applied sciences

MDPI

Article

A Deep Reinforcement Learning Approach for Active SLAM

Julio A. Placed *, **José A. Castellanos**

Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, C/ María de Luna 1, 50018 Zaragoza, Spain; jacaste@unizar.es

* Correspondence: jplaced@unizar.es

Received: 26 October 2020; Accepted: 23 November 2020; Published: 25 November 2020

Abstract: In this paper, we formulate the active SLAM paradigm in terms of model-free Deep Reinforcement Learning, embedding the traditional utility functions based on the Theory of Optimal Experimental Design in rewards, and therefore relaxing the intensive computations of classical approaches. We validate such formulation in a complex simulation environment, using a state-of-the-art deep Q-learning architecture with laser measurements as network inputs. Trained agents become capable not only to learn a policy to navigate and explore in the absence of an environment model but also to transfer their knowledge to previously unseen maps, which is a key requirement in robotic exploration.

Keywords: active SLAM; robotic exploration; optimality criteria; deep reinforcement learning; deep Q-networks

1. Introduction

Simultaneous Localization and Mapping (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. Since its statement, more than three decades ago, it has significantly drawn the attention of the robotics community and numerous approaches have been developed, either based on novel filter techniques or optimization methods where inertial and visual measurements may be jointly optimized. See [1–4] and references there in.

Active localization was first introduced by Burgard et al. [5], where it was proven that picking actions to minimize the localization's uncertainty would result in a better localization than using a passive approach (e.g., Markov localization or adaptive Monte Carlo localization). Active SLAM augments this approach to the SLAM problem, and it can be defined as the paradigm of controlling a robot which is performing SLAM so as to reduce the uncertainty of its localization and the map's representation [6,7]. Typically, it consists of three stages [8]: (i) the identification of all possible locations to explore (ideally infinite), (ii) the computation of the utility or reward generated by the actions that would take the robot from its current position to each of those locations and (iii) the selection and execution of the optimal action. In the second step, the utility of each action is traditionally computed by quantifying the uncertainty in the estimation of the two target random variables: the robot's pose and the map's representation. This matrix quantification can be done on the basis of either Theory of Optimal Experimental Design (TOED) or Information Theory (IT). According to the first one, four optimality criteria can be inferred from the generic covariance matrix $\Sigma \in \mathbb{R}^{\ell \times \ell}$ of the state vector $x = (x_1 ... x_\ell)^T \in \mathbb{R}^\ell$, with eigenvalues $\lambda_1 ... \lambda_\ell$:

- T-optimality criterion: captures the average variance,

$$T\text{-}opt \triangleq \frac{1}{\ell}\sum_{k=1}^{\ell}\lambda_k \tag{1}$$

- A-optimality criterion: captures the harmonic mean variance, which is sensitive to a lower-than average value,

$$A\text{-}opt \triangleq \left(\frac{1}{\ell}\sum_{k=1}^{\ell}\lambda_k^{-1}\right)^{-1} \tag{2}$$

- D-optimality criterion: captures the whole covariance (hyper) ellipsoid,

$$D\text{-}opt \triangleq \exp\left(\frac{1}{\ell}\sum_{k=1}^{\ell}\log(\lambda_k)\right) \tag{3}$$

- E-optimality criterion: captures the minimum eigenvalue,

$$E\text{-}opt \triangleq \min(\lambda_k : k = 1...\ell) \tag{4}$$

Within IT, Shannon's entropy (or its variation) is the most commonly used metric, which can be expressed for an ℓ-dimensional multivariate Gaussian distribution, $X \sim \mathcal{N}^{\ell}(\mu, \Sigma)$, as follows:

$$\mathcal{H}(X) \triangleq \frac{\ell}{2}(1 + \log(2\pi)) + \frac{1}{2}\log(\det(\Sigma)) \tag{5}$$

The proper choice of the criterion is still an open research issue. Since each of them captures a different metric of Σ, they do not lead to the same behavior neither require the same computational resources. It is worth noticing that every criterion is right-unbounded and that the following relationship holds for every well-defined covariance matrix: $E\text{-}opt \leq A\text{-}opt \leq D\text{-}opt \leq T\text{-}opt$. Moreover, the robot's state space representation (using quaternions, rotation matrices or Lie algebras) and the uncertainty representation have recently been shown to be of utmost importance in the criteria properties and therefore in the algorithm performance [9]. While differential state representations allow the use of any criterion, absolute representations can only rely on D-opt (or Shannon's entropy, equivalently); being the other criteria unable to capture a true metric of the covariance matrix nor to ensure certain key properties such as their monotonicity during exploratory trajectories. Further discussion about the monotonocity of optimality criteria under Lie algebra representations can be found in Appendix A, where the proof of their monotonocity under differential uncertainty representations is presented.

Aside from traditional methods, such as particle filters [10] or model predictive control [11] where frontier-based or random tree strategies are generally used, the problem of selecting the most informative action in robotic exploration has been lately addressed with Deep Neural Networks (DNN), often equating to Convolutional Neural Networks (CNN). Notably, Deep Reinforcement Learning (DRL) is suitable for such problems in which the agent must learn by directly interacting with the environment under partial observability, and where its *a priori* knowledge is nonexistent. Thus, some DRL algorithms and strategies originally developed for other decision making problems (e.g., video-games) have already been successfully applied to robotic navigation and exploration, such as Deep Q-Networks (DQN) [12] and their double [13], dueling [14] and Rainbow [15] successors, the Asynchronous Advantage Actor-Critic (A3C) architecture [16], the Soft Actor-Critic (SAC) [17] or the Prioritized Experience Replay (PER) buffer [18].

A first example of those methods appears in [19], where Tai and Liu used a large CNN to extract the feature maps of a front-view camera that would be later fed into a 2-layer DQN capable of selecting the best next action to execute. In this case, experiments were carried out in a complex simulator,

but convergence of training was only achieved for topologically simple scenarios (e.g., a straight corridor). Moreover, the same simple scenarios were used in both training and testing stages, being the agent's generalization ability unclear. Work in [20] was among the firsts showing that navigation via DRL was also achievable using the A3C algorithm, notably improving speed convergence with respect to DQN. However, the acquired knowledge was never tested in environments different from the training one, hence the environment was again known in all cases. A critic generalization experiment, though, was carried out in [21], showing that trained agents do not perform as well in previously unseen maps, and could even demonstrate behaviors comparable to random agents in certain scenarios. [22] shows a similar approach in which obstacle avoidance problem is tackled using the D3QN architecture and FastSLAM backend to map the environment and locate the robot on it.

The aforementioned works used simple extrinsic rewards to encourage obstacle avoidance. However, once navigation was proved to be achievable, motivation and curiosity capabilities [23] were added to the agents in order to expand the problem to robotic exploration. While some approaches, frequently called *curiosity-driven*, encourage the agent to visit user-defined or novel states [24] or to maximize the coverage of a known map [25], others motivate those actions that minimize the environment's uncertainty (i.e., the agent's knowledge of the environment is maximized). The latter, that will be referred as *uncertainty-based*, have been commonly addressed by encouraging the visit of those states which are more difficult to predict [26,27], although they have been recently tackled on the basis of true uncertainty metrics. In [28] and [29] actions are selected so as to maximize the entropy reduction (information gain or Kullback-Leibler divergence) of a 2-dimensional occupancy grid map; either by learning in a supervised fashion with labelled cells, or by introducing a metric of it in the DRL's reward function, respectively. Another approach to solve the active localization problem that also acts directly on the reward function design was proposed in [30,31], where the entropy reduction was achieved by including on it the maximum likelihood (ML) of being in any state (belief accuracy). Recently, Chen et al. [32] trained both DQN and A2C agents using the underlying SLAM pose-graph through Graph Neural Networks (GNN). In this case, they combined DRL with exploration graphs in order to achieve true exploration policies based on the map's uncertainty (T-optimality), outperforming random and nearest frontier agents. They showed good generalization results with higher dimensional state spaces. However, the simulation environment used was a simple grid world in which obstacle avoidance was not needed and only landmark positions were changed.

In the literature, experiments are usually conducted in extremely-simplified simulation environments, such as grid-worlds, where the agent moves between cells in a discrete fashion, and sensors are never modelled. In these cases, the inputs to the DNN range from simple occupancy-grid maps [33] to combinations of frontier's and robot's groundtruth locations [32,34]. More complex and realistic scenarios, although seldom used, are required to feed images [19,22] or laser measurements [35] to the network. In such cases, it is worth noticing the complexity tackled; not only in the interaction between the simulator and the training algorithm but also in the physics of the robot movement, the environment itself and the sensors modelled (and thus their probabilistic behavior). Knowledge transfer to real environments is yet to be evaluated due to the stage these approaches are at, and only a few of them have effectively addressed it for navigation –and not exploration–purposes [19,35].

Over the past few years, the basics of DRL have proved to have a great potential in navigation, mapping and even exploration tasks. In this work, we aim to study that potential for the active SLAM decision making problem. In this way, one of the main drawbacks of traditional active SLAM methods is relaxed by transferring the intensive computations to the DNN training phase, requiring only feed-forward propagation during evaluation. The paradigm is formulated as a multi-reward DRL problem, on the basis of classical multi-objective optimization approaches and TOED. On top of Gazebo simulator, a state-of-the-art DQN is embedded within a Robot Operating System (ROS) framework. Thus, the use of both an *uncertainty-based* reward and a complex simulator are jointly taken into account. Trained agents become capable of making quasi-optimal decisions in order to navigate

and explore an environment, just from raw laser measurements. Acquired knowledge transfer is tested in previously unseen scenarios to clarify their generalization ability. Also, traditionally extrinsic and *uncertainty-based* rewards are compared to evidence the value of the proposed approach. Table 1 summarizes the differences between our approach and other mentioned literature. It can be noticed that true uncertainty metrics are needed in order to tackle Active SLAM.

Table 1. Comparison with prior methods.

	Task Accomplished	Architecture	Reward Design	Complex Environment	Generalization Unknown env.	Partial Observability
Tai and Liu [19]	Navigation	DQN	Extrinsic	✓		
Mirowski et al. [20]	Navigation	A3C	Extrinsic	✓		
Wen et al. [22]	Navigation	D3QN	Extrinsic	✓		
Zhelo et al. [26]	Goal navigation	A3C	Intrinsic		✓	
Oh and Cavallaro [27]	Goal navigation	A3C	Intrinsic	✓		
Tai et al. [35]	Goal navigation	ADDPG	Intrinsic	✓	✓	
Zhu et al. [33]	Frontier selection	A3C	Intrinsic		✓	
Niroui et al. [34]	Frontier selection	A3C	Intrinsic	✓	✓	✓
Gottipati et al. [31]	Active localization	A2C	Posterior belief	✓	✓	
Chaplot et al. [30]	Active localization	A3C	Posterior belief	✓	✓	
Chen et al. [29]	Exploration	DQN	Entropy		✓	✓
Chen et al. [32]	Active SLAM	DQN+GNN	T-opt		✓	✓
Ours	Active SLAM	D3QN	D-opt	✓	✓	✓

This section provided a brief but comprehensive introduction to the problem tackled. The rest of the paper is organized as follows. In Section 2, active SLAM problem is formulated and the proposed DRL approach is presented. The experimental setup (consisting of the simulation, SLAM and decision making modules) and he results obtained and their discussion are shown in Sections 3 and 4. Finally, the paper conclusions are in Section 5.

2. Problem Formulation

2.1. Traditional Approach

The active SLAM problem is included within a wider mathematical framework, Partially Observable Markov Decision Processes (POMDP), that generalize such decision making processes under both action and observation uncertainties. A POMDP is formally defined as a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, \mathcal{R}, \gamma)$ consisting of the finite agent's state space $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a transition function between states $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ where $\Pi(\mathcal{S})$ is the probability distribution over $\mathcal{S}$, a finite observation space Ω and its conditional probabilities $\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and the discount factor γ which allows to work with finite rewards even when planning has an infinite time horizon. Every time step, the agent selects an action to execute $a_t \in \mathcal{A}$ based on the current policy π, generating a transition from s_t to $s_{t+1} \equiv s'$, both contained in $\mathcal{S}$, where a new observation $o_t \in \Omega$ is made. The goal of the agent is to find the optimal policy π^* that maximizes a metric of the reward function, usually defined as the future expected discounted reward $R_t \triangleq \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$.

Resolution of POMDP can be based on several methods, such as Monte Carlo, dynamic programming or temporal differences. Reinforcement Learning (RL) algorithms, based on the latter and Bellman equations, solve the problem by iterating either directly over the policy or over a value function. This function is just a mapping to real numbers from the value encoded in being at a certain state, or in a more complex fashion, from the value of being at a certain state and also executing a certain action according to π; and are denoted as $V(s)$ and $Q(s, a)$, respectively. Both value- and policy-based trends allow a POMDP resolution when the process itself (i.e., $\mathcal{T}$ and $\mathcal{O}$) is not known nor modelled (model-free).

In any case, RL algorithms are just recurrences over the expected rewards, as shown in Algorithm 1 of Q-learning, or more specifically, in Equation (6)–the Q-function's Bellman update. The design of

the reward function is then extremely important in order to achieve the desired goal, as it occurs in optimization problems with cost functions.

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s') \max_a Q^*(s',a) \tag{6}$$

Algorithm 1 Q-learning (for deterministic environment).

Parameters: $\alpha \in (0,1]$, $\gamma \in (0,1]$
Initialize Q-table with arbitrary Q-values
for episode $\leftarrow$ 1 to max episodes **do**
  Perceive s_t
  while s_t not terminal **do**
    Select $a_t \leftarrow \pi(s_t)$
    Take a_t, get r_t and perceive s_{t+1}
    if s_{t+1} is terminal **then** $Q_t \leftarrow r_t$
    else $Q_t \leftarrow r_t + \gamma \max_a Q(s_{t+1}, a)$
    end if
    $Q(s_t, a_t) \leftarrow (1-\alpha) Q(s_t, a_t) + \alpha Q_t$
    $s_t \leftarrow s_{t+1}$
  end while
end for

Furthermore, active SLAM can be formulated as a multi-objective optimization problem [7] where the cost function $\mathcal{J}$ must contain the cost of performing a free-collision trajectory ($\mathcal{F}$) and a metric of the covariance matrix ($\mathcal{U}$), which represents the uncertainties in the estimates of the robot's pose and the map. As shown hereafter, both terms should be scaled by task-dependent coefficients, and can be considered in an n-step horizon,

$$\mathcal{J} = \sum_n \beta_n \mathcal{F}_n + \sum_n \alpha_n \mathcal{U}_n \tag{7}$$

Such formulation can be straightforward generalized to RL problem, just by designing the reward function as follows,

$$\mathcal{R}_{aug} = \mathcal{R}_{\mathcal{F}} + \mathcal{R}_{\mathcal{U}} \tag{8}$$

where the first term refers to a fully extrinsic reward that accounts for the collision-free trajectories (i.e., navigation), and the second one is an intrinsic reward, that can be defined inversely proportional to a metric of the covariance matrix $f(\Sigma)$, e.g., the aforementioned utility functions. This definition would motivate the robot not only to navigate the environment but also to explore it, in terms of improving the estimate of the map's representation and the robot's localization on it.

2.2. Neural Network Approach

In Deep Q-learning formulation, the analytical computation of the action-value function (Q-function) is approximated by a DNN which coefficients θ are iteratively updated by using the well-known back-propagation algorithm [36]. Let the loss function of this new optimization problem be the quadratic loss,

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \tag{9}$$

where n is the batch size, $\hat{y} = Q(s,a|\theta)$ the estimated Q-value by the DNN, and y the target Q-value, that can be computed using an auxiliary identical neural network, following [13], with parameters θ^*, as:

$$y = \mathcal{R}(s,a) + \gamma Q(s', \arg\max_a Q(s',a|\theta)|\theta^*) \tag{10}$$

Therefore, just by feeding reward function from Equation (8) in the previous equation, the neural network is recursively taught to visit those states that contributes to an uncertainty reduction. For the sake of understanding of the approach, Algorithm 2 shows the proposed learning process, using a dueling double DQN with PER embedded in Gazebo simulation environment.

Algorithm 2 Dueling Double Deep Q-learning with PER in simulation environment.

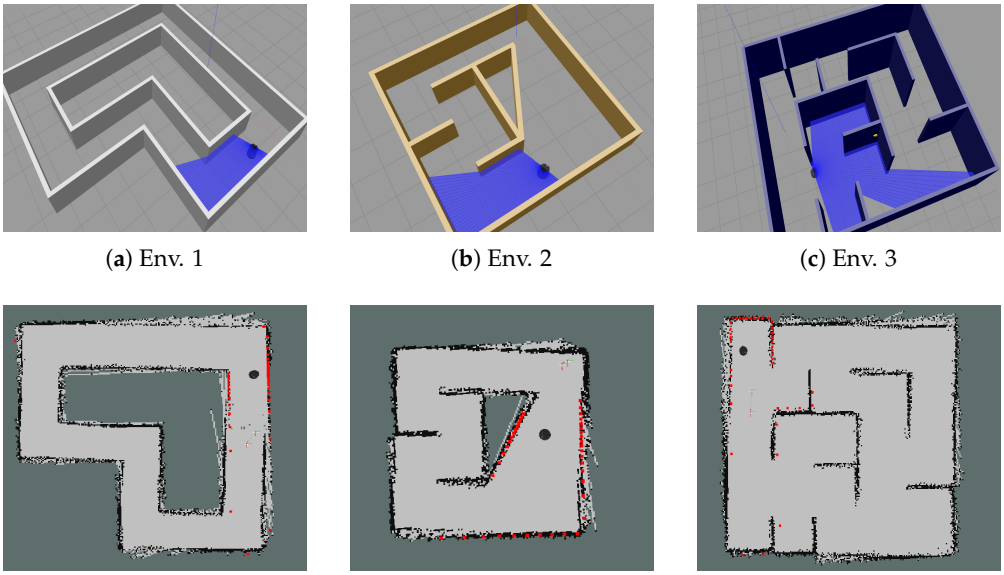
Parameters: $\alpha \in (0,1]$, $\gamma \in (0,1]$, U, U_e, ε_0, ε_f, $\varepsilon \leftarrow \varepsilon_0$
Initialize memory $\mathcal{M}$, $\delta \approx 0$ and $\alpha' \in (0,1]$
Initialize DNN with parameters θ and $\theta^* \leftarrow \theta$
Initialize ROS master, Gazebo and SLAM algorithm
for episode $\leftarrow$ 1 to max episodes **do**
  ▶ Unpause simulation
  Reset robot pose, map and SLAM algorithm
  Perceive s_t
  ▶ Pause simulation
  while s_t not terminal **do**
    Select $a_t \leftarrow \pi(s_t|\varepsilon)$
    ▶ Unpause simulation
    Execute a_t during t_a seconds
    Recover Σ, compute desired $f(\Sigma)$ and r_t
    Perceive s_{t+1}
    ▶ Pause simulation
    if s_{t+1} is terminal **then** $e_t = 1$ **else** $e_t = 0$
    Store tuple $(s_t, a_t, r_t, s_{t+1}, e_t)$ in $\mathcal{M}$ with $\max(\mathfrak{p})$
    Sample minibatch from $\mathcal{M}$
    for each tuple in minibatch **do**
      if s_{i+1} is terminal **then** $Q_{target} \leftarrow r_i$
      else $Q_{target} \leftarrow$
        $r_i + \gamma Q(s_{i+1}, \arg\max_a Q(s_{i+1}, a|\theta)|\theta^*)$
      end if
      $\Omega_i \leftarrow Q_{target} - Q(s_i, a_i|\theta)$
      $\mathfrak{p}_i \leftarrow (\Omega_i + \delta)^{\alpha'}$
    end for
    Perform optimization on Ω^2 w.r.t. θ
    Every U steps set $\theta^* \leftarrow \theta$
    $s_t \leftarrow s_{t+1}$
    $\varepsilon \leftarrow \min\left(\varepsilon_0, \varepsilon - (\varepsilon_0 - \varepsilon_f)/U_e\right)$
  end while
end for

3. Experimental Setup

3.1. Environment Setup

In order to evaluate the proposed approach, several experiments have been conducted in Gazebo robotics simulator (http://gazebosim.org/). A Turtlebot robot has been used, consisting of with two differential wheels and a modelled laser sensor that generates n rays equally distributed in the 180° front field of view of the robot, each of them with a minimum range of 0.1 m, a maximum range of 10 m, two-digit decimal precision and a Gaussian noise model with zero mean and small variance. Three different environments (Figure 1), adapted from [37], have been used during training and testing stages. The first one, where the agent is trained, consists of a simple maze with 90° turnarounds. Second and third environments have a more complex topology, including consecutive turnarounds and dead-ends, and have been used only during testing (i.e., agents do not improve their policy anymore, see Figure 2). A training/testing stage consists of a certain number of episodes, in which the robot moves until either a collision occurs (i.e., gets closer to any obstacle than a defined

threshold of 0.2 m) or 500 steps (i.e., decisions) are consumed. A ROS framework has been used to connect the simulation environments with the SLAM algorithm and the decision making module, by using GymGazebo (https://github.com/erlerobot/gym-gazebo) library, based on OpenAI Gym (https://gym.openai.com/).



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

(**d**) Map 1 (**e**) Map 2 (**f**) Map 3

Figure 1. Gazebo simulation environments (top) and their corresponding maps generated by the SLAM algorithm (bottom).

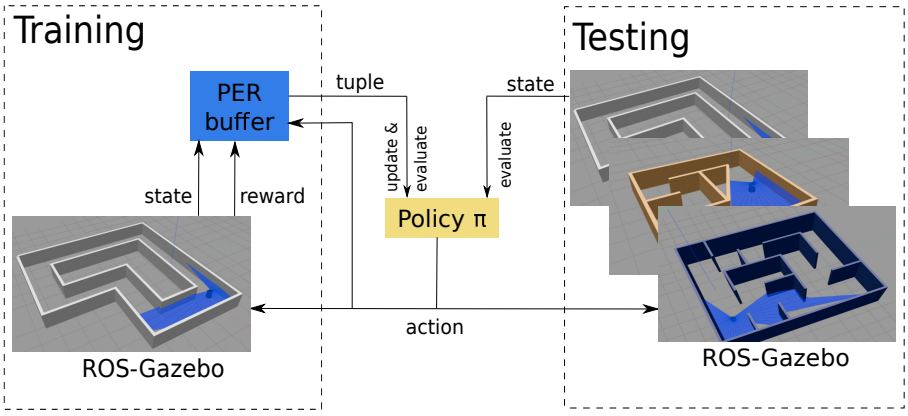


Figure 2. Training (**left**) and testing (**right**) processes.

Every step, the DNN receives the n-dimensional sensed vector and estimates the Q-values for each possible action, namely going forward, turning right and turning left:

$$a_1 = \{v = 0.3, \ \ \omega = 0\}$$
$$a_2 = \{v = 0.05, \omega = 0.3\}$$
$$a_3 = \{v = 0.05, \omega = -0.3\}$$

where v is the linear velocity in m/s and ω is the angular velocity in rad/s; and $v \neq 0$ in a_2 and a_3 to avoid continuous turnings on the same spot. By using a fully extrinsic approach, after each decision

making, the environment would reward the agent with a large negative value if a collision occurred, or with a small positive value otherwise:

$$\mathcal{R}_{\mathcal{F}} = \begin{cases} -\ 100 & \text{if collision} \\ \quad 1 & \text{if } \omega = 0 \\ -\ 0.05 & \text{if } \omega \neq 0 \end{cases} \tag{11}$$

Note that the reward values have been designed empirically, and the penalty to nonzero angular velocities prevents continuous spins and uneven trajectories.

The proposed *uncertainty-based* approach requires, following Equation (8), its augmentation as follows,

$$\mathcal{R}_{aug} = \begin{cases} -\ 100 & & \text{if collision} \\ \quad 1 & +\tanh\left(\dfrac{\eta}{f(\Sigma)}\right) & \text{if } \omega = 0 \\ -\ 0.05 & +\tanh\left(\dfrac{\eta}{f(\Sigma)}\right) & \text{if } \omega \neq 0 \end{cases} \tag{12}$$

where η is a task-dependent scale factor and $f(\Sigma)$ is the D-optimality criterion. Since this criterion –likewise any other of the presented criteria– is right-unbounded, the $\tanh(\cdot)$ function has been used to bound it in the interval $[0, 1]$ and therefore to maintain a relationship between the free-collision trajectory and uncertainty terms.

3.2. SLAM

From previous formulation it is clear that the covariance matrix recovery is a key requirement. In this sense, a slightly modified *gmapping* [2] algorithm runs on background of the training, re-initializing its map every episode and communicating the covariance estimate to the learning module via the ROS architecture (see Algorithm 2). Despite other SLAM algorithms (e.g., [4,38]) would able to compute more accurate state and covariance estimates, *gmapping* has been chosen because of its simplicity and low computational load, which were requirements that have been considered essential for the stage the study is at.

Every time the algorithm processes a laser scan, it recovers the distribution of the particles' state as a Gaussian with mean $\mu \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times3}$:

$$\mu = \sum_{i=1}^{n_p} \bar{w}_i x_i \tag{13}$$

$$\Sigma = \sum_{i=1}^{n_p} \bar{w}_i (x_i - \mu)(x_i - \mu)^T \tag{14}$$

where n_p is the total number of particles, index i represents a single particle, $\bar{w}_i = w_i / \sum_{j=1}^{n_p} w_j$ its normalized weight and $x_i = (x_i, y_i, \theta_i)^T$ its state vector. After its retrieval, the covariance matrix is referenced to the previous pose following the absolute formulation available in [9], and published in ROS.

The main parameters of the algorithm have been tuned to achieve a balance between performance and low computational load. Since each particle has to carry its own map, only five particles have been used. The minimum effective number of particles, a key parameter to detect loop closures, has been set to one-fourth of the total number of particles.

3.3. Decision Making Module

Hitherto, the simulation environment and the SLAM algorithm have been described. Decision making is the third and last module of the approach, and it has been done via deep Q-learning due to the high computational load that multi-agent training (e.g., A3C) would require in Gazebo.

First of all, a vanilla DQN and a double DQN (DDQN henceforth) have been implemented in TensorFlow. The networks contain two hidden layers, each of them with 24 hidden units and LeakyReLU activation with negative slope $f(x) = -0.01x$, and a linear output layer with a number of hidden units equal to the dimension of the action set. Secondly, a dueling double DQN architecture (D3QN henceforth) with PER has been implemented, akin to the state-of-the-art network described by Hessel et al. [15]. It contains two parallel separate flows that encode the value function $V(s)$ and the advantage function $A(s, a)$. Each flow contains only one hidden layer, with the same structure as the previous networks, and an intermediate output linear layer (1-dimensional for V and 3-dimensional for A). Both flows are then non-trivially aggregated to generate the estimate $Q(s, a)$ in the output layer, again of dimension equal to the size of the action set. In addition, the weights of this network have been non-randomly initialized to known values that encode a reasonable policy, so as to speed convergence. In all cases, the target network is hard-updated every 10000 steps, the training algorithm uses RMSProp optimizer ($\epsilon = 1 \times 10^{-6}$, $\rho = 0.9$), and the agents follow an ε-greedy policy with decreasing ε in the interval $(1, 0.02]$ during the first 50 training episodes. This policy is nevertheless changed to a greedy one during testing. The main parameters of the learning algorithm and the simulator are listed in Table 2.

Table 2. Training and simulation main (hyper)parameters.

Parameter	Description	Value
α	Learning rate	0.00025
γ	Discount factor	0.99
M	Memory buffer size	20000
b	Minibatch size	64
n_{in}	Number of DNN inputs	100
n_{out}	Number of DNN outputs	3
U_e	Exploration episodes	50
U	Steps between θ^* updates	10000
η	Scaling factor of $f(\Sigma)$	0.01
t_{sim}	Gazebo real-time factor	10
t_a	Seconds to apply the action	0.1
s	Max. no. of steps per episode	500

4. Discussion of Results

4.1. Reinforcement Learning

To conceptually prove the validity of the approach, a preliminary simple experiment has been performed where decision making is done via Q-learning. In this case, only five laser measurements have been used to reduce the computational complexity of this tabular approach, and the reward function of Equation (12) has been modified so $f(\Sigma)$ equals to Shannon's entropy and $\eta = 1$. The agent has been pre-trained in the first scenario to learn some general skills and then trained again in the second environment, which is more complex. Figure 3 shows the entropy evolution along episodes during the training phase for the traditional (left) and *uncertainty-based* (right) approaches. Color maps indicate the logarithmic occurrence of entropy (darker means lower). In the left figure, entropy trend is not ordered and any improvements are achieved accidentally, since the algorithm does not bother on its minimization. The rightmost figure, however, shows a decreasing entropy trend as training progresses, which matches with the proposed formulation.

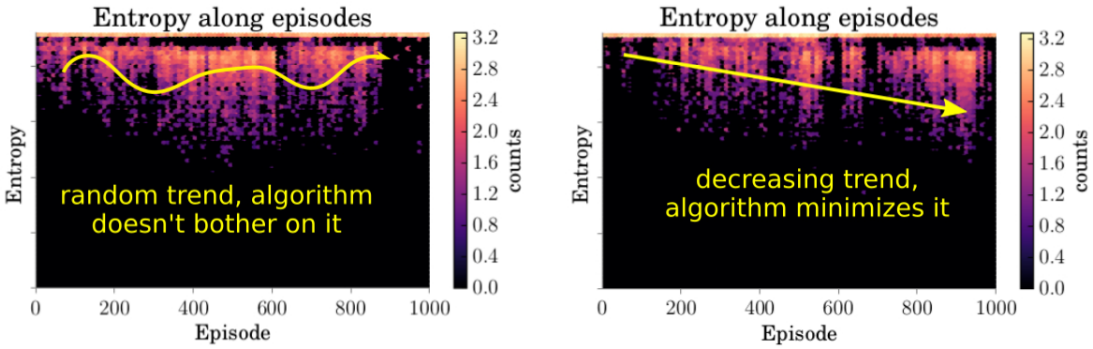


Figure 3. Logarithmic occurrence of entropy along episodes in the second scenario. Results are shown for traditional (**left**) and *uncertainty-based* (**right**) RL's reward functions. Darker colors denote lower occurrence.

4.2. Deep Reinforcement Learning

Once the suitability of the approach has been conceptually proved, it has been extended to include the benefits of DL. First of all, the three agents (DQN, DDQN and D3QN) have been trained in the simplest environment with an extrinsic reward, in order to compare their performance, study their generalization ability and prove that exploration is not achievable with such formulation. Figure 4 contains their training curves (i.e., the cumulative reward after each episode of the training) after roughly 30 h and 500 episodes (all experiments have been performed on a laptop with Intel Core i7-7500U CPU, Nvidia Quadro M520 (2GB) GPU and a 32GB RAM). Light-colored curves correspond to raw data while bold ones do to outlier-filtered moving averages. DQN agent, depicted in blue, demonstrates permanent instabilities due to the memory saturation with useless experiences, resulting in continuous cycles of forgetting and relearning similar policies. DDQN agent (red) mitigates them, although converged maximum values are slightly lower (i.e., policies are less optimal and thus trajectories become rougher). Finally, D3QN curve (green) has a significantly more stable behavior attributable to the PER, and also higher maximum converged values. Note that, in this case, convergence is much faster because of the non-random initialization of the network weights.

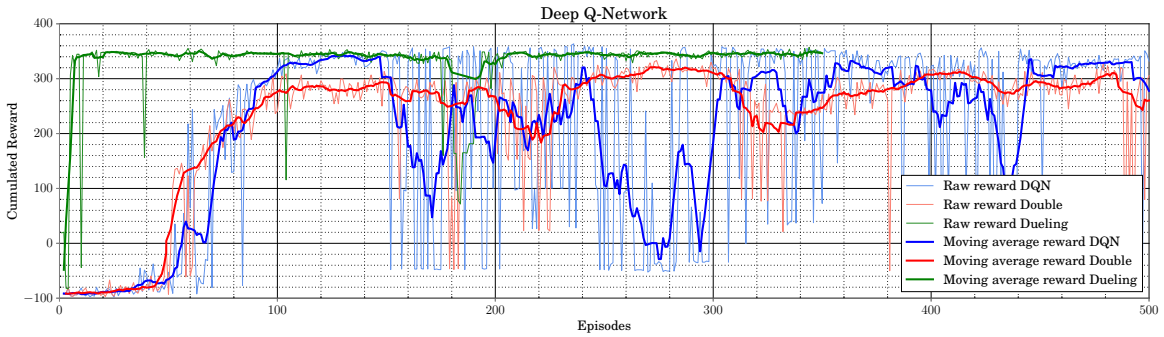


Figure 4. Cumulative reward during training in environment 1 for DQN (blue), DDQN (red) and D3QN (green) agents. Light-colored curves correspond to raw data while bold ones do to outlier-filtered moving averages.

Table 3 shows the performance of each agent in the three scenarios during testing (generalization), in terms of success ratio (SR, i.e., number of times over all trials that they sense more than 95% of the map at least once, where 95% have been used instead of 100% due constraints of the laser sensor), mean number of steps per episode and mean reward per episode. In addition, results have been averaged over five experiments with different simulation random seeds, and their standard deviation is presented in parenthesis. D3QN results are the most remarkable in the first environment, as it

outperforms the reward that a human would obtain by manually controlling the robot (≈350). In the second environment both DDQN and D3QN show a good behavior. Despite DDQN have higher SR (and mean steps, thus), the higher mean reward obtained by D3QN proves the generation of more optimal trajectories: smoother movements and less spins. The third environment presents quite a challenge not only for being unknown but also because of its complex topology. None of the agents trained with $\mathcal{R}_{\mathcal{F}}$ is capable of going across the whole map, leaving some areas such as the top right corner and the middle dead-end unvisited (see Figure 1f). Besides that, D3QN navigation performance is exceedingly superior, with mean rewards several times higher than the ones obtained by DQN and DDQN.

The three trained agents performed well during generalization in the second environment, due to its similarity to the training one. After the training, the agents have learned to decide when to go forward and when to steer when an obstacle is close enough. As the complexity of the agent's architecture increases, the learned skills complexity grows too. For instance, it can be noticed how the D3QN agent learned the exact moment to steer so as to maximize rewards (i.e., minimize the traveled distance). In this sense, the second scenario keep this topology and thus the agent is capable of transferring the learned skills. The third scenario, however, is more challenging. Several dead-end corridors appear and also a corridor with two ways. Therefore, the agent must be capable of choosing the right path or, at least, being able to return when a wrong action is selected. Note that these are skills that were no taught in the first scenario. In the third environment, only the D3QN agent is capable of generalizing the learned skills to those needed in order to complete it.

For the purpose of showing the strong effect of not having *a priori* knowledge of the environment, a second experiment has been conducted where the simplest trained agent was allowed to store information of the second environment during a few episodes before being tested on it (denoted as DQN† in Table 3). Results of the retraining stage are shown in Figure 5. In the beginning, the agent worsens its performance while the network weights are adjusted, but after about 65 episodes, it is capable to outperform agents with a more complex architecture, as it is also shown in Table 3. It is worth noticing too how the standard deviation of the reward significantly increases when the environment is unknown for the agents, since the input states are unfamiliar. This experiment has not been performed in the first scenario since DQN agent has been already trained on it. In the third environment, the required amount of warm-up steps needed for the neural network to adapt was too high, so it is not presented either. Despite demonstrating a superior behavior, the agent would be learning a whole new policy based on the information gathered in the new scenario and forgetting the old one.

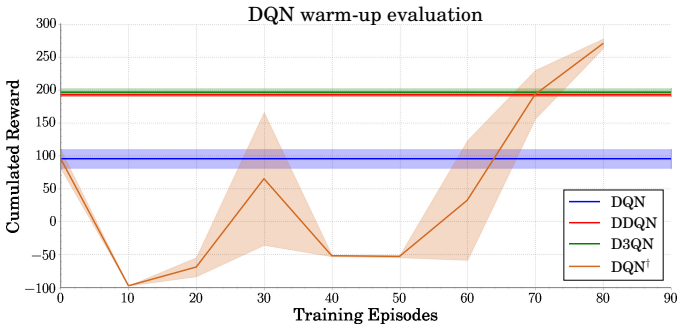
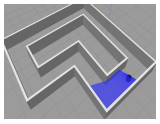
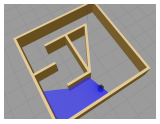
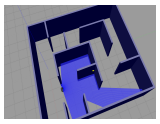


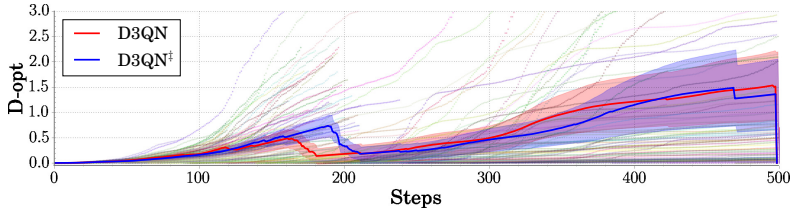
Figure 5. Mean cumulative reward and its deviation for DQN (blue), DDQN (red) and D3QN (green) agents in the second environment. Also, evolution of the cumulative reward for DQN† (orange) over the number of retraining episodes.

Table 3. Mean evaluation results in all environments for DQN, DDQN and D3QN agents trained with $\mathcal{R}_{\mathcal{F}}$. Superscript † refers to an agent that was allowed to train briefly before its evaluation on that scenario, while ‡ does to an agent retrained with $\mathcal{R}_{aug}$.

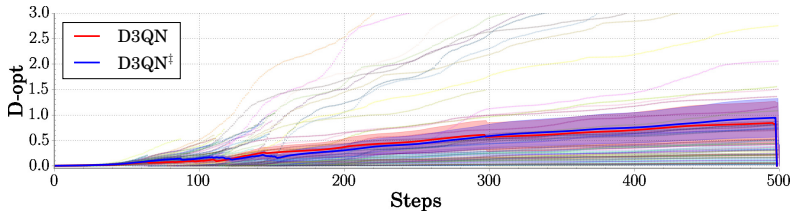
		Agent	SR(%)	Steps	R
	Env. 1	DQN	100	500 (0)	352.14 (0.1)
		DDQN	100	500 (0)	337.56 (2.2)
		D3QN	100	500 (0)	**355.31** (0.3)
		D3QN ‡	100	500 (0)	300.1 (9.3)
	Env. 2	DQN	72.8	312 (13)	95.55 (14.4)
		DDQN	100	500 (0)	192.52 (3.0)
		D3QN	89.3	419 (9.3)	196.5 (5.7)
		DQN †	100	500 (0)	**269.98** (4.4)
		D3QN ‡	100	500 (0)	**241.1** (29.2)
	Env. 3	DQN	0	170 (15)	−24.89 (8.3)
		DDQN	0	253 (16)	−42.15 (4.8)
		D3QN	0	432 (18)	241.56 (16.5)
		D3QN ‡	26	459 (97)	**261.62** (57.5)

Lastly, an experiment has been carried out using the reward proposed in Equation (12). The previous D3QN agent has been trained again in the first environment during 250 episodes (reaching a tradeoff between performance and low training time). Evaluation results are shown in Table 3 under D3QN‡ notation. Note that, for comparison purposes, the mean rewards shown correspond only to the extrinsic term of $\mathcal{R}_{aug}$, albeit the complete function was used during training. Although in the first environment the mean extrinsic reward is significantly lower; in the second one the agent improves the performance of its predecessors, achieving a 100% SR and obtaining mean rewards close to the ones that an agent with previous knowledge of the environment would obtain. In the third environment, the increase in mean steps and reward is also remarkable. However, the most notable fact is the nonzero SR, meaning that the agent is capable of visiting the whole scenario in some occasions: a fact that any agent had achieved due the topological complexity of the environment and since exploration was not encouraged until now. In all cases, the inclusion of $\mathcal{R}_{\mathcal{U}}$ has led to the selection of more optimal actions in terms navigation and robotic exploration; but also to a significant increase of the standard deviation. This could be mitigated with longer training stages, since the optimization problem becomes more complex with multi-reward functions.

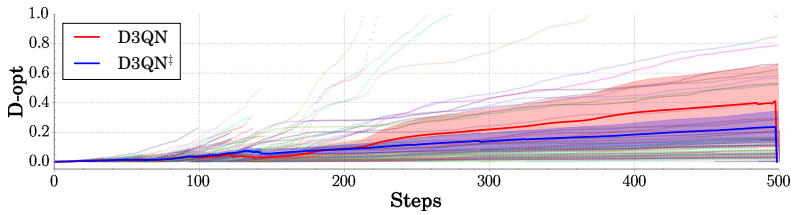
Previous data has proved the effectiveness of the proposed approach only in terms of navigation, though. Next, it is shown how uncertainty evolves in both the traditional and the proposed methods. Figure 6 shows the evolution of the D-optimality criterion during evaluation of D3QN (red) and D3QN‡ (blue) agents in the three environments. This figure contains the mean behavior (bold curves) of 50 different episodes (shown in the background), and their confidence interval at 95% (light-colored areas). In the first two environments the evolution is similar with both agents, however, there is a big difference on how they achieve such behavior: while D3QN‡ agent looks for the uncertainty decrease, D3QN achieves it accidentally due to the reduced topological complexity of the environments. In the third scenario, which is topologically more complex and where D3QN left several areas unvisited (and therefore loop closures), the use of D3QN‡ results on a significant reduction of *D-opt* ($\approx$ 34%). Also, note how the increasing monotonicity property is hold and how discontinuities occur due to loop closures.



(**a**) Environment 1



(**b**) Environment 2

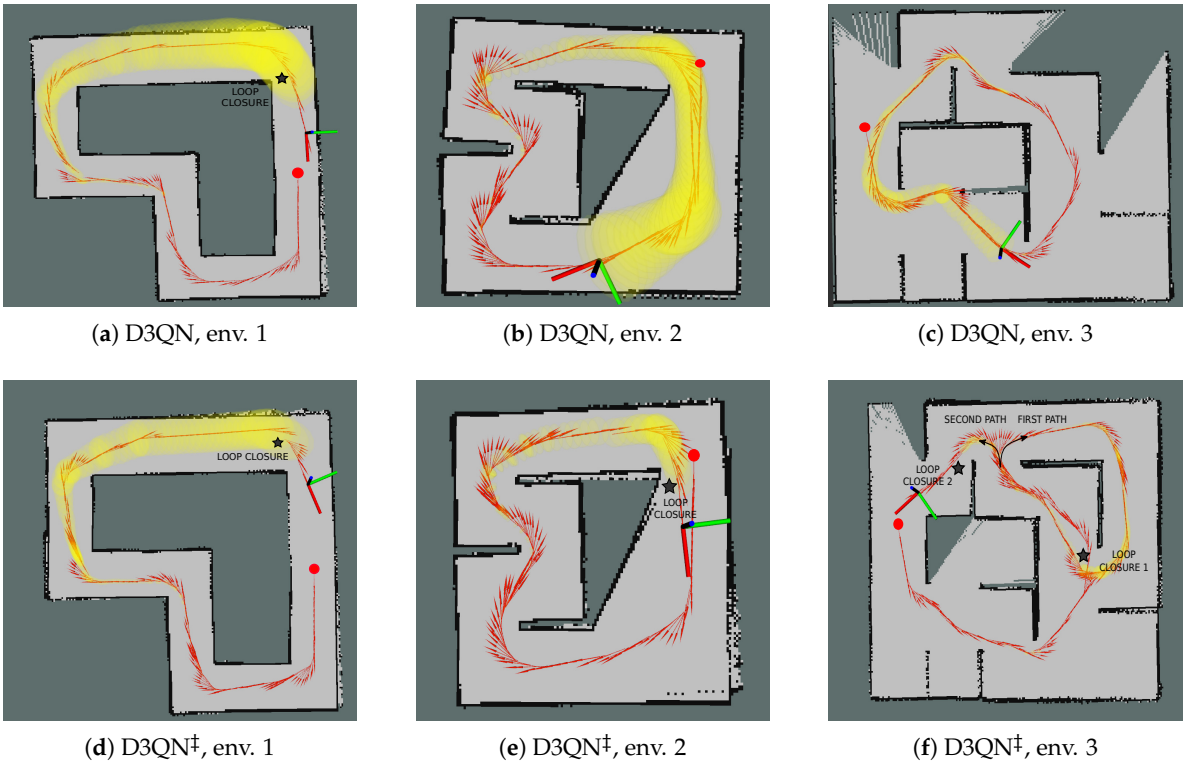


(**c**) Environment 3

Figure 6. Evolution of the D-optimality criterion during evaluation for the agents trained with $\mathcal{R}_{\mathcal{F}}$ (D3QN, red) and $\mathcal{R}_{aug}$ (D3QN‡, blue).

Figure 7 contains the maps generated online by the SLAM algorithm in the three environments after about 150 testing episodes. Images also contain starting positions (red circles), the detection of loop closures (black stars), the robot's trajectory (red arrows) and its uncertainty in X and Y directions (yellow ellipses). Maps are depicted for D3QN (top) and D3QN‡ (bottom) agents. Firstly, it is notable the lower uncertainty achieved by the proposed agent in all cases, and the effectiveness of resampling in comparison to the traditional agent. Moreover, trajectories followed are more optimal and angular velocity is applied more sparsely (source of uncertainty increase). In the third environment, D3QN‡ agent is strikingly capable of selecting different paths to explore a greater area and resample the particles (i.e., close the loop) twice. Note that although these maps have been hand-selected to represent a common behavior of the agent's policy, do not always reproduce equally due to the environment and simulation's randomness.

Finally, a discussion on the complexity of this approach with respect to the traditional active SLAM algorithm is presented in Appendix B.



(**a**) D3QN, env. 1 (**b**) D3QN, env. 2 (**c**) D3QN, env. 3

(**d**) D3QN‡, env. 1 (**e**) D3QN‡, env. 2 (**f**) D3QN‡, env. 3

Figure 7. Maps built by D3QN (top) and D3QN‡ (bottom) agents in the three environments during testing. Red circles indicate the starting position, red arrows the trajectory followed, black stars point out a resample of algorithm particles (loop closures) and yellow ellipses illustrate a measurement of the uncertainty of robot's 2D position.

5. Conclusions

In this paper, we have presented a novel approach to solve the Active SLAM paradigm by using model-free DRL. In contrast to prior approaches that try to effectively address autonomous exploration with intrinsic rewards, we embedded true uncertainty metrics in the reward function, that stem from traditional TOED. Despite many works claim to succeed in exploration with extrinsic rewards, we have proved that performing Active SLAM with such rewards is not feasible. We embedded a state-of-the-art deep Q-learning architecture on top of Gazebo simulator; and executed a lightweight 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 a complex environment with realistic models for the robot, the sensors and the environment physics, unlike most related works. Trained agents are able to select actions so as to reduce the uncertainty via quasi-optimal trajectories and place recognition. More interestingly, they are also able to transfer the acquired knowledge to previously unseen maps, a key requirement to solve Active SLAM.

As future work, we aim to embed a more complex state-of-the-art SLAM system to obtain a more precise uncertainty estimate which also includes a state representation based on Lie algebras, e.g., ORB-SLAM2 [39], in order to accurately compute optimality criteria, see [9]. To our understanding, the main limitations of our approach are the DRL architecture and the environments used. Thus, we plan to test actor-critic algorithms and explore new inputs to feed the networks, e.g., images or the underlying pose-graphs. Finally, experiments in real world will be conducted where our approach will be compared to classical active SLAM algorithms, although for now, further theoretical exploration with simulator validation seems to be more valuable.

Author Contributions: Conceptualization, J.A.P. and J.A.C.; methodology, J.A.P. and J.A.C.; software, J.A.P.; validation, J.A.P.; formal analysis, J.A.P. and J.A.C.; investigation, J.A.P.; resources, J.A.P.; data curation, J.A.P.;

writing—original draft preparation, J.A.P.; writing—review and editing, J.A.P. and J.A.C.; visualization, J.A.P.; supervision, J.A.C.; project administration, J.A.C.; funding acquisition, J.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the MINECO-FEDER project DPI2015-68905-P and DGA Group T04-FSE.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of the Monotonicity Property of the Robot's Pose Uncertainty

Appendix A.1. Representation of the State

The goal of a SLAM framework is to simultaneously reconstruct both the 3D trajectory of the system and a sufficiently accurate model of the navigation area, from the observations provided by the available sensors. Following a reduced-order notation of [40], the state of the system at time k is given by the tuple:

$$x_k = (R_k,\ p_k) \tag{A1}$$

where $R_k \in \mathrm{SO}(3)$ is the rotation matrix and $p_k \in \mathbb{R}^3$ is the translation vector.

Appendix A.2. Representation of the Pose's Uncertainty

Based on a matrix Lie group representation, at any given time step k, the real location of the system w.r.t. the global reference frame, T_k, can be represented by the product of a "large" estimated location, $\bar{T}_k \in \mathrm{SE}(3)$, and a "small" perturbation, $\hat{d}_k \in \mathfrak{se}(3)$:

$$T_k \equiv T_k(R_k,\ p_k) = \exp(\hat{d}_k)\bar{T}_k \tag{A2}$$

where the perturbation is expressed w.r.t. the first reference frame [41,42]. Considering the perturbation as a Gaussian random vector $d_k = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T \in \mathbb{R}^6$ for the 3D and $d_k = (dx, dy, d\theta)^T \in \mathbb{R}^3$ for the 2D case, its mean and covariance matrix are given by:

$$\bar{d}_k = \mathbb{E}[d_k] \tag{A3}$$

$$\Sigma_k = \mathbb{E}[(d_k - \bar{d}_k)(d_k - \bar{d}_k)^T] \tag{A4}$$

For notational convenience, the hat operator of Equation (A2) converts the vector d_k to a member of the Lie algebra $\mathfrak{se}(n)$. That is, for the 2 and 3D cases, respectively:

$$\hat{d}_k^{2D} = \left(\begin{array}{c|c} [d\theta]_\times & \begin{matrix} dx \\ dy \end{matrix} \\ \hline \begin{matrix} 0 & 0 \end{matrix} & 0 \end{array}\right) = \begin{pmatrix} 0 & -d\theta & dx \\ d\theta & 0 & dy \\ 0 & 0 & 0 \end{pmatrix} \tag{A5}$$

$$\hat{d}_k^{3D} = \left(\begin{array}{c|c} & \begin{matrix} dx \\ dy \\ dz \end{matrix} \\ [d\omega]_\times & \\ \hline \begin{matrix} 0 & 0 \end{matrix} & 0 \end{array}\right) = \begin{pmatrix} 0 & -d\omega_z & d\omega_y & dx \\ d\omega_z & 0 & -d\omega_x & dy \\ -d\omega_y & d\omega_x & 0 & dz \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A6}$$

where $[d\theta]_\times$ and $[d\omega]_\times$ are the $\mathbb{R}^{2\times2}$ and $\mathbb{R}^{3\times3}$ skew symmetric matrices corresponding to the hat operators in $\mathfrak{se}(2)$ and $\mathfrak{se}(3)$.

Elements of Equations (A5) and (A6) from the tangent space $\mathfrak{se}(n)$ can be associated to their underlying Lie groups $\mathrm{SE}(n)$ by the group exponential maps [43]. Note that an alternative representation is also possible by considering the "small" perturbation being expressed in the k-th reference frame $T_k = \bar{T}_k \exp(\hat{d}'_k)$ where d'_k and d_k are related by the adjoint action on the Lie

group [40,44] as $d'_k = Ad_{T_{k1}}(d_k)$. However, for the discussion below on monotonicity, it is required that every perturbation correspond to the same tangent space to be comparable.

Appendix A.3. Uncertainty Propagation

Let's now consider two noisy poses, T_{AB} and T_{BC}, both contained in SE(3) with their associated "small" perturbation errors d_A and d_B. The composed pose, T_{AC}, can be estimated by the matrix product,

$$T_{AC} = T_{AB}\, T_{BC} = \exp(\hat{d}_A)\bar{T}_{AB} \exp(\hat{d}_B)\bar{T}_{BC} \tag{A7}$$

Using the following definition of the adjoint,

$$\exp\left(Ad_X Y\right) \triangleq X \exp\left(Y\right) X^{-1} \tag{A8}$$

or, equivalently,

$$\exp\left(Y\right) = X^{-1} \exp\left(Ad_X Y\right) X \tag{A9}$$

and setting $Y = \hat{d}_B$ and $X = \bar{T}_{AB}$, Equation (A7) results in,

$$T_{AC} = \exp\left(\hat{d}_A\right) \exp\left(Ad_{T_{AB}} \hat{d}_B\right) \bar{T}_{AB} \bar{T}_{BC} \tag{A10}$$

$$= \exp\left(\hat{d}_A\right) \exp\left(Ad_{T_{AB}} \hat{d}_B\right) \bar{T}_{AC} \tag{A11}$$

Finally, using the first-order approximation [45–47] of the Baker-Campbell-Haussdorf formula for the product of the exponential maps, it can be inferred that,

$$T_{AC} \simeq \exp\left(\hat{d}_A + Ad_{T_{AB}} \hat{d}_B\right) \bar{T}_{AC} \tag{A12}$$

Thus, the perturbation of the composed pose T_{AC} will be:

$$\hat{d}'_A \simeq \hat{d}_A + Ad_{T_{AB}} \hat{d}_B \tag{A13}$$

Assuming that the perturbation errors d_A and d_B are uncorrelated with one another, the uncertainty of $\hat{d}'_A$ is,

$$\Sigma'_A \simeq \Sigma_A + Ad_{T_{AB}} \Sigma_B Ad^T_{T_{AB}} \tag{A14}$$

where $Ad_{T_{AB}}$ is the adjoint representation of transformation $\bar{T}_{AB}$, defined, in SE(2) and SE(3) respectively, as,

$$Ad^{2D}_{T_{AB}} = \begin{pmatrix} \cos\phi_{AB} & -\sin\phi_{AB} & y_{AB} \\ \sin\phi_{AB} & \cos\phi_{AB} & -x_{AB} \\ 0 & 0 & 1 \end{pmatrix} \tag{A15}$$

$$Ad^{3D}_{T_{AB}} = \begin{pmatrix} R_{AB} & D_{AB} R_{AB} \\ 0 & R_{AB} \end{pmatrix} \tag{A16}$$

where R_{AB} is an orthogonal rotation matrix and D_{AB} is the 3×3 skew symmetric matrix related to the translational part of $\bar{T}_{AB}$.

Appendix A.4. Monotonicity of Utility Functions

In the literature different utility functions to quantify the uncertainty of the robot's pose have been proposed both from the perspective of the TOED and IT: (i) *T-opt* (trace of the covariance matrix, or sum of its eigenvalues) [11,48–50]; (ii) *D-opt* (determinant of the covariance matrix, or product of its

eigenvalues) [51–53]; (iii) *E-opt* (maximum or minimum eigenvalue, depending on the notation) [54] criteria, and (iv) Shannon's entropy [55,56].

As reported in [9,57,58], monotonicity (the property that states that uncertainty increases as the robot moves during exploratory trajectories) is essential for adequate decision making in the context of active SLAM. Moreover, Rodriguez-Arevalo et al. [9] demonstrated that monotonicity holds for any utility function when uncertainty is represented differentially, regardless of the chosen representation for the robot's orientation. Also, Kim and Kim [58] demonstrated that using a Lie-group based representation monotonicity also holds.

Notably, from Equation (A14),

$$\Sigma'_A - \Sigma_A \simeq Ad_{T_{AB}} \Sigma_B Ad^T_{T_{AB}} \tag{A17}$$

it holds that, because Σ_B is a positive semi-definite matrix, $Ad_{T_{AB}}$ is a non-singular matrix (with $\det(Ad_{T_{AB}}) = 1$) and $Ad_{T_{AB}} \Sigma_B Ad^T_{T_{AB}}$ is a positive semi-definite matrix. Also, from [59], if A and B are symmetric matrices contained in $\mathbb{R}^{n \times n}$ it can be stated that,

- $A \geq B \Rightarrow \text{tr}(A) \geq \text{tr}(B)$.
- $A \geq B \geq 0 \Rightarrow \det(A) \geq \det(B)$.
- $A \geq B \Rightarrow \lambda_i(A) \geq \lambda_i(B)\ \forall i = (1, ..., n)$ with λ_i the i-th largest eigenvalue of A and B, respectively.

Therefore, because $\Sigma'_A \geq \Sigma_A \geq 0$, the four utility functions would satisfy monotonicity and, therefore, would result in a reliable decision making.

Appendix B. On the Complexity

Consider a neural network with n the number of inputs, m the number of outputs, H the number of hidden layers, m_i the intermediate outputs of the i-th hidden layer, and h_i the number of neurons in the i-th hidden layer. The computations required to propagate the information trough the network (feed-forward) during testing are one matrix multiplication and one activation function per layer. Thus, the number of multiplications performed will be,

$$n \times m_1 + m_1 \times m_2 + \cdots + m_{H-1} \times m_H + m_H \times m \tag{A18}$$

$$= n \times m_1 + m_H \times m + \sum_{i=2}^{H-1} m_i \times m_{i-1} \tag{A19}$$

which allows to express the complexity as:

$$\Theta\left(n \times m_1 + m_H \times m + \sum_{i=2}^{H-1} m_i \times m_{i-1} \right) \tag{A20}$$

where $\Theta(\cdot)$ is a strict bound, following the Bachmann–Landau or asymptoticwhere $\Theta(\cdot)$ is a strict bound, following the Bachmann–Landau or asymptotic notation.

Consider now the number of hidden units per hidden layer to be equal for all hidden layers, and denoted as $\hat{m}$, and the number of hidden layers to be $H = 2$. Then, Equation (A20) can be rewritten as:

$$\Theta\left((n + m + \hat{m}) \times \hat{m} \right) \tag{A21}$$

On the other hand, the complexity of applying the activation function can be expressed, under the previous assumptions, as $\mathcal{O}(\hat{m})$, where $\mathcal{O}(\cdot)$ is the upper bound. Therefore, the global complexity of the problem will be $\mathcal{O}\left((1 + n + m + \hat{m}) \times \hat{m}\right)$, or, as a function of only the number of inputs and outputs, $\Theta(n + m)$. The number of outputs is equal to the dimension of the action set,

$\dim(A) \equiv |A| = m$, and the number of inputs directly affects the observation set, O. For example, in the studied case where a laser scan is used, the dimension of the observation set would be,

$$\dim(O) \equiv |O| = \left((\max_{\text{range}} - \min_{\text{range}}) \times 10^2 \right)^n = k^n \tag{A22}$$

Thus, the time complexity of solving the problem will be linear in the action set and logarithmic in the observation set according to $\Theta\left(|A| + \log(|O|)\right)$.

Traditional resolution of the problem may result in different complexities depending on the method selected. For example, value iteration for a MDP is known to be $\mathcal{O}\left(|A| \times |S|^2\right)$ complex, where $|S|$ is the dimension of the state set; and for a POMDP, $\mathcal{O}\left(c^{|A|} \times c^{|O|}\right)$ with $c > 1$, i.e., exponentially complex in the action and observation sets. Note that the previous complexity corresponds to the most demanding stage, being lower order terms ignored, such as the search over the action set, or the computation of the utility function that would be, for example, for D-opt, $\Omega(|S|^{2.373})$, where $\Omega(\cdot)$ is a lower bound.

References

1. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110, doi:10.1109/MRA.2006.1638022.
2. Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot.* **2007**, *23*, 34–46, doi:10.1109/TRO.2006.889486.
3. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332, doi:10.1109/TRO.2016.2624754.
4. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163, doi:10.1109/TRO.2015.2463671.
5. Burgard, W.; Fox, D.; Thrun, S. Active mobile robot localization. In Proceedings of the 1997 International Joint Conferences on Artificial Intelligence, Nagoya, Japan, 25 February 1997; pp. 1346–1352.
6. Feder, H.J.S.; Leonard, J.J.; Smith, C.M. Adaptive mobile robot navigation and mapping. *Int. J. Robot. Res.* **1999**, *18*, 650–668, doi:10.1177/02783649922066484.
7. Carrillo, H.; Reid, I.; Castellanos, J.A. On the comparison of uncertainty criteria for active SLAM. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, USA , 14–18 May 2012; pp. 2080–2087, doi:10.1109/ICRA.2012.6224890.
8. Makarenko, A.A.; Williams, S.B.; Bourgault, F.; Durrant-Whyte, H.F. An experiment in integrated exploration. In Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 534–539, doi:10.1109/IRDS.2002.1041445.
9. Rodriguez-Arevalo, M.L.; Neira, J.; Castellanos, J.A. On the importance of uncertainty representation in Active SLAM. *IEEE Trans. Robot.* **2018**, *34*, 829–834, doi:10.1109/TRO.2018.2808902.
10. Carlone, L.; Du, J.; Ng, M.K.; Bona, B.; Indri, M. Active SLAM and exploration with particle filters using Kullback-Leibler divergence. *J. Intell. Robot. Syst.* **2014**, *75*, 291–311, doi:10.1007/s10846-013-9981-9.
11. Leung, C.; Huang, S.; Dissanayake, G. Active SLAM using model predictive control and attractor based exploration. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, 9–15 October 2006; pp. 5026–5031, doi:10.1109/IROS.2006.282530.
12. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; others. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529, doi:10.1038/nature14236.
13. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.

14. Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; De Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the 2016 International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1995–2003.
15. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, LA, USA, 2–7 February 2018; pp. 3215–3222.
16. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the 2016 International Conference on Machine Learning, New York, NY, USA, 19– 24 June 2016; pp. 1928–1937.
17. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv* **2018**, arXiv:1801.01290.
18. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. In Proceedings of the 2016 International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–2.
19. Tai, L.; Liu, M. Mobile robots exploration through cnn-based reinforcement learning. *Robot. Biomim.* **2016**, *3*, 24, doi:10.1186/s40638-016-0055-x.
20. Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A.J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. Learning to navigate in complex environments. In Proceedings of the 2017 International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
21. Dhiman, V.; Banerjee, S.; Griffin, B.; Siskind, J.M.; Corso, J.J. A critical investigation of deep reinforcement learning for navigation. *arXiv* **2018**, arXiv:1802.02274.
22. Wen, S.; Zhao, Y.; Yuan, X.; Wang, Z.; Zhang, D.; Manfredi, L. Path planning for active SLAM based on deep reinforcement learning under unknown environments. *Intell. Serv. Robot.* **2020**, 1–10, doi:10.1007/s11370-019-00310-w.pdf.
23. Ryan, R.M.; Deci, E.L. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemp. Educ. Psychol.* **2000**, *25*, 54–67, doi:10.1006/ceps.1999.1020.
24. Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; Munos, R. Unifying count-based exploration and intrinsic motivation. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 1471–1479.
25. Chaplot, D.S.; Gandhi, D.; Gupta, S.; Gupta, A.; Salakhutdinov, R. Learning To Explore Using Active Neural SLAM. In Proceedings of the 2019 International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
26. Zhelo, O.; Zhang, J.; Tai, L.; Liu, M.; Burgard, W. Curiosity-driven exploration for mapless navigation with deep reinforcement learning. *arXiv* **2018**, arXiv:1804.00456.
27. Oh, C.; Cavallaro, A. Learning Action Representations for Self-supervised Visual Exploration. In Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 5873–5879, doi:10.1109/ICRA.2019.8794401.
28. Bai, S.; Chen, F.; Englot, B. Toward autonomous mapping and exploration for mobile robots through deep supervised learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2379–2384, doi:10.1109/IROS.2017.8206050.
29. Chen, F.; Bai, S.; Shan, T.; Englot, B. Self-learning exploration and mapping for mobile robots via deep reinforcement learning. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; doi:10.2514/6.2019-0396.
30. Chaplot, D.S.; Parisotto, E.; Salakhutdinov, R. Active neural localization. In Proceedings of the 2019 International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
31. Gottipati, S.K.; Seo, K.; Bhatt, D.; Mai, V.; Murthy, K.; Paull, L. Deep Active Localization. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4394–4401, doi:10.1109/LRA.2019.2932575.
32. Chen, F.; Martin, J.D.; Huang, Y.; Wang, J.; Englot, B. Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs. *arXiv* **2020**, arXiv:2007.12640.
33. Zhu, D.; Li, T.; Ho, D.; Wang, C.; Meng, M.Q.H. Deep reinforcement learning supervised autonomous exploration in office environments. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 7548–7555, doi:10.1109/ICRA.2018.8463213.

34. Niroui, F.; Zhang, K.; Kashino, Z.; Nejat, G. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 610–617, doi:10.1109/LRA.2019.2891991.
35. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36, doi:10.1109/IROS.2017.8202134.
36. LeCun, Y.; Touresky, D.; Hinton, G.; Sejnowski, T. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School*; Morgan Kaufmann: Pittsburgh, PA, USA, 1988; Volume 1, pp. 21–28.
37. Zamora, I.; Lopez, N.G.; Vilches, V.M.; Cordero, A.H. Extending the OpenAI Gym for robotics: A toolkit for reinforcement learning using ROS and Gazebo. *arXiv* **2016**, arXiv:1608.05742.
38. Kaess, M.; Johannsson, H.; Roberts, R.; Ila, V.; Leonard, J.J.; Dellaert, F. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Int. J. Robot. Res.* **2012**, *31*, 216–235.
39. Mur-Artal, R.; Tardós, J.D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262.
40. Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Trans. Robot.* **2017**, *33*, 1–21.
41. Wang, Y.; Chirikjian, G.S. Error propagation on the Euclidean group with applications to manipulator kinematics. *IEEE Trans. Robot.* **2006**, *22*, 591–602.
42. Barfoot, T.D.; Furgale, P.T. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Trans. Robot.* **2014**, *30*, 679–693.
43. Murray, R.M.; Li, Z.; Sastry, S. *A Mathematical Introduction to Robotic Manipulation*; CRC Press: Boca Raton, FL, USA, 1994.
44. Bourmaud, G.; Megret, R.; Arnaudon, M.; Giremus, A. Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions. *J. Math. Imaging Vis.* **2015**, *51*, 209–228.
45. Brossard, M.; Bonnabel, S.; Condomines, J.P. Unscented Kalman filtering on Lie Groups. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 2485–2491.
46. Bergerl, J.; Lenzen, F.; Becker, F.; Neufeld, A.; Schnoerr, C. Second-Order Recursive Filtering on the Rigid-Motion Lie Group SE(3) Based on Nonlinear Observations. *J. Math. Imaging Vis.* **2017**, *58*, 102–129.
47. Barrau, A.; Bonnabel, S. Invariant Kalman Filtering. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 237–257.
48. Chernoff, H. Locally optimal designs for estimating parameters. *Ann. Math. Stat.* **1953**, 24, 586–602.
49. Kollar, T.; Roy, N. Trajectory optimization using reinforcement learning for map exploration. *Int. J. Robot. Res.* **2008**, *27*, 175–196.
50. Meger, D.; Rekleitis, I.; Dudek, G. Heuristic search planning to reduce exploration uncertainty. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, 22–26 September 2008; pp. 3392–3399.
51. Wald, A. On the efficient design of statistical investigations. *Ann. Math. Stat.* **1943**, *14*, 134–140.
52. Vidal-Calleja, T.; Davison, A.J.; Andrade-Cetto, J.; Murray, D.W. Active control for single camera SLAM. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL, USA, 15–19 May 2006; pp. 1930–1936.
53. Kim, A.; Eustice, R.M. Perception-driven navigation: Active visual SLAM for robotic area coverage. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 3196–3203.
54. Ehrenfeld, S. On the efficiency of experimental designs. *Ann. Math. Stat.* **1955**, *26*, 247–255.
55. Stachniss, C.; Grisetti, G.; Burgard, W. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. *Robot. Sci. Syst.* **2005**, *2*, 65–72.
56. Blanco, J.L.; Fernandez-Madrigal, J.A.; Gonzalez, J. A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao-Blackwellized Particle Filters. *Int. J. Robot. Res.* **2008**, *27*, 73–89.
57. Carrillo, H.; Latif, Y.; Rodriguez-Arevalo, M.L.; Neira, J.; Castellanos, J.A. On the monotonicity of optimality criteria during exploration in active SLAM. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1476–1483.

58. Kim, Y.; Kim, A. On the Uncertainty Propagation: Why Uncertainty on Lie Groups Preserves Monotonicity? In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3425–3432.
59. Seber, G.A. *A Matrix Handbook for Statisticians*; John Wiley & Sons: Hoboken, NJ, USA, 2008; Volume 15.