

Depth-CUPRL: Depth-Imaged Contrastive Unsupervised Prioritized Representations in Reinforcement Learning for Mapless Navigation of Unmanned Aerial Vehicles

Junior C. de Jesus¹, Victor A. Kich², Alisson H. Kolling²,
Ricardo B. Grando³, Rodrigo S. Guerra², Paulo L. J. Drews-Jr¹

Abstract—Reinforcement Learning (RL) has presented an impressive performance in video games through raw pixel imaging and continuous control tasks. However, RL performs poorly with high-dimensional observations such as raw pixel images. It is generally accepted that physical state-based RL policies such as laser sensor measurements give a more sample-efficient result than learning by pixels. This work presents a new approach that extracts information from a depth map estimation to teach an RL agent to perform the mapless navigation of Unmanned Aerial Vehicle (UAV). We propose the Depth-Imaged Contrastive Unsupervised Prioritized Representations in Reinforcement Learning (Depth-CUPRL) that estimates the depth of images with a prioritized replay memory. We used a combination of RL and Contrastive Learning to lead with the problem of RL based on images. From the analysis of the results with Unmanned Aerial Vehicles (UAVs), it is possible to conclude that our Depth-CUPRL approach is effective for the decision-making and outperforms state-of-the-art pixel-based approaches in the mapless navigation capability.

SUPPLEMENTARY MATERIAL

Video of the experiments is available at: <https://youtu.be/-iNnP3HLXmI>. Released code at: https://github.com/dranaju/curl_navigation.

I. INTRODUCTION

The problem of autonomous robotics navigation can be defined in the idea of perceiving the environment and planning a path that leads a robot to avoid places that may represent a specific danger [1]. This first principle can be solved by using senses to tell the robot's environment; examples are laser sensors, cameras, or depth maps. In the second principle for an autonomous vehicle, it is necessary to use systems to plan the robot's path to avoid dangerous situations, such as collisions and unsafe conditions. Linking these principles of navigation problems to UAVs, it is possible to see the great difficulty found in contrasting means of performance and perception [2].

Deep Reinforcement Learning (Deep-RL) is a promising approach to tackle the problem of autonomous robotics navigation. In this so-called mapless navigation [3], the

¹Junior C. de Jesus and P. L. J. Drews-Jr are with the NAUTEC, Centro de Ciencias Computacionais, Universidade Federal do Rio Grande - FURG, RS, Brazil. dranaju@gmail.com

²Victor A. Kich, Alisson H. Kolling and Rodrigo S. Guerra are with the Universidade Federal de Santa Maria - UFSM, RS, Brazil. rodrigo.guerra@ufsm.br

³Ricardo B. Grando is with Technological University of Uruguay, Rivera, Uruguay. bedingrando@gmail.com

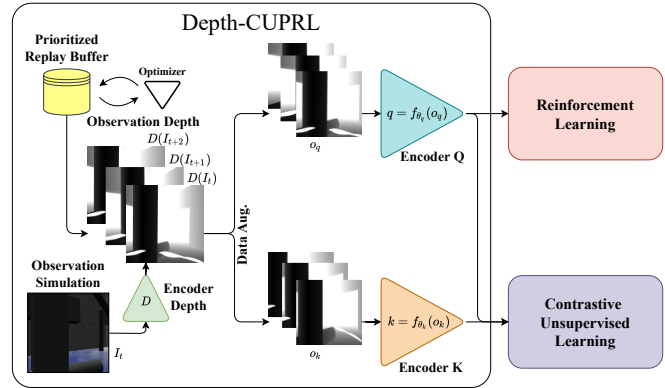


Fig. 1: System of Depth-CUPRL: The observations are passed through a network that generates the depth images. These depth images are stored in a replay memory, and it is sampled memories in the buffer to make the Contrastive and Reinforcement Learning.

methods are capable of providing an action response for continuous systems [4], [5], as well as the Deep-RL control of UAVs in simulated environments [6]. It is possible to see the application of these techniques for the mapless navigation of UAVs in models of unknown dynamics in many environments [6]–[10]. However, one of the significant challenges of Deep-RL techniques is the limitation given by the immense need for labeled data which is difficult to collect in the real world [11]. It has been empirically observed that high-dimensional Deep-RL observations with images of raw pixels are inefficient in terms of sample [12]. Knowing this inefficiency case in RL, better techniques to extract information from images are required.

Since depth maps are a well-known source of perception data for navigation, we propose a Deep-RL system to perform the mapless navigation of an UAV through features extracted from a depth map image. We propose the Depth-Imaged Contrastive Unsupervised Prioritized Representations in Reinforcement Learning (Depth-CUPRL) that considers information from the depth of images with a prioritized replay memory to perform the navigation of an UAV. We show that a combination of Deep-RL and Contrastive Learning leads to better performance when compared with other approaches of Deep-RL based on images. Fig. 1 shows the system of states $D(I_t, I_{t-1}, I_{t-2})$ that will be used in networks for training and its architecture.

The main contributions of this work are listed as follows:

- We present a novel methodology based on depth images and contrastive learning that can successfully perform mapless navigation and obstacle avoidance for an UAV that manages to outperform image-based Deep-RL algorithms;
- We also show that taking into account information from **depth images we are able to increase the performance by 43.9% in success rate in the best case (ours)** and by 26.9% in the worst depth-based method used by comparison.
- We also show that **prioritized experience replay** memory further enhances contrastive learning and generalization ability in deep-RL networks, **increasing the performance by 5.6% in success rate.**

The paper is divided as follows: Section II presents a review of the relevant works in the field. After that, Sections III and IV present details related to the proposed methodology as well as a discussion of the developed tools. Section V, the description of the vehicle and the environment are made and the results obtained are presented. Finally, our contribution and future works are summarized in Section VII.

II. RELATED WORKS

The study around mapless navigation is extensively explored using terrestrial mobile robots [3]. Autonomous navigation of aerial mobile robots using Deep-RL approaches is less frequent and mainly focused on approaches that avoid the use of visual information [6], [10] or using simplified information, without contrastive learning [7]–[9], [13].

Rodriguez *et al.* [7] implemented a Deep-RL approach to solve the problem of landing a UAV on a base that could be in motion or static. Sampedro *et al.* [8] suggested that an approach based on the Deep-RL technique could perform a task of Search and Rescue in closed scenarios. They are modified through a Deep Deterministic Policy Gradient (DDPG) [4]. In the work of [14], it was shown that approaches based on the Soft Actor-Critic (SAC) algorithm [5] is more effective for robot navigation than approaches based on the DDPG algorithm.

Knowing that Deep-RL from high-dimensional observations (state) with raw pixel images is inefficient in terms of sample [15], Laskin *et al.* [16] presented a technique capable of extracting high-level characteristics from images with raw pixels for the control of Deep-RL networks through these extracted features. In addition, the CURL method showed a better performance than the state-of-the-art algorithms such as SAC when dealing with images. However, CURL is a method that was only applied in game-like environments with a third-person view in the environment.

For navigation for mobile aerial vehicles, it is possible to find works like Thomas *et al.* [17] which presents an algorithm for an autonomous UAV using RL with self-attentive models. It showed that it can effectively complete the vehicle navigation even when subjected to varying inputs in the algorithm. This work highlights how the navigation works with the input state data with noise or modifications. He *et al.* [9] used a Lobula Giant Moment Detector (LGMD)

to simplify vision information for Deep-RL in navigation and obstacle avoidance of UAV. It performed missions in a complex environment with 80% of success rate.

The present work differs from the related works by using information from a depth image in a contrastive learning-based approach. Our distance sensing information is yielded from a neural network capable of depth estimation in monocular images instead of a distance sensor such as a Lidar. We also differ by avoiding the high-dimensional observations problem by presenting a contrastive learning structure. Also, we outperform the state-of-the-art by proposing a prioritized experience replay memory system that further enhances our approach. To the best of our knowledge, we are the first to propose such a contrastive learning system with prioritized experience replay.

III. THEORETICAL BACKGROUND

This section details the information of tools and techniques used to develop our work.

A. SAC-Based Approach

The main idea of the SAC technique is to use approximation functions that can learn policies of continuous action space, characterizing this method as a stochastic actor-critic [5]. The SAC algorithm uses a neural network to approximate a policy to the actor-network. The state value is used to estimate a value network V and a value Q with a critic network. These three networks calculate the action prediction for the current state and generate a time difference error signal for each time interval step. In addition, intending to seek the maximization of the system’s rewards, the SAC also seeks to maximize the entropy of the policy. Entropy refers to how unpredictable a variable can be. If a random variable still assumes a single value, then the strategy has zero entropy, encouraging exploration by the agent.

In general, it is recommended to make use of a replay memory that can save the experiences of the agent during the [18] training session. The states, actions, rewards, and new states that an agent has experienced in previous episodes are saved in a memory *buffer*, being marked as an experience. This replay memory technique can disrupt temporal correlations between different episodes in the training phase. A significant performance increase in off-policy Deep-RL algorithms is noticeable even with short memories. Regardless of the increase in application capabilities with Deep-RL, replay memories can affect an agent’s learning time.

Despite the use of replay memories, another network for the target is used with a time delay, called the target network. The update of target network weights can be defined as:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta' \quad (1)$$

with $\tau \ll 1$. This equation tells us that the target network is just a copy of the main network using a Poliak-averaging [19] to update its parameters that affect the learning stability of Deep-RL algorithms.

The SAC technique was used in the CURL as the learning method, and we used it in our Depth-CUPRL approach as

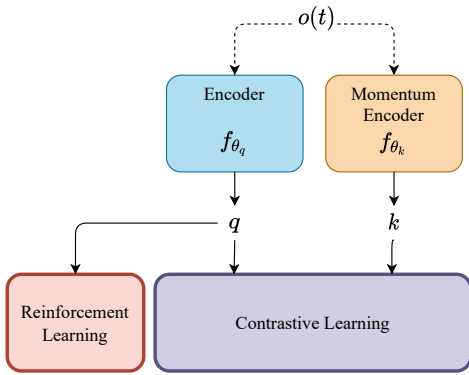


Fig. 2: CURL structure combines instance Contrastive Learning and Reinforcement Learning.

well. We also used this technique in a CNN-based approach with the SAC algorithm (SAC CNN Prio.) to further enhance the comparison with our proposed system.

B. Contrastive Unsupervised Representations for Reinforcement Learning

Contrastive Unsupervised Representations for Reinforcement Learning (CURL) is a general framework for combining contrastive learning with Deep-RL [16]. In principle, any RL algorithm in the CURL pipeline can be used, either on-policy or off-policy. The SAC technique is primarily used in this case [16]. CURL has been shown to significantly improve sample efficiency over previous pixel-based methods by performing contrastive learning simultaneously with an off-policy RL algorithm.

A key component of CURL is the ability to learn rich representations of dimensional data using contrastive unsupervised learning. Given a *query* q and *key* k , the objective of contrastive learning is to secure that q matches with k . A *query* and *key* are positive pairs if they are modified data from the same observation $o(t)$, like an image observation. Fig. 2 shows that CURL trains a visual representation *encoder*, ensuring that embeddings of augmented versions of data o_q and o_k from observation $o(t)$ match using a contrastive loss of the work from Van der Oord *et al.* [20].

The observations from the *query* o_q are handled by *encoder* f_{θ_q} , returning $q = f_{\theta_q}(o_q)$. All observations are built from the minibatch (i) sampled from replay memory for the Deep-RL update. The *key* is encoded and return $k = f_{\theta_k}(o_k)$, where θ_k is a Polyak-averaging version of the *encoder query* or $\theta_k \leftarrow \tau\theta_k + (1 - \tau)\theta_q$. The Deep-RL policy and value function are built on top of the *query* encoder that is trained together with the objectives of reinforcement learning and contrastive learning:

$$\mathcal{L}_{oss_{\theta_q}} = \log \frac{\exp(q^T k_0)}{\sum_0^i \exp(q^T k_i)} \quad (2)$$

We further enhanced the CURL approach and added a prioritized replay memory system to create ours. We implemented a version of the CURL classic and a version with Depth Maps for a more extensively comparison. Both

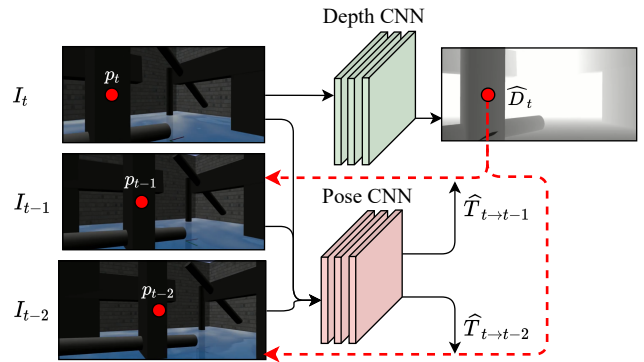


Fig. 3: Unsupervised Depth estimation architecture. The depth network takes the current as input and outputs a depth map \widehat{D}_t . The pose network assists in the loss for the depth map estimation.

prioritized replay memory system and unsupervised depth estimation techniques are described below.

C. Prioritized Replay Memory

In the replay of experience, the agent can use a replay memory that allows data efficiency by storing past experiences of the agent to have the opportunity to reprocess the data later. The data saved in memory refers to the current state s_t , action a_t , reward r_t and next state s_{t+1} of the agent. Furthermore, replay memories also ensure that updates are made from reasonably stable data stored in memory which helps with network convergence.

While replay memory allows processing data transitions in different orders than if they were experienced, there is also the possibility to use [18] prioritized memories. These prioritized experiences allow data transitions with frequencies different from those experimentally tested depending on their importance. That is to say, which experience is stored and which one is repeated.

We added a prioritized replay memory system to our approach and to some of the approaches that we used compared to. The idea is that such a prioritized system increases the ability of the Deep-RL algorithm to learn the depth representation.

D. Unsupervised Depth Estimation

One hypothesis is that the human being developed a rich, structural understanding of the world through our previous visual experience, which consisted mainly of moving around and observing large numbers of scenes and developing consistent models of our observations. From millions of such observations, we learn about the regularities of the world - roads are flat, buildings are straight, roads support cars, etc. We can apply this knowledge to perceiving a new scene, even from a single monocular image [21].

Given this, works such as Bian *et al.* [22] explore learning to estimate depth maps through image sequences. His approach is based on the insight that a geometric view synthesis system performs consistently well when its intermediate predictions of scene geometry and camera pose match the

fundamental physical truth. They proposed a framework for jointly training a single-view depth CNN and a camera pose estimation CNN from unlabeled video sequences, shown in Fig. 3. We adopted the network proposed by Bian *et al.* [22] to learn depth maps from monocular color images in our framework.

IV. METHODOLOGY

In this work, we propose a contrastive prioritized Deep-RL system that considers information from depth maps to perform the mapless navigation and obstacle avoidance of an UAV. We named our system of Depth-CUPRL, as this method is based on a CURL structure [16] using depth maps and prioritized memories for navigation of an aerial vehicle. In the sequence, we explain our proposed approach and the other approaches used in this work for comparison.

The Depth-CUPRL equation of motion is defined as:

$$v_t = f^{Depth-CUPRL}(I_t) \quad (3)$$

where I_t is the raw pixel-colored monocular input image and v_t is the velocity applied to the UAV. For Depth-CUPRL, the Equation 3 passes through $Depth_{CNN}$ which generates a depth image of I_t , the CURL-based network extracts information from this depth map and then passes through a SAC-based network which gives the robot velocity. A schematic diagram of the Depth-CUPRL architecture is illustrated in Fig. 1 and pseudo-code can be seen in Algorithm 1. It uses a [22] dense block-based encoding network to generate depth maps. The KITTI dataset [23] is used to train this network.

After training the Depth-CUPRL with the KITTI dataset, the network is *finetuned* to improve depth estimation for the navigation environment. For this, images are extracted in sequences of random actions to improve the network's performance that generates the scene's depth. After that, the depth estimated is stacked in 3 consecutive temporal frames as the input from the encoder *query* and *key*, as shown in Figure 1. The images received in the encoders that extract information return the latent spaces q and k , where q is used in the Deep-RL network. The outputs of the Depth-CUPRL network are the linear and angular velocities are used to high-level control the UAV.

The network architecture used by the Deep-RL resembles the SAC network presented in the works of de Jesus *et al.* [24], and Grando *et al.* [25]. Fig. 4 shows the architecture including the *query* encoders as convolutional layers. The depth maps are processed by four convolutional layers and three fully connected layers. The number of layers and nodes are based on the proposal of Laskin *et al.* [16]. The output layer estimates angular and linear velocity to be used in the robot. However, the action value is normalized between $(-1;1)$ because of the hyperbolic tangent function (\tanh), which is the activation function. Therefore, the angular velocity and the linear velocity are also constrained between the ranges -0.25 to 0.25 rad/s and 0 to 0.22 m/s , respectively.

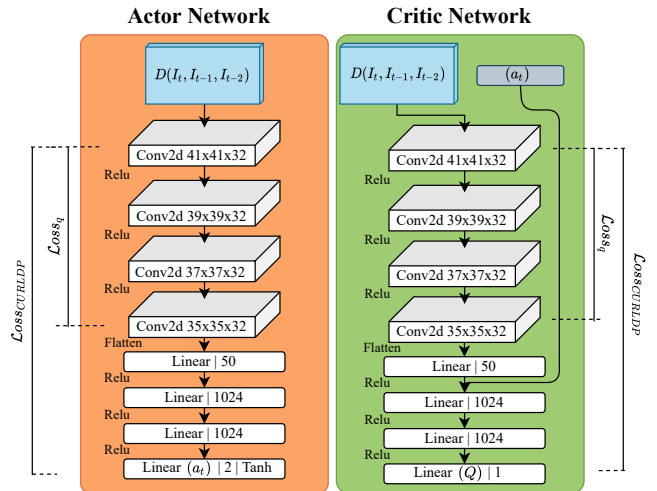


Fig. 4: Depth-CUPRL network architecture.



(a) Real vehicle.

(b) Simulated vehicle.

Fig. 5: Simulated vehicle based on the real Hydrone Vehicle.

A. Reward Function

The rewards and penalties system given to the agent is a simple one, based on empirical knowledge created during the problem-solving process. Thus, the network does a feed-forward and back-propagation step to learn the hyperparameters. The reward function is formulated for the obstacle avoidance task, and it is defined as:

$$r(s_t, a_t) = \begin{cases} r_{navigating} & \text{if } \min_x \geq c_o, \\ r_{collision} & \text{if } \min_x < c_o, \end{cases} \quad (4)$$

where a negative reward ($r_{collision} = -1$) is given if the robot collides with an obstacle. c_o is equivalent to the robot's distance of $62cm$ from an obstacle or wall. These collision conditions cause the agent to be reset to the initial training position. Furthermore, $r_{navigating}$ is given when the robot is navigating the map, whose value was decided to be 0.01 . The observation that the only thing we want the robot to avoid is colliding with objects or walls defined the rewards without creating a misleading reward for the agent to interpret the map [26].

B. Experimental Setup

The simulations are run with the Robot Operating System (ROS) [27] as backbone managing the connections between the proposed methods and the simulations. The chosen simulator is Gazebo for an easy connection with ROS and its realistic physics simulation. The algorithms are developed using the PyTorch library and OpenCV [28]. The robotic UAV platform used in the simulation is designed by Grando

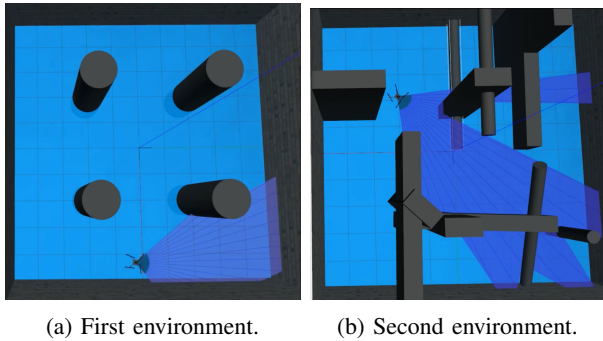


Fig. 6: Simulated Environments used on Gazebo simulation.

et al. [29]. The vehicle is based on a real vehicle called Hydrone [30], [31]. The Hydrone vehicle is a quadrotor-like hybrid unmanned aerial underwater vehicle, but only the aerial capabilities are evaluated here. Fig. 5 shows the real vehicle and the final vehicle described in simulation. The model in ROS can be controlled by linear and angular velocity in 6 degrees of freedom ($x, y, z, roll, pitch, yaw$).

For the vehicle’s base frame, the value of 1.93 kg for mass is used. For the *collision* of the frame, a box with dimensions of 0.47 meters on a side and 0.09 meters in height was estimated. The four rotors are placed symmetrically with an angle of 90° between each and an exact distance of 0.27 meters from the center of the vehicle.

C. Simulated Environments

Both environments created in Gazebo can be seen in Figure 6. The proposed environments are generated to show that the approach used in this work can navigate without colliding with walls and obstacles. In the first environment, four obstacles are allocated as illustrated in Figure 6a, the addition of these fixed obstacles produce a more challenging scenario for the robot. Therefore, if the vehicle collides with the wall or any obstacle, a negative reward is given for this action and the current episode stops. Fig. 6b show a more complex navigation scenario. This environment presents a higher degree of complexity than the first one since we add several obstacles.

V. EXPERIMENTAL RESULTS

This section presents the discussion of the results. Firstly, we present the reward gathered during the training. This information allows the estimation of the agent’s degree of learning in the environment, as the improved in terms of reward is intrinsically linked to the agent’s performance in the environment it must navigate. All environments used for training the network are provided by the Grando *et al.* [29]. However, some changes are made in the source code of the Gazebo simulation to use the camera of the simulated UAV.

It is important to note that a Deep-RL agent is trained for each proposed environment. We compared our method Depth-CUPRL with the *CURL* [16] having as input the raw pixel image, and with an own version of the *CURL* called here *CURL (Depth)* with depth maps as input but without prioritized memory. We also compared with a *SAC (CNN*

prio.) as adopted in Grando *et al* [25] but with convolutional layers. The *SAC (CNN prio.)* uses depth maps as inputs of the network and has prioritized memory. All networks tested in the work followed the architecture proposed in Fig. 4. The initial position of the vehicle for both training and testing in each of the scenarios is defined as the Cartesian position (3.6, -2.6, 2.0) for the aerial environment. Only linear and angular velocities are applied to the *yaw* of the vehicle.

Our method is configured with a prioritized replay memory of 35000 samples for training in the first environment. Each episode has 1000 steps t . The network is evaluated at each 1000 step. The evaluation of the networks is carried out in such a way that it uses only the deterministic response of the network without any noise for 10 episodes. With this, it is possible to evaluate the network’s response by an average of tests and having as the maximum reward for evaluation of 10.00.

The results of training the reward function of the first environment for the proposed and tested networks are shown in Figure 7a. In the first episodes of training networks, negative rewards are noted. This condition is because the algorithm has started and is still learning. This episode reward means the robot is trying to maximize the task reward. All networks are trained for 100000 steps. It is possible to conclude that the Depth-CUPRL, *CURL (Depth)*, *SAC (CNN prio.)* networks successfully completed the proposed task in this environment. Depth-CUPRL presents higher efficiency to converge to the optimal reward. *CURL* based on RGB image [16] remains unstable, failing to converge to an optimal result.

The results gathered in the second environment are shown in Fig. 7b. Given a higher complexity, a more significant number of training interactions is necessary for this environment. We adopted a replay memory of 140000 and all methods are trained for almost 300000 steps. If compared with the previous reward functions, from Fig. 7a, it is possible to notice a more unstable reward for the trained networks. Despite this, the Depth-CUPRL obtains the highest average reward. Thus, we can see the advantage of using our technique in navigation and obstacle avoidance.

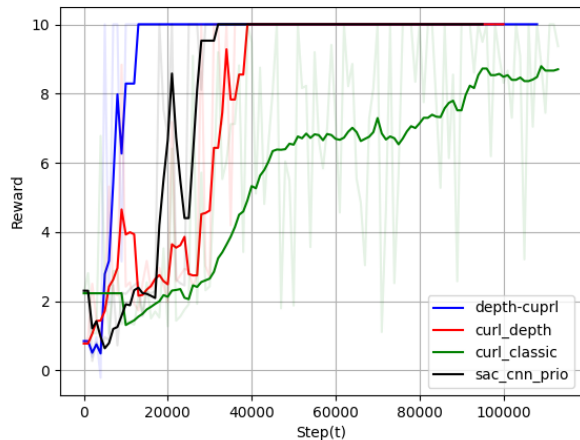
VI. DISCUSSION

After training in the first environment, a final model evaluation is performed after the 1000 episodes. Table I shows the results obtained for all methods.

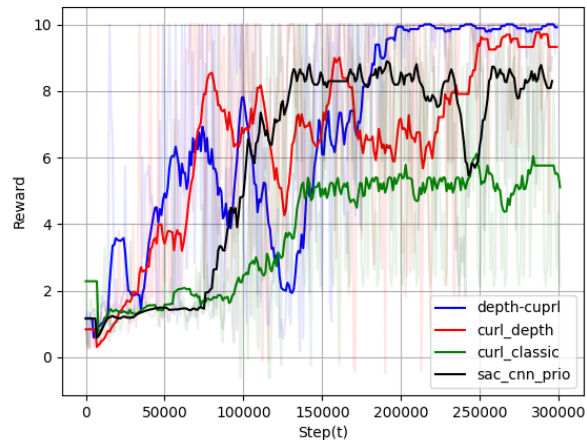
In this evaluation, it is possible to see that the algorithms that used information from depth images were able to

TABLE I: Comparison between our approach and other approach in the first environment. Bold letters highlights the best results.

Algorithm	Image	Success	Crash	Success Rate (%)
<i>CURL (Depth)</i>	Depth	1000	0	100%
<i>CURL (Classic)</i>	Pixel	856	144	85.6%
<i>SAC (CNN prio.)</i>	Depth	1000	0	100%
Depth-CUPRL	Depth	1000	0	100%



(a) First environment.



(b) Second environment.

Fig. 7: Moving average of the reward over 100000 and 300000 training steps for the first and second environment, respectively.

complete the 1000 tests with 100% success. Specifically, CURL (Classic), which uses pixel-based images, crashed many times in the evaluation. We can see in these results that the depth map images approach had a better efficiency in navigating through the first environment.

After training the approaches in the second environment, a final evaluation is performed at the end of the 1000 episodes. Table II shows the obtained results for all approaches.

In this evaluation, the depth image approaches, collaborating with the results of Table I, had a better performance. Yet, none of the approaches were completed the scenario without a crash. Our approach Depth-CUPRL got the best performance crashing only 8 times of 1000, showing an overall better performance of 5.6% in success rate. Only the approach that could not get a success rate above 80% was the pixel-based method.

Taking into account information from depth images, we are able to increase the performance by 43.9% in success rate in the best case (ours) and by 26.9% in the worst depth-based method used by comparison (SAC (CNN prio)).

VII. CONCLUSION AND FUTURE WORK

In this work, we presented a novel Deep-RL based on the information of the estimated image depth of a simple monocular camera system to perform the mapless navigation and obstacle avoidance of UAVs. We evaluated our approach

TABLE II: Comparison between proposal and other networks in the second environment. Bold letters highlights the best results.

Algorithms	Image	Success	Crash	Success Rate (%)
CURL (Depth)	Depth	936	64	93.6%
CURL (Classic)	Pixel	553	447	55.3%
SAC (CNN prio.)	Depth	822	178	82.2%
Depth-CUPRL	Depth	992	8	99.2%

in two realistic scenario showing the Depth-CUPRL approach outperforms the state-of-the-art in terms of image-based RL approach. The use of depth maps is a commonplace in robotics and we show the advantage to use it in Deep-RL methods instead of raw pixel images.

With our approach, the generated depth maps are used as inputs in a network that learns these representations by image sequences. After that, this information is extracted and the Deep-RL agent trained to perform the vehicle’s navigation and make it wise enough to avoid collisions with objects. The results show that the reward increases with the number of training episodes in the environment.

The next steps will be focused in apply our approach also to the underwater navigation of the Hydrono [25], [30] using underwater depth maps [32]. For these underwater scenarios, we intended to improve the depth estimation method to handle a high turbidity environment such as water and still be effective when in an aerial environment. Furthermore, real-world results are being conducted.

ACKNOWLEDGEMENTS

The authors would like to thank the VersusAI team. This work was partly supported by the Technological University of Uruguay (UTEC), Federal University of Santa Maria (UFSM), Federal University of Rio Grande (FURG), and PRH-ANP.

REFERENCES

- [1] S. F. Alves, J. M. Rosario, H. Ferasoli Filho, L. Rincon, R. Yamasaki, and A. Barrera, “Conceptual bases of robot navigation modeling, control and applications,” *Advances in Robot Navigation*, p. 26, 2011.
- [2] R. B. Grando, P. M. Pinheiro, N. P. Bortoluzzi, C. B. da Silva, O. F. Zauk, M. O. Piñeiro, V. M. Aoki, A. L. Kelbouscas, Y. B. Lima, P. L. Drews-Jr, and A. A. Neto, “Visual-based autonomous unmanned aerial vehicle for inspection in indoor environments,” in *IEEE LARS/SBR*, 2020, pp. 1–6.
- [3] L. Tai and M. Liu, “Towards cognitive exploration through deep reinforcement learning for mobile robots,” *arXiv preprint arXiv:1610.01733*, 2016.

- [4] 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," in *ICLR*, 2016.
- [5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*, vol. 80, 2018, pp. 1861–1870.
- [6] R. B. Grando, J. C. de Jesus, and P. L. Drews-Jr, "Deep reinforcement learning for mapless navigation of unmanned aerial vehicles," in *IEEE LARS/SBR*, 2020, pp. 1–6.
- [7] A. Rodríguez-Ramos, C. Sampedro, H. Bavle, I. G. Moreno, and P. Campoy, "A deep reinforcement learning technique for vision-based autonomous multirotor landing on a moving platform," in *IEEE/RSJ IROS*, 2018, pp. 1010–1017.
- [8] C. Sampedro, A. Rodríguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, pp. 601–627, 2019.
- [9] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Integrated moment-based LGMD and deep reinforcement learning for UAV obstacle avoidance," in *IEEE ICRA*, 2020, pp. 7491–7497.
- [10] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, and P. L. J. Drews-Jr, "Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, pp. 1–14, 2022.
- [11] R. Bonatti, R. Madaan, V. Vineet, S. Scherer, and A. Kapoor, "Learning visuomotor policies for aerial navigation using cross-modal representations," in *IEEE/RSJ IROS*, 2020, pp. 1637–1644.
- [12] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.
- [13] B. Li, Z. Gan, D. Chen, and D. Sergey Aleksandrovich, "Uav maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning," *Remote Sensing*, vol. 12, no. 22, p. 3789, 2020.
- [14] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, pp. 1–11, 2021.
- [15] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.
- [16] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *ICML*, 2020, pp. 5639–5650.
- [17] D.-G. Thomas, D. Olshanskyi, K. Krueger, T. Wongpiromsarn, and A. Jannesari, "Interpretable uav collision avoidance using deep reinforcement learning," *arXiv preprint arXiv:2105.12254*, 2021.
- [18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th Int. Conference on Learning Representations, ICLR*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: <http://arxiv.org/abs/1511.05952>
- [19] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SICON*, vol. 30, no. 4, pp. 838–855, 1992.
- [20] A. Van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
- [21] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *IEEE CVPR*, 2018, pp. 1983–1992.
- [22] J.-W. Bian, H. Zhan, N. Wang, Z. Li, L. Zhang, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised scale-consistent depth learning from video," *IJCV*, pp. 1–17, 2021.
- [23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *IJRR*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [24] J. C. de Jesus, J. A. Bottega, M. A. Cuadros, and D. F. Gamarra, "Deep deterministic policy gradient for navigation of mobile robots in simulated environments," in *19th ICAR*, 2019, pp. 362–367.
- [25] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, N. P. Bortoluzzi, P. M. Pinheiro, A. Alves Neto, and P. L. J. Drews-Jr, "Deep reinforcement learning for mapless navigation of a hybrid aerial underwater vehicle with medium transition," in *IEEE ICRA*, 2021, pp. 1088–1094.
- [26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," in *IEEE ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.
- [28] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.
- [29] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, and P. L. J. Drews-Jr, "Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, pp. 1–14, 2022.
- [30] P. L. Drews-Jr, A. A. Neto, and M. F. Campos, "Hybrid unmanned aerial underwater vehicle: Modeling and simulation," in *IEEE/RSJ IROS*, 2014, pp. 4637–4642.
- [31] A. C. Horn, P. M. Pinheiro, R. B. Grando, C. B. da Silva, A. A. Neto, and P. L. Drews-Jr, "A novel concept for hybrid unmanned aerial underwater vehicles focused on aquatic performance," in *IEEE LARS/SBR*, 2020, pp. 1–6.
- [32] P. L. Drews-Jr, E. R. Nascimento, S. S. Botelho, and M. F. M. Campos, "Underwater depth estimation and image restoration based on single images," *IEEE computer graphics and applications*, vol. 36, no. 2, pp. 24–35, 2016.

Algorithm 1 Depth-Imaged Contrastive Unsupervised Prioritized Representations in Reinforcement Learning

- 1: Input: initial policy parameters ϕ , Q-function parameters θ_1, θ_2 , empty replay buffer M , maximum number of episodes E , maximum number of steps T
- 2: Image network: depth estimation network trained D , contrastive encoder ψ_q and ψ_k ;
- 3: Set target parameters equal to main parameters $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$
- 4: **for** $episode = 1$ until E **do**
- 5: reset environment
- 6: **for** $t = 1$ until T **do**
- 7: Observe image state I_t
- 8: Image observation passed to depth map $d = D(I_t)$
- 9: Depth map passed to $s = \psi_q(d)$
- 10: Observe state s and select action $a \sim \pi_\phi(\cdot|s)$
- 11: Execute a in the environment
- 12: Observe next state s' , and reward r
- 13: Store (s, a, r, s', d) in replay buffer M
- 14: **if** update Reinforcement Learning **then**
- 15: Randomly sample a batch of transitions, $B = \{(s, a, r, s')\}$ from M
- 16: Compute targets for the Q functions:

$$y(r, s') = r + \gamma \left(\min_{i=1,2} Q_{\theta'_i}(s', a') - \alpha \log \pi_\phi(a'|s') \right)$$

- 17: Update the critic:

$$\nabla_{\theta_{i=1,2}} \frac{1}{|B|} \sum_{(s,a,r,s') \in B} (Q_{\theta_i}(s, a) - y(r, s'))^2$$

- 18: Update policy:

$$\nabla_\phi \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\theta_i}(s, a_\phi) - \alpha \log \pi_\phi(a_\phi | s) \right)$$

- 19: Updates target:

$$\theta'_{i=1,2} \leftarrow \tau \theta_i + (1 - \tau) \theta_i$$

- 20: **end if**
- 21: **if** update Contrastive Learning **then**
- 22: Randomly sample batch of depth map d , $B(d) = \{(s, a, r, s', d)\}$ from M
- 23: Random augmentation $aug()$ sample for *query* and *key*:
 - $d_q = aug(d)$
 - $d_k = aug(d)$
- 24: Computes the latent space of *query* and *key*:
 - $q = \psi_q(d_q)$
 - $k = \psi_k(d_k)$
- 25: Updates *query*:

$$\mathcal{L}_{oss_{\psi_q}} = \log \frac{\exp(q^T k_0)}{\sum_{i \in B(d)} \exp(q^T k_i)}$$

- 26: Updates *key*:

$$\psi_k \leftarrow \tau \psi_k + (1 - \tau) \psi_k$$

- 27: **end if**
 - 28: **end for**
 - 29: **end for**
-