CURIOSITY-DRIVEN REINFORCEMENT LEARNING AGENT FOR MAPPING UNKNOWN INDOOR ENVIRONMENTS

N. Botteghi[1]*, R. Schulte[1], B. Sirmacek[2*], M. Poel[3], C. Brune[4]

[1]Robotics and Mechatronics, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands (e-mail: n.botteghi@utwente.nl)
[2]Smart Cities, School of Creative Technology, Saxion University of Applied Sciences, The Netherlands (e-mail: b.sirmacek@saxion.nl)
[3]Datamanagement and Biometrics, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands (e-mail: m.poel@utwente.nl)
[4]Applied Analysis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands (e-mail: c.brune@utwente.nl)

KEY WORDS: Reinforcement Learning, Simultaneous Localization and Mapping, Mobile Robotics, Indoor Mapping

ABSTRACT:

Autonomously exploring and mapping is one of the open challenges of robotics and artificial intelligence. Especially when the environments are unknown, choosing the optimal navigation directive is not straightforward. In this paper, we propose a reinforcement learning framework for navigating, exploring, and mapping unknown environments. The reinforcement learning agent is in charge of selecting the commands for steering the mobile robot, while a SLAM algorithm estimates the robot pose and maps the environments. The agent, to select optimal actions, is trained to be *curious* about the world. This concept translates into the introduction of a curiosity-driven reward function that encourages the agent to steer the mobile robot towards unknown and unseen areas of the world and the map. We test our approach in explorations challenges in different indoor environments. The agent trained with the proposed reward function outperforms the agents trained with reward functions commonly used in the literature for solving such tasks.

1. INTRODUCTION

The problem of autonomous robot navigation is traditionally tackled by employing environment representations, i.e. maps, that are used to plan a collision-free path to reach specific target locations. These indoor maps are usually constructed using Simultaneous Localization and Mapping, or SLAM, algorithms (Thrun et al., 2005). SLAM algorithms estimate the robot sensor's location and construct the map of the environment simultaneously using a sequence of sensor measurements, with the underlying assumption of a known and pre-defined sequence of control commands or actions, e.g. velocity commands for a mobile robot carrying the sensor. The indoor maps are not always available beforehand, and even when maps are known, they do not include, for example, details about obstacles in a dynamic environment.

Therefore, especially in the last decade, thanks to the advances in reinforcement learning (Sutton and Barto, 2018) map-less path planning solutions took the attention of the scientists. Successful reinforcement learning-based solutions are proposed in (Wu et al., 2018), (Tai et al., 2017), (Zhelo et al., 2018), (Pfeiffer et al., 2017),(Zhang et al., 2018), and (Zhang et al., 2020). However, such map-less path planners often require long training times and a high amount of data to perform well. Moreover, when maps are available, reinforcement learning-based planners can benefit from the information contained in the maps and often outperforms map-less approaches. Examples of learning-based planners relying, fully or partially, on maps are proposed in (Zhang et al., 2016), (Brunner et al., 2017) and (Mustafa et al., 2019).

When the environments are unknown, efficiently constructing the maps by selecting the optimal sequence of control actions is a very challenging problem. This is commonly referred to in literature as the *active* SLAM problem, in which not only pose and map are estimated, but also the control actions are chosen in order to explore the whole environment. The simplest solution one could think of is to rely on pre-coded navigational directives based on landmark positions and/or specific features, e.g. turn left when recognizing a certain landmark such as the corner of a room. However, coding these directives is problematic if the environment is not known beforehand or not static and often these solutions are brittle and struggle to adapt to environments with different topologies. Another approach for active SLAM is the so-called frontier-based exploration (Yamauchi, 1997a). This method relies on the identification of special navigational targets, i.e. the frontiers, sitting in-between the known and unknown regions of the map, selecting the most promising[1] one and eventually navigate to it. Once the chosen frontier is reached, the procedure is repeated until all the frontiers are visited and the map is complete. This approach is mostly suitable for occupancy grid-based maps, where each cell of the grid corresponds to a portion of the environment. Each cell of the grid map is either classified as free, occupied, or unknown. However, the overall path the robot follows to complete the map is usually not optimal in terms of distance traveled, even in the case the closest frontier is repeatedly selected.

As in the case of path planning, reinforcement learning-based methods can be used to select the control commands to steer the robot for fully exploring unknown environments such that SLAM algorithms can reconstruct their maps. However, the problem of exploratory navigation for constructing maps is more challenging, than the navigation-to-a-target problem. Because

* Corresponding authors

[1] The most promising frontier can be chosen in different ways, such as the closest frontier, the most informative one or combinations of the two objectives.

there is not an explicit navigation target, the reinforcement learning agent needs to learn to explore. Defining a suitable non-sparse reward function is not straightforward in this case. The problem of learning to explore, in the context of reinforcement learning, can be tackled by reward shaping. The goal is to reward the agent to be *curious*, i.e. to choose actions that allow traveling to novel or never-visited states of the environment. This is usually referred to in literature as *curiosity-driven* reinforcement learning (Pathak et al., 2017).

In this work, we propose a reinforcement learning and SLAM-based framework for efficiently exploring and building maps of unknown indoor environments based on the robot's on-board sensory readings and the information coming from the SLAM algorithm, such as the pose estimate and the map's completeness. The proposed approach is shown in Figure 1. In particular, we introduce a curiosity-driven reward function dependent on the novelty of the position visited by the robot that allows fast learning of the exploration policy and its generalization to untrained environments. Additionally, the proposed reward function is also independent of the type of map, either feature-based or location-based map, built using the SLAM algorithm.

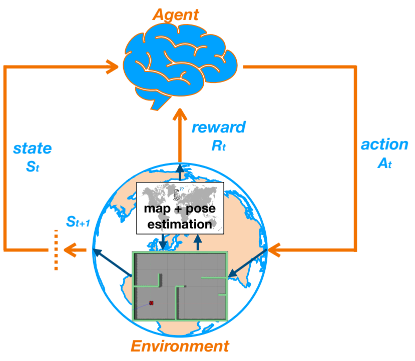


Figure 1. Proposed framework combining reinforcement learning and SLAM where the reconstructed 2D map and the pose estimation of the SLAM algorithm contributes to both state estimation and reward shaping process. As a result, the agent can choose the best action to steer the robot and to support the continuation of the active SLAM algorithm.

The paper is organized as follows: Section 2 introduces the background information regarding reinforcement learning and SLAM, Section 3 discusses the related work to this research and Section 4 presents the proposed approach. Then, Section 5 shows the experimental design, followed by Section 6 presenting the result and discusses the findings. Eventually, Section 7 concludes the paper.

2. BACKGROUND

2.1 Reinforcement Learning

Reinforcement learning, or RL, (Sutton and Barto, 2018) studies the problem of learning optimal behaviors through the interaction, occurring at discrete time-steps, of the agent, i.e. the learning entity, with an unknown world, i.e. the environment. At each time-step t, the agent is in a certain state $s_t \in \mathcal{S}$, determined by the information perceived from the environment, performs an action $a_t \in \mathcal{A}$ and receives a scalar reward r_t signal assessing the quality of the action taken. The goal of a reinforcement learning agent is to find the best policy for solving a given task by accumulating the highest total reward at the end of the run $\sum_{k=0}^{N} \gamma^k r_{t+k}$.

2.1.1 Deep Deterministic Policy Gradient

Deep deterministic policy gradient, or DDPG, (Lillicrap et al., 2015), is an actor-critic reinforcement learning algorithm that can deal with problems with continuous state and action spaces. DDPG estimates the action-value function Q, or critic, and a deterministic policy π, or actor, using two neural networks. The deterministic policy $\pi(s;\theta^\pi)$, parametrized by the parameter vector θ^π, is made stochastic during the training phase of the algorithm with the addition of Ornstein–Uhlenbeck noise(Uhlenbeck and Ornstein, 1930). The actor network $\pi(s;\theta^\pi)$ is updated with the deterministic policy gradient theorem and it is shown in Equation (1).

$$\nabla_\theta J(\theta^\theta) = \mathbb{E}_{s\sim\rho,a\sim\pi}[\nabla_{\theta^\pi}\pi_\theta(s;\theta^\pi)\nabla_a \mathrm{Q}(s,a;\theta^{\mathrm{Q}})] \quad (1)$$

where ρ is the state distribution induced by the policy π and a is the action chosen accordingly to the policy π.

The critic network $\mathrm{Q}(s,a;\theta^{\mathrm{Q}})$, parametrized by the parameter vector θ^{Q}, is updated using the temporal-difference error computed by a target network $\mathrm{Q}^-(s,a;\theta^{\mathrm{Q}^-})$ with the same approach used in (Mnih et al., 2015). The loss for updating the critic network is shown in Equation (2).

$$\mathcal{L}(\theta^{\mathrm{Q}}) = \mathbb{E}_{s\sim\rho,a\sim\pi}[(r(s,a)+\gamma\mathrm{Q}'(s',a',;\theta^{\mathrm{Q}^-})-\mathrm{Q}(s,a;\theta^{\mathrm{Q}}))^2] \quad (2)$$

where ρ is the state distribution induced by the policy π and $a' = \pi^-(s';\theta^{\pi^-})$ is the next action and it is chosen accordingly to the target policy network $\pi^-(s';\theta^{\pi^-})$.

We update the target networks with soft updates using the exponential moving average of the original parameters θ, as in Equation (3).

$$\theta_{\text{ema}} = (1-\lambda)\sum_{i=0}^{N}\lambda^i\theta_{N-i} \quad (3)$$

where $\lambda \in [0,1]$ is the decay rate and θ_N are the neural network weights after N update steps.

2.1.2 Exploration by Reward Shaping

The trade-off between exploration and exploitation is one of the biggest challenges in reinforcement learning, the agent has to explore to find higher rewards, but its ultimate goal is to exploit and collect the highest possible rewards. Especially in all the situations in which the environment is complex and a sparse reward function is used, random exploration is not sufficient, and improving the policy requires a long training time.

It is possible to formulate curiosity-driven reward functions that motivate the agent to explore uncertain regions of the state space, e.g. states that are badly predicted, accordingly to a learned forward dynamical model, as in (Pathak et al., 2017) and (Zhelo et al., 2018). These approaches drastically improve the convergence of the algorithms and can solve sequential decision making processes even in the cases of very sparse reward functions. When policies are learned through such reward functions, these tend to indiscriminately explore the whole state space and they do not necessarily focus on exploring more interesting areas. This may be a limitation in very large environments. Moreover, if the environment is highly stochastic or random, the forward model can only poorly predict the next state and, consequently, the reward is not informative enough to allow good exploration and fast policy learning (Burda et al., 2018).

Alternatively, the novelty can be defined in terms of reachability, i.e. how many steps the agent needs to reach a certain state (Savinov et al., 2018). This approach stores a special sub-set of novel observations and compares them with the most recent observation of the agent in order to determine the novelty of such observation. In their approach, the authors use a neural network to determine if the current observation can be reached within the k-steps from any of the stored observations and consequently generate a reward proportional to that distance. This method shows significant improvements in terms of convergence speed with respect to the forward model prediction and suffers less from the problem of the environment stochasticity.

2.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) algorithms are designed for creating 2D or 3D maps of the environment while simultaneously estimating the pose of the sensor used for the measurements. Depending on the measurement sensor of choice, the scale of the environment to be mapped, computational limitations, and level of details wanted from the mapping and pose estimation outputs, a large variety of SLAM methods are proposed in the literature (Wang et al., 2017). For indoor robot navigation problems, the most frequently chosen robust and fast methods have been; FastSLAM (Montemerlo et al., 2003), RGB-D SLAM (Endres et al., 2014), ORB-SLAM (Mur-Artal et al., 2015), GraphSLAM (Thrun and Montemerlo, 2005) and Rao Blackwellized particle filter (RBPF) (Murphy, 2000) approaches. Even though RGB-D SLAM provided fast and robust localization and detailed map generation possibilities, lower availability and higher prices of the RGB-D sensors made other SLAM algorithms more interesting for most of the researchers. FastSLAM and ORB-SLAM relied on the availability of trackable visual features in the environment. GraphSLAM and RBPF SLAM offered comparable performances even when different sensor measurements are involved (Kümmerle et al., 2009). GraphSLAM provided slightly better performance when the sensor turns back to the same starting point (called loop-closure). On the other hand, RBPF SLAM enabled storing the environment information in a smaller memory size which is very advantageous for mobile robot systems to run active SLAM and other complex decision making algorithms (such as path planning with a RL-based algorithm) real-time on a small embedded platform. Because of these mentioned advantages, in this study, we have used RBPF SLAM to solve the simultaneous localization and mapping problem of the mobile robot by building an occupancy grid-based map. However, we need to highlight that the proposed curiosity-driven reinforcement learning-based path planning framework is fully independent of the SLAM algorithm of choice as it requires only the pose estimate. The SLAM module of our framework can be assumed as a functional module that can easily be replaceable with another SLAM algorithm that provides the environment map and the sensor pose as a time series.

2.2.1 Rao-Blackwellized Particle Filter

The goal of any SLAM algorithm is to estimate the pose χ of the sensor and map m of the environment, given its measurements $z_{1:t}$ and control actions $a_{0:t-1}$:

$$p(\chi_{1:t}, m | z_{1:t}, a_{0:t-1}) \tag{4}$$

RBPF SLAM simplifies the estimation of the posterior probability, shown in Equation (4), by factoring it in two terms:

$$p(\chi_{1:t}, m | z_{1:t}, a_{0:t-1}) = p(\chi_{1:t} | z_{1:t}, a_{0:t-1}) p(m | \chi_{1:t}, z_{1:t}) \tag{5}$$

where $p(\chi_{1:t} | z_{1:t}, u_{0:t-1})$ is the probability of the trajectory conditioned to the sequence of measurements $z_{1:t}$ and control actions $a_{0:t-1}$ and $p(m | \chi_{1:t}, z_{1:t})$ is the probability of the map conditioned to the trajectory $\chi_{1:t}$, assumed to be known[2], and measurements $z_{1:t}$.

The sensor's pose $p(\chi_{1:t} | z_{1:t}, a_{t-1})$ is estimated by the RBPF using a finite set of N particles:

$$\mathcal{X}_t = \{\chi_t^{(1)}, \chi_t^{(2)}, ..., \chi_t^{(N)}\}, \tag{6}$$

where each particle $\chi_t^{(n)}$ represents the belief of what the true pose may be at time step t. Moreover, at each iteration of the particle filter algorithm, a new set of particles $\mathcal{X}_t$ is recursively computed from the previous set $\mathcal{X}_{t-1}$. To do so, first, a new set of particles $\mathcal{X}_t$ is sampled from the probabilistic motion model $s_t^{(n)} \sim p(\chi_t^{(n)} | \chi_{t-1}^{(n)}, a_{t-1})$ given the previous set of particles $\mathcal{X}_{t-1}$. Then, by means of the observation model $p(o_t | \chi_t^{(n)})$ the *importance* weight $w_t^{(n)}$ of each particle is computed. Eventually, we apply the so-called *selective resampling* (Grisetti et al., 2005) to sample a new set of N particles in accordance to their importance weight.

Our implementation of the RBPF SLAM algorithm utilizes an occupancy grid map representation. The untrackable posterior probability of the map $p(m | z_{1:t}, \chi_{1:t})$ can be computed as the product of the posterior of each cell m_i of the grid. This factorization is shown in Equation (7).

$$p(m | z_{1:t}, \chi_{1:t}) = \prod_{i=1}^{M} p(m_i | z_{1:t}, \chi_{1:t}) \tag{7}$$

where M is the total number of cells in the map.

3. RELATED WORK

3.1 Reinforcement Learning-Based Active SLAM in Robotics

Reinforcement learning has been used in active SLAM for learning optimal exploration strategies for steering a mobile robot, carrying sensors, in unknown environments, and constructing their maps. In (Kollar and Roy, 2008), reinforcement learning is used to find the most informative trajectories to reach a set of given target locations and to consequently, improve the quality of the map. However, differently from our work, we do

[2] This term is commonly known in literature as *mapping with known poses*.

not assume a known set of navigation targets, but we rely only on onboard sensor readings and information coming from the SLAM algorithm for steering the robot into the unknown areas of the environments.

A reinforcement learning-based active SLAM architecture is proposed in (Chen et al., 2019), where an agent is trained to explore the environments in order to build their maps by relying on raw RGB-D sensor readings, a top-view image of the map, and a bump sensor for detecting collisions. The approach relies on expert trajectories to initialize the agent's policy. Differently for them, we do not rely on expert data and high-dimensional sensory reading, but only on LiDAR. Additionally, the focus of our work is to design a reward function that encourages exploration.

A similar architecture is proposed in (Dooraki et al., 2018), where a reinforcement learning agent is trained on depth images and range sensory readings for navigating without collisions and constructs maps of different environments. The reward function is shaped using multiple objectives and ultimately promotes moving forward to free space. However, this reward function has no terms related to the map coverage, therefore the agent is not necessarily rewarded for exploring quickly and/or choosing short trajectories.

In (Botteghi et al., 2020) and (Placed and Castellanos, 2020), the authors propose a reinforcement learning approach for active SLAM that relies on 2D LiDAR readings and information coming from the SLAM algorithm to select the best steering command, out of a discrete action space, for a mobile robot to explore different environments. While we use similar state space definition, we employ a continuous action space for obtaining smoother trajectories. Both approaches focus on the shape of the reward function. Differently, in our work, we propose a new curiosity-based reward function that drastically improves the generalization skills of the agent.

Eventually, the authors of (Niroui et al., 2019) propose a hybrid reinforcement learning and frontier-based exploration framework, where the agent is not directly choosing velocity commands nor trajectories for the robot, but simply selects the frontier to visit. An A*-based path planner is then in charge of steering the robot to the selected frontier. Despite the success, the method reduces, only slightly, the total traveled distance by the robot with respect to the original frontier-based exploration approach (Yamauchi, 1997b). In our approach, we do not rely on the explicit computation of the frontier, which may be computationally expensive. Moreover, visiting all the frontiers may cause longer trajectories than necessary.

All these reinforcement learning-based approaches rely on random noise on the action space to explore action space and, consequently, explore the environment. We believe that is not enough for fast learning and generalization and we employ a curiosity-based reward function. The proposed reward function may be beneficial to the performance of all the mentioned approaches.

4. METHODOLOGY

4.1 Proposed approach

We aim at solving the active SLAM problem using reinforcement learning to select the best actions to explore indoor environments and construct their maps. We assume this problem to have a finite horizon, i.e. we assume the existence of a terminal state, reachable in a finite number of steps, corresponding to the environment being fully explored and the map being fully constructed. The reinforcement learning algorithm, i.e. DDPG, only relies on 80 2D-LiDAR readings, the robot's pose estimate coming from the RBPF SLAM algorithm, the previous action taken by the agent, the percentage of the map to be explored, and the time steps left before the end of the episode. The time steps left and the percentage of the map left before completeness help to make the algorithm aware of the terminal conditions and consequently help the performance and stability of the learning algorithms (Pardo et al., 2017). Moreover, these two assumptions allow us to study and tackle the problem without the need for an explicit memory structure in the reinforcement learning algorithm. The state vector s_t at a given time instant t is shown in Equation (8).

$$s_t = [z_t, \chi_t^{(x,y,\theta)}, a_{t-1}, \zeta_t^{\neg}, \tau_t^{\neg}] \tag{8}$$

where z_t corresponds to the LiDAR readings uniformly spread on a 360° range, $\chi_t^{(x,y,\theta)}$ to the pose estimate, i.e. x, y-position in the Cartesian plane and the robot's orientation θ, a_{t-1} to the previous action taken by the agent, $\zeta_t^{\neg}$ to the remaining map percentage before completeness and $\tau_t^{\neg}$ to the time steps left before the end of the episode. The agent's actions are continuous linear and angular velocity set-points that are sent and tracked by the low-level controllers on the robot.

4.2 Exploration by Reward Shaping

We propose an adaptation of the episodic curiosity reward introduced by (Savinov et al., 2018) for improving the exploration skills of the reinforcement learning algorithm in the context of active SLAM and we investigate its effect on the generalization skills of the trained agent to different environment topologies. We propose a reward function that combines a bonus when the environment is fully explored and the map is completed, a penalty when collisions occur, and a curiosity reward promoting exploration of unknown locations. The propose reward function, named *Curiosity*, is shown in Equation (9).

$$\mathrm{R}(s_t) = \begin{cases} r_{\text{map completed}}, & \text{if } \zeta_t \geq C \\ r_{\text{crashed}}, & \text{if } z_t \leq z_{\text{min}} \\ r_t^c, & \text{otherwise} \end{cases} \tag{9}$$

where $r_{\text{map completed}}$ is the bonus given to the agent when the percentage of map completeness ζ_t is above a threshold[3] C, r_{crashed} is a negative penalty for getting too close, accordingly to a fixed threshold z_{min}, to obstacles according the current LiDAR reading z_t and r_t^c is the curiosity term promoting exploration.

To compute the curiosity-based reward, we first define a finite-size buffer $\mathcal{M}$ with cardinality M containing novel robot's positions $\chi_t^{(x,y)}$, estimated via the SLAM algorithm. A position is novel if it is at least at a distance k from all the other positions visited, with k a parameter of our algorithm. If a novel position is found during exploration, this is added to the novelty buffer. A graphical illustration of the curiosity reward is shown in Figure 2.

[3] Due to small mapping error, the map is rarely completed exactly at 100%, therefore we set the map-completeness threshold to 93% to prevent never reaching such a condition.

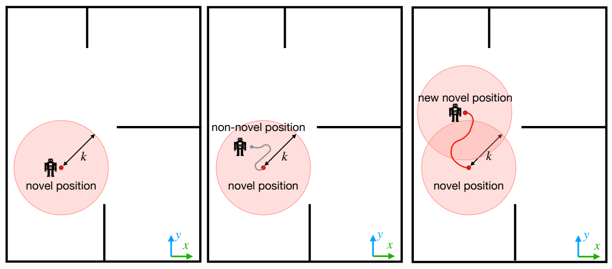


Figure 2. Intuitive representation of the novelty circle with radius k.

At each time step of the algorithm, the curiosity reward term is computed by averaging the sum of the Euclidean distance between the current robot position estimate $\chi_t^{(x,y)}$ and all the novel position $\chi_{1 \dots M}^{(x,y)} = \chi_1^{(x,y)}, \dots, \chi_M^{(x,y)}$ in the novelty buffer, as shown in Equation (10).

$$r_t^c = \frac{\alpha}{M} \sum_{i=1}^{M} d(\chi_t^{(x,y)}, \chi_i^{(x,y)}) \tag{10}$$

where α is a constant scaling the novelty reward term and controlling the urgency in reaching novel positions, M is the number of elements of the novelty buffer and d is the Euclidean distance operator. If r_t^c is higher than a certain threshold for a given position $\chi_t^{(x,y)}$, then this is added to the novelty buffer.

If the buffer is full an older novel position is randomly discarded. In this way, anything that is easily reachable by the agent becomes quickly uninteresting since no further reward can be obtained from it, whilst reaching positions further away from the known positions, i.e. at least distant k, is encouraged. This reward drives the robot to unexplored areas by leveraging on the pose estimate from the SLAM algorithm. Moreover, because only the pose estimate is used the proposed approach is independent of the kind of map the SLAM algorithm is constructing, by making the approach suitable to different scenarios.

Positions close to the walls are not considered as novel positions since this would encourage undesired behaviors. A benefit of this is that the agent gains a natural tendency to select actions keeping the robot in the middle of the rooms. The choice of k is intuitively crucial for the fast learning of the reinforcement learning algorithm. If k is chosen too small, any position can, in principle, become novel and appended to the novelty buffer. This would slow down the exploration and consequently reduce convergence speed. On the other side, if k is picked too large the curiosity reward term tends to disappear because novel positions may be too difficult to reach as too distant.

The complete algorithm used for computing the curiosity reward is presented in Algorithm 1 in Appendix A.

4.3 Neural networks architecture

In the DDPG algorithm, both the policy and value function are approximated by neural networks. In particular, the state vector is fed to the actor network which is composed of three fully connected hidden layers with ReLU activations and 512 neurons each. Eventually, the output layer has dimensionality 2 and outputs the linear and angular velocities for the robot. To constrain the velocities to feasible values for the robot's actuators, we use a sigmoid and a tanh activation for the linear and angular velocity respectively. In this way, we constrain the robot to move only forward, but we leave the possibility to turn left or right.

The critic network is also composed of the three fully connected hidden layers with ReLU activations and 512 neurons each. However, while the state vector is fed to the first hidden layer of the network, the action is only fed into the second layer. The output layer has linear activation and a single output, i.e. the estimated Q-value of the input state-action pair.

5. EXPERIMENTAL DESIGN

5.1 Setup

The experiments are performed using the Robot Operating System (ROS) and the simulation platform Gazebo. Gazebo allows simulating robots, their dynamics, realistic environments, and sensors, while ROS allows the communication among all the elements of the simulation platform, e.g. robot and sensors, and the learning algorithm. Additionally, ROS allows easy integration with different SLAM packages and the *gmapping* package that is used in this research.

The learning algorithm is written in Python using the Tensorflow library for the construction and training of the neural networks, and OpenAI-gym for the reinforcement learning environment.

5.2 Baselines

We compare the proposed reward function, in Equation (10), with three different reward functions used in literature and in Equation (11)-(13). In particular, we focus on the learning speed of the agents, trained with different reward functions, the length and quality of the trajectory after training, the percentage of map completed, and the generalization to untrained environments.

We first compare with a sparse reward function, named *Sparse*:

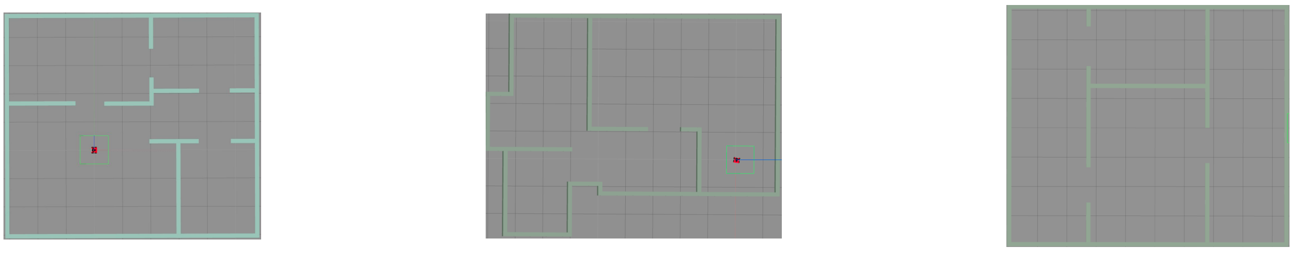
$$\mathrm{R}(s_t) = \begin{cases} r_{\text{map completed}}, & \text{if } \zeta_t \geq C \\ r_{\text{crashed}}, & \text{if } l_t \leq l_{\min} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Then, we compare with the reward function based on the map completeness gain, named *Oracle*, similar to the one adopted by (Chen et al., 2019):

$$\mathrm{R}(s_t) = \begin{cases} r_{\text{map completed}}, & \text{if } \zeta_t \geq C \\ r_{\text{crashed}}, & \text{if } l_t \leq l_{\min} \\ \zeta_t - \zeta_{t-1}, & \text{otherwise.} \end{cases} \tag{12}$$

Moreover, we compare with a reward function based on the entropy, named *Information-gain*, similar to the one used in (Botteghi et al., 2020):

$$\mathrm{R}(s_t) = \begin{cases} r_{\text{map completed}}, & \text{if } \zeta_t \geq C \\ r_{\text{crashed}}, & \text{if } l_t \leq l_{\min} \\ H_t - H_{t-1}, & \text{otherwise.} \end{cases} \tag{13}$$



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

Figure 3. The training environment, *Env-1*, and testing environments, *Env-2* and *Env-3* used for training and evaluating the proposed approach.

where H_t is the map's entropy at time t and H_{t-1} is the map's entropy at time $t-1$.

Eventually, all the learning-based approaches are compared with the frontier-based exploration (Yamauchi, 1997a).

5.3 Training and Testing

We are interested in studying the effect of the reward function on the performances and generalization skill of the agents. To do that, we train the agents is a single environment, *Env-1*, and evaluate their performances on two unknown ones. The training environments is the *Env-1*, shown in Figure 3a, while the testing environments, *Env-2* and *Env-3*, are shown in Figure 3b and 3c respectively.

The training environment *Env-1*, is based on a real $65\,\mathrm{m}^2$ apartment from the Dutch housing website *funda* (funda, 2020). Mapping the whole environment provides a significant challenge since a random policy is not enough for exploring the whole space and construct its map efficiently. The second and the third environments, *Env-2* and *Env-3*, have total areas of $68.5\,\mathrm{m}^2$ and $75\,\mathrm{m}^2$ and they are, again, based on real apartments from (funda, 2020). These two environments are only used for testing the generalization capabilities of the trained agents and no policy-learning is performed here.

We train each agent for the same amount of episodes on environment *Env-1*, by employing two different training strategies. Firstly, the robot position, at the beginning of each episode, is kept the same for the whole training process[4]. Secondly, at the beginning of each episode, the robot is randomly spawned in one of four possible starting positions.

After training, each agent is tested in the unseen a priori environment *Env-2* and *Env-3*, and the map completeness and the trajectory length and quality are recorded. For each testing environment, four different starting positions are selected and three experiments per position are performed.

6. RESULTS AND DISCUSSIONS

6.1 Training Results

The map-completeness[5] and the number of actions per episode of the different agents during training is presented in Figure 4a and 4b. When only a single fixed starting position for the robot, in each training episode, is chosen, the agent trained with the curiosity-driven reward function outperforms the other in terms of map-completeness rate, by quickly converging after ~ 150 episodes, and the number of actions compared to the other agents. The proposed curiosity-based reward function quickly motivates the agent to choose actions that can steer the robot to unseen locations of the environment, as far as possible from the known ones.

On the other side, when the robot is spawned at different starting locations, the agent trained with the information-gain reward function, in Equation (13) and the one trained with the map-completeness reward function, in Equation (12), achieve performance comparable to the curiosity, if not even slightly superior, to the proposed curiosity-based reward function in terms of map-completeness during training. These two methods benefit the most from the random initialization of the robot's position. However, when we analyze the number of actions taken in each training episode, the proposed reward function can reduce the actions taken at a higher rate than the other reward functions. In this context, the sparse reward function, in Equation (11), without a memory structure, e.g. a recurrent policy network, does not converge to a good solution in the time span of 400 episodes.

6.2 Generalization Results

We test how well the planners trained in environment *Env-1* perform in the unseen a priori environment *Env-2* and *Env-3*. Additionally, we compare the learning-based planners with frontier-based exploration. In Table 1, the generalization results in environment *Env-2* and the ones related to environment *Env-3* are shown, where we compare the planners by recording the average map-completeness and trajectory length in twelve different runs for each planner with four possible starting positions.

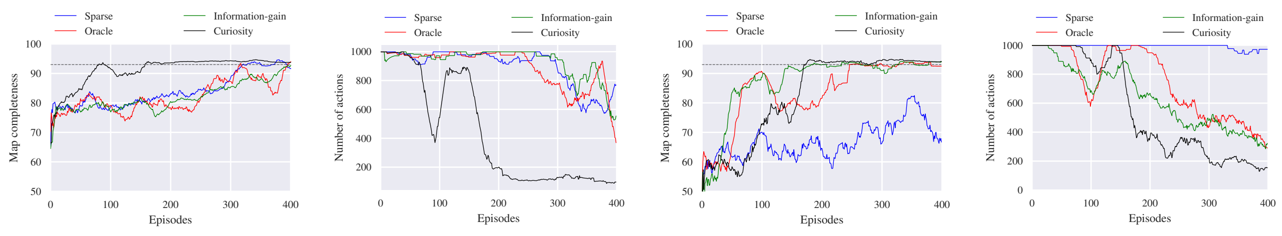
	approach	map-completeness % (mean ± std)	traj.length (m) (mean ± std)
Env-2	Sparse	55.86 ± 17.44	11.86 ± 3.35
	Oracle	65.28 ± 18.53	15.89 ± 4.4
	Information-gain	93.48 ± 8.94	24.31 ± 5.87
	Curiosity	99.09 ± 0.65	17.65 ± 5.14
	Frontier	98.45 ± 0.366	15.68 ± 3.6
Env-3	Sparse	37.08 ± 7.88	10.7 ± 2.25
	Oracle	66.83 ± 11.29	18.87 ± 10.33
	Information-gain	88.03 ± 6.54	18.78 ± 4.04
	Curiosity	92.58 ± 6.69	25.23 ± 7.93
	Frontier	99.15 ± 0.29	25.38 ± 4.08

Table 1. Generalization results in the untrained environment *Env-2* and *Env-3*.

When compared to the other agents, the one trained with the proposed reward function achieves the highest map-completeness in both environments and travels smoother and shorter paths

[4] The chosen starting position is a point in the middle of the bottom-left room of environment *Env-1*.

[5] Because we are employing different reward functions, the reward values obtained by the agents are different. Therefore, we compare the training performances based on how fast the map is completed.



(a) (b) (c) (d)

Figure 4. The training results in *Env-1* when the robot starts each training episode in the same position, Figure 4a and 4b, and when its pose is randomly selected, Figure 4c and 4d. The *Sparse* corresponds to the sparse reward function, the *Oracle* to the map-completeness reward function, the *Information-gain* to the entropy based reward function and *Curiosity* to the proposed reward function.

compared to the others. It is worth mentioning that when trained with the *Sparse* and *Oracle* reward functions, the agent never and rarely, respectively, completes the maps and they show a circling behavior on the spot with low-velocity. Because we fix the maximum number of actions in an episode to 500, the distance traveled by such planners is low.

The curiosity-driven agent achieves performances comparable to the frontier based explorations in terms of average map completeness and trajectory length. However, it is worth to mention that the proposed planner has higher computational efficiency compared to the frontier, where more operations have to be performed at each time step, such as detecting the frontiers, choosing one accordingly to a pre-determined criterium, and navigating to the frontier without collisions[6].

7. CONCLUSION

The key element of the success of the approach is to transform the problem of exploration of unknown environments into the problem of visiting novel locations of the world and the map. This is done by shaping the reward function, used to train the reinforcement learning agent, to encourage its curiosity into unseen locations and, consequently, features. The reinforcement learning agent, trained with the proposed curiosity-driven 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. The proposed approach achieves performance comparable to the frontier-based exploration method, but with lower computation cost. Moreover, the approach is not limited to occupancy-grid maps, this is the case of the frontier-based exploration, but can cope with any type of map representation and SLAM algorithm. This is due to the fact that the algorithm for training and testing only requires the robot's pose estimate and the completeness of the map.

REFERENCES

Botteghi, N., Sirmacek, B., Schulte, R., Poel, M., Brune, C., 2020. REINFORCEMENT LEARNING HELPS SLAM: LEARNING TO BUILD MAPS. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2020, 329–335. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLIII-B4-2020/329/2020/.

[6] The current implementation of the frontier-based exploration algorithm mostly uses C++ code, while our proposed approach used mostly Python code. A fair quantitative comparison is not possible due to the well-known, higher speed of C++ code.

Brunner, G., Richter, O., Wang, Y., Wattenhofer, R., 2017. Teaching a Machine to Read Maps with Deep Reinforcement Learning. *arXiv:171107479*.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A. A., 2018. Large-scale study of curiosity-driven learning.

Chen, T., Gupta, S., Gupta, A., 2019. Learning exploration policies for navigation.

Dooraki, R., Amir, Lee, Deok-Jin, 2018. An end-to-end deep reinforcement learning-based intelligent agent capable of autonomous exploration in unknown environments. *Sensors*, 18(10), 3575.

Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W., 2014. 3-D Mapping With an RGB-D Camera. *IEEE Transactions on Robotics*, 30(1), 177-187.

funda, 2020. `https://www.funda.nl/` (accessed: January 2021). House finding website in the Netherlands.

Grisetti, G., Stachniss, C., Burgard, W., 2005. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. *Proceedings of the 2005 IEEE international conference on robotics and automation.*

Kollar, T., Roy, N., 2008. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27(2), 175–196.

Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., Kleiner, A., 2009. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27, 387-407.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., 2003. Fast-SLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *Proc. IJCAI Int. Joint Conf. Artif. Intell.*

Mur-Artal, R., Montiel, J. M. M., Tardos, J. D., 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5), 1147–1163. http://dx.doi.org/10.1109/TRO.2015.2463671.

Murphy, K., 2000. Bayesian Map Learning in Dynamic Environments.

Mustafa, K. A. A., Botteghi, N., Sirmacek, B., Poel, M., Stramigioli, S., 2019. TOWARDS CONTINUOUS CONTROL FOR MOBILE ROBOT NAVIGATION: A REINFORCEMENT LEARNING AND SLAM BASED APPROACH. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13, 857–863.

Niroui, F., Zhang, K., Kashino, Z., Nejat, G., 2019. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2), 610–617.

Pardo, F., Tavakoli, A., Levdik, V., Kormushev, P., 2017. Time Limits in Reinforcement Learning. *CoRR*, abs/1712.00378. http://arxiv.org/abs/1712.00378.

Pathak, D., Agrawal, P., Efros, A. A., Darrell, T., 2017. Curiosity-driven exploration by self-supervised prediction.

Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., Cadena, C., 2017. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. http://dx.doi.org/10.1109/ICRA.2017.7989182.

Placed, J. A., Castellanos, J. A., 2020. A Deep Reinforcement Learning Approach for Active SLAM. *Applied Sciences*, 10(23), 8386.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T. P., Gelly, S., 2018. Episodic Curiosity through Reachability. *CoRR*, abs/1810.02274. http://arxiv.org/abs/1810.02274.

Sutton, R. S., Barto, A. G., 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA : The MIT Press.

Tai, L., Paolo, G., Liu, M., 2017. Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. *International Conference on Intelligent Robots and Systems*, 31–36.

Thrun, S., Burgard, W., Fox, D., 2005. *Probabilistic robotics*. MIT Press, Cambridge, Mass.

Thrun, S., Montemerlo, M., 2005. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. *International Journal on Robotics Research*, 25(5/6), 403–430.

Uhlenbeck, G. E., Ornstein, L. S., 1930. On the Theory of the Brownian Motion. *Phys. Rev.*, 36, 823–841. https://link.aps.org/doi/10.1103/PhysRev.36.823.

Wang, Y., Zhang, W., An, P., 2017. A survey of simultaneous localization and mapping on unstructured lunar complex environment. *AIP Conference Proceedings*, 1890(1), 030010. https://aip.scitation.org/doi/abs/10.1063/1.5005198.

Wu, Y., Wu, Y., Gkioxari, G., Tian, Y., 2018. Building Generalizable Agents with a Realistic and Rich 3D Environment. *arXiv:180102209*.

Yamauchi, B., 1997a. A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 146–151.

Yamauchi, B., 1997b. A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 146-151.

Zhang, J., Springenberg, J., Boedecker, J., Burgard, W., 2016. Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments. *arXiv:161205533*.

Zhang, Q., Zhu, M., Zou, L., Li, M., Zhang, Y., 2020. Learning Reward Function with Matching Network for Mapless Navigation. *Sensors*, 20(13), 3664.

Zhang, W., Zhang, Y., Liu, N., 2018. Danger-aware Adaptive Composition of DRL Agents for Self-navigation. *arXiv:180903847*.

Zhelo, O., Zhang, J., Tai, L., Liu, M., Burgard, W., 2018. Curiosity-driven Exploration for Mapless Navigation with Deep Reinforcement Learning. *arXiv:180400456*.

APPENDIX A

Curiosity Reward Computation

The outer loop iterates over the number of episodes, the middle loop iterates over the number steps allowed in an episode and the inner loop iterates over the novelty buffer $\mathcal{M}$.

Algorithm 1: Episodic Curiosity Reward

```
Initialize novelty buffer $\mathcal{M}$ with size $M$ ;
for episode in $N_{episode}$ do
    for t in $N_{step}$ do
        $r^{\text{temp}} = 0$ ;
        for i in $M$ do
            if $d(\chi_t^{(x,y)}, \chi_i) > k$ then
                $r^{\text{temp}} = r^{\text{temp}} + d(\chi_t^{(x,y)}, \chi_i)$;
            else
                $r^{\text{temp}} = 0$ ;
            end
        end
        $r_t^c = \frac{\alpha}{M} r^{\text{temp}}$
    end
end
```

APPENDIX B

Experiments Settings

The parameters used in our experiments are shown in Table 2.

RL and SLAM parameters	Value
optimizer	ADAM
actor learning rate	10^{-3}
critic learning rate	10^{-4}
discount factor γ	0.99
batch size	64
replay buffer size	10^6
novelty threshold k	$1.8\,\mathrm{m}$
particles	80
process scan threshold translation	0.05
process scan threshold rotation	0.05
grid cell size	$0.05\,\mathrm{m}\times0.05\,\mathrm{m}$
occupancy threshold	0.60
LiDAR max. range	$10\,\mathrm{m}$
LiDAR min. range	$0.2\,\mathrm{m}$
collision threshold	$0.2\,\mathrm{m}$
map completeness threshold	93 %

Table 2. Parameters of the experiments.