2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE) | 978-1-6654-6280-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/LARS/SBR/WRE56824.2022.9996006

DA-SLAM: Deep Active SLAM based on Deep Reinforcement Learning

Martin Alcalde
Instituto de Computación
Facultad de Ingeniería
Montevideo, Uruguay
martin.alcalde@fing.edu.uy

Matias Ferreira
Instituto de Computación
Facultad de Ingeniería
Montevideo, Uruguay
mferreira@fing.edu.uy

Pablo González
Instituto de Computación
Facultad de Ingeniería
Montevideo, Uruguay
pablo.gonzalez.morelli@fing.edu.uy

Federico Andrade
Instituto de Computación
Facultad de Ingeniería
Montevideo, Uruguay
fandrade@fing.edu.uy

Gonzalo Tejera
Instituto de Computación
Facultad de Ingeniería
Montevideo, Uruguay
gtejera@fing.edu.uy

***Abstract*—This work presents improvements to the state-of-the-art algorithms for path planning and exploration of unknown and complex environments using Deep Reinforcement Learning. Our novel approach takes into consideration: (i) map information, built online by the robot using a Simultaneous Localization and Mapping algorithm and (ii) uncertainty of the robot's pose, which leads to active loop-closing to encourage exploration and better map generation within two agents. The results show that the map completeness-based reward function outperforms literature's results on shorter trajectories, thus, better performance; while uncertainty-based with loop-closing reward function improves map generation. Both agents showed the ability, to perform Active SLAM over complex environments and generalization to unseen maps capabilities.**

***Index Terms*—SLAM, Active SLAM, AI, Machine Learning, Computer Vision, Robotics, RL, DRL, PPO, Open AI, ROS**

I. INTRODUCTION

Autonomous exploration and mapping are one of the open challenges of robotics and artificial intelligence. Especially when environments are unknown, choosing the optimal navigation policy is not easy [1]. The Autonomous robot's navigation problem is usually tackled using route planning algorithms based on representations of the environments: the maps. These maps are usually built using Simultaneous Localization and Mapping (SLAM) algorithms [5]. However, in many applications like surveillance, cleaning and searching, a complete map of the environment is expensive to obtain or difficult to keep up to date.

SLAM refers to the problem of incrementally building the map of a previously unseen environment while at the same time locating the robot on it [10].

In the last few years, Reinforcement Learning (RL) [15] has become more relevant to address and solve several different robotics tasks, such as stabilization, manipulation, locomotion, and navigation. In the context of navigation, RL-based route planners do not typically rely on any maps and, while very successful, do not take advantage of the important information stored in them.

Within the classifications of the SLAM problem is Active SLAM. Active SLAM aims to find the sequence of control outputs, i.e. actions, to fully explore and build the maps of the environments. This sequential decision-making process can be formulated as a Reinforcement Learning (RL) problem, where the RL agent has to learn the actions that maximize the accumulated reward. Regarding the problem of Active SLAM with RL, there is a need to have reward functions that allow to reduce map uncertainty and motivate map exploration.

Over the past few years, the basics of Deep Reinforcement Learning (DRL) have proved to have great potential in navigation, mapping and even exploration tasks. In this way, one of the main drawbacks of traditional Active SLAM methods is relaxed by transferring the intensive computations to the Deep Neural Network (DNN) training phase, requiring only feed-forward propagation during evaluation [10].

It is important that the agent learns to navigate in complex environments since in these it can learn most of the necessary skills to navigate in other environments. Having the map information allows the agent to have the uncertainty value quite accurately at all times [10]. Rewarding low uncertainty has the advantage that the agent prioritizes not getting lost in map generation, which results in a more robust mapping and more accurate maps.

To the best of the authors' knowledge, this is the first approach including uncertainty-based reward and map information over topologically complex environments.

The objective of this research is to present improvements to the state-of-the-art algorithms of Active SLAM that use Deep Reinforcement Learning (DRL). The main contributions of the Active SLAM DRL-based solution presented in this work are:

[1]Code gitlab: **https://gitlab.fing.edu.uy/matias.ferreira/da-slam**
[2]Demo video: **https://youtu.be/lR7QvRIdJkI**

978-1-6654-0761-8/21/$31.00 ©2022 IEEE

- Two Active SLAM DRL agents, which became capable of making quasi-optimal decisions on unseen environments.
- Agents navigate and explore environments whose complexity exceeds the presented by state-of-the-art literature [1] [3] [10].
- Only 5 LiDAR data points were used (front view), which reduces the dimension of the observation space. This has several advantages: avoiding the curse of dimensionality, reducing computational cost, and faster training.

The experimental results show improvements in the performance of the proposed agents concerning other state-of-the-art approaches while decreasing the dimension of the observation space as well as increasing the precision of the maps obtained.

This section provided a brief but comprehensive introduction to the problem tackled. The rest of the paper is organized as follows. In section II we make a brief mention of related works. The problem statement is described in section III. Our solution, including reward function, observation space, action space, and state space, is introduced in section IV. The experimental setup is in Section V, while obtained results and their discussion is shown in Section VI. Finally, the paper's conclusions are in Section VII, and the future work in Section VIII.

II. Related Work

In the context of path planning algorithms based on DRL for navigation in unknown environments, Bottegi et. al [3] present the idea of shaping the reward function based on the online-acquired knowledge gained by the robot about the environment during the training phase. This approach attempts to enhance the obstacle awareness of the agent and relies on 40-dimensional raw laser data, odometry information and continuous action space. The RL algorithm chosen for the high-level controller of the robot is Deep Deterministic Policy Gradient (DDPG). In the same way, another work by the same author [2] introduces one of the first approaches in which the map's entropy, built online by the robot using a Simultaneous Localization and Mapping algorithm, is utilized during the training phase to shape the reward function. Map's entropy is used to improve the exploration skills of the agent and its ability to escape local minima that occur when the environments are more complex and realistic. In this case, they rely on 100 LiDAR data points of the 360° range, continuous action space and also chose DDPG as the RL algorithm.

In another approach to solve Active SLAM by using model-free DRL, Placed et. al [10] add true uncertainty metrics in the reward function that stem from traditional Theory of Optimal Experimental Design (TOED). This approach is based on sensory data obtained from n rays equally distributed in the 180° front field of view of the robot and a discrete action space (forward, turn right, turn left). Even though they were able to truly perform Active SLAM generating quasi-optimal trajectories and proved to have a generalizable approach, the algorithm didn't manage very well over complex scenarios.

In most recent works focused on improving exploration of unknown environments, Bottegi et. al [1] present an agent trained to be curious about the world. This concept translates into the introduction of a curiosity-based reward function that encourages the agent to visit unknown and unseen areas of the world and the map. The RL algorithm (DDPG) relies on 80 LiDAR data points uniformly spread over a 360° range and the percentage of the map to be explored between other input data, and action space is continuous. Results show that the agent trained with a curiosity-based reward function outperforms (in terms of generalization to untrained environments, map-completeness, and trajectory length and smoothness) the agents trained with commonly used reward functions for such tasks.

In the context of reducing uncertainty and improving map generation, Stachniss et. al [14] introduces an algorithm for generating trajectories to actively close loops during SLAM. This approach forces the robot to traverse previously visited loops again, reducing the uncertainty in the pose estimation and, as a result, obtains more accurate maps compared to standard combinations of SLAM algorithms with exploration techniques.

Approaches presented before show that trained agents became capable not only of learning a policy to navigate and explore in the absence of an environment model but also of transferring their knowledge to previously unseen maps, a key requirement to solve Active SLAM. On the other hand, their main limitation relies on the topological complexity of environments, they are all based on simple ones.

This paper presents algorithms to solve the Active SLAM paradigm with model-free DRL in complex environments combining both navigation and exploration capabilities. Our approach takes into consideration: (i) map completeness and (ii) uncertainty of the robot's pose (which leads to active loop-closing). Loop-closing refers to the situation in which the robot must be able to recognize a place that has already been visited; this helps the robot improve the entropy of its pose, thus obtaining a lower average entropy which allows the generation of a better map. For that purpose we, developed two Active SLAM DRL-based agents trained and evaluated over environments whose complexity exceeds the presented by state-of-the-art literature. Both agents, trained in a single environment, became capable of making quasi-optimal decisions regarding the navigation of unseen environments. To achieve that, we used only 5 LiDAR data points equally distributed in the 180° front field of view of the robot and defined a discrete action space (forward, turn left, turn right). This reduced input data leads to computational performance improvements in the RL algorithm, thus faster training.

Finally, empirical results show our agents' ability to perform Active SLAM over complex environments and, also, outperform literature's results as follows: (i) map completeness-based agent gets shorter trajectories, which implies better performance; while (ii) uncertainty-based agent with active loop-closing improves complex exploration, leading to better map generation.

III. PROBLEM STATEMENT

In this research, the problem of active exploration and mapping in a two-dimensional indoor home-like environment is addressed. The problem is modeled as a Partially Observable Markov Decision Process (POMDP) [15] defined by a six-element tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \omega)$ where: (i) $\mathcal{S}$ is the state space that can not be observed, (ii) $\mathcal{A}$ is defined as three actions discrete action space, (iii) $\mathcal{P}$ is the transition probability, (iv) $\mathcal{R}$ is the reward function, (v) $\mathcal{O}$ is defined as a continuous observation space based on laser (LiDAR), and (vi) ω is the observation model. The robot's goal is to maximize the expected return, as shown in equation 1,

$$E[\sum_{t=0}^{\inf} \gamma^t r_t] \tag{1}$$

where r_t is the robot's return at time t and γ is the discount factor.

IV. DEEP ACTIVE SLAM SOLUTION

In this section, our solution proposal is presented. The solution is based on a DNN trained with a RL paradigm. The system aims to learn a policy π, which leads the robot to explore the environment and keep the pose uncertainty low. The rest of this section shows the main components of the proposed solution.

A. Reward function

The design of the reward function is an important aspect of RL, as it should incorporate task-specific knowledge that the agent uses to learn the optimal behavior. Here we present two different reward functions.

The first one (2) is:

$$\mathcal{MC}(s_t) = \begin{cases} r_{crashed} & \text{if collision} \\ c_t - c_{t-1} & \text{otherwise.} \end{cases} \tag{2}$$

where $r_{crashed}$ is a negative scalar ($r_{crashed} = -100$) if the terminal collision state is reached and c_t is the map completeness at time t and c_{t-1} is the map completeness at time $t-1$ [4]. Map completeness is calculated at each step by obtaining the percentage of cells that are not -1 in the occupancy grid.

The second one (3) [10] is:

$$\mathcal{UB}(s_t) = \begin{cases} r_{crashed} & \text{if collision} \\ 1 + \tanh\left(\frac{\eta}{f(\Sigma)}\right) & \text{otherwise} \end{cases} \tag{3}$$

where $r_{crashed}$ is a negative scalar ($r_{crashed} = -100$) if the terminal collision state is reached, η is a scalar ($\eta = 1$) and $f(\Sigma)$ is the D-optimally criterion [10].

B. Observation space

Of the sensed data provided by the 360 available laser sensors, we consider (in the observation space) only 5 frontal sensors corresponding to the following angles: 0, 45, 90, 225 (-45), 270 (-90).

Be $O = [o_0, o_{45}, o_{90}, o_{225}, o_{270}]$ the observation space vector where o_x is the distance value sensed by the laser from position x, and max_{range} is the maximum distance value that could be sensed by the laser, then it's considered O_N as the normalized observation space vector, where each of the entries of O is divided by max_{range}, as a result, all its entries will be between 0 and 1.

Normalizing the observation space is very important because increases convergence speed, aids computer precision, prevents divergence of parameters, allows for easier hyper-parameter tuning, etc. Experimentally we couldn't achieve convergence if we didn't normalize.

C. Action space

In order to keep the system and training phase simple, we decided to use a discrete action space, which makes the use of a continuous action space unnecessary. Actions defined were:

- forward: the linear speed is 0.5 m/s and the angular speed is 0.
- turnleft: the linear speed is 0.05 m/s and the angular speed is 0.3 m/s.
- turnright: the linear speed is 0.05 m/s and the angular speed is -0.3 m/s.

D. State space

The agent selects actions based on the information contained within the state vector s_t at time t (equation 4):

$$s_t = [O_t, a_{t-1}, r_{t-1}] \tag{4}$$

where O_t is the normalized observation space vector, a_{t-1} is the previous action chosen by the robot and r_{t-1} is the previous percentage of completeness of the map in the case of the agent with $\mathcal{MC}$ reward function or the previous value of entropy in the case of the agent with $\mathcal{UB}$ reward function.

V. EXPERIMENTS DESIGN

In order to evaluate the proposed approach, several experiments have been conducted in a 3D simulation environment Gazebo robotics simulator [7] on top of Robot Operating System (ROS) [12], with a differential drive mobile robot ROBOTIS TurtleBot3-Burger, controlled through velocity commands and equipped with 360 laser range scanner (LiDAR) and wheels odometry. The LiDAR range is between 0.12m and 3.5m.

The general architecture presented in figure 1 shows how the ROS Noetic Ninjemys framework has been used to connect the simulation environments with the Passive SLAM algorithm [16] and the decision-making module, by using openai_ros [9] library, based on OpenAI [8] Gym. For the DRL algorithm we use Stable-Baselines3 [13].

A. Environments design

Four different environments (figure 2) have been used during the training and testing stages. The first one, where the agent was trained, consists of a maze with several obstacles and high symmetry. The second is a simple maze, the third

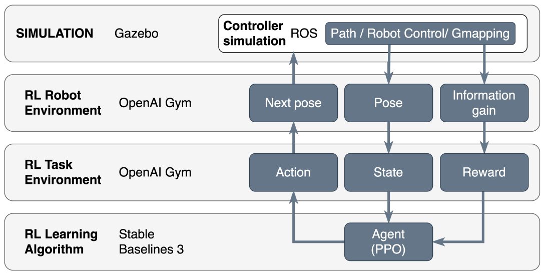
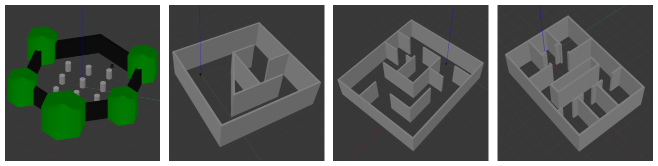


Fig. 1: Architecture



(a) *Env-1* (b) *Env-2* (c) *Env-3* (d) *Env-4*

Fig. 2: Environments

and fourth environments have a more complex topology and also are very symmetrical, including consecutive turnarounds and dead-ends. The Last three described environments have been used during testing phase only. It Is important to mention that *Env-3* and *Env-4* present higher complexity than the environments used in the state-of-the-art articles [1] [3] [10].

B. Decision Making Module

Decision-making has been done via DRL using Proximal Policy Optimization algorithm (PPO)[11]. Experimentally, we found that PPO had the most stable training and reliable outcomes, therefore we choose PPO as our DRL algorithm. Since we used Stable-Baselines3 implementation, we will not discuss its details. The main hyper-parameters used are the defaults provided by Stable-Baselines3 [13].

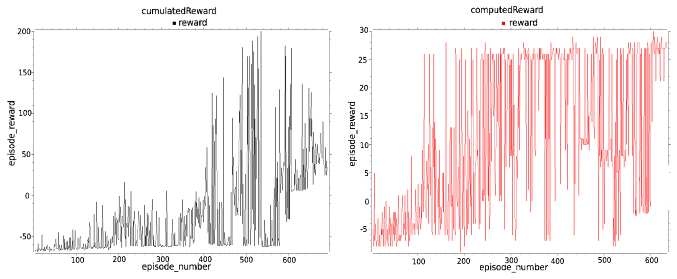
C. Training

A training/testing stage consists of a certain number of episodes, in which in each episode, the robot moves until either a collision occurs (i.e., gets closer to any obstacle than a defined threshold of 0.12 m) or 1000 steps (i.e., decisions) are consumed.

Agents were trained in *Env-1* (Height: 5.6m, Width: 5.0m), with a total of 6x10^6 time steps and 1000 maximum steps per episode. In figure 3 the accumulated reward graphs of the training are shown. The number of episodes is assigned on the x-axis (episode_number) and the reward for each episode is assigned on the y-axis (episode_reward).

D. Evaluation

Trained agents were evaluated in *Env-2* (Height: 7.5m, Width: 7.5m), *Env-3* (Height: 10.2m, Width: 8.9m) and *Env-4* (Height: 8.8m, Width: 5.8m) based on the path length, time, the uncertainty average of the distribution over the robot's pose, and the map completeness percentage of every episode.



(a) Completeness agent (b) Uncertainty-based agent

Fig. 3: Accumulated reward

TABLE I: Training results in *Env-1*.

Agent	Path length (m)	Time (s)	AVG Uncertainty	Map completeness (%)
Completeness-based	12.83	105.61	4.78	98.9
Uncertainty-based	20.14	164.71	4.59	100

Furthermore, we compared our trained agents with frontier-based exploration [17] and an advanced Active SLAM agent (EALC) that combines exploration and active loop-closing based on [14].

EALC uses frontier-based exploration to explore the map. In parallel, the topological map $\mathcal{G}^{[x]}$ is stored, where x_0 is the initial pose of the robot. Each node $x^{[t]}$ of $\mathcal{G}$ has assigned the entropy of the pose, the real distance to $x^{[t-1]}$ and the topological distance to $x^{[t-1]}$. If exists at least one frontier to explore, then, to determine whether a loop can be closed we compute the set $\mathcal{I}$ of poses of interest, which contains all nodes that are close to current pose x_t for the real distance but are far away given the topological $\mathcal{G}^{[x]}$ and the entropy is higher than the entropy of x_t. If $\mathcal{I} \neq \emptyset$, then the robot goes to the node with the closest real distance and then the pose is stored to $\mathcal{G}^{[x]}$; else the robot goes to the pose provided by frontier-based. Finally, if there is no frontier to explore, the robot re-explores each pose of the inverse of $\mathcal{G}^{[x]}$ where the entropy was bigger than the actual entropy average.

We use ROS package explore-lite [6] for the implementation of frontier-based exploration.

VI. RESULTS

A. Training results

After the training, the completeness-based agent and the uncertainty-based agent were evaluated 10 times each inside the *Env-1* as a way to validate the training phase. Table I shows the results.

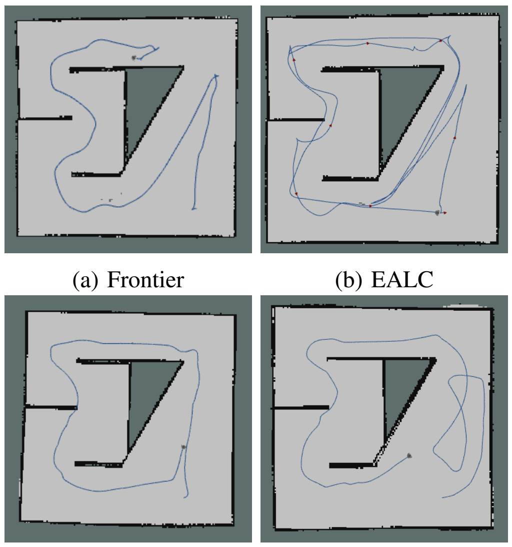
From these results, it can be said that both agents managed to complete the map almost completely, avoiding collisions.

The time it takes to end the map and the length of the path generated by the uncertainty-based agent is bigger than the one generated by the completeness agent. However, the average uncertainty is smaller and the generated map is better.

Even on this first map, it can be seen the difference in the objectives of the agents. The completeness-based agent tries to complete the map as quickly as possible, while the uncertainty-

TABLE II: Evaluation results in *Env-2*, *Env-3* and *Env-4*.

Env	Agent	Path length (m)	Time (s)	AVG Uncertainty	Map completeness (%)
2	Frontier	25.02	134-69	4.67	100
	EALC	55.61	293.49	4.05	100
	Completeness-based	20.34	181.53	4.76	100
	Uncertainty-based	30.69	241.42	4.57	100
3	Frontier	76.04	397.68	4.20	100
	EALC	113.01	717.91	4.05	100
	Completeness-based	37.32	294.19	4.78	95.3
	Uncertainty-based	139.15	921.52	4.51	100
4	Frontier	29.95	179.33	4.72	99.02
	EALC	50.92	293.77	4.11	99.28
	Completeness-based	42.67	325.77	4.50	100
	Uncertainty-based	63.41	412.12	4.52	100


(a) Frontier (b) EALC

(c) Completeness-based agent (d) Uncertainty-based agent

Fig. 4: Evaluation in *Env-2*

based revisits places that have been already seen to lower the uncertainty.

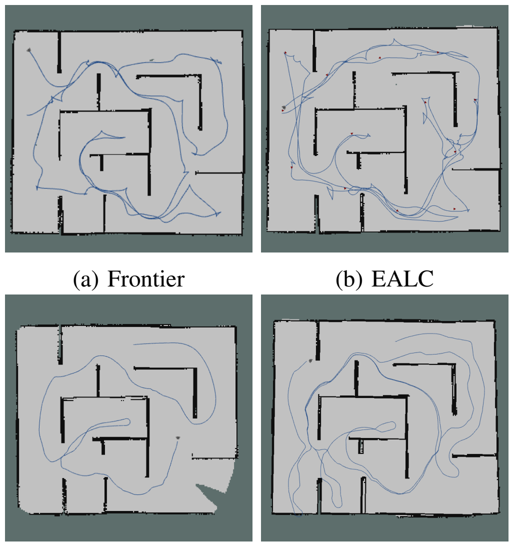
B. *Evaluation tests*

The agents were evaluated 10 times each inside the *Env-2*, *Env-3* and *Env-4*. Table II shows the results in *Env-2*, *Env-3* and *Env-4*. Figure 4 shows the maps with their respective generated paths in *Env-2*. Figure 5 shows the maps with their respective generated paths in *Env-3*. Figure 6 shows the maps with their respective generated paths in *Env-4*. In each map, the end position of the agent is represented by the black figure (the start position is on the other end of the path). In EALC-generated maps, there is also red dots which are the points of the topological map.

From these results, it could be said that both agents managed to learn to perform Active SLAM over complex environments and generalization to unseen maps.

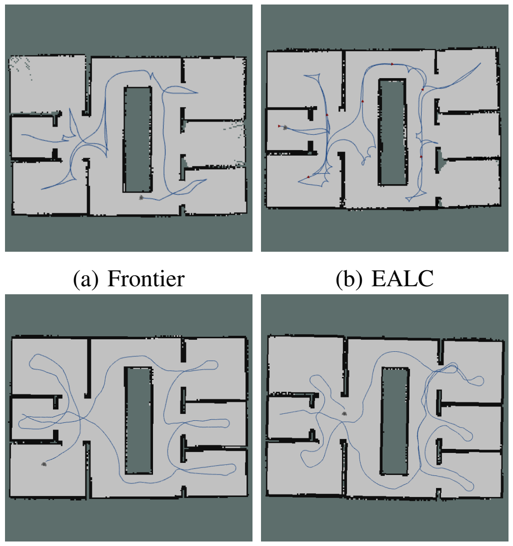
The map completeness-based agent has a shorter path and takes less time to complete the map. However, the uncertainty-based agent has a lower average pose uncertainty, that is because it frequently visits already visited positions, resulting in the generation of more accurate maps.

This difference in the paths relates to the different objectives of the agents, the completeness-based agent tries to complete


(a) Frontier (b) EALC

(c) Completeness-based agent (d) Uncertainty-based agent

Fig. 5: Evaluation in *Env-3*


(a) Frontier (b) EALC

(c) Completeness-based agent (d) Uncertainty-based agent

Fig. 6: Evaluation in *Env-4*

the maps as quickly as it can, while the uncertainty-based tries to not get lost in the map it's generating.

Although there is no tool to compare the maps, the reader can do a visual analysis and will notice a better map generated by the uncertainty-based agent.

One thing to consider is that frontier-based exploration and EALC are not concerned about optimal paths, their objective is just to reach the largest frontier.

If after shrinking the current frontier it is exploring, it finds out that there is a larger frontier than the current one, on the other side of the map, the robot will go to that position, resulting in the robot going back and forth from one end of the map to the other.

In that sense, a completeness-based agent generates shorter paths as the map is larger and more complex and therefore has less exploration time. Finally, the uncertainty-based agent has a better average uncertainty than frontier-based exploration and the generated map closely resembles that of EALC.

VII. Conclusions

In this paper, we have presented a novel approach to solve the Active SLAM paradigm by using model-free DRL. We have embedded a state-of-the-art Proximal Policy Optimization architecture on top of the Gazebo simulator, and executed a lightweight Passive SLAM algorithm back-end that allowed the retrieval of the robot's uncertainty and therefore, the computation of D-optimality. Thus, we were able to truly perform Active SLAM in complex environments with realistic models for the robot, the sensors and the environment physics. As far as we know, there are no Active SLAM agents trained with DRL that have been evaluated with good results in scenarios of the difficulty presented in this work.

We presented two DRL agents with different reward function approaches. Both agents learned a policy that allows them to avoid collisions to complete the map, but the agent with a reward function based on map completeness learned short paths to complete the map. Learning a short path implies that the time to complete the map is less, but this does not mean that the resulting map is better. The agent with an uncertainty-based reward function has a better resulting map, this is due to the fact that the agent tries to not get lost in the map (its reward function uses the D-opt and the pose entropy is embedded in the observation space). Thanks to this, the agent not only learns to avoid collisions and complete the map but also learns to close loops.

Both DRL agents showed performance comparable or better than EALC which is an advanced Active SLAM agent developed for comparison proposes.

Something to highlight is the case of *Env-4*, where due to the design of the map, the robot tends to involuntarily close loops when entering and leaving rooms. This causes both agents to behave similarly in terms of average entropy and resulting map, maintaining a shorter path and less time for the map completeness agent. This is the only special case where the map completeness agent has better overall performance than the uncertainty-based agent.

Also, agents were trained in a single environment and showed, a posteriori, the ability to transfer the acquired knowledge to previously unseen maps, which is a key requirement to solve Active SLAM.

VIII. Future Work

In future work, there are several desirable lines of work like creating an agent that aggregates the two proposed ones (seeking to obtain a good map with shorter paths), re training on complex environments and testing them in real-world environments.

References

[1] N. Botteghi et al. "CURIOSITY-DRIVEN REINFORCEMENT LEARNING AGENT FOR MAPPING UNKNOWN INDOOR ENVIRONMENTS". In: *ISPRS* V-1-2021 (2021), pp. 129–136.

[2] N. Botteghi et al. "Entropy-Based Exploration for Mobile Robot Navigation: A Learning-Based Approach". English. In: PlanRob 2020; Conference date: 22-10-2020 Through 23-10-2020. 2020.

[3] N. Botteghi et al. *On reward shaping for mobile robot navigation: A reinforcement learning and SLAM based approach*. English. WorkingPaper. arXiv.org, 2020.

[4] N. Botteghi et al. "Reinforcement learning helps slam: Learning to build maps". English. In: 43.B4 (Aug. 2020). XXIVth ISPRS Congress 2020, ISPRS 2020 ; Conference date: 04-07-2020 Through 10-07-2020, pp. 329–336. ISSN: 1682-1750.

[5] Wolfram Burgard and Dieter Fox. "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)". In: Jan. 2005. ISBN: 0-262-20162-3.

[6] *explore_lite - ROS Wiki*. http://wiki.ros.org/explore_lite. (Visited on 02/16/2022).

[7] *Gazebo*. https://gazebosim.org/home. (Visited on 02/16/2022).

[8] *Gym Documentation*. https://www.gymlibrary.ml/. (Visited on 02/16/2022).

[9] *openai_ros - ROS Wiki*. http://wiki.ros.org/openai_ros. (Visited on 02/16/2022).

[10] Julio A. Placed and José A. Castellanos. "A Deep Reinforcement Learning Approach for Active SLAM". In: *Applied Sciences* 10.23 (2020). ISSN: 2076-3417.

[11] *PPO - Stable Baselines3 1.5.1a8 documentation*. https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html. (Visited on 02/16/2022).

[12] *ROS: Home*. https://www.ros.org/. (Visited on 02/16/2022).

[13] *Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 1.5.1a8 documentation*. https://stable-baselines3.readthedocs.io/en/master/. (Visited on 02/16/2022).

[14] C. Stachniss, D. Hahnel, and W. Burgard. "Exploration with active loop-closing for FastSLAM". In: *2004 IEEE/RSJ IROS (IEEE Cat. No.04CH37566)*. Vol. 2. 2004, 1505–1510 vol.2.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[16] *turtlebot3_slam - ROS Wiki*. http://wiki.ros.org/turtlebot3_slam. (Visited on 02/16/2022).

[17] B. Yamauchi. "A frontier-based approach for autonomous exploration". In: *Proceedings 1997 IEEE CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. 1997, pp. 146–151.