attention mechanism [19]. The module periodically selects transition samples from the experience buffer and updates the parameters of the Actor, Critic 1 and Critic 2 networks.

In conclusion, the operation of the entire system can be described by the following sequence of steps:

- the simulation environment generates sensor data during the robot's movement;
- this data is sent via ROS to the GazeboEnv module, where it is converted into a state vector and reward value;
- the DRL agent, having received the state, selects an action and transmits it to ROS as a movement command;
- information about the state, action and reward is stored in the experience buffer;
- the training module, using the accumulated experience, periodically updates the parameters of the neural networks.

The described architecture is the basis for further research, where simulation environments, the structure of the state vector, the reward function and the features of the implementation of agent training algorithms are considered in more detail.

3.2 Mobile robot navigation simulation environments

For training and further testing of the agent, four simulation worlds were created in the Gazebo environment. All of them are rectangular areas divided into a square grid with a step of 1 m. In each world, the location of stationary obstacles is given, while the initial position of the robot and the coordinates of the target are partially or completely randomized.

The training world has a size of 10×10 m and is used at the training stage of all three networks. The obstacles are placed in such a way as to create several typical situations: direct movement to the target, bypassing individual objects, passing narrow corridors [20]. At the beginning of each episode, the initial position of the robot and the target point are randomly selected, and the coordinates of the obstacles are slightly changed if necessary. Such randomization increases the variety of situations and prevents retraining on one fixed route. A schematic representation of the training world is shown in Figure 3.

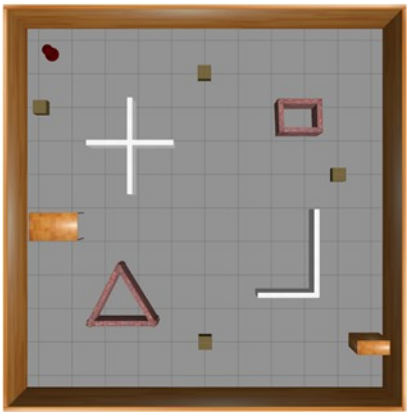

Figure3: Primary light at the middle Gazebo

Test world I also has a size of 10×10 m, but differs from the training world in a different configuration of obstacles. In this world, the agent's ability to generalize the navigation policy to a new environment with similar scales is assessed. For each episode, the positions of the robot, the target, and the obstacles are randomly assigned within the allowed area. The appearance of test world I is shown in Figure 4.

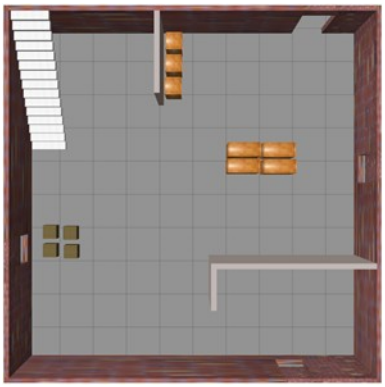

Figure 4: Test light I size 10x10 m

Test world II has a larger size — 20×20 m — and an increased number of obstacles. Typically, the initial position of the robot and the target point are fixed near opposite corners of the scene, so the robot has to cover a significant distance, bypassing obstacles. This environment is used to test how well the learned policy retains its effectiveness when changing the task scale and trajectory length. A schematic representation of test world II is shown in Figure 5.

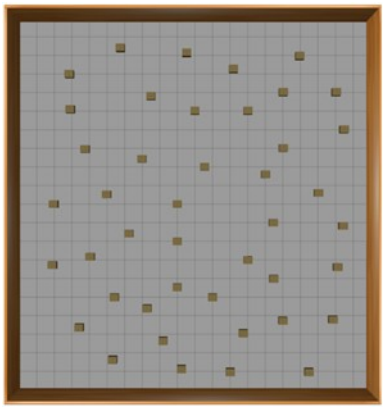

Figure 5: Test light II size 20x20 m

Test world III also has a size of 20×20 m, but is characterized by a different arrangement of obstacles. The structure of the environment is complicated: there are narrow passages, “pockets” in which the robot can get stuck, and alternative paths to the goal. It is in this world that the agent’s ability to choose trajectories close to the shortest and avoid local traps is tested. The appearance of test world III is shown in Figure 6.

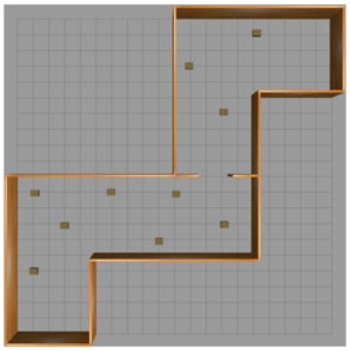

Figure 6: Test light III size 20x20 m

The same kinematic model of a differentially controlled robot is used in all worlds. The linear velocity limit is from 0 to 1 m/s, the angular velocity limit is from — 1 to 1 rad/s. The maximum measurement range of the laser scanner is 10 m, the simulation frequency is 30 Hz. Network training is performed only in the training world, while testing is performed in three other environments, which allows us to evaluate the generalization ability of the algorithm.

3.3 State vector formation and reinforcement learning structure of an agent

The navigation process of a mobile robot can be represented as a Markov decision process [21]. At each time step *t*, the agent observes the current state, performs an action *at*, receives a scalar reward *rt*, and transitions to a new state *st*+1. The corresponding sequence of transitions can be written as formula (3.1):

$$\left(s_t, a_t, r_t, s_{(t+1)}\right), \tag{3.1}$$

where s_t is a state of the environment at a time step t; a_t is an agent action performed in the state s_t; r_t is a reward received by an agent for performing an action a_t; s_{t+1} is a new state of the environment after applying the action a_t.

According to the scheme, the simulation environment, based on the current position of the mobile robot, the configuration of obstacles and the parameters of the movement, forms a state vector *s*. This vector is sent to the agent (mobile robot), which, using the current policy, chooses an action *a*. The action vector is transmitted to the simulation environment via ROS and is implemented in the form of linear and angular commands speed. As a result of the action, the environment updates the robot's position, forms a new state s_{t+1} and calculates the reward value r_t, which is returned to the agent. Thus, the closed loop "state-action-reward-new-state" is implemented, which is the basis of the reinforcement learning process.

The state vector in the developed system has dimension 24 and includes the following components:

- 20 measurements of the distance $r_1, r_2, \ldots, r_{20}$ from the laser sensor, evenly distributed in the angular sector in front of the robot;
- the distance to the target *d*, calculated using the Euclidean metric between the current
- position of the robot and the target point;
- the angle to the target φ — the difference between the current direction of the robot and the direction to the target;
- the linear velocity v of the robot;
- the angular velocity ω.

In compact form, the state vector is given by the formula (3.2).

$$s=[r_1, r_2, \ldots, r_{20}, d, \varphi, v, \omega]^T, \tag{3.2}$$

where r_i is a measured distance in i^{th} angular sector of the laser scanner, i = 1, ..., 20; d is a distance to the target point; φ is an angle to the target; v is a current linear velocity of the mobile robot; ω is a current angular velocity of the mobile robot.

Before being fed to the input of the neural network, all components of the state vector are normalized to the range [0;1], which increases the stability and convergence speed of the learning process.

The action vector has dimension 2 and describes the continuous control effects on the mobile robot according to formula (3.3):

$$a=[v, \omega]^T, \tag{3.3}$$

where v is the desired linear speed at which the robot should move; ω is the desired angular speed of rotation of the robot.

In such a reinforcement learning setting, the agent gradually learns to choose actions a_t based on current states s_t, that maximize the total expected reward in the mobile robot navigation task [22].

3.4 Reward function

The design of the reward function plays a key role in reinforcement learning, as it determines what behavior the agent considers desirable. The developed system uses a reward function built on the logic of the basic scientific work on which the implementation is based. The main idea is to reward

the robot for approaching the goal and moving safely, and severely penalize for collisions with obstacles or inefficient trajectories [23]. In general, the reward at time step t is given by a function in the form of the formula (3.4):

$$r_t = \begin{cases} r_g, d_t < \eta \\ r_c, collision \\ v_t - |w_t|, \end{cases} \quad (3.4)$$

where r_t is the reward value at time step t; d_t is the current distance to the target point; η is the threshold distance to the target at which the robot is considered to have reached the target position; r_g is the positive reward for successfully reaching the target; r_c is the penalty for colliding with an obstacle; v_t is the linear velocity of the mobile robot at step t; ω_t is the angular velocity of the mobile robot at step t.

The implemented system uses the following specific parameter values, as presented in formula (3.5):

$$r_t = \begin{cases} 80, d_t \le 0{,}1, \\ -100, collision \\ v_t - |w_t| \end{cases} \quad (3.5)$$

where η = 0,1 m is the robot is considered to have reached the goal if it has approached it closer than 10 cm; r_g = 80 is a large positive reward that fixes a successful trajectory; r_c = −100 is a significant penalty for a collision, which forces the agent to actively avoid obstacles.

In the "normal" mode, when neither the goal was reached nor the collision occurred, the reward is determined by the expression $vt - |\omega t|$. This means that the robot is encouraged to move with a higher linear speed and with fewer sharp turns. Thus, the agent learns to choose trajectories that combine speed and smoothness of movement.

In the software implementation, this logic is expressed as a conditional operator, which at each step checks whether the goal has been achieved, whether there is a collision, and whether the maximum episode time has elapsed, and changes the value of *rt* accordingly. This allows you to modify the reward parameters if necessary when conducting additional experiments.

3.5 Implementation of TD3, GRU-TD3 and GRU-Attention TD3 algorithms

The autonomous navigation agent is trained using three variants of the actor-critical algorithm in a continuous action space: the basic TD3, two extended GRU-TD3, and GRU-Attention TD3. All three approaches use the same input/output structure (state of dimension 24, action of dimension 2) and the same reward function, which ensures the correctness of their comparison [24]. In the main variant of TD3, the actor-network consists of several sequential fully connected layers with ReLU activation. The input is a state vector s, and the output is an action vector $a = (v, \omega)$, o limited to the permissible range of mobile robot speeds. The critical part of the algorithm consists of two independent networks Critic 1 and Critic 2, which calculate the estimate of the action-state function $Q(s,a)$. n the training process, a minimum of these two estimates is used to form the target value, which allows reducing the overestimation of actions characteristic of DDPG and making policy updates more stable [25].

In the GRU-TD3 variant, a recurrent block based on the Gated Recurrent Unit (GRU) is added to the architecture of the actor and critical networks. The GRU input is fed with a sequence of the last L states ($st–L+1,…,st$). This allows the network to take into account the robot's motion history and previous sensor measurements, which is important in situations where the current observation itself may be ambiguous or incomplete. The output of the GRU is combined with fully connected layers, forming a summary representation, on the basis of which both the action a and the value of the function $Q(s,a)$ are calculated.

In the GRU-Attention TD3 variant, the GRU-TD3 architecture is additionally equipped with an attention mechanism that is applied to the sequence of GRU hidden states. For each step in the

history, an importance coefficient is calculated, after which a weighted sum of the hidden states is formed. Thanks to this, the network can focus on the most informative moments of the trajectory and reduce the influence of noisy or poorly informative observations. This is especially useful in complex environments with a large number of obstacles and changing routes.

During the training phase, all three networks interact with the simulation environment according to a single scheme. At each step, the actor network receives the current state vector, generates an action, which is transmitted to Gazebo via ROS as a speed command, after which the environment returns a new state and reward value. Transitions are formed that accumulate in the experience buffer; based on samples from this buffer, gradient updates of the parameters of the actor and critical networks are periodically performed. Thus, a closed loop "state-action-reward-parameter update" is implemented, within which the navigation policy is gradually improved [26].

Figure 7 shows an example of the visualization of the agent training process in the Gazebo/RViz environment. The right part of the window displays the projection of the scene in the form of a grid and obstacles; the colored segments correspond to the measurements of the laser scanner, and the purple broken line shows the trajectory of the robot's movement during several training episodes. The lower part of the window shows the image from the robot's front camera. Such visualization allows you to monitor the agent's behavior in real time, check the correct operation of the sensors, and timely detect possible errors in the environment or algorithm settings [27].

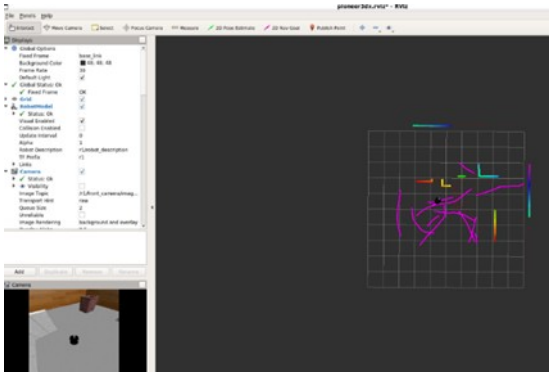

Figure 7: Visualization of the process of starting the navigation agent of a mobile robot in the middle of RViz

3. Testing and evaluating system performance

3.1. Organization of training and test experiments

This section presents the organization of experiments, within which the effectiveness of the proposed mobile robot navigation system was tested. Three implemented networks TD3, GRU-TD3 and GRU-Attention TD3 are compared.

All three networks were trained in a training world of 10×10 m. At the beginning of each episode, the following were randomly assigned:

- the initial position of the robot;
- the coordinates of the target point;
- if necessary, the configuration of some obstacles within the permissible area.

Such randomization provides a variety of situations that the agent encounters and reduces the risk of overtraining on individual trajectories [30].

In the process of interaction with the environment, for each step of the episode, information was stored about the current state, action, received reward, new state, and sign of episode completion (goal achievement, collision, timeout). All these transitions were accumulated in an experience buffer with a fixed maximum capacity: after its achievement, the oldest records were replaced by new ones. To update the network parameters, mini-samples of a fixed size were randomly selected from the buffer, based on which the actor and critical networks were trained and their target copies were "softly" updated [28].

Following each training epoch, a brief evaluation was conducted within the training environment without additional action noise. This involved a series of test episodes, from which the average cumulative return per episode (epoch reward) was calculated. These values serve as the basis for constructing convergence plots [29].

Upon completion of the training phase, the policies of the three networks were evaluated across three simulation environments: Test World I, Test World II, and Test World III. For each environment, a series of test episodes was performed without stochastic noise added to the actions, enabling an accurate comparison of the quality of the developed navigation strategies.

3.2. Navigation quality assessment metrics

To compare the three implemented networks TD3, GRU-TD3 and GRU-Attention TD3, a set of quantitative indicators characterizing the quality of mobile robot navigation was determined. The main metrics are summarized in Table 2.

In the following analysis, these metrics play different roles. In Section 4.3, the average reward per epoch $\overline{R}_{\text{epoch}}$ використовується як основний is used as the main indicator of convergence and stability of learning. In Sections 4.4.1-4.4.3, where the behavior of the agent in test simulation environments is considered, the main attention is paid to the total reward Rep, the number of steps Nstep and the number of collisions Ncoll, which allows us to assess how efficiently and safely the robot performs the navigation task [31].

Table 2. Metrics for assessing navigation performance

Indicator	Marking	Description
Total reward per episode	R_{ep}	The sum of all instantaneous rewards within a single episode. It is an integral assessment of the quality of the trajectory from the point of view of the reinforcement learning problem.
Average reward per era	$\overline{R}_{\text{epoch}}$	Average value R_{ep} for several test episodes after each training epoch is complete. Used to plot convergence graphs in figures 8-11.
Number of steps to achieving the goal	N_{step}	The number of simulation steps from the initial state until the goal is reached. Characterizes the length and efficiency of the constructed trajectory: a smaller value corresponds to a shorter and faster route.
Number of collisions	N_{coll}	The number of episodes that ended in collisions with obstacles. It is an indicator of navigation safety and the sustainability of the established policy.

3.3. Agent learning outcomes in the learning world

The learning dynamics of the three networks in the training environment are illustrated in Figures 8-11, constructed based on the results of the experiments. Each graph shows the change in the total reward per episode over the training epochs and its smoothed value (moving average).

Figure 8 shows the epoch reward curve for the basic TD3 network. It can be seen that at the initial epochs the reward values are low and fluctuate strongly due to the random choice of actions and a large number of collisions. During the training process, the average reward gradually increases and reaches a moderate but relatively stable level. The moving average demonstrates a clear upward trend, which indicates the formation of a workable, although not optimal, navigation policy.

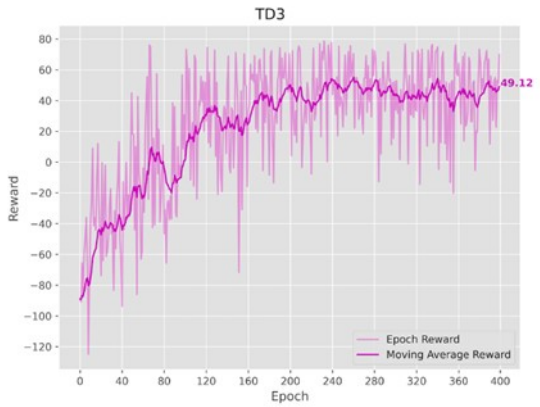

Figure 8: Epochal wine towns for the basic range TD3 in the initial middle

Figure 9 shows the results for the GRU-TD3 network. In the first epochs, it learns faster than the base TD3: the reward curve quickly rises from negative values to the positive region. At the same time, the graph is more "noisy", with pronounced fluctuations. In later epochs, there is an alternation of periods of improvement and deterioration, and the final level of the moving average turns out to be lower than in the base TD3. This indicates some instability of the policy, despite the use of a recurrent layer.

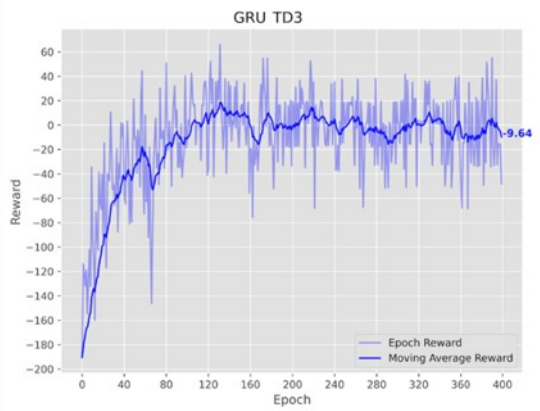

Figure 9: Epochal wine city for the GRU-TD3 border in the initial middle

Figure 10 shows the epoch reward curve for the GRU-Attention TD3 network. This network shows the fastest growth of the indicator already in the first tens of epochs: the curve quickly passes into the region of high positive values. After entering the stationary mode, the moving average level significantly exceeds the similar values for TD3 and GRU-TD3, and the fluctuations around this level are relatively small. This indicates that the combination of GRU with the attention mechanism allows for more effective use of the temporal context and muffles noisy observations.

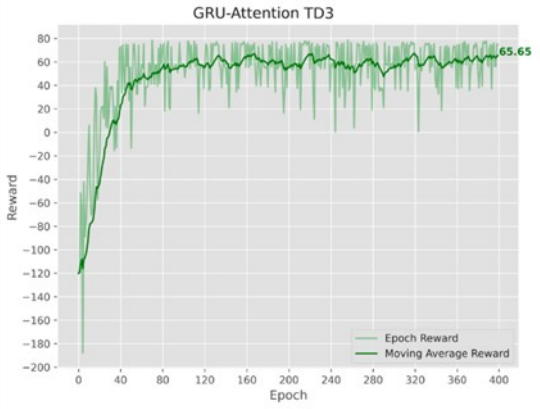

Figure 10: A milestone award for the GRU-Attention TD3 network in a learning environment

For a visual comparison, Figure 11 shows the moving average of the epoch reward for all three networks simultaneously. The curve for GRU-Attention TD3 is located higher than for TD3 and GRU-TD3 for almost the entire training interval, which confirms its superiority in terms of convergence speed and achieved level of policy quality [32]. The baseline TD3 occupies an intermediate position, while GRU-TD3 demonstrates the lowest average reward values and the highest instability [33].

Figure 11: Comparison of moving average epoch reward for TD3, GRU-TD3 and GRU-Attention TD3 networks

3.4. Test results in simulation environments

After training in the Training World environment, the parameters of all three networks were fixed, and further experiments were conducted in three test scenarios: Test Environment I, Test Environment II, and Test Environment III. For each scenario, the networks were run with the same initial conditions, and comparisons were made on average reward, trajectory length, and number of collisions.

3.4.1. Results in the test environment I

The test environment I has a size of 10×10 m and differs from the training one in a different configuration of obstacles. For each of the three networks, 100 testing episodes were performed, after which the average reward and the number of collisions were calculated.

Figure 12 shows the average reward for every ten episodes. It can be seen that the GRU-Attention TD3 network maintains the highest values throughout the test: its average reward is about 64.24. The basic TD3 network demonstrates a slightly worse, but still consistently positive result — about 52.91. In contrast, GRU-TD3 lags significantly behind: the average reward is only about 8.15, and individual groups of episodes even give negative values. This means that without the attention mechanism, the recurrent network is not able to confidently generalize the navigation strategy to a new environment.

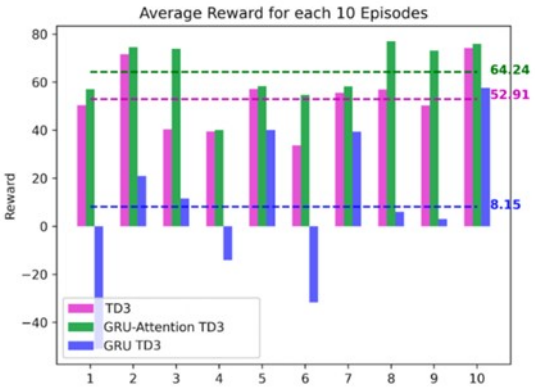

Figure 12: Average reward for every 10 testing episodes in test environment I for TD3, GRU-TD3 and GRU-Attention TD3 networks

Figure 13 illustrates the number of collisions for the same 100 episodes. The base TD3 allowed 10 collisions, the GRU-Attention TD3 network only 5, while GRU-TD3 had 37 collisions. That is, GRU-Attention TD3 not only receives the highest reward, but also ensures the safest robot behavior, while GRU-TD3 often leads to emergency termination of episodes.

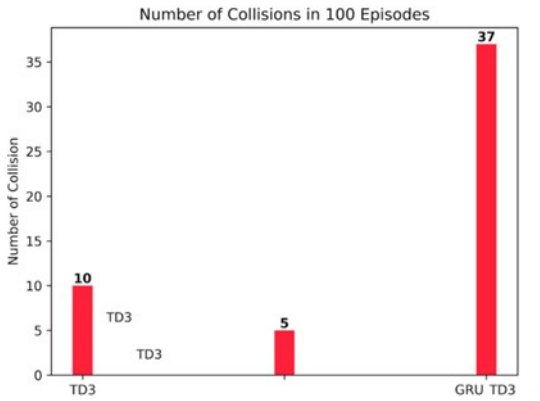

Figure 13: Number of collisions per 100 testing episodes in test environment I for different networks

In summary, we can say that in the test environment I, GRU-Attention TD3 demonstrates the best compromise between navigation efficiency and reliability. The basic TD3 generates acceptable but lower-quality policies, while GRU-TD3 turns out to be unstable and significantly inferior in all key indicators.

3.4.2. Results in Test Environment II

Test environment II is 20×20 m in size and contains more obstacles. The robot's starting position and the target point are usually located near opposite corners of the scene, so the trajectory is longer and more complex. The summarized numerical test results are given in Table 3.

Table3. Network testing results in the test environment II

Chain	Number of steps	Total reward	Collision
GRU-Attention TD3	242	82,48	No
TD3	393	52,90	No
GRU-TD3	12	-102,73	Yes

The table shows that:

- GRU-AttentionTD3 reaches the goal in 242 steps with the highest total reward of 82.48 and without any collisions;
- The baseline TD3 also completes the task without collisions, but requires significantly more steps and receives a lower reward, which means a less efficient route;
- GRU-TD3 takes only 12 steps, but immediately leads to a collision and receives a strong negative reward

The typical trajectories of the robot in the test environment II are shown in Figures 14-17. Figure 14 shows the trajectory of the baseline TD3 network. The robot reaches the goal, but the route contains several unnecessary bends and deviations from the shortest path.

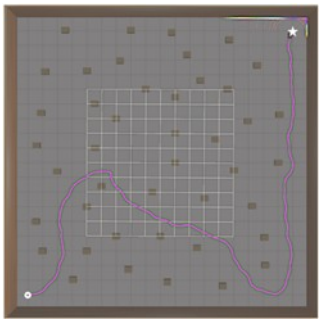

Figure 14: Robot movement trajectory in test environment II for TD3 network

Figure 15 shows the trajectory of the GRU-Attention TD3 network. In this case, the path is more rectilinear and smooth, the robot enters the “correct” corridor earlier and overcomes obstacles with minimal detours.

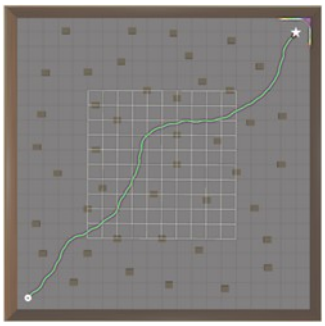

Figure 15: Robot movement trajectory in test environment II for the GRU-Attention TD3 network

Figure 16 demonstrates the movement of the GRU-TD3 network: the robot quickly deviates from the safe trajectory and encounters an obstacle at the beginning of the path, which is consistent with the negative reward and the “Collision” label in Table 3.

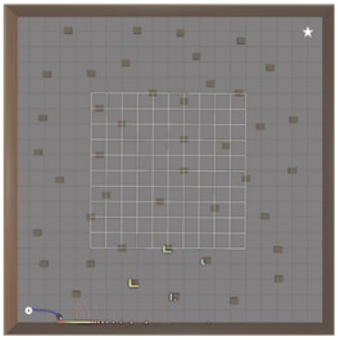

Figure 16: Robot movement trajectory in test environment II for the GRU-TD3 network

Figure 17 shows a combined image of the TD3 and GRU-Attention TD3 trajectories. It is clearly seen that the green trajectory (GRU-Attention TD3) is shorter and closer to the optimal one, while the baseline TD3 trajectory has extra “loops”, especially in the central part of the scene.

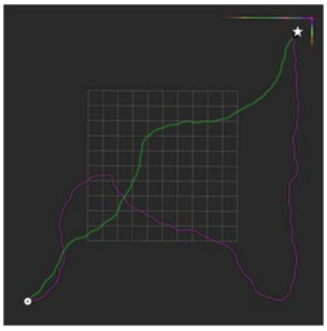

Figure 17: Comparison of TD3 and GRU-AttentionTD3 trajectories in test environment II

3.4.3. Results in Test Environment III

Test environment III has the same size of 20×20 m as test environment II, but a more complex configuration of walls and obstacles: narrow corridors, "pockets" and several alternative routes to the target. The initial position of the robot and the position of the target point were chosen to be the same as in the previous environment to ensure a correct comparison of the results.

The summarized numerical indicators are given in Table 4.

Table 4. Network testing results in test environment III

Chain	Number of steps	Total reward	Collision
GRU-Attention TD3	153	84,96	No
TD3	207	60,81	No
GRU-TD3	5	-100,75	Yes

As can be seen from the table, the GRU-Attention TD3 neural network again demonstrates the best results. It brings the robot to the goal in 153 steps, receiving the highest total reward of 84.96 without any collisions. The basic TD3 also successfully performs the task, but it takes longer (207 steps) and receives a lower reward, which indicates less optimal trajectories. The GRU-TD3 network in this environment actually does not cope with the task: the robot takes only 5 steps, encounters an obstacle and receives a significant negative reward. Typical movement trajectories are shown in Figures 18-21.

Figure 18 shows the route formed by the TD3 base network: the robot reaches the goal, but in narrow sections the trajectory has noticeable "loops" and deviations from the shortest path.

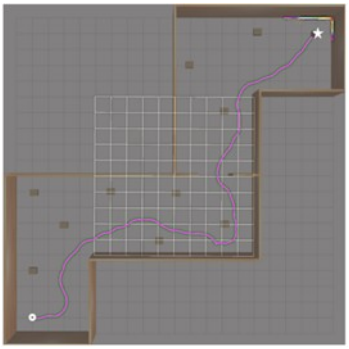

Figure 18: Robot movement trajectory in test environment III for TD3 network

Figure 19 shows the trajectory of the GRU-Attention TD3 network, which is significantly shorter and smoother: the robot confidently passes narrow corridors, without falling into "pockets" and without making unnecessary maneuvers.

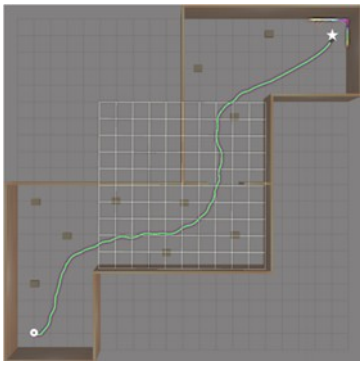

Figure 19: Robot movement trajectory in test environment III for the GRU-Attention TD3 network

Figure 20 shows the motion obtained from the GRU-TD3 network; it quickly leads to a collision with an obstacle, which is consistent with the data in Table 4.

Figure 20: Robot movement trajectory in test environment III for the GRU-TD3 network

Figure 21 shows a joint display of the TD3 and GRU-Attention TD3 trajectories, where it is clearly seen that the second approach forms a shorter and more natural route to the target.

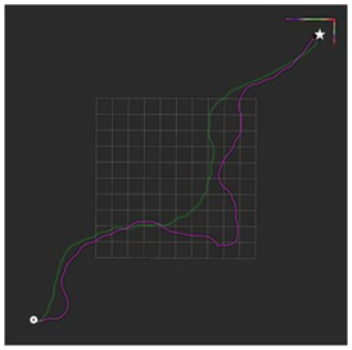

Figure 21: Comparison of TD3 and GRU-Attention TD3 trajectories in test environment III

In this part of the work, a complete experimental verification of the developed mobile robot navigation system and three variants of the reinforcement learning algorithm: TD3, GRU-TD3 and GRU-Attention TD3 was carried out. The obtained results allow us to draw a number of generalizing conclusions regarding the quality of training, the ability to generalize and the safety of robot movement in different simulation environments.

First, the chosen scheme for organizing training and test experiments turned out to be correct and sufficiently representative. Training all networks in one training world with randomization of initial conditions and obstacle configuration provided a variety of scenarios that the agent encounters and reduced the risk of overtraining. Further testing in three environments differing in structure and scale showed that not only adaptation to a specific map is assessed, but also the real ability of the navigation policy to be transferred to new conditions.

Secondly, the set of metrics used — total reward per episode, average reward per epoch, number of steps to achieve the goal and number of collisions — gave a holistic picture of the quality of the system. The training graphs showed how the effectiveness of the policy changes over time, and the test results made it possible to assess how quickly, confidently and safely the robot moves towards the goal under conditions of varying complexity.

Thirdly, the study of the learning dynamics revealed significant differences between the three networks. The basic TD3 gradually reaches a stable, but not the highest level of policy quality; GRU-TD3, despite a fast start, exhibits instability and significant fluctuations in quality from epoch to epoch. In contrast, GRU-Attention TD3 demonstrates the fastest and most stable convergence: the average reward grows faster and remains at the highest level for a long time among all the compared options.

Fourthly, the test results in three simulation environments confirmed the superiority of the GRU-Attention TD3 architecture. In each environment, this network provided the highest values of the total reward, a smaller or comparable number of steps to the goal and the smallest number of collisions. The basic TD3 formed an acceptable policy, but required longer trajectories and more often chose not the most optimal paths. GRU-TD3 in complex environment configurations often led

to collisions already at the initial stages of movement, which indicates insufficient stability of this modification.

Fifth, the analysis of the movement trajectories further emphasized the qualitative differences between the approaches. TD3 is characterized by unnecessary “loops” and deviations from the shortest path, especially in narrow corridors. GRU-TD3 is characterized by errors that quickly end in collisions. In contrast, GRU-Attention TD3 builds more direct, smooth and “conscious” trajectories, successfully bypasses local traps and confidently passes complex sections with a large number of obstacles.

In summary, it can be stated that the developed system based on GRU-Attention TD3 provides the best balance between efficiency, speed and safety of navigation. It not only successfully learns in one environment, but also demonstrates the ability to generalize the acquired experience to new, more complex scenes. This confirms the feasibility of using a combination of recurrent networks and the attention mechanism in the tasks of autonomous navigation of mobile robots in an unknown environment and creates a basis for further development of such systems in the direction of real robotic applications.

4. Conclusions

This article addresses the scientific and applied task of developing an intelligent navigation system for a mobile robot in unknown environments using deep reinforcement learning. An approach is proposed where the base TD3 algorithm is integrated with GRU recurrent memory and an attention mechanism. The entire system was implemented and evaluated within the ROS+Gazebo simulation suite. The results demonstrate the effectiveness of this architecture for autonomous navigation tasks under conditions of partial observability.

Domain analysis indicates that classical path planning and obstacle avoidance algorithms, despite their maturity and widespread adoption, exhibit limited effectiveness in dynamic and partially observable environments where comprehensive spatial maps are unavailable and obstacle configurations shift. In contrast, deep reinforcement learning (DRL) methods have demonstrated superior adaptive potential, though they necessitate specialized architectures to account for temporal dependencies and noisy sensor data. This provides the rationale for selecting TD3 as the baseline algorithm and underscores the feasibility of enhancing it with recurrent and attention-based mechanisms.

The study establishes the requirements for the intelligent navigation system, the reinforcement learning agent, and the simulation environment. We define the structure of the state vector and the reward function, which simultaneously encourages goal-oriented movement while penalizing collisions and inefficient trajectories. The implementation tools were selected and justified, leading to a software architecture with a clear separation of concerns between ROS nodes, sensor data processing modules, and the agent's learning components. This ensured consistency between the theoretical problem formulation and its practical execution.

Three autonomous navigation agents were implemented — TD3, GRU-TD3, and GRU-Attention TD3 — alongside several simulation environments of varying complexity. Experimental results demonstrate that the modified GRU-Attention TD3 agent achieves higher cumulative rewards, fewer collisions, and shorter, more stable trajectories in most scenarios compared to the baseline variants. However, limitations were identified, including sensitivity to hyperparameter selection, slower convergence, or susceptibility to local minima in highly complex environments. Furthermore, all experiments were conducted solely in simulation, without validation on a physical robotic platform. These findings provide a realistic assessment of the proposed approach.

The synthesis of the conducted research confirms that the results align with the diploma project's objectives and the requirements established in the introduction. The scope of work encompassed an analysis of existing solutions, the development of system requirements, the design and implementation of the software suite, agent training and testing, and the preparation of the textual and graphical components of the explanatory note. The primary goal — the development,

modeling, and experimental evaluation of an intelligent mobile robot navigation system in unknown environments based on the GRU-Attention TD3 architecture — has been successfully achieved.

The feasibility of practical implementation is underscored by the integration of industry-standard tools within the robotics ecosystem and the modular, universal design of the navigation component. The developed system is adaptable to various types of wheeled mobile robots, while the simulation framework serves as a robust foundation for platform-specific algorithm tuning and as a versatile research and educational environment.

The scientific and technical significance of this work resides in the advancement of the TD3 architecture through the synergy of recurrent memory and attention mechanisms. Furthermore, the study contributes a comprehensive suite of metrics for evaluating navigation quality and provides new empirical data regarding the influence of agent architecture on training stability and trajectory characteristics.

The prospects for further research are associated with the transition to more complex simulation scenarios and the application of domain randomization techniques to facilitate the transfer of learned policies into the real world (Sim-to-Real). Additionally, future efforts will focus on integrating the navigation module with SLAM (Simultaneous Localization and Mapping) algorithms and optimizing computational complexity for efficient execution on the embedded platforms of mobile robots. Implementing these directions will contribute to the development of increasingly reliable and practically-oriented autonomous navigation systems.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] S. Fujimoto, H. Hoof, D. Meger, Addressing Funct. Approximation Error in Actor-Critic Methods, International Conference on Machine Learning, PMLR, 2018, pp.1587—1596

[2] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous Control with Deep Reinforcement Learning. arXiv:1509.02971, 2015

[3] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, 2018

[4] ROS Wiki: ROS Noetic Documentation, https://wiki.ros.org/noetic

[5] B. Zhurakovskyi, V. Poltorak,S. Toliupa, O. Pliushch, A. Platonenko, Process. and Analyzing Images Based on a Neural Network, CEUR Workshop Proceedings, vol. 3654, pp. 125–136, 2024, Workshop Cybersecur. Providing in Inf. and Telecommun. Systems, CPITS 2024 Virtual, Online 28 February 2024, Germany, https://ceur-ws.org/Vol-3654/paper11.pdf

[6] B. Zhurakovskyi, V. Shutenko, A. Makarenko, O. Pliushch, R. Zakharov. A Security System with Automated Person Identif. Based on Computer Vision Technologies. CEUR Workshop Proceedings, CPITS-2025-II: Cybersecur. Providing in Inf. and Telecommun. Systems, October 26, 2025, Kyiv, Ukraine, pp. 228–245, https://ceur-ws.org/Vol-4145/paper15.pdf

[7] B. Zhurakovskyi, O. Nedashkivskiy, M. Klymash, O. Pliushch, V. Saiko, Traffic Control System Based on Neural Network. Digital Ecosystems: Interconnecting Advanced Networks with AI Applications. TCSET 2024, Lecture Notes in Electrical Engineering, vol. 1198, 522—542, Springer, Cham, doi:10.1007/978-3-031-61221-3_25

[8] N. Gulak, S. Ilyenko, E. Dubchak, A. Maistrenko, B. Zhurakovskyi. Software Module Protecting Web Applications from Several Attacks Types using Formal Logic, CEUR Workshop Proceedings, CPITS-2025-II: Cybersecur. Providing in Inf. and Telecommun. Systems, October 26, 2025, Kyiv, Ukraine, pp. 35–45, https://ceur-ws.org/Vol-4145/paper3.pdf

[9] B. Zhurakovskyi, O. Pliushch, V. Saiko, V. Shutenko, Machine Learning-based Environmental Monitoring and Analysis System. Proceedings of the Workshop on Cybersecur.

Providing in Inf. and Telecommun. Systems (CPITS 2025). Kyiv, Ukraine, February 28, 2025 (online), Vol-3991, pp. 183—203, https://ceur-ws.org/Vol-3991/paper14.pdf

[10] K. Maatoug, M. Njah, M. Jallouli, Autonomous Wheelchair Navigation in Indoor Environment Based on Fuzzy Logic Controller and Intermediate Targets, 2017, International Conference on Advanced Systems and Electric Technologies (IC_ASET), pp. 55—59, 2017

[11] R. Ravi, A. Kos, Intelligent Mobile Robot Navigation in Unknown and Complex Environment using Reinforcement Learning Technique. Scientific Reports, vol. 14, article number: 22852, 2024, doi:10.1038/s41598-024-72857-3

[12] I. Liminovych, V. Poltorak, N. Kushnir, B. Zhurakovskyi, S. Obushnyi, Protection System for Analysis of External Link Placing, CEUR Workshop Proceedings, vol. 3654, pp. 179–188, 2024 Workshop Cybersecur. Providing in Inf. and Telecommun. Systems, CPITS 2024 Virtual, Online28 February, 2024, Germany, https://ceur-ws.org/Vol-3654/paper15.pdf

[13] D. Virovets, S. Obushnyi, B. Zhurakovskyi, P. Skladannyi, V. Sokolov, Integr. of Smart Contracts and Artificial Intelligence Using Cryptographic Oracles, CQPC-2024: Classic, Quantum, and Post-Quantum Cryptography, August 6, 2024, Kyiv, Ukraine, vol. 3829, 39–46, https://ceur-ws.org/Vol-3829/short5.pdf

[14] RViz – ROS Visualization Tool. ROS Wiki, https://wiki.ros.org/rviz

[15] B. Zhurakovskyi, S. Toliupa, A. Bondarchuk, V. Druzhynin, M. Stepanov, Calculation of Qual. Indicators of the Future Multiservice Network. Future Intent-Based Networking, 2022. – (Part of the Lecture Notes in Electrical Engineering book series, vol. 831, pp. 197–209, doi:10.1007/978-3-030-92435-5_11

[16] Clearpath Robotics, RViz tutorial, https://docs.clearpathrobotics.com/docs/ros/tutorials/rviz/

[17] Gazebo Simulator Documentation, https://gazebosim.org/

[18] M. Vaskovic, M. Jurisevic, N. Babajic, M. Matijevic, Pioneer 3-DX Distance Control Using Different Type of Sensors, Acta Technica Corviniensis – Bulletin of Engineering, 2014, vol. 7, No. 2, p. 31–35

[19] B. Zhurakovskyi, V. Poltorak, S. Toliupa, O. Pliushch, O. Nesterova, Data Protection in the Automated Agribusiness Manag, System, CPITS-II 2024: Workshop on Cybersecur, Providing in Inf. and Telecommun. Systems II, October 26, 2024, Kyiv, Ukraine, vol. 3826, pp. 267—275, https://ceur-ws.org/Vol-3826/short16.pdf

[20] B. Zhurakovskyi, S. Otrokh, M. Poliakov, O. Poliakov and P. Skladannyi, Enhancing Inf. Transmission Secur. with Stochastic Codes, CQPC-2024: Classic, Quantum, and Post-Quantum Cryptography, 2024, Kyiv, Ukraine, vol. 3829, 62–69, https://ceur-ws.org/Vol-3829/short8.pdf

[21] M. Lagoudakis, A. Maida, Neural Maps for Mobile Robot Navigation, In Proceedings of the 1999 International Joint Conference on Neural Networks, 1999

[22] N. Thelasingha, S. Ekanayake, B. Udugama, G. Godaliyadda, M. Ekanayake, B. Samaranayake, J. Wijayakulasooriya, Autonomous Exploration Planning Strategy for a Reconnaissance Agent, 2017, IEEE International Conference on Industrial and Inf. Systems (ICIIS), pp.1—6, 2017

[23] M. Quigley, et al., ROS: an Open-Source Robot Operating System, ICRA Workshop on Open Source Software, 2009, vol. 3, No. 3.2

[24] T. Oliphant, Guide to NumPy. 2nd ed. USA: Continuum Press, 2015, p. 378

[25] A. Paszke, S. Gross, F. Massa, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Inf. Processing Systems 32, NeurIPS 2019

[26] G. Kim, A. Kim, LT-mapper: A Modular Framework for Lidar-Based Lifelong Mapping. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: New York, NY, USA, 2022; pp. 7995–8002

[27] A. Abeysekara, D. Liyanage, W. Welikala, G. Godaliyadda, M. Eakanayake, J. Wijayakulasooriya, Depth Map Generation for a Reconnaissance Robot Via Sensor Fusion. In IEEE 10th International Conference on Industrial and Inf. Systems , p. 320–325, Dec 2015

[28] B. Liu, X. Xiao, P. Stone, A Lifelong Learning Approach to Mobile Robot Navigation, IEEE Robot, Autom. Lett, 2021, 6, 1090–1096

[29] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning. MIT Press, 2016, p. 127

[30] A. Victorino, P. Rives, J. Borrelly, Safe Navigation for Indoor Mobile Robots, Part II: Explor., Self-Localization and Map Building, 2004

[31] I. Ariffin, A. Baharudin, H. Yussof, M. Miskam. Reactive Obstacle Avoidance Implement. for Humanoid Controlled Mobile Platform Navigation. In 2016 Asia-Pacific Conference on Intelligent Robot Systems (A CIRS), p. 192–196, July 2016

[32] B. Zhurakovskyi, I. Averichev, I. Shakhmatov, Using the Latest Methods of Cluster Analysis to Identify Similar Profiles in Leading Social Networks, *CEUR* Workshop Proceedings,T. 3646, pp. 116–126, 2023, 10th International Scientific Conference "Inf. Technology and Implemen.", 2023, Kyiv, 20 Novemb. 2023—21 Novemb. 2023, https://ceur-ws.org/Vol-3646/Paper_12.pdf

[33] R-FCN: Object Detect. via Region-based Fully Convolutional Networks / Dai, J., Li, Y., He, K., Sun, J. Neural Inf. Processing Systems (NIPS), 2016, p. 379—387