Laser-Based Reactive Navigation for Multirotor Aerial Robots using Deep Reinforcement Learning

Carlos Sampedro[1], Hriday Bavle[1], Alejandro Rodriguez-Ramos[1], Paloma de la Puente[1] and Pascual Campoy[1]

Abstract— Navigation in unknown indoor environments with fast collision avoidance capabilities is an ongoing research topic. Traditional motion planning algorithms rely on precise maps of the environment, where re-adapting a generated path can be highly demanding in terms of computational cost. In this paper, we present a fast reactive navigation algorithm using Deep Reinforcement Learning applied to multirotor aerial robots. Taking as input the 2D-laser range measurements and the relative position of the aerial robot with respect to the desired goal, the proposed algorithm is successfully trained in a Gazebo-based simulation scenario by adopting an artificial potential field formulation. A thorough evaluation of the trained agent has been carried out both in simulated and real indoor scenarios, showing the appropriate reactive navigation behavior of the agent in the presence of static and dynamic obstacles.

I. INTRODUCTION

Autonomous multirotor navigation through unknown, cluttered indoor scenarios is a complex task which traditionally involves an efficient combination of different robotic modules such as perception, path planning, and state estimation. The complexity of this task increases when moving objects are placed in the scenario. Navigating in dynamic scenarios requires computationally efficient algorithms capable of re-adapting the path in real-time when a new obstacle appears in the field of view of the robot. Traditional motion planning algorithms exhibit some limitations in terms of reactive behavior owing to the computational effort required to re-plan a navigation path.

Moreover, another key aspect in path planning is the representation of the map (e.g. grid-based, polygon-based, etc). Grid-based methods can suffer from imprecise obstacle representations or big memory demands especially when the scenario is substantially large. On the other hand, geometric representations of the map can be more efficient in terms of memory consumption while suffering from a high computational cost when representing complex shaped obstacles. In addition, several traditional path planning algorithms may require a fine-tuning stage of their parameters in order to adapt the algorithm to a previously unseen scenario.

In this work, the research effort is focused on the development of an efficient reactive navigation algorithm able to deal with the aforementioned limitations, with special attention to unknown scenarios with dynamic obstacles. In this direction, the main contribution of this work is the development and

experimental validation of a mapless reactive navigation algorithm applied to multirotor aerial robots. For this purpose, we adopt a recent *actor-critic* deep reinforcement learning algorithm named Deep Deterministic Policy Gradients (DDPG) [1] which is successfully trained in simulation and directly transferred to a real aerial robotic platform. One of the key properties of the proposed system is the agent's state definition which integrates the relative distance to a predefined goal together with the linear velocities of the aerial robot in the x and y directions and the laser scan data organized in circular sectors. This representation provides the aerial robot with the ability to deal with obstacles of different shape and size. The second main contribution of this work is related to the training process of the agent. To this aim, we adopt an Artificial Potential Field (APF) formulation in order to design the reward function, which has proven to reduce by a large margin the training process of the agent as compared to similar state-of-the-art approaches. Moreover, the computational cost of the algorithm is reduced to a feed-forward pass through the *actor* network, which leads to fast avoidance maneuvers. The experiments conducted in cluttered and dynamic scenarios demonstrate the appropriate navigation capabilities of the proposed approach, especially in the presence of obstacles that exhibit sudden movements.

The remainder of this document is organized as follows: Section II provides an overview of the related work; Section III introduces the proposed reactive navigation approach. The experiments conducted and the results obtained in simulated and real scenarios are presented in Section IV before we discuss the results in Section V and highlight the conclusions and future research works in Section VI.

II. RELATED WORK

Autonomous navigation in scenarios with unexpected obstacles is a key challenge for advanced mobile systems [2]. This problem has traditionally been addressed by integrating localization, global planning, and local planning or reactive control. We focus on the latter and refer the reader to [3] for a global planning literature review.

The simplest methods for local motion planning are based on the well-known Bug algorithm [4]. Other classic techniques are related to artificial potential field concepts, applying the idea that the goal generates attractive forces while the obstacles create repulsive forces for the robot [5]. Elastic band methods [6] consider that the path provided by a global planner is subject to deformations when objects are encountered along the way. Alternatively, in narrow cluttered scenarios the Nearness Diagram representation [7]

---

[1]All authors are with the Computer Vision and Aerial Robotics group, Centre for Automation and Robotics, Universidad Politécnica de Madrid (UPM-CSIC), Calle José Gutiérrez Abascal 2, Madrid (Spain). `carlos.sampedro@upm.es`, ORCID: https://orcid.org/0000-0003-2414-2284

for different divide-and-conquer strategies has provided very good results. Recent works have achieved faster and less oscillatory motion for ground robots [8].

In aerial robotics, several approaches create local maps and find free-space corridors online [9], [10], but in the case of platforms with limited resources building complex models of the environment for reactive control may be unaffordable. A novel contribution by Lopez and How [11] presented how using raw point clouds as representation resulted in impressive aggressive flights of quadrotor vehicles.

Learning abilities for reactive navigation, which is the main topic of this paper, is showing great potential too. One of the first ideas was about learning to imitate reactive human control of Micro Aerial Vehicles (MAVs) from visual features, based on heading commands provided by an expert pilot [12]. Gandhi *et al.* [13] argued that requiring expert data for learning is a significant limitation and contributed a dataset of crashes in real world scenarios for vision-based self-supervised Deep Learning (DL) of heading control. Crashes in real world scenarios are undoubtedly useful as negative examples, but sometimes it is not acceptable to damage the robot in the training phase.

Deep Reinforcement Learning (DRL) has been successfully employed to solve difficult navigation problems by means of trial and error experiences. Going beyond heading commands, Xie *et al.* [14] present a DRL strategy to explore vision-based obstacle avoidance policies. The D3QN architecture was trained in simulation to infer discretized linear and angular velocities from raw images. Zhu *et al.* [15] pursued reactive navigation towards a visual goal, achieving fast convergence and good generalization results.

Zhang *et al.* [16] solve navigation problems in simple maze-like scenarios as a sequence of related reinforcement learning tasks with four discrete commands as possible outputs. Mirowski *et al.* [17] deal with challenging auxiliary functions such as depth prediction and loop closure detection. Kahn *et al.* [18] proposed a method to combine model-free and model-based DRL for sample-efficient visual self-supervised learning of continuous actions in the real world. Uncertainty was considered so as to improve safety.

Regarding the use of laser sensors for learning navigation strategies, Pfeiffer *et al.* [19] developed an end-to-end DL approach for goal-based local navigation of mobile robots. The proposed model was trained with data from a global planner and good reactions to sudden changes were shown. The authors highlight that for fully operational navigation in complex environments their motion planner should be integrated with a global planner providing intermediate targets.

The work by Tai *et al.* [20] is the most closely related to ours. By means of an Asynchronous DDPG, DRL is applied to learn continuous control actions for a non-holonomic differential drive robot from 10 sparse laser measurements and the relative position of the goal. In contrast, our approach introduces a method based on potential fields to guide and improve the learning process and provides enhanced robustness against outliers and flawed measurements.

III. REACTIVE NAVIGATION APPROACH

A. Background

The principal components in the reinforcement learning paradigm are the *agent* and the *environment*. In this paper, we model the problem of reactive navigation by means of a Markov Decision Process (MDP) in which at each time step t, the agent being at state s_t executes an action a_t which causes its transition to a next state s_{t+1} and receives a reward r_t from the environment. The aim of the agent is to learn the appropriate *policy* in order to maximize the expected return $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ given the reward function $r(s_i, a_i)$ and a discount factor $\gamma \in [0, 1]$. In order to evaluate the policy, the action-value function (1) is usually utilized, which computes the expected return starting from state s_t, taking action a_t, and then following policy π afterwards.

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R_t | s_t, a_t] \qquad (1)$$

where $Q^\pi(s_t, a_t)$ is the action-value function.

In contrast to traditional reinforcement learning techniques, deep reinforcement learning algorithms include neural networks in order to estimate the action-value function [1], [21], [22] and learn the policy to be executed by the agent [1], [21]. Concretely, the DDPG was conceived as a model-free, off-policy algorithm based on an actor-critic architecture, where the actor network is in charge of learning the policy that directly maps a state to a continuous action for the agent, and the critic network is used as a non-linear function approximator for estimating the action-value function.

B. Reactive Navigation Architecture

The reactive navigation system proposed in this work is based on the architecture presented in Fig. 1. The proposed architecture has been designed in order to integrate a reinforcement learning agent (e.g. DDPG) with an aerial robotics simulator (e.g. RotorS Gazebo [23]), using the Robot Operating System (ROS) [24] for communication between the different components. In the next paragraphs, we explain the configuration of the selected agent (*Agent2*), obtained after the evaluations performed in Sections IV-B and IV-C.

As can be seen in Fig. 1, the environment receives the raw laser range findings coming from a 2D laser range sensor, and the position of the aerial robot and the goal. Based on the previous information, the environment is in charge of computing the 14-dimensional state (observation) and the reward to be sent to the agent. The first four components of the state consist of the x and y relative position of the aerial robot with respect to the desired goal and the x and y linear velocities of the former. The rest 10 components of the state are based on laser data, and are calculated using the following sequence of computations:

1) Preprocessing: This step consists in saturating the laser range findings up to a *maximum virtual range* in order to reduce the influence of the roll and pitch angles of the robot (see Fig. 2b). This computation has proven to be crucial in the learning process of the agent.
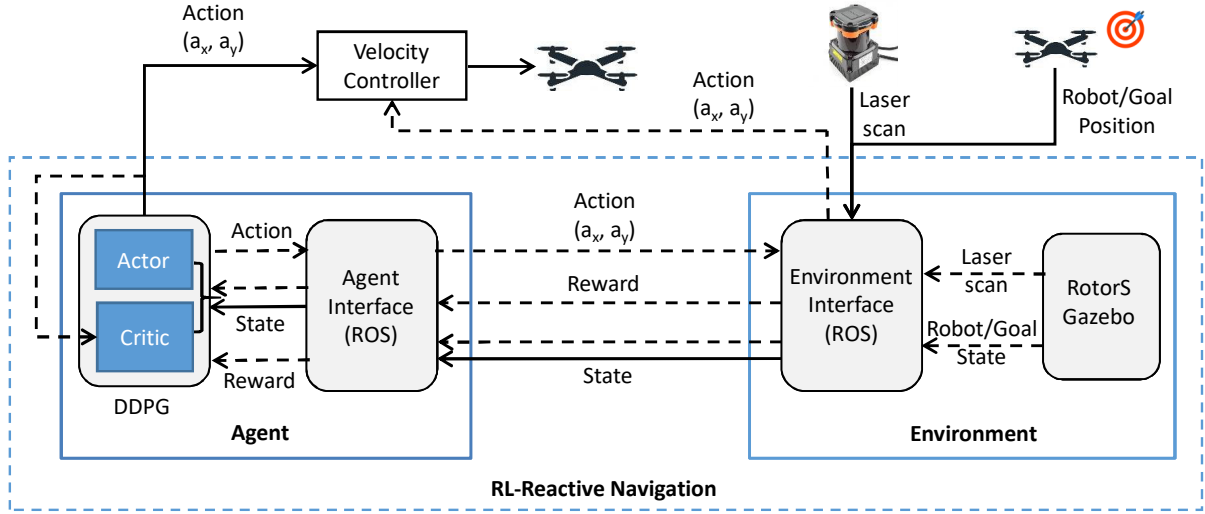
Fig. 1: Architecture of the proposed RL-Reactive Navigation system. The *enviroment* receives as input raw laser range findings and the aerial robot position from a state estimator module. The *agent* which implements the DDPG algorithm, receives an observation and a reward from the environment and computes the action (linear velocity in the x and y directions) to be commanded to a velocity controller. The dotted lines represent interactions between the components in training mode (simulation), while the continuous lines depict the interactions in test mode (e.g. real flights).

2) State term based on laser information: Based on the preprocessed laser range findings from the previous step, we divide the whole laser scan into 10 evenly-spaced circular sectors (see blue lines in Figures 2b and 2c). After that, the average of the laser range findings within each sector is computed (see black lines in Fig. 2c). This method provides a more stable representation of the state, reducing its vulnerability to noisy isolated ranges or possible failures in the sensor.

Regarding the agent's side of the proposed RL architecture, the DDPG algorithm with *batch normalization* is used in order to learn the appropriate policy for generating linear velocity commands to the velocity controller proposed in [25]. In this work, the *actor* and *critic* neural networks of the DDPG consist of feed-forward neural networks with 3 hidden layers of 400 units each. The activation function implemented in each hidden unit is the Rectified Linear Unit (ReLU). The output layer of the actor network is composed of two units with a $\tanh$ activation function in order to provide a continuous linear velocity command within the range $[-1, 1]$ m/s, whereas the output layer of the critic network is composed of a single linear unit used to predict the action-value function.

C. Reward function design

In order to achieve the two main objectives of our navigation system, i.e. reaching a predefined goal while providing an obstacle avoidance behavior, the reward function has been designed taking into account an APF formulation. Thus, the reward function is mainly influenced by two terms which define the attractive and repulsive potential fields.

In order to calculate the attractive potential field, the system relies on a state estimation algorithm which can provide the position of the robot in the world frame of reference. Based on this information and using (2), the attractive potential field is computed based on the Euclidean distance between the position of the aerial robot ($\mathbf{t_r}$) and the goal ($\mathbf{t_g}$) at the current time step:

$$U_{att} = \alpha \rho_{goal}(\mathbf{t_r}) \qquad (2)$$

where $\rho_{goal}(\mathbf{t_r}) = \|\mathbf{t_r} - \mathbf{t_g}\|_2$ and α is a positive gain and has been empirically obtained to be 100.

Regarding the calculation of the repulsive potential field, we want that each obstacle contributes with only one component to the repulsive field term (3). To this aim, we first identify the obstacles in the surroundings of the aerial robot up to the *maximum virtual range* defined previously. For this, an artificial image is generated (see Fig. 2d) by projecting the laser ranges into an image of predefined resolution, where the robot is always in the image center. Using this virtual image, we use computer vision techniques in order to find contours in the image which will represent obstacles in the surroundings of the robot. Once the obstacles (contours) have been identified, the distance from the center of the image (aerial robot) to the closest point in each contour is calculated. The vectors formed by the minimum distance to each obstacle in the image (see red arrows in Fig. 2d) represent the laser range findings, within the corresponding circular sector, used to compute the repulsive potential field:

$$U_{rep} = \beta \sum_{i=1}^{N} \left(\frac{1}{k + l_i} - \frac{1}{k + l_{max}} \right) \qquad (3)$$

where β is a positive gain computed using (4), N is the number of detected obstacles at the current time step, k is a constant used to limit the repulsive field and has been empirically set to 0.04, l_i is the minimum laser range towards the obstacle i within the corresponding circular sector, and l_{max} is the maximum virtual range value. The last term in
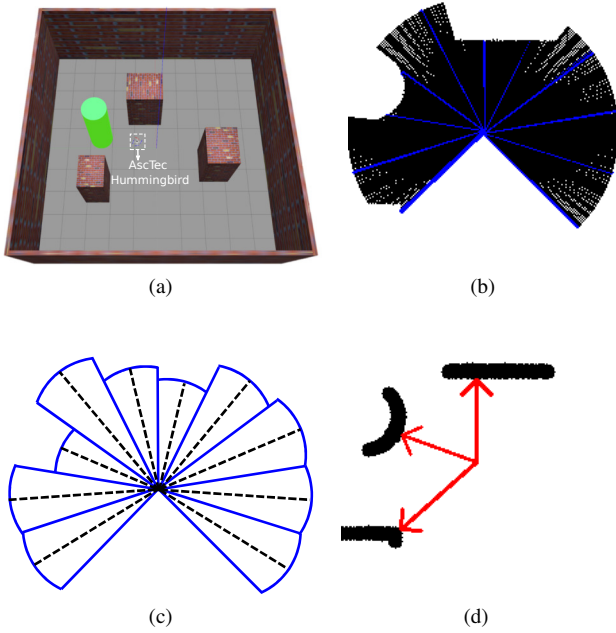
Fig. 2: State and repulsive field computation steps (best seen in color). (a) Example of a Gazebo test scenario for illustrating the process. (b) Saturated laser ranges up to a *max* virtual range. (c) Circular sectors (blue color) and average of the laser ranges within each sector (black lines). (d) Virtual image generated in order to identify near obstacles, quantify them, and assign them to the corresponding circular sector.

the equation $1/(k+l_{max})$ is used to soften the repulsive field in the instant when new obstacles appear in the field of view of the robot.

$$\beta = \begin{cases} \delta & \text{if } \rho_{goal}(\mathbf{t_r}) > d_{infl} \\ \frac{\delta}{\exp[4(d_{infl}-\rho_{goal})]} & \text{if } \rho_{goal}(\mathbf{t_r}) \le d_{infl} \end{cases} \quad (4)$$

where δ is a positive gain and has been empirically found being $\delta = 2$, and $d_{infl} = 0.75l_{max}$ is the distance of influence from which the repulsive field starts decreasing. This term is used for reducing the influence of the repulsive field when the aerial robot is close to the goal.

Once the attractive and repulsive potential fields are calculated, we compute the reward term using (6) by considering first a *shaping* function (5) which provides information to the agent about its instantaneous progress while speeding up the learning process [26]. The reward term is finally computed taking into account the evolution of the shaping function in two consecutive time steps:

$$shaping_t = -U_{att} - U_{rep} \quad (5)$$

$$r_t = shaping_t - shaping_{t-1} \quad (6)$$

where r_t is the reward obtained at time step t, and U_{att} and U_{rep} stand for the attractive and repulsive potential fields respectively.

IV. EXPERIMENTS AND RESULTS

A video demonstration of the reported experiments an results is available in: `https://vimeo.com/259398134`.

A. Experimental Setup

In order to train and test the proposed RL-Reactive Navigation system in simulation and real flight scenarios, it has been integrated within the Aerostack framework [27] using ROS as the communication middleware. Aerostack is an open source framework for aerial robotics and is used in this work in order to provide additional software components (e.g. state estimator, velocity controller, etc) which are necessary for simulation and real flight experiments. In all the experiments, the frequency of the agent has been set to 20 Hz.

Simulation experiments have been conducted using the RotorS Gazebo simulator, in which an AscTec Hummingbird quadrotor has been used as the aerial robotic platform. For training purposes, ChainerRL[1] library has been utilized, which is built on top of the Python-based deep learning framework Chainer[2]. Using these libraries, all the models have been trained on a GPU Nvidia GeForce GTX 970.

Real flight experiments have been performed in order to evaluate the capabilities of the RL-Reactive Navigation system in unknown indoor scenarios with static and dynamic obstacles. The aerial robotic platform utilized for these experiments is the DJI Matrice 100 quadrotor, which is equipped with a DJI Manifold (ARM-architecture) computer and a Hokuyo laser rangefinder UTM-30LX (see Fig. 7a). In all the experiments the only information that the robot receives at each time step is the laser scan measurements, limited to the maximum virtual range of 2 m, and the current position of the aerial robot provided by a state estimator. Then, the velocities of the aerial robot in the x and y coordinates are computed by differentiating the position of the former.

B. Training Methodology

The simulation scenario utilized for training the agent is shown in Fig. 3 and consists of an 8 m × 8 m square area with 3 main obstacles inside. The simplistic configuration of the proposed scenario has been purposely designed in order to evaluate the generalization capabilities of the agent in further test experiments.

In order to train the aerial robotic agent, the simulation has been configured in an episodic setting, where each episode is composed of a sequence of steps. At the beginning of each episode, the aerial robot and the goal are placed in a random position of the scenario, where the altitude of the aerial robot always remains constant at 1.2 m.

At each time step during the training process, the agent executes an action with added noise according to an Ornstein-Uhlenbeck distribution and receives a reward given by (6). This process is repeated until the maximum number of steps (400 steps/episode) is reached or when the aerial robot reaches a terminal state. A terminal state occurs when the aerial robot collides with an obstacle or when it reaches the goal. In the first situation, a negative reward of -100 is given to the agent, while in the second situation, which represents the main objective, a positive reward of $+10$ is provided.
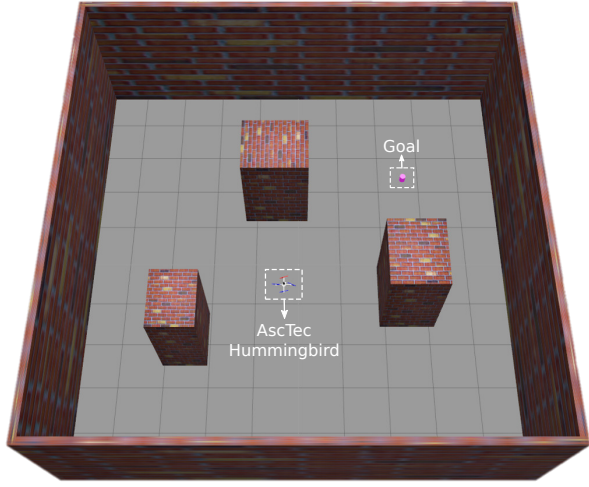
[1]https://github.com/chainer/chainerrl
[2]https://github.com/chainer/chainer

Fig. 3: Gazebo-based simulation scenario used for training the RL-Reactive Navigation system. Pink cylinder represents the goal.
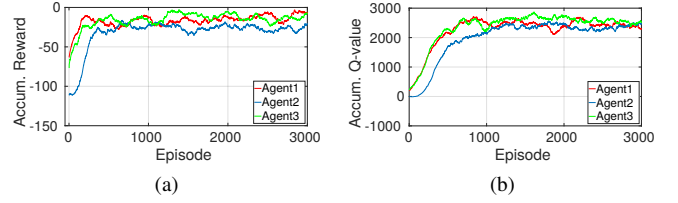

Fig. 4: Training curves (best seen in color). (a) Accumulated reward per episode. (b) Accumulated action-value function (Q) per episode. Each curve is depicted considering a moving average of 200 episodes.

During the training process of the actor and the critic neural networks (see Fig. 4), Adam optimizer [28] has been utilized with a base learning rate of 10^{-4} and a minibatch size of 64. The rest of the hyperparameters are the same as those presented in the DDPG original work. Using this configuration, and the reward design presented in Section III-C, the agent is able to learn an appropriate reactive navigation policy in 600 episodes, taking approximately 147k simulation steps (2 hours). In order to select the most appropriate agent, in this work we perform a thorough comparison between three different configurations of the DDPG agent varying the number of neurons in each hidden layer (actor and critic) and the number of circular sectors that make up the part of the state corresponding to laser measurements. The first configuration studied, referred to as *Agent1*, has been trained with the actor and critic neural networks being composed of 3 hidden layers of 400 units each. The state considered in Agent1 is a 12-dimensional state with 8 circular sectors as the laser term of the state. The second configuration, referred to as *Agent2*, has been trained using the same actor and critic configuration as *Agent1* and a state composed of 10 laser circular sectors. The final configuration considered, referred to as *Agent3*, has been conceived as a configuration for analyzing the influence of the number of hidden units in the performance of the agent. Thus, *Agent3* has been trained with an actor and critic of 300 hidden units in each hidden layer. In this case, the state has the same configuration as *Agent2*. Table I summarizes the three agent's configurations analyzed.

TABLE I: Configuration of the different agents analyzed.

Agent	Hidden units (actor)	Hidden units (critic)	state dim.
1	3 × 400	3 × 400	12
2	3 × 400	3 × 400	14
3	3 × 300	3 × 300	14

C. Simulation Experiments

The main objective of the simulation experiments presented in this section is to select the most appropriate agent configuration for the proposed RL-Reactive Navigation system while measuring its generalization capability. For this purpose, a benchmark of three Gazebo-based scenarios has been created with different levels of complexity. The first two scenarios contain static obstacles (see Figures 5a and 5b), while the last one is made up of dynamic obstacles (see Fig. 5c). The last designed scenario is of special interest owing to the different type of movement that the obstacles can exhibit (one cylindrical obstacle moves according to a sinusoidal trajectory, while a quadrilateral obstacle performs sudden movements in front of the aerial robot). It has to be noted that, in all the simulation scenarios, the obstacles are significantly smaller than the ones used in the training scenario (see Figures 3 and 5) with the aim of evaluating the generalization capabilities of the proposed system.

In order to obtain a complete evaluation of the different agents presented in Table I, we perform 100 tests in each of the three simulation scenarios. In each test, the aerial robot is initialized at the bottom part of the scenario while the goal is located at the upper part at 5 m distance in the y axis. Both of them are randomly placed within a section of 2 m in the x axis. In all the tests the velocity of the aerial robot is limited within the range $[-0.6, 0.6]$ m/s.

The results obtained during the execution of the experiment are presented in Table II. In this table, four main metrics are shown: PF stands for the performance or percentage of accuracy, which measures the number of tests in which the agent reached the goal successfully. We consider the goal reached when the distance between the agent and the goal is less than 0.3 m. On the other hand, the test is considered failed when the agent collides with any of the obstacles. This situation occurs when any of the laser ranges provides a measurement less than 0.3 m. The variable PL represents the average path length measured in meters, TG is the average time to reach the goal, and finally, MD provides a measurement of the mean minimum distance to the obstacles during the corresponding test.

Based on the results presented in Table II, *Agent2* configuration is adopted in order to conduct further real flight experiments, and its selection will be further discussed in Section V.
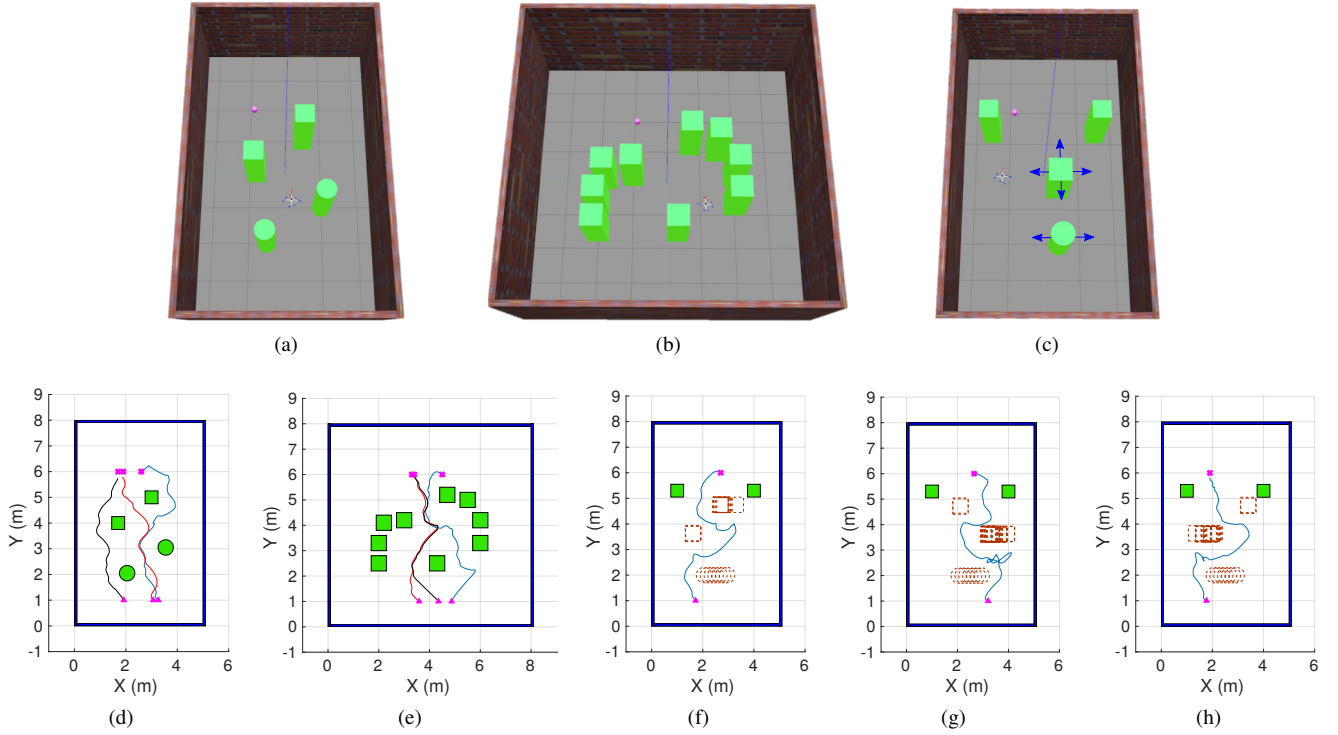
Fig. 5: Simulation experiments. (a), (b), (c) Gazebo simulation scenarios created for evaluating the agents. (d) Three illustrative trajectories generated by *Agent2* in scenario 1. (e) Three representative trajectories generated by *Agent2* in scenario 2. (f), (g), (h) Three representative trajectories generated by *Agent2* in scenario 3. The dotted line shapes represent moving obstacles, while shapes with solid lines indicate static obstacles. The pink triangle represents the takeoff point of the agent and the pink cross indicates the location of the goal.

TABLE II: Evaluation of the different agents in the scenarios of Fig. 5. S stands for scenario and A represents the agent.

S	A	$PF(\%)$	PL (m)	TG (s)	MD (m)
1	1	86	5.88 ± 0.40	11.57 ± 0.85	0.7 ± 0.03
	2	98	5.87 ± 0.24	11.99 ± 1.44	0.71 ± 0.06
	3	89	5.81 ± 0.47	10.92 ± 0.71	0.71 ± 0.05
2	1	78	6.05 ± 0.29	11.31 ± 0.83	0.81 ± 0.03
	2	100	5.99 ± 0.21	12.28 ± 1.06	0.75 ± 0.03
	3	67	5.76 ± 0.23	10.88 ± 0.73	0.78 ± 0.04
3	1	75	6.28 ± 0.47	14.1 ± 2.02	0.73 ± 0.07
	2	81	6.88 ± 0.81	16.3 ± 3.49	0.73 ± 0.06
	3	51	5.84 ± 0.16	11.82 ± 0.6	0.70 ± 0.05

D. Real Flight Experiments

Three experiments of increasing difficulty have been considered. The first experiment has been designed in order to evaluate the long-term navigation capabilities of the proposed RL-Reactive Navigation system. For this purpose, the objective of the aerial robot is to navigate through an *a priori* unknown indoor scenario in order to reach a sequence of goals which are defined before takeoff. The scenario consists of an 11 m × 10 m area and is composed of static obstacles of varying shape (cylindrical and quadrilateral) and size (see Fig. 6). Fig. 6b shows the trajectory generated by the aerial robot while reaching the sequence of goals.

The same scenario is used in the second experiment, which additionally introduces a mobile obstacle that is handled by a human operator from within. In this experiment, the moving obstacle tries to block the trajectory of the aerial robot during some period of time in two sections of the scenario. The main objective of this experiment is to evaluate the long-term and reactive navigation capabilities of the proposed system as a whole. Fig. 6c depicts the trajectory generated by the aerial robot while reaching each of the commanded goals. In this figure, sections A and B of the trajectory are highlighted for further discussion, where the aerial robot suddenly encounters the moving obstacle and performs reactive maneuvers in order to avoid it.

Finally, the third experiment has been designed for testing the reactive behavior of the proposed approach in the presence of a fast-moving obstacle. For this purpose, a 4 m × 3 m area is utilized in which OptiTrack motion capture system is used for capturing the state of the aerial robot and the moving obstacle (see Fig. 7). For a better understanding of the last two experiments involving moving obstacles, we refer the reader to the video demonstration.

V. DISCUSSION

The experiments presented in simulation and real flight scenarios reveal the outstanding navigation capabilities of the mapless RL-Reactive Navigation system proposed in this work. The proposed agent has been only trained in a simple simulation scenario in order to demonstrate its generalization
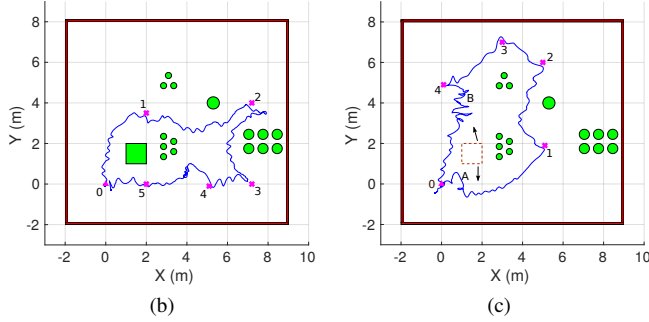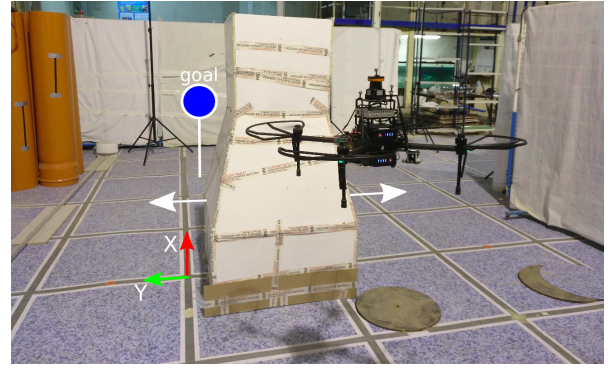
(a)

(b)

(c)
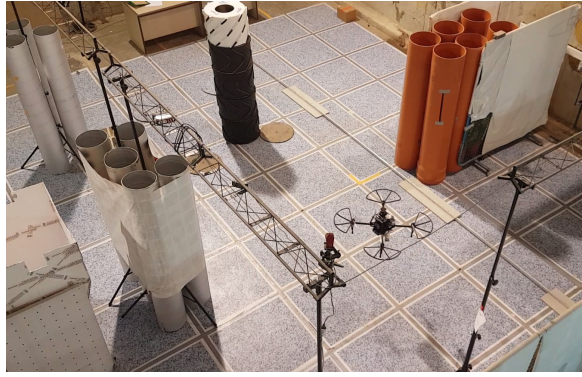
Fig. 6: Real Flight experiments designed for evaluating the proposed RL-Reactive Navigation system in long missions with static and dynamic obstacles. (a) Indoor scenario created for the evaluation. (b) Trajectory generated by the aerial robot in the first real flight experiment (static obstacles). (c) Trajectory generated by the aerial robot in the second real flight experiment (static and dynamic obstacles).
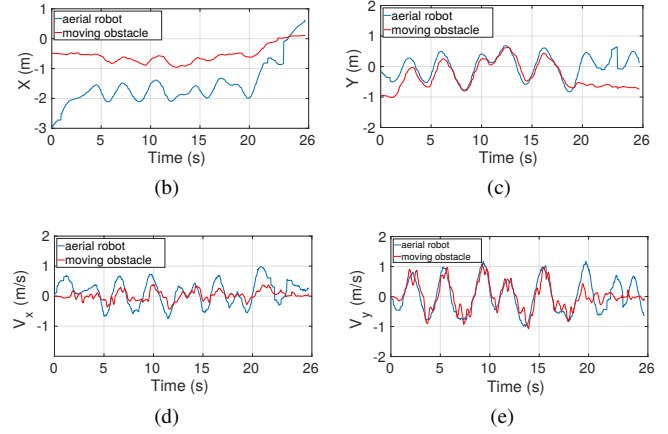

(a)

(b)

(c)

(d)

(e)

Fig. 7: Real flight experiment designed for evaluating the reactive behavior of the proposed RL-Reactive Navigation system (best seen in color). (a) Indoor scenario created for the evaluation. (b), (c) Positions of the aerial robot and the moving obstacle. (d), (e) Linear velocities reached by the aerial robot and the moving obstacle during the experiment.

capabilities. The reward function design proposed in this work for training the agent is based on an artificial potential field formulation, which introduces attractive and repulsive field terms. Unlike similar approaches of the state-of-the-art [20] in which the agent is rewarded negatively only in the moment of collision, the inclusion of the repulsive field term allows providing feedback regarding the proximity of obstacles in each time step. This fact considerably reduces the training time of the agent as shown in Fig. 4.

The simulation results presented in Table II show the considerable influence of the number of parameters of the model in the performance of the agent. A reduction of 10^6 in the number of weights of the actor and critic neural networks (*Agent2* to *Agent3*) led to learning a policy with limited navigation and generalization capabilities. The final agent configuration adopted in this work (*Agent2*) is based on an actor network with 400 units in each of the 3 hidden layers. Furthermore, the inclusion of more laser information in the state of the agent (10 laser circular sectors as compared to the 8 circular sectors of *Agent1*) proved to be beneficial. The results regarding the time to reach the goal (TG) are of special interest. In all the simulation scenarios, *Agent2* obtained higher TG than the other two agents owing to the complex behaviors learned by this agent. One of these behaviors has been confirmed in a visual manner and consists

in performing a hard braking when an obstacle appears suddenly in front of the aerial robot. These learned maneuvers lead to a safer behavior which is translated into a higher performance, sacrificing the time to reach the goal.

Regarding the results obtained in real flight experiments, we would like to emphasize the versatility of the proposed RL-Reactive Navigation system, which has been only trained in a Gazebo simulation scenario using a quadrotor (AscTec Hummingbird) with different dynamics as compared to the one utilized in real flights (DJI Matrice 100). The results presented in Fig. 6, demonstrate the appropriate reactive maneuvers generated by the proposed system in the presence of a dynamic obstacle which suddenly appears in front of the aerial robot in the sections A and B of its trajectory. These reactive maneuvers are further demonstrated in the results presented in Fig. 7, where the aerial robot is able to maintain a safety distance in the x direction from the moving obstacle (see Fig. 7b), which performs aggressive maneuvers (> 1 m/s) trying to block the trajectory of the aerial robot towards the goal. The fast response of the aerial robot is evidenced in Figures 7c and 7e, where the plots are almost coincident with a little delay in time revealing the avoidance maneuver of the aerial robot. It should be remarked that,

even though the agent runs at 20 Hz, the inference time of the actor network has been measured in 8 ms on average.

Finally, it should be noted that some limitations that might occur owing to the use of an APF formulation such as dead ends in U-shaped obstacles can be easily mitigated by integrating the proposed RL-Reactive Navigation system with a robust global planner. Despite the mentioned limitation, the results presented throughout this paper demonstrate that the proposed approach can be utilized as an effective long-term and fast-reactive navigation system in scenarios with static and dynamic obstacles. Furthermore, the versatility of the proposed approach allows its operation in previously unseen scenarios independently of the obstacle configurations in terms of number, shape, and size.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a novel reactive navigation system for multirotor aerial robots based on Deep Reinforcement Learning is proposed. Taking as input only laser information and the position of the aerial robot and the goal, the proposed agent is successfully trained in a Gazebo simulation scenario. Furthermore, by using a potential field formulation in the reward function the training process of the agent is accelerated by a large margin as compared to similar state-of-the-art approaches. An extensive evaluation of the proposed RL-Reactive Navigation system has been conducted in simulation and real flight experiments, demonstrating outstanding long-term and reactive navigation capabilities, especially in the presence of obstacles that execute sudden movements.

Future work aims towards the extension of the proposed system with memory-based functionalities (e.g. Long-Short Term Memory) in order to provide long-term navigation capabilities in maze-like scenarios. Furthermore, the integration of the proposed system with a global planner will be considered in order to leverage the benefits of both systems.

ACKNOWLEDGMENT

REFERENCES

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[2] V. Kumar and N. Michael, *Opportunities and Challenges with Autonomous Micro Aerial Vehicles*. Springer International Publishing, 2017, pp. 41–58.

[3] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[4] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987.

[5] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp. 1398–1404 vol.2.

[6] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 802–807 vol.2.

[7] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, Feb 2004.

[8] M. Mujahed, D. Fischer, and B. Mertsching, "Tangential gap flow (TGF) navigation," *Robotics and Autonomous Systems*, vol. 84, no. C, pp. 15–30, Oct. 2016.

[9] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1484–1491.

[10] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1476–1483.

[11] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5759–5765.

[12] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[13] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3948–3955.

[14] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *arXiv preprint arXiv:1706.09829, RSS workshop New Frontiers for Deep Learning in Robotics*, 2017.

[15] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3357–3364.

[16] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2371–2378.

[17] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[18] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," *arXiv preprint arXiv:1709.10489*, 2017.

[19] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1527–1533.

[20] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *arXiv preprint arXiv:1703.00420*, 2017.

[21] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine learning*, vol. 84, no. 1-2, pp. 137–169, 2011.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors a modular gazebo mav simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[24] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."

[25] J. Pestana, I. Mellado-Bataller, J. L. Sanchez-Lopez, C. Fu, I. F. Mondragón, and P. Campoy, "A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 387–400, 2014.

[26] M. Dorigo and M. Colombetti, *Robot shaping: an experiment in behavior engineering*. MIT press, 1998.

[27] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. S. Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 683–709, 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.