DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

Comparing Continuous Single-Agent Reinforcement Learning Controls in a Simulated Logistic Environment using NVIDIA Omniverse

Mike Wesselhöft
Philipp Braun
Jochen Kreutzfeldt

Institut of Technical Logistics
Technical University of Hamburg

With the transition to Logistics 4.0, the increasing demand for autonomous mobile robots (AMR) in logistics has amplified the complexity of fleet control in dynamic environments. Reinforcement learning (RL), particularly decentralized RL algorithms, has emerged as a potential solution given its ability to learn in uncertain terrains. While discrete RL structures have shown merit, their adaptability in logistics remains questionable due to their inherent limitations. This paper presents a comparative analysis of continuous RL algorithms - Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO) - in the context of controlling a Turtlebot3 within a warehouse scenario. Our findings reveal A2C as the frontrunner in terms of success rate and training time, while DDPG excels step minimization while PPO distinguishes itself primarily through its relatively short training duration. This study underscores the potential of continuous RL algorithms, especially A2C, in the future of AMR fleet management in logistics. Significant work remains to be done, particularly in the area of algorithmic fine-tuning.

[Keywords: logistics 4.0, autonomous mobile robots, reinforcement learning, artificial intelligence, robotics]

Mit dem Übergang zur Logistik 4.0 hat der zunehmende Bedarf an autonomen mobilen Robotern (AMR) in der Logistik die Komplexität der Flottensteuerung in dynamischen Umgebungen erhöht. Reinforcement Learning (RL), insbesondere dezentrale RL-Algorithmen, haben sich aufgrund ihrer Fähigkeit, in unsicheren Umgebungen zu lernen, als potenzielle Lösung erwiesen. Während sich diskrete RL-Strukturen bewährt haben, bleibt ihre Anpassungsfähigkeit in der Logistik aufgrund ihrer inhärenten Einschränkungen fraglich. In diesem Beitrag wird eine vergleichende Analyse kontinuierlicher RL-Algorithmen - Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG) und Prox-imal Policy Optimization (PPO) - im Kontext der Steuerung eines Turtlebot3 in einem Lagerszenario vorgestellt. Unsere Ergebnisse zeigen A2C als Spitzenreiter in Bezug auf Erfolgsrate und Trainingszeit, während DDPG bei der Minimierung der Episodenlänge punktet und PPO lediglich mit einer geringen Trainingsdauer aufwarten kann. Diese Studie unterstreicht das Potenzial von kontinuierlichen RL-Algorithmen, insbesondere A2C, für die Zukunft des AMR-Flottenmanagements in der Logistik, wobei gerade im Bereich des Finetunings der Algorithmen noch viel Arbeit zu tun ist.

[Schlüsselwörter: Logistik 4.0, Autonome Roboter, Reinforcement Learning, Künstliche Intelligenz, Robotik]

1 Introduction

With the advancement towards Industry 4.0, there has been a rapid increase in the demand for autonomous mobile robots in various sectors, notably in logistics [1, 2]. This can be attributed to the high degree of automation offered, significantly enhancing the efficiency of operations and reducing human error. However, as this sector evolves, so does the complexity of controlling fleets of autonomous mobile robots. The dynamic nature of logistics environments necessitates control algorithms that are not only efficient but also highly adaptable. Increasingly, warehouses and distribution centers operate around the clock, with many unforeseen changes in their environment. Therefore, managing these fleets of autonomous mobile robots demands sophisticated multi-agent path planning algorithms.

In recent years, reinforcement learning (RL) has shown substantial promise in addressing these challenges. RL algorithms possess the ability to learn optimal strategies in complex, dynamic, and uncertain environments [3.,4.], making them a suitable choice for multi-agent path planning problems [5.,6.]. However, as the complexity and scale of such systems increase, the computational requirements also become a concern.

In this regard, decentralized reinforcement learning algorithms are seen as an ideal choice, as they reduce the computational effort while handling the flexibility [7.] of the system [8]. The scalability of the algorithms controlling

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

the individual agents significantly depends on the efficiency of the algorithms. Hence, choosing the optimal RL algorithm for controlling the single agents becomes paramount to ensure overall system efficiency. Thus, this work will perform a comparative analysis the three state of the art reinforcement learning algorithms with continuous action space - Advantage Actor-Critic (A2C, [9]), Proximal Policy Optimization (PPO, [10}), and - Deep Deterministic Policy Gradient (DDPG, [11]) in the context of controlling a turtlebot3 in a warehouse scenario. The goal is to determine the most suitable RL algorithm that strikes a balance between efficiency (maximize payload, while minimizing time, energy consumption and costs), flexibility and computational effort, thereby meeting the demanding needs of logistics 4.0.

For this purpose, the work is structured as follows: The second section introduces some basic definitions of single agent path finding problems and basics of RL algorithms. In the third section, related work is presented, before the methodology and approach follow in the fourth section. The work then closes with the results in the fifth and a summary in the last section.

2 Reinforcement Learning

Reinforcement learning is a type of machine learning where an agent learns to make decisions by taking actions in an environment to maximize some cumulative reward. The agent's goal is to learn a policy π, which is a mapping from states to actions that maximizes the expected sum of rewards. The cumulative reward, denoted by G_t, is computed as follows in [12]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where R_{t+k+1} is the reward at time $t+k+1$ and $\gamma \in (0,1)$ is the discount factor. Two fundamental concepts in reinforcement learning are the value function $V_\pi(s)$ and the action-value function $Q_\pi(s,a)$, defined as (the subscripted π denotes an explicit policy the value and action-value function are computed on):

$$V_\pi(s) = E_\pi[G_t \mid S_t = s]$$

$$Q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

$V_\pi(s)$ is the expected return from state $S_t = s$ under policy π, and $Q_\pi(s,a)$ is the expected return after taking action $A_t = a$ in state $S_t = s$ under policy π.

2.1 Advantage Actor Critic

Actor-critic methods are a type of reinforcement learning algorithms that maintain two models: an actor, which determines the policy of the agent, and a critic, which estimates the value function associated with the policy of the actor. The name actor-critic is based on these two primary components. The actor's job is to select actions, and the critic's job is to estimate the value function used to criticize the actions made by the actor. The actor updates its policy in response to this criticism, hence the name actor-critic.

The Advantage Actor-Critic (A2C) method is a variation of the actor-critic approach, which introduces the concept of an advantage function [9]. The advantage function $A(s,a)$ essentially quantifies how much better it is to take a certain action a in a certain state s compared to the average action in that state under the current policy. Intuitively, the advantage function measures the relative quality of a certain action in a given state. The advantage function is generally defined as the difference between the action-value function $Q_\pi(s,a)$ and the state-value function $V(s)$:

$$A(s,a) = Q_\pi(s,a) - V_\pi(s)$$

In the case of Advantage Actor-Critic, the advantage function $A(s,a)$ is used in place of the full action-value function $Q(s,a)$ in the policy gradient update rule.

2.2 Proximal policy Optimization

Proximal Policy Optimization (PPO) is a type of policy gradient method for reinforcement learning. It optimizes the policy by maximizing an objective function that includes a clipped surrogate objective to control the policy update size. This objective function is as follows [10]:

$$L(\theta) = E_t\,[\,min(\rho_t(\theta)A_t, clip(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)\,]$$

where $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio of the old and new policy for the update, A_t is the advantage function at timestep t and the $clip$ function chooses $\rho_t(\theta)$ if it is between $1-\varepsilon$ and $1+\varepsilon$ and either of them if $\rho_t(\theta)$ is bigger or smaller ($\varepsilon > 0$).

2.3 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy, actor-critic algorithm that uses deep function approximators to learn policies in continuous action spaces. In DDPG, the actor policy function and the critic action-value function are updated according to the following rules [11]:

$$\theta_\pi \leftarrow \theta_\pi + \alpha \nabla_{\theta_\pi} Q(s,a|\theta_Q)|s = S_t, a = \pi(S_t)$$

$$\theta_Q \leftarrow \theta_Q + \alpha \nabla_{\theta_Q} (r + \gamma Q(s', \pi(s')|\theta_Q) - Q(s,a|\theta_Q))\text{^}2$$

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

where θ_π and θ_Q (∇_{θ_π} the gradient with respect to the parameters) are the policy parameters and the action-value parameters, respectively, and $\pi(s)$ is the policy function.

3 Related Work

Reinforcement learning (RL) based control approaches present a promising direction in robotics due to their inherent capacity to solve complex tasks. A body of evidence demonstrating the potential of these algorithms has been growing in the recent literature. [13] for instance, explored the implementation of deep RL for drone navigation tasks using sensor data, yielding insightful outcomes and further highlighting the capabilities of these algorithms. In the domain of multi-agent pathfinding (MAPF), RL-based algorithms have begun to show noteworthy results in comparison to traditional state-of-the-art approaches. Established path planning algorithms such as Conflict-Based Search (CBS) [14] and Reciprocal n-Body Collision Avoidance (ORCA) [15] have seen competition from more recent approaches like Transformer-based Imitative Reinforcement Learning (TIRL) [16]. Several other RL-based approaches, like Distributed Heuristic Multi-Agent Path Finding with Communication (DHC) [17], Learning Selective Communication for Multi-Agent Path Finding (DCC) [18], and Pathfinding via Reinforcement and Imitation Multi-Agent Learning (PRIMAL and PRIMAL 2) [5, 6], have demonstrated similar successes when compared to aforementioned state of the art algorithms. Two key features have been identified among these RL-based approaches, which offer a high potential for controlling fleets of autonomous mobile robots (AMR). The first feature is decentralization (when agents are controlled decentralized to reduce computational effort but have a communication mechanism to ensure successful fleet behaviour), which is becoming increasingly recognized as an efficient strategy for controlling fleets of AMR [8]. [8] note that such decentralized control mechanisms allow for improved scalability and adaptability, particularly critical in dynamic environments like logistics. The second feature these algorithms have in common is that they are discrete, hence primarily applied in graph-based structures or similar frameworks. While this structure simplifies the problem space and allows for certain computational efficiencies, it imposes a limitation on the flexibility of the algorithm [19]. This restriction is potentially consequential in real-world scenarios that often require a high degree of adaptability and will therefore harm especially the flexibility of the algorithm which is needed for high dynamic environments like logistics or production.

Given the identified gaps and potentials, this paper takes a step forward by exploring the application of continuous RL approaches for single-agent robot control. The aim is to establish an optimized basis for a decentralized and continuous RL-based control system for AMR fleets in logistics scenarios. By comparing these continuous RL algorithms in a common environment, we seek to contribute valuable insights towards achieving more efficient and flexible AMR fleet management.

4 Implementation

The present work follows a systematic approach to evaluate the performance of reinforcement learning (RL) algorithms in a warehouse navigation task using an autonomous mobile robot. The fundamental components of the employed methodology are the simulated environment, a customized control interface for the robot, and the RL algorithms. For the simulation part NVIDIA Isaac Sim and Gym are used, due to their comprehensive capabilities and extensive reinforcement learning support.

Environment Setup

Incorporating a simulated environment facilitates the isolation and manipulation of task-specific variables in a controlled manner, thus allowing for a more focused study. Utilizing the NVIDIA Isaac Sim, a robotics simulator offering highly realistic physics and exceptional rendering fidelity, the warehouse environment's nuances could be simulated effectively. Integration with NVIDIA Isaac Gym, a toolkit designed specifically for the development and comparison of reinforcement learning algorithms, allowed for a comprehensive exploration of algorithmic performance in the given task.

The warehouse environment was established by importing a Universal Scene Description Format (USD) model of the TurtleBot3 robot into the Isaac Sim environment, equipped with a 360-degree rotating Lidar sensor. The environment therefore is a small warehouse consisting out of 4 walls and two large racks with pallets, cartons and other stationary objects. As a goal a small cube has been integrated for the turtlebot3 agent to reach.

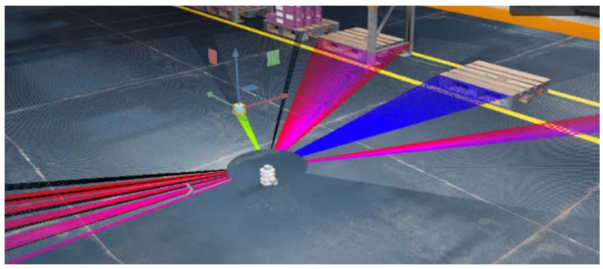

Figure 1 Simulated Turtlebot3 in Nvidia Isaac Sim environment equipped with a 360-degree Lidar

Control Mechanism Implementation

The control mechanism acts as the interface between the RL algorithms and the simulated environment, allowing

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

commands to be translated into specific robot actions. This involved in-depth consideration of the robot's physical parameters and constraints, ensuring accurate and feasible execution of the actions recommended by the RL algorithms. To maneuver the Turtlebot3, a library from the Omniverse Isaac Core extension was utilized. This library features a wheeled robot interface paired with a differential controller, designed to translate specified angular and linear velocities into actuation signals for the robot's wheels, enabling precise navigation within the environment.

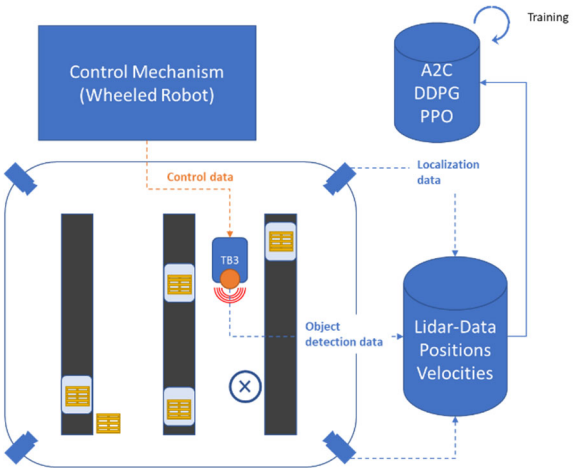

Figure 2 Overview of the experimental setup

Algorithm Implementation and Training

The RL algorithms, PPO, DDPG and A2C were sourced from the Stable Baselines 3 library [20]. Stable Baselines 3 offers Python implementations of state-of-the-art reinforcement learning algorithms, ensuring consistent access to the latest algorithmic advancements.

Each algorithm was manipulated to the specific requirements of the warehouse navigation task. This involved modification of the action- and observation-spaces of the algorithms. Actions were formulated within a continuous interval [0, 1] and subsequently scaled by the robot's maximum linear and angular velocities. The observation space encompassed the robot's current positional coordinates, goal coordinates, angular and linear velocities, and the proximate distance to the goal. This observational data was further enriched with readings from a 360-degree LiDAR sensor. However, due to the increased dimensionality introduced by the LiDAR data, only every fourth ray was incorporated into the algorithm's observational input. Subsequent to the initial setup, the algorithms were trained on an NVIDIA RTX Titan graphics card, undergoing a total of 5 million training steps each.

Hyperparameter Tuning and Reward Optimization

Following initial training, a rigorous hyperparameter tuning phase was conducted. This process involved adjusting the algorithms' basic hyperparameters like the learning rate or the discount factor to enhance the learning process's efficiency and the resulting performance. The neural networks for each algorithm were constructed with uniform sizes to enhance comparability between them and to limit parameter tuning options, ensuring the scope of this work was maintained.

Parallel to the hyperparameter tuning, the reward structure of the environment was refined to maximize the agent's goal-oriented learning. The design of the reward structure is crucial in reinforcement learning, as it defines the signals based on which the agent learns its policy. To tackle the reward-sparseness (only achievements for the robot are picking the object) of the environment the reward-function was separated into four parts, a part that measures the difference in distance to the target regarding the last update ($\Delta_d = d_{before} - d_{now}$), a term that rewards close distances to the target ($\alpha * \frac{1}{1+d_{now}}$), where $\alpha \in [0,1]$ scales the term with the distance update. Additionally, a weighted translational distance, denoted as d_{trans}, is incorporated into the reward function to ensure that the TurtleBot3's field of view aligns with its target objective. This term is scaled by a factor $\beta \in [0,1]$, which harmonizes the magnitude of this component with the other elements of the reward structure. In the end a case is added which leads to certain bonus-rewards, which brings the reward-function down to the following:

$$R(x) = \begin{cases} \Delta_d + \alpha\left(\frac{1}{1+d_{now}}\right) - d_{trans} + 3 & if\ d_{now} < \tau_1 \\ \Delta_d + \alpha\left(\frac{1}{1+d_{now}}\right) - d_{trans} + 25 & if\ d_{now} < \tau_2 \\ \Delta_d + \alpha\left(\frac{1}{1+d_{now}}\right) - d_{trans} - 15 & if\ d_{now} > \tau_3. \\ \Delta_d + \alpha\left(\frac{1}{1+d_{now}}\right) - d_{trans} - 20 & if\ epis > \tau_4 \\ \Delta_d + \alpha\left(\frac{1}{1+d_{now}}\right) - d_{trans} & else \end{cases}$$

In this case τ_1, τ_2 and τ_3 are thresholds measuring if the algorithm reached a curtain milestone the first time (or in the case of τ_3 if the algorithm is to far away to the target). In the fourth case $epis > \tau_4$ signifies that the algorithm has exceeded the maximum predefined episode length.

5 Results

In this section the capability, to control a turtlebot3 in a logistic environment, for every algorithm will be presented. The performance metrics focused on four essential areas: mean episode length, mean reward per episode, success rate and the training time.

In analyzing the mean reward and episode lengths, the A2C algorithm demonstrated superior performance by achieving the highest mean reward at -12.2, while DDPG has the shortest mean episode length at 232 (seemingly exploiting

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

the reward structure, the agent appears to strategically opt for actions that increase the distance between itself and the goal, as these actions incur a less severe penalty of -15 compared to potentially larger negative rewards from other non-successful steps). Conversely, A2C exhibited a commendable episode length of 238, making it the second shortest, while DDPG is the runner up in case of mean reward with -12.7. The PPO algorithm is a distant last with a mean reward of -14 and a mean episode length of 357. The initial superior performance of the agent suggests one of two possibilities: either the algorithm is not sufficiently robust for task completion, or it is currently in an exploratory phase that may require a significantly larger number of training steps for convergence towards optimal behavior. The respective illustrations for PPO, DDPG, and A2C can be found in Figure 3, Figure 4, and Figure 5.

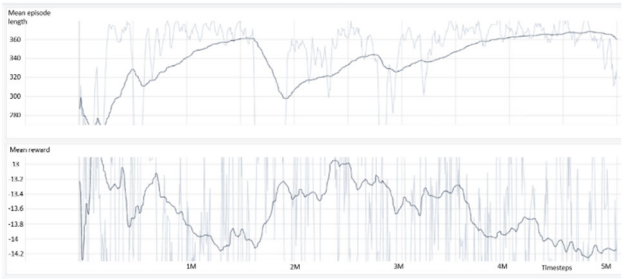

Figure 3 Mean episode-length and reward of the PPO algorithm

From a training efficiency standpoint, the A2C algorithm was quickest, wrapping up its training in a mere 40.63 hours. The PPO algorithm took a slightly longer duration, finishing its training in 40.68 hours, and DDPG was the most time-consuming, necessitating 49.8 hours for training completion.

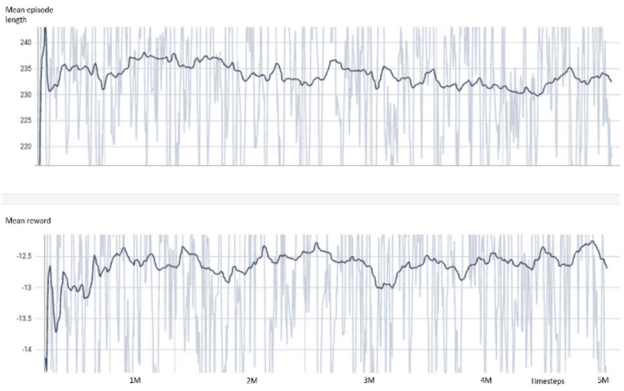

Figure 4 Mean episode-length and reward of the DDPG algorithm

A significant measure of performance, the success rate, showed that A2C outperformed the other two with a notable success rate of 64 percent. This was closely followed by the DDPG algorithm which achieved 56 percent, while PPO lagged slightly behind at 42 percent. These detailed success rates and the training time are further tabulated in Table 1 for clarity.

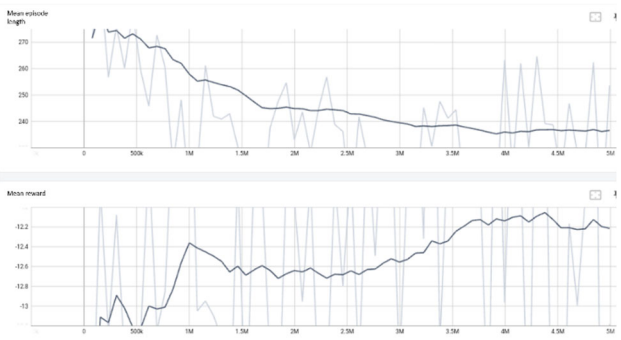

Figure 5 Mean episode-length and reward of the A2C algorithm

The A2C algorithm emerged as the most proficient, boasting the highest success rate among its counterparts. Moreover, its training efficiency was evident as it completed its learning in the shortest duration (almost identical to the training duration of the PPO).

Table 1 Success rates of the PPO, DDPG and A2C algorithm in reaching the goal in the logistic scenario

Algorithm	A2C	DDPG	PPO
Success Rate	0.64	0.56	0.42
Time to train	40,63 h	49,80 h	40,68 h

While the DDPG secured a commendable mean episode lengths, its overall performance ranked it second, just after A2C. PPO lagged in all performance metrics, positioning it behind both A2C and DDPG. In summary, the A2C algorithm demonstrated superior performance in the given path planning logistics scenario, followed by DDPG, while PPO took a distant third place. It's imperative to note that the selection of an algorithm should consider specific requirements and the nature of tasks, but for this scenario, A2C has shown the best performance. It should be noted that further refinement is necessary to enhance the algorithm's performance in the given environment. Specifically, additional hyperparameter tuning, extended training durations, and optimization of the reward structure are requisite steps for achieving a higher success rate.

6 Conclusion

In the dawn of Industry 4.0, the increasing demand for autonomous mobile robots (AMR) in logistics, propelled by the quest for efficiency and a reduction in human error, has brought to the fore the intricacies of controlling expansive fleets of such robots. Navigating the dynamic terrains of logistics environments, these robots require agile, adaptable, and computational efficient control algorithms. Rein-

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

forcement learning (RL), especially in its decentralized incarnation, emerges as an attractive solution, bringing with it the promise of optimal strategies within unpredictable, evolving contexts.

The survey of literature reflected a growing consensus on the capabilities of RL in robot control, from drone navigation to multi-agent pathfinding (MAPF). Pioneers in this field have moved from traditional path-planning mechanisms to RL-based ones, which demonstrate significant promise in comparison. Among these, decentralization and a focus on discrete structures stood out as two discernible trends. The former offers scalability and adaptability, invaluable in the changeable arenas of logistics, while the latter, despite its computational advantages, imposes limits on the flexibility of the RL algorithms. In light of this, this work journeyed into the realms of continuous RL algorithms for single-agent robot control as a needed basis for decentralized fleet controls. The objective was clear: to lay the foundation for a system that is both decentralized and continuous, addressing the previously identified limitations and leveraging the strengths of RL. Our analysis of the A2C, DDPG, and PPO algorithms for controlling a Turtlebot3 in a logistics scenario produced interesting findings. In our exploration of continuous RL for single-agent robot control, we comparatively analyzed the A2C, DDPG, and PPO algorithms for the Turtlebot3 within a logistics context. A2C notably outperformed the others in terms of success rate and training time. DDPG followed closely in terms of rewards and episode lengths, while PPO's strength lay primarily in its computation time. Future avenues for research should delve deeper into hyperparameter tuning and reward structures that further optimize the Turtlebot3's success rate. Additionally, extending training over more timesteps and analyzing data efficiency will provide a more comprehensive understanding of these algorithms' potential.

In summary, our study emphasizes the promise of continuous RL algorithms, especially A2C, in managing AMRs in logistics. As the logistics industry continues its rapid transformation, the findings here offer valuable insights for the next generation of AMR fleet control solutions.

LITERATURE

[1] International Federation of Robotics. World Robotics Report 2020; International Federation of Robotics: 2020.

[2] International Federation of Robotics. Robot Sales Rise Again; International Federation of Robotics: 2021.

[3] Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 2019, 575, 350–354.

[4] Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; et al. Solving Rubik's Cube with a robot hand. arXiv 2019, arXiv:1910.07113.

[5] Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.K.S.; Koenig, S.; Choset, H. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. IEEE Robot. Autom. Lett. 2019, 4, 2378–2385. https://doi.org/10.1109/LRA.2019.2903261.

[6] Damani, M.; Luo, Z.; Wenzel, E.; Sartoretti, G. PRIMAL2: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning Lifelong. IEEE Robot. Autom. Lett. 2021, 6, 2666–267

[7] Golden, W. and Powell, P., 2000. Towards a definition of flexibility: in search of the Holy Grail?. Omega, 28(4), pp.373-384.

[8] Wesselhöft, M., Hinckeldeyn, J. and Kreutzfeldt, J., 2022. Controlling fleets of autonomous mobile robots with reinforcement learning: a brief survey. Robotics, 11(5), p.85.

[9] Mehta, D., 2020. State-of-the-art reinforcement learning algorithms. International Journal of Engineering Research and Technology, 8, pp.717-722.

[10] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[11] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M., 2014, January. Deterministic policy gradient algorithms. In International conference on machine learning (pp. 387-395). Pmlr.

[12] Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

[13] Hodge, V.J., Hawkins, R. and Alexander, R., 2021. Deep reinforcement learning for drone navigation using sensor data. Neural Computing and Applications, 33, pp.2015-2033.

[14] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in European Conference on Artificial Intelligence, pp. 961–962, 2014.

DOI: 10.2195/lj_proc_wesselhoeft_en_202310_01
URN: urn:nbn:de:0009-14-58257

[15] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in Robotics research, pp. 3–19. Springer, 2011.

[16] Chen, L., Wang, Y., Miao, Z., Mo, Y., Feng, M., Zhou, Z. and Wang, H., 2023. Transformer-based Imitative Reinforcement Learning for Multi-Robot Path Planning. IEEE Transactions on Industrial Informatics.

[17] Ma, Z., Luo, Y. and Ma, H., 2021, May. Distributed heuristic multi-agent path finding with communication. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 8699-8705). IEEE.

[18] Ma, Z., Luo, Y. and Pan, J., 2021. Learning selective communication for multi-agent path finding. IEEE Robotics and Automation Letters, 7(2), pp.1455-1462.

[19] Eliasmith, C. and Furlong, P.M., 2022, January. Continuous then discrete: A recommendation for building robotic brains. In Conference on Robot Learning (pp. 1758-1763). PMLR.

[20] Antonin Raffin and Ashley Hill and Adam Gleave and Anssi Kanervisto and Maximilian Ernestus and Noah Dormann, 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research, 22, (pp1-8)

Mike Wesselhöft, M.Sc., Research Assistant at the Institute of Technical Logistics, Technical University Hamburg. Mike Wesselhöft was born 1993 in Henstedt-Ulzburg, Germany. Between 2013 and 2019 he studied Mathematics at the University of Hamburg.

Address: Institut für Technische Logistik, Technische Universität Hamburg, Theodor-Yorck-Straße 8, 21073 Hamburg, Germany, E-Mail: mike.wesselhoeft@tuhh.de