Generalization in Deep Reinforcement Learning for Robotic Navigation by Reward Shaping

Victor R. F. Miranda, Armando A. Neto, Gustavo M. Freitas, Member, IEEE, and Leonardo A. Mozelli

Abstract—This article addresses the application of deep reinforcement learning (DRL) methods in the context of local navigation, i.e., a robot moves toward a goal location in unknown and cluttered workspaces equipped only with limited-range exteroceptive sensors. Collision avoidance policies based on DRL present advantages, but they are quite susceptible to local minima, once their capacity to learn suitable actions is limited to the sensor range. We address this issue by means of reward shaping in actor–critic networks. A dense reward function, that incorporates map information gained in the training stage, is proposed to increase the agent’s capacity to decide about the best action. Also, we offer a comparison between the twin delayed deep deterministic policy gradient and soft actor–critic algorithms for training our policy. A set of sim-to-sim and sim-to-real trials illustrate that our proposed reward shaping outperforms the compared methods in terms of generalization, by arriving at the target at higher rates in maps that are prone to local minima and collisions.

Index Terms—Deep reinforcement learning (DRL), local navigation, mobile robots, unknown cluttered environments.

I. INTRODUCTION

Among the open problems in mobile robotics, safe navigation and exploration in unknown scenarios are two of the most challenging ones. Several questions still demand better solutions when a robot is faced with the task of moving to a relative target position based on the local perception of the environment, especially in cluttered spaces or when optimized trajectories are required in real time. In the last few years, propelled by advancements in deep neural networks, deep reinforcement learning (DRL) techniques have gained relevance for autonomous systems [1].

Manuscript received 15 December 2022; revised 8 March 2023 and 24 May 2023; accepted 16 June 2023. Date of publication 5 July 2023; date of current version 19 January 2024. This work was supported in part by CAPES - Finance Code 001, CNPq under Grant 310446/2021-0, Grant 306286/2020-3, and Grant 407063/2021-8, in part by the FAPEMIG under Grant 3APQ-0228-22, and in part by NVIDIA Corporation for the donation of the Jetson TX2 GPU used for this research. (Corresponding author: Victor R. F. Miranda.)

The authors are with the Graduate Program in Electrical Engineering (PPGEE), Department of Electronics Engineering (DELT), Department of Electrical Engineering (DEE), Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG 31270-901, Brazil (e-mail: victormrfm@ufmg.br; aaneto@cpdee.ufmg.br; gustavomfreitas@ufmg.br; mozelli@cpdee.ufmg.br).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIE.2023.3290244.

Digital Object Identifier 10.1109/TIE.2023.3290244

Unlike more classical approaches, which split the navigation problem into simpler tasks, DRL techniques are used as end-to-end approaches, mapping the raw sensor data directly to input commands to the robots. Conceptually, DRL is more robust to other machine learning paradigms, since the agent learns from its mistakes during the training stage and can extrapolate this knowledge to unseen situations, avoiding overfitting. However, there is the risk of sample inefficiency and simulations are more time-consuming. Therefore, it is quite challenging to train model-free policies in growing spaces, increasing the generalization with a large set of possible obstacle configurations.

To address the generalization problem in DRL, some solutions have been proposed: increase the similarity between training and execution environments with data augmentation [2], use domain randomization or generate different environments along training [3], [4], and handle differences between environments with regularization techniques [5]. However, these strategies require wide data variation and training steps to reduce sample inefficiency. Besides, the encoded behavior in a simulation may underperform or even fail when deployed in the real world.

In this article, we address the generalization of robot navigation using DRL in mapless, obstacle-filled environments. We propose a new state representation, continuous action space, and reward function in the DRL framework to improve sim-to-sim and sim-to-real transfer capability. Our strategy increases the exploration of actions during training, avoiding local minima in cluttered environments, i.e., actions that maintain the robot stuck or moving around the same place. This new approach endows the robot with some capabilities for untrained scenarios, such as moving in obstacle-free directions and performing progressive movements (not standing still). By balancing between behaviors that increase information in a local map or reduce the distance to the target, we can improve generalization. When compared with the current literature, we aim for the following contributions:

1) shape the reward function to incorporate exteroceptive information (newer perception information from the environment map) and to improve the agent’s robustness to the local minima of the environment;
2) a comparative analysis among the model trained by the proposed reward shaping and others from the literature;
3) a comparative analysis using different training algorithms;
4) a set of sim-to-sim and sim-to-real experiments to illustrate the generalization of our proposed approach.

II. RELATED WORK

There are many strategies to safely (and/or efficiently) navigate a robot toward a target point in the environment, ranging from map-based approaches to model-free deep reinforcement learning (DRL). Map-based approaches, such as occupancy grids and potential fields, provide a clear advantage in terms of safety and robustness, but they require a significant amount of computational resources and are not scalable to large or complex environments [6].

On the other hand, model-free approaches, such as deep Q-learning [2] and actor–critic algorithms [7], have gained popularity due to their ability to learn policies directly from raw sensor data without the need for a map. However, they suffer from sample inefficiency, especially in high-dimensional and complex environments, leading to long training times and limited generalization to unseen situations [8].

To address these limitations, recent research has focused on improving the generalization capabilities of deep reinforcement learning methods. Techniques such as domain randomization [9], data augmentation [10], and regularization [5] have been proposed to enhance the ability of DRL algorithms to generalize from training environments to real-world scenarios. These strategies require additional data or computational resources, which may not be feasible in all cases [11].

More recently, there has been a trend towards sim-to-real transfer learning, where the goal is to transfer the learned policies from simulations to real-world environments. This approach has shown promising results, especially in scenarios where high-quality simulations are available, allowing for efficient exploration and data collection [12]. However, sim-to-real transfer remains a challenging problem, and further research is needed to address the limitations of this approach, such as the need for high-quality simulations and the potential for sample inefficiency in real-world settings [13].

Despite the progress made in these areas, there is still a lack of consensus on the best practices for achieving effective generalization in deep reinforcement learning. This article contributes to the ongoing discussion by proposing a novel approach to reward shaping in the context of local navigation, which aims to improve generalization by incorporating information from the environment map and increasing the similarity between training and execution environments.

References

[1] A. A. Neto, L. A. Mozelli, and G. M. Freitas, “Generalization in deep reinforcement learning for robotic navigation by reward shaping,” IEEE Trans. Ind. Electron., vol. 71, no. 6, pp. 6013–6022, June 2024.
[2] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Apr. 2015.
[3] J. Schulman et al., “Equivalence of policy gradients,” in Proc. 34th Int. Conf. Mach. Learn. (ICML), 2017, pp. 121–130.
[4] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, Nov. 2016.
[5] M. G. Bellemare et al., “The arcadia project: Model-free deep reinforcement learning,” in Proc. 31st Int. Conf. Mach. Learn. (ICML), 2014, pp. 121–129.
[6] C. B. Schoenauer and J. J. Laumond, “Obstacle avoidance for nonholonomic mobile robots,” IEEE Trans. Robot. Autom., vol. 12, no. 5, pp. 646–658, Oct. 1996.
[7] S. P. Singh and M. R.avery, “Generalization and reinforcement learning,” in Proc. 15th Int. Joint Conf. Artif. Intell., 1997, pp. 1122–1127.
[8] D. Precup et al., “Addressing the exploration problem throughطة الاستكشاف في المواقف الفصلية,” in Proc. 15th Int. Joint Conf. Artif. Intell., 1997, pp. 1122–1127.
[9] A. A. Neto, L. A. Mozelli, and G. M. Freitas, “Generalization in deep reinforcement learning for robotic navigation by reward shaping,” IEEE Trans. Ind. Electron., vol. 71, no. 6, pp. 6013–6022, June 2024.
[10] A. A. Neto, L. A. Mozelli, and G. M. Freitas, “Generalization in deep reinforcement learning for robotic navigation by reward shaping,” IEEE Trans. Ind. Electron., vol. 71, no. 6, pp. 6013–6022, June 2024.
[11] A. A. Neto, L. A. Mozelli, and G. M. Freitas, “Generalization in deep reinforcement learning for robotic navigation by reward shaping,” IEEE Trans. Ind. Electron., vol. 71, no. 6, pp. 6013–6022, June 2024.
from reactive to deliberative hierarchies. Classical planning methods generally provide collision-free, shortest distance or time-efficient paths [6], [7], [8], [9], but they often require prior knowledge of the map and the obstacles’ location. Alternative approaches were proposed to meet the requirement of navigating environments with obstacles. One such approach is to use bug algorithms that provide actions to bypass obstacles obstructing the path to the destination [10]. Another method uses potential field-based computation to derive robot velocities for collision-free paths [11]. However, these strategies may experience local minimal issues when operating in complex environments due to the nullification of antagonistic forces.

The utilization of DRL algorithms as the fundamental control [12], navigation [13], localization [14], and planning [15] systems has gained notoriety recently. The adaptability of artificial neural network (ANN), combined with the ability to simulate and train, as well as employ them as end-to-end solutions, has caught the attention of the research community, compelling them to integrate these advancements into the realm of robotics. Autonomous robot navigation is an area of interest for DRL systems, which examine the effectiveness of policies trained through reinforcement learning and the different algorithms specifically designed for this purpose [16], [17].

Ruan et al. [18] used discrete actions with a double deep Q-network (DDQN) algorithm for navigating an environment using only camera images and avoiding obstacles. Although effective in some situations, discrete actions can hinder performance in complex cluttered environments, and their reward function neglects local minimum problems. Similarly, Chen et al. [19] also used a DDQN algorithm with discrete actions for navigation, relying on distance-based rewards for goal achievement and collision avoidance, but the method requires an occupancy grid map as input, which increases computational costs at runtime. Liu et al. [20] proposed a navigation policy that uses discrete actions with the asynchronous advantage actor–critic algorithm, allowing for global agent training from parallel agents. This approach facilitates generalization across a variety of maps and navigation scenarios. In contrast, our proposed strategy focuses on obstacle avoidance using data from an occupancy map and light detection and ranging (LiDAR), which requires increased computation processing for both training and runtime.

Considering continuous actions, Grando et al. [21] proposed a method for training a policy that controls a hybrid unmanned aerial vehicle (UAV) to navigate toward a target point while avoiding obstacles using the deep-deterministic policy gradient (DDPG) algorithm. The model employs the robot’s states, LiDAR measurements, and distance to the target point, to select the best control actions for the UAV. Despite the qualitative results, the chosen reward function is too sparse and may result in poor performance in complex scenarios. In [22], raw depth images captured at four steps are fed to a double soft actor–critic (SAC) architecture, where a primary network trains the navigation policy and a secondary network deals with obstacle avoidance.

Similarly, Zhu and Hayashi [23] adopted a hierarchical approach, where a low-level policy is responsible for navigation toward the target, and a high-level policy ensures safety. Our proposal is different in that we use raw laser data and perform all tasks with the same network. Hierarchical networks generally help to improve learning efficiency and stability when the agent is facing complex tasks, but there are disadvantages when compared with single networks, such as an increase in the complexity of the algorithms, demand for more training, higher risk of overfitting, and more difficulty of hyperparameters tuning.

In the map exploration context, the authors of [24] and [25] presented strategies to explore the space using a trained policy navigation system. The first one employs the twin delayed deep-deterministic policy gradient (TD3) algorithm and offers a reward function that observes only arrivals, collisions, and nonprogressive movement conditions. For the second one, the DDPG was used for training, and a reward that includes a safety clearance term, avoiding obstacles, and moving in the target direction was designed. Since the robot must navigate long distances, local minima issues are frequently seen in map exploration tasks. Cimurs et al. [24] proposed auxiliary algorithms to help the target destination avoid local minimum and increase the method generalization, and Hu et al. [25] used a safety clearance reward to prevent a high attraction of the robot to the target. However, the algorithms used for training do not present high action exploration during agent learning, and the proposed reward function does not deal with this issue directly.

Once local minima represent a hard difficulty in the learning process, we have to seek forms to facilitate the training, and a good alternative is the shaping technique. According to [26], in reward shaping: 1) we can modify the reward function to make it less sparse, given the agent intermediate rewards more frequently, or 2) we can progressively increase the environment’s complexity until we reach the final target. Hu et al. [27] also referred to reward shaping as the act of modifying the original reward function with a shaping reward function that incorporates domain knowledge. The authors in [28] used the shaping strategy to position a grip’s manipulator around a cube, grasp it, and lift it off, which can be considered a complex task concerning real-world robots.

When facing the local minima problem, we also expect to generalize our trained policy to scenarios with distinct configurations. In the literature, this generalization is defined as a paradigm for neural network model training. In the context of DRL, several studies discuss the capability of models in actuating on untrained situations and address methods to improve their efficiency [29]. Some papers present forms of quantifying, measuring, and characterizing the generalization in DRL methods [30], [31], and others (as the present article) focus on developing strategies to increase the generalization capability of the policies by modifying the learning algorithms. Another concern in terms of generalization is the transfer of simulation-trained models to real-world situations. Similarly to [32], we also address the sim-to-real problem.
III. DRL SETUP

The navigation problem generally consists of computing a sequence of commands that moves the robot along a path represented by a curve or a sequence of waypoints in the workspace. Furthermore, we can incorporate safe navigation into this problem when facing environments with obstacles and then using obstacle-avoidance techniques. The most common strategies for safe navigation involve planning and motion control steps, which often require the environment map and impose a high computational cost. In our approach, an end-to-end neural network, that receives laser data, robot pose, and the target position as inputs, commands the robot to safely navigate in the unknown space, avoiding obstacles until reaching the goal. Fig. 1 shows the complete procedure of our proposed navigation system. First, a goal point is defined and sensors collect data from the robot’s states and the environment. Second, the system processes these data to create the observation state $s$ for the trained navigation policy model. Finally, the system defines actions $a$ (velocities) for the robot based on the observed states. Mathematically, the desired policy is given by $a = \mu_\theta(s)$, with $\theta$ being the parameters of the ANN.

### A. Observation State and Action Representation

The observation state vector $s$ includes the robot’s relative position to the target region $p \in \mathbb{R}^2$, absolute linear $v$ and angular $\omega$ velocities, and the information data provided by the robot’s planar LiDAR $l \in \mathbb{R}^L \subset \mathbb{R}^l$ with $l$ the number of laser beams. Formally, it can be represented as

$$s = [p \ v \ \omega \ l]_T^T.$$  

Some approaches feed the laser data to a neural network to extract features that are then passed to the policy. Instead, we are pooling the raw laser data, using the $l_c$ shortest distances, which allows more efficient training and avoidance of overfitting to some repetitive obstacles.

The continuous action vector includes the commanded linear velocity $\bar{v} \in \mathbb{R}^+$ and an angular velocity $\bar{\omega} \in \mathbb{R}$, represented by

$$a = [\bar{v} \ \bar{\omega}]_T^T.$$  

### B. Network Structure

We consider the actor–critic framework, where value function and policy are optimized jointly. Fig. 2 shows the proposed actor network structure composed of the observation state $s$ input layer, three hidden dense rectified linear unit (ReLU) layers with 512 nodes each, and the action output generated by merging values from a linear layer with a sigmoid and hyperbolic tangent activation functions. Critic network: input layer formed by the observation space merged with the action space, followed by three dense ReLU layers of 512 nodes and the output $Q$-value generated by a dense linear layer.

### C. Reward Shaping Strategy

In DRL, a critical aspect involves the definition of a reward function $r(s, a)$ that guides the agent’s decision-making process based on the observation state. Sparse and delayed rewards pose significant challenges in robotic navigation when local information is limited. For instance, rewarding the agent only upon reaching the goal or penalizing it for colliding with obstacles can result in several problems. In some scenarios, a trained policy can select actions that keep the agent stuck to a local minimum in the environment, performing movements in the target direction while also taking actions in the opposite direction to avoid obstacles. Moreover, learning effective escape strategies may require a vast number of scenarios. Therefore, we address this challenge by implementing a reward shaping technique that can enhance generalization, encouraging the agent to follow intermediate strategies that mitigate the issue of being trapped in a local minimum.

We start by defining an original reward function, hereinafter called the sparse term

$$r_{\text{sparse}}(s, a) = \begin{cases} r_a, & \text{if } d \leq d_{\text{min}} \\ r_c, & \text{if collision} \\ r_t, & \text{if steps} \geq \text{Timeout} \end{cases}$$

in which $r_a$ denotes a positive reward assigned when the Euclidean distance $d$ between the robot and the target is less than or equal to a minimal value $d_{\text{min}} > 0$, $r_c$ signifies a penalty imposed in the event of a collision, and $r_t$ represents a negative reward given when the number of steps exceeds a predefined timeout limit. Such a reward function is sparse because it only feeds the agent at terminal state conditions.

Next, we define a shaping function as the dense term

$$r_{\text{dense}}(s, a) = r_{Gd} + r_t + r_v$$

where

$$r_{Gd} = \frac{G}{d}$$

represents the ratio between the increased map information $G$ since the last measurement (with regards to newly mapped areas) and $d$ (that will never be less or equal to zero due to the stop criteria $d_{\text{min}} > 0$)

$$r_t = \min l \frac{l_2}{l_1}$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
is the minimum value measured by the LiDAR \( I \) in the interval \([l_1, l_2]\) representing a read in the robot’s front side, and

\[
r_v = v - |\omega|
\]

is a reward based on linear and angular velocities.

The initial coefficient \( r_{Gn} \) serves to reduce issues associated with local minima by providing reinforcement exclusively when the robot navigates to novel locations while decreasing the magnitude of this reinforcement as the distance between the robot and the target diminishes. This approach also helps to prevent the robot from revisiting locations that have previously been explored but do not facilitate advancement toward the target destination. The increased map information \( G \) is computed using data from the map occupancy grid created while the robot navigates through the environment. Basically, we calculate the difference between the sum of all occupied cells (\( Oc \)) and free cells (\( Fc \)) in the grid created at the current step \( t \) and the one in the last step \( t-1 \), as follows:

\[
G_t = \sum (Oc_t + Fc_t) - \sum (Oc_{t-1} + Fc_{t-1}).
\]

In addition, to promote motion into unoccupied regions, the coefficient \( r_1 \) offers a reinforcement proportional to the minimum reading of the LiDAR sensor ahead of the robot. To discourage unproductive movements, such as cyclic behaviors, the coefficient \( r_s \) bestows a reinforcement linked to the robot’s velocities [24]. Finally, we set our reward functions \( r(s, a) \) as the accumulated sum of \( r_{sparse} \) and \( r_{dense} \).

### IV. Results

This section evaluates our proposed approach by comparing it with the current literature. Two main aspects have been analyzed: 1) the impact of employing twin delayed deep-deterministic policy gradient (TD3) or soft actor–critic (SAC) algorithms in the context of safe local navigation for mobile robots, and 2) the effect of reward shaping. To do so, we have first trained our policies in different simulated cluttered (sparse and complex) scenarios. Then, concerning the success rate (the number of times the robot reaches the goal region without collision in a limited time), we compare our method with others in environments distinct from those used in the training stage. Next, to advance the generalization analysis of the solution, we added comparative trials in a second simulator (sim-to-sim analysis). Finally, we demonstrate the effectiveness with real-world experiments (sim-to-real analysis).

#### A. Training Setup

The training process was performed on a laptop with Ubuntu 20.04, equipped with an NVIDIA GTX 3060 graphics card, 16 GB of RAM, and an Intel Core i7-11800H CPU. The agent is represented by a differential wheeled robot navigating in limited workspaces of the standalone simulator presented in [33]. The three maps illustrated in Fig. 3, whose dimensions are 12 m×12 m, 20 m×20 m, and 40 m×40 m, respectively, have been employed at this step. All policies used in the experiments were modeled according to the actor–critic deep networks illustrated in Fig. 2 and trained by using the SAC and the TD3 algorithms, both with 20 000 episodes. Each training episode ends when the robot reaches the goal region, collides with some obstacle, or violates the Timeout of 500 steps. The map changes randomly every 500 episodes among those shown in Fig. 3. Besides that, the actions were limited to \( v \in [0, 0.5] \) m/s and \( \omega \in [-1, 1] \) rad/s. Also, \( d_{min} = 0.5 \). The LiDAR reads between \(-135^\circ\) and \(135^\circ\) with \( l = 684 \), pooled into \( l_c = 57 \) groups of 12 measures, selecting the minimal distances of each group [34].

Table I gives the learning parameters used for the training of the neural networks. In addition, we have set the constant reward values: \( r_n = 100 \), \( r_c = -200 \), and \( r_l = -200 \). In the case of TD3, the delayed rewards were updated over the last ten steps. In addition, from (6), \( l_1 = 336 \) and \( l_2 = 348 \), representing the LiDAR beam interval for the robot’s front-side read.

#### B. Comparative Analysis

Our main goal here is to evaluate the generalization capacity of the proposed method in contrast with the baseline methods. Therefore, after obtaining the trained policies, we execute some tests to evaluate the performance in untrained scenarios. Fig. 4 illustrates the tested maps, relatively larger than those of Fig. 3, with dimensions 40 m×40 m, 40 m×40 m, and 60 m×60 m, respectively.

In our comparative analysis, we have incorporated three different reward functions from the state of the art. In [21], the reward is the most sparse, using only collision information and the arrival at the goal. In [24], the reward adds the robot’s linear speed to the goal, but they incorporate intermediate rewards from the exteroceptive (LiDAR) data to compute a safety clearance reward term for collision avoidance purposes. Our trials were separated into two groups. First, for all rewards functions (including ours), we have applied the TD3 algorithm proposed in [35]. Then, for the same reward functions,
TABLE II

| Reward/Reference | Strategy-[24] | Strategy-[25] | Strategy-[21] |
|------------------|---------------|---------------|---------------|
| Collision        | $r_c = -100$  | -             | $r_c = -10$   |
| Arrival          | $r_a = 80$    | $r_a = 40$    | $r_a = 100$   |
| Others           | -             | $r_{cp} = 0, r_{cwp} = 0, r_{sw} = -1$ | -             |

![Graph](image)

Fig. 5. Moving average of all compared reward functions for the TD3 (in red) and the SAC (in blue) algorithms. (a) Reward function in [24]. (b) Reward function in [25]. (c) Reward function in [21]. (d) Our reward shaping function.

we have replaced TD3 with SAC algorithm. It is important to highlight that Cimurs et al. [24] used the TD3 to learn the navigation policy, whereas the authors in [21] and [25] used the DDPG [36], a predecessor of the TD3 and SAC. The learning parameters used for training all structures were the same in Table I and the same 20,000 episodes in approximately 40 training hours for each one, significantly less time than when using other simulators for training as in [24], which uses a similar machine setup. However, as we have considered different reward functions for comparing, the reward constants received values according to the authors’ proposal of each paper in analysis, as given in Table II. Fig. 5 presents the moving average evolution for all aforementioned reward functions, each one trained with both, TD3 and SAC algorithms. In most cases, the rewards evolve side by side (with some variations) throughout the training. The periodic oscillations that appear in the graphs refer to each change in the training map, where the value decreases for a while and increases according to the train evolution in the new map.

Table III compiles the results obtained for the first set of simulations using the TD3. The percentage value for each entry scenario/method was calculated by counting the number of trials in which the robot reached the goal position before the timeout of 1500 steps or colliding with an obstacle. For each attempt, we set the starting and goal locations following 15 pairs of positions that were repeated during all experiments. The first observation is that our reward function outperforms all others in terms of completed missions. The results were significantly superior, although [25] also stood out from the other two. The second observation is that the results follow a progression: making the reward less sparse improves the generalization capacity. The results also allow us to conclude that Map 1 in Fig. 4(a) is the least complex environment, whereas Map 3 in Fig. 4(c) is the most complex regarding obstacles and local minima situations, once all reward functions follow this pattern. Still evaluating the reward functions, an outlier appears in scenario 3 when solely comparing the performance obtained by the rewards presented in [21] and [24]. Although [24] is better in maps 1 and 2, the performance deteriorates sharply in map 3 even with the simpler reward function proposed in [21]. Table IV contains the results obtained using the SAC, following the same test parameters used for TD3. Results show an increase in the number of successfully completed trials when using SAC for training compared with TD3. This performance shows that, for the robot navigation policies, increasing the exploration of actions during training helps the agent to select better actions in untrained scenarios, that is, increasing the generalization of the trained network. Therefore, adopting exploration techniques as including the entropy maximization on training, which is done in the SAC algorithm, helps to increase the generalization. Although most of the results present this performance increase comparing the model trained using SAC instead of TD3, the proposed rewards functions presented in [21] and [25] decreased only in scenario 3. However, these results can be observed as outliers since, in all other experiments, the SAC results were better than TD3, and also maintaining the performance results observed about the different rewards.

Regarding the proposed reward function, some benefits can be commented on. Even when using the TD3 algorithm, our

TABLE III

| Scenario/Method | TD3-[24] | TD3-[25] | TD3-[21] | TD3-Ours |
|-----------------|---------|---------|---------|---------|
| Map 1 – Fig.4(a) | 63.4%   | 81.6%   | 56.2%   | 86.4%   |
| Map 2 – Fig.4(b) | 51.4%   | 54.2%   | 36.6%   | 57.2%   |
| Map 3 – Fig.4(c) | 18.2%   | 38.4%   | 33.8%   | 42.0%   |

TABLE IV

| Scenario/Method | SAC-[24] | SAC-[25] | SAC-[21] | SAC-Ours |
|-----------------|---------|---------|---------|---------|
| Map 1 – Fig.4(a) | 80.2%   | 88.8%   | 64.8%   | 59.2%   |
| Map 2 – Fig.4(b) | 61.0%   | 61.6%   | 45.2%   | 83.0%   |
| Map 3 – Fig.4(c) | 35.6%   | 34.2%   | 31.6%   | 59.4%   |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
approach outperforms SAC-[24] and SAC-[21], for map 1, SAC-[21] for map 2, and all methods for map 3. It is slightly worse than SAC-[25], for maps 1 and 2 (under 4.5% difference), and than SAC-[24], for map 2 (under 4% difference). When the best combinations of reward and training algorithms are compared, our approach outperforms all the others. Compared with the second best method, it is 7.2%, 34.7%, and 54.7% better for maps 1–3, respectively.

C. Sim-to-Sim Analysis

Sim-to-sim analysis is often an intermediate step to sim-to-real transfer tests since, in robotics, collecting and validating data in the real world is more costly [37]. Although we have demonstrated in the previous section that our approach is more generalizable than others in the context of robot navigation in cluttered environments, it has been done using the standalone simulator presented in [33], which is more simple in terms of dynamics, friction effects, and disturbances in general. Our choice was based on the fact that a simpler simulator tends to be more efficient in the training stage, allowing greater exploration of the environment in a shorter time. Therefore, in this section, we present a comparative analysis concerning a more realistic simulator, the CoppeliaSim, integrated with robot operating system (ROS) Noetic running in the same hardware from the previous section. The simulations rely on a Pioneer P3DX equipped with a planar LiDAR. The 2-D scans also have readings between −135° and 135° with 684 distance measurements and decimated in 57 values of minimum distances detected, as described in Section III-A.

Here, the set of trials was performed in the three environments illustrated in Fig. 6, two with dimensions 40 m × 30 m and another maze style with 24 m × 24 m. In the last section, the reward function in [25] shows to be the second most efficient; then we incorporate it in this new test in two versions, one with the TD3 and another with the SAC. Also, we have limited our proposed method to the SAC method, which presented a better performance in the previous section.

Table V gives the success rates, median of steps, and the first and third percentiles for the simulations in the CoppeliaSim using the same policies trained in the previous section. This time, due to the higher cost of running the simulator, we executed 200 trials for each table entry and increased the timeout to 4000 steps due to the size and complexity of the scenarios. As can be seen, our method also outperforms [25], both with the TD3 and the SAC, in all scenarios.

Concerning the binomial probability of the success rate for the first map [see Fig. 6(a)] the SAC-[25] presents 90% confidence interval of approximately [78%, 87%], while our method falls between [87%, 94%]. Results were even better for the second map [see Fig. 6(b)] where the 99% confidence interval is [67%, 83%] for the SAC-[25], and [88%, 94%] for our method. Finally, for the third map [see Fig. 6(c)], the 99% confidence interval is [17%, 33%] for the SAC-[25], and [67%, 83%] for our method.

Table V also shows that our approach presents the lowest expected values for the number of steps until the target, demonstrating that it is more efficient than others. At the same time, as the maps are more complex, they presented a more significant variation in the distribution. This may reflect that the proposed approach actively tries to explore the environment when faced with the local minimum traps, leading to longer paths to the goal. The other methods accommodate under this adversity since most of the mission fail because of timeout. The main conclusion at this point is that our proposed approach was capable of dealing better with the differences between the previous simulator (in which all policies have been trained) and the CoppeliaSim, at least for the used maps.

D. Sim-to-Real Experiments

Finally, to evaluate our proposed method, we have applied it to navigate a robot in the real world. The main idea is to qualitatively assess the sim-to-real transfer capacity, which are concrete instances of the generalization problem [29]. Here, it is important to highlight that our evaluation is performed in a sim-to-real zero-shot fashion since no fine-tuning adjustment is realized over our trained policy [4].

In order to keep evaluating the comparison made in the sim-to-sim analysis, we performed the same zero-shot experiment for the method SAC-[25].

We also repeated the experiment using the potential field controller (PFC) as a baseline method, a classical navigation strategy that seeks to reach goals using attractive potential fields and avoiding obstacles using repulsive fields.

The experiments involve navigating an office-like environment, with multiple rooms and heterogeneous obstacles, to some target points. The robot selected for the task was an autonomous vacuum cleaner Roomba Irobot Create 1 equipped with a Hokuyo URG-04LX-U01, and a laptop with Ubuntu 20.04 and ROS Noetic, fitted with an NVIDIA GTX 3060, 16 GB of RAM, and an Intel Core i7-11800H CPU.

Fig. 7 shows the performed experiment in a sequence of frames. Fig. 8 shows the path performed by the robot in three experiments, starting on the blue points and with the target defined to the red points. The experiments were conducted at...
the same place and with the same target, modifying only the control strategy and the start position. In addition, a map was created only for visual purposes. In the experiment in Fig. 8, the robot navigates to the goals avoiding obstacles (black points on the map) and local minima. Both goals were selected to emulate local minima issues, but the robot does not get stuck when running with the proposed method. However, the results in Fig. 8 show that the PFC method has difficulty avoiding local minima. We modified the test’s start point with the PFC to make navigation easier and avoid starting in local minima.

In addition, in the third experiment using SAC-[25], although the robot moved in the target direction, the algorithm presented difficulty in avoiding obstacles using the zero-shot transfer method, colliding when trying to reach the target. This behavior should be related to the generalization of the model and performance in a zero-shot sim-to-real transfer.

In the first experiment, the robot navigates a distance of 21.54 m in 120 s, the second run 3.75 m in 30 s and gets stuck, and in the third run 6.34 m in 33 s and finished colliding with an obstacle.

Using the proposed local navigation policy, the robot could successfully navigate toward the goals region without knowledge of the map of this indoor environment. A video illustrating the training, sim-to-sim, and sim-to-real experiments can be seen in https://youtu.be/2t_n66_uCSc.

V. CONCLUSION AND FUTURE WORK

We have proposed a method with a continuous action space and a shaping reward function in a DRL framework to have sim-to-sim and sim-to-real transfer capability in the context of local navigation for autonomous mobile robots in unknown cluttered environments. Our method improved the generalization of the policy, in the sense that it could navigate the robot throughout the workspace with higher success rates, i.e., the ability to reach the target region avoiding collisions and local minima, than other existing approaches. Our first conclusion was that using the local information to make the reward less sparse significantly improved the success rate when compared with other methods. By this reward shaping strategy, the map information gained during navigation prevented actions that did not explore new places, and the distance to the target prevented navigation away from it. The number of steps to reach the goal was also significantly smaller for our method, demonstrating its efficiency. Another conclusion was that the SAC outperforms the TD3 for almost all reward functions, in the proposed shaping and in previous studies, justifying the use of learning algorithms that adopt techniques for increasing action exploration.

In future works, we intend to study the use of asynchronous algorithms for learning by using the proposed reward function to optimize the process, including a high number of scenarios, and evaluate its influence on generalization. In addition, we want to perform experiments as an ablation study, checking the effects of noise in the sensor measures in the policy.

REFERENCES

[1] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[2] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” J. Big Data, vol. 6, no. 1, 2019, Art. no. 60.
[3] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2017, pp. 23–30.
[4] X. B. Peng, M. Andrzychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 3803–3810.
[5] K. Wang, B. Kang, J. Shao, and J. Feng, “Improving generalization in reinforcement learning with mixture regularization,” in Proc. Adv. Neural Inf. Process. Syst. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020, pp. 7968–7978.
[6] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, “A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization,” IEEE Trans. Ind. Electron., vol. 66, no. 11, pp. 8557–8566, Nov. 2019.
[7] A. M. C. Rezende et al., “Indoor localization and navigation control strategies for a mobile robot designed to inspect confined environments,” in Proc. IEEE 16th Int. Conf. Autom. Sci. Eng., 2020, pp. 1427–1433.
[8] L. Zhang, Z. Lin, J. Wang, and B. He, “Rapidly-exploring random trees multi-robot map exploration under optimization framework,” Robot. Autom. Syst., vol. 131, 2020, Art. no. 103565.
[9] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, “A generalized Voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots,” IEEE Trans. Ind. Electron., vol. 69, no. 5, pp. 4926–4937, May 2022.
[10] S. S. D. A. Nandesh, R. Raman K., G. K, and R. R, “An investigation of bug algorithms for mobile robot navigation and obstacle avoidance in two-dimensional unknown static environments,” in Proc. IEEE Int. Conf. Commun. Inf. Comput. Technol., 2021, pp. 1–6.
[11] J. Steir, M. Saad, and H. Saliah-Hassane, “An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment,” in Proc. IEEE Int. Symp. Robot. Sens. Environ., 2011, pp. 208–213.
[12] I. Carlucho, M. Paula, S. Wang, Y. Petillot, and G. G. Acosta, “Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning,” Robot. Auton. Syst., vol. 107, pp. 71–86, 2018.
[13] H. Kayakoku, M. S. Guzel, E. Bostanci, I. T. Medeni, and D. Mishra, “A novel behavioral strategy for RoboCode platform based on deep q-learning,” Complexity, vol. 2021, pp. 1–14, 2021.
[14] X. Gao et al., “RL-AKF: An adaptive Kalman filter navigation algorithm based on reinforcement learning for ground vehicles,” Remote Sens., vol. 12, no. 11, 2020, Art. no. 1704.

[15] C. You, J. Lu, D. Fivel, and P. Tsiotras, “Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning,” Robot. Auton. Syst., vol. 114, pp. 1–18, 2019.

[16] K. Zhu and T. Zhang, “Deep reinforcement learning based mobile robot navigation: A review,” Tsinghua Sci. Technol., vol. 26, no. 5, pp. 674–691, Oct. 2021.

[17] H. Jiang, H. Wang, W.-Y. Yau, and K.-W. Wan, “A brief survey: Deep reinforcement learning in mobile robot navigation,” in Proc. IEEE 15th Conf. Ind. Electron. Appl., 2020, pp. 592–597.

[18] X. Ruan, D. Ren, X. Zhu, and J. Huang, “Mobile robot navigation based on deep reinforcement learning,” in Proc. IEEE Chin. Conf. Decis. Control, 2019, pp. 6174–6178.

[19] G. Chen et al., “Deep reinforcement learning of map-based obstacle avoidance for mobile robot navigation,” SN Comput. Sci., vol. 2, no. 6, 2021, Art. no. 417.

[20] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, “Robot navigation in crowded environments using deep reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 5671–5677.

[21] R. B. Grando et al., “Deep reinforcement learning for mapless navigation of a hybrid aerial underwater vehicle with medium transition,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 1088–1094.

[22] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, “Learn to navigate autonomously through deep reinforcement learning,” IEEE Trans. Ind. Electron., vol. 69, no. 5, pp. 5342–5352, May 2022.

[23] W. Zhu and M. Hayashibe, “A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation,” IEEE Trans. Ind. Electron., vol. 70, no. 5, pp. 4962–4971, May 2023.

[24] R. Cimurus, I. H. Suh, and J. H. Lee, “Goal-driven autonomous exploration through deep reinforcement learning,” IEEE Robot. Automat. Lett., vol. 7, no. 2, pp. 730–737, Apr. 2022.

[25] J. Hu, H. Niu, I. Carrasco, B. Lennox, and F. Arvin, “Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning,” IEEE Trans. Vect. Technol., vol. 69, no. 12, pp. 14413–14423, Dec. 2020.

[26] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[27] Y. Hu et al., “Learning to utilize shaping rewards: A new approach of reward shaping,” in Proc. 34th Int. Conf. Neural Inf. Process. Syst., 2020, Art. no. 1336.

[28] A. A. Shahid, D. Piga, F. Braghin, and L. Roveda, “Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning,” Auton. Robots, vol. 46, no. 3, pp. 483–498, Mar. 2022.

[29] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktaschel, “A Survey of Zero-Shot Generalisation in Deep Reinforcement Learning,” J. Artif. Int. Res., vol. 76, Feb. 2023, doi: 10.1613/jair.1.14741.

[30] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, “Quantifying generalization in reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 1282–1289.

[31] S. Witty et al., “Measuring and characterizing generalization in deep reinforcement learning,” Appl. AI Lett., vol. 2, no. 4, 2021, Art. no. e45.

[32] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2017, pp. 31–36.

[33] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, “Zero-Shot Generalisation in Deep Reinforcement Learning,” J. Artif. Int. Res., vol. 2, no. 6, 2023, doi: 10.1613/jair.1.14741.

[34] J. Choi et al., “Fast adaptation of deep reinforcement learning for real autonomous mobile robot navigation in indoor environments,” 2020, arXiv:2005.13857.

[35] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in Proc. Int. Conf. Mach. Learn., ser. Mach. Learn. J. Res. Dy, and A. Krause, Eds., 2018, pp. 1578–1597.

[36] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in Proc. 4th Int. Conf. Learn. Representations, May 2–4, 2016. [Online]. Available: https://dl.acm.org/doi/10.1613/jair.1.14741.

[37] S. James et al., “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 12627–12637.