

## Dynamic Reward in DQN for Autonomous Navigation of UAVs using Object Detection

Lagoda, Adam; Fatemeh Mahdavi Sharifi, Seyedeh; Pedersen, Thomas Aagaard; Ortiz Arroyo, Daniel; Chang, Shi; Durdevic, Petar

*Published in:*

9th 2023 International Conference on Control, Decision and Information Technologies, CoDIT 2023

*DOI (link to publication from Publisher):*

[10.1109/CoDIT58514.2023.10284087](https://doi.org/10.1109/CoDIT58514.2023.10284087)

*Publication date:*

2023

*Document Version*

Accepted author manuscript, peer reviewed version

[Link to publication from Aalborg University](#)

*Citation for published version (APA):*

Lagoda, A., Fatemeh Mahdavi Sharifi, S., Pedersen, T. A., Ortiz Arroyo, D., Chang, S., & Durdevic, P. (2023). Dynamic Reward in DQN for Autonomous Navigation of UAVs using Object Detection. In *9th 2023 International Conference on Control, Decision and Information Technologies, CoDIT 2023* (pp. 2372-2377). Article 10284087 IEEE (Institute of Electrical and Electronics Engineers). <https://doi.org/10.1109/CoDIT58514.2023.10284087>

### General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

### Take down policy

If you believe that this document breaches copyright please contact us at [vbn@aub.aau.dk](mailto:vbn@aub.aau.dk) providing details, and we will remove access to the work immediately and investigate your claim.



# Dynamic Reward in DQN for Autonomous Navigation of UAVs using Object Detection

Adam Lagoda<sup>1</sup>  
alagod19@student.aau.dk

Seyedeh Fatemeh Mahdavi Sharifi<sup>1</sup>  
smahda17@student.aau.dk

Thomas Aagaard Pedersen<sup>1</sup>  
tape19@student.aau.dk

Daniel Ortiz-Arroyo<sup>1</sup>  
doa@et.aau.dk

Shi Chang<sup>2</sup>  
sc2892@cornell.edu

Petar Durdevic<sup>1</sup>  
pdl@et.aau.dk

**Abstract**—This paper discusses the implementation of a Deep Reinforcement Learning policy, based on DQN, which optimizes the navigation of the UAV to the front of wind turbine blades. The UAV was trained in simulation using Unreal Engine V4.27 coupled with AirSim. The action space of the UAV was discretized while allowing 6 different actions to be executed. A YoloV5 network trained with images of simulated wind turbines was used for detection and tracking, providing the DQN policy with state information, upon which it has been trained. In addition to this, the dynamic reward has been implemented, which combined both navigation and inspection objectives in the final evaluation of actions. Our tests showed that after 7500 time-steps the exploration rate reached near 0, the mean length of the episodes increased from 10 down to 30, but the mean reward increased from around -60 to stabilizing the output at 26. These results suggest that the proposed method is a promising solution to optimizing the autonomous inspection of wind turbines with UAVs.

**Index Terms**—Deep Reinforcement Learning, Deep Q-network, UAV, Dynamic Reward, Condition Monitoring, Path-planning, Inspection, Simulation

## I. INTRODUCTION

The global need for sustainable energy has led to the increased use of wind energy, where wind turbines account for 7% of the worldwide power demand [1]. In order to maintain their high efficiency, regular inspections are necessary to prevent damages [2], increase their lifespan [3], and increase annual energy production by up to 5% [4]. Traditional inspection methods for offshore wind turbines involve human observations with professional climbers [5]. Alternatively, unmanned aerial vehicles (UAVs) controlled by external operators have been proposed to decrease inspection costs and improve time efficiency [6]. Automation of the inspection process can minimize dependence on human interference and further decrease the expenses [7]. Challenges arise when fully automating the control of an UAV in an unfamiliar or unstructured environment [8] [9]. Therefore to ensure safe and reliable operation is essential to obtain a comprehensive description of the environmental perception. Including the UAVs ability to promptly and accurately react to changes, for example, variation in wind speed and lighting levels.

Achieving full automation of offshore inspections presents a new challenge due to the diverse range of navigational tasks that the UAV is required to perform. To address this challenge, the controller must be equipped with supervisory rules that guide the UAV's behavior, including tasks such as maintaining a specified altitude, a safe distance from wind turbine blades, and returning to the launch site in the event of an emergency or failure.

To enable a UAV to execute a multitude of complex tasks in a 3D space, an approach could be to use low-level flight control coupled with a supervisory rule-based or fuzzy controller to optimize the drone's navigation [10]. However, traditional controllers, such as PID, are not effective in handling dynamic changes such as wind gusts or turbulence, which can significantly affect the UAV's flight trajectory [11]. Optimizing UAV's trajectories is one of the aims of automating inspections, with the goal of enhancing its efficiency. Rather than employing a rule-based approach, we propose utilizing Deep Reinforcement Learning (DRL) to optimize searching the task space.

DRL combines deep and reinforcement learning enabling the UAV to learn complex behaviors through trial and error [12]. The UAV learns by interacting with the environment and receiving feedback in the form of rewards or penalties, making it effective in handling complex tasks in dynamic and uncertain environments. Deep neural networks can leverage their ability to approximate complex functions to learn, represent, and utilize high-dimensional state and action spaces effectively. As a result, they are well-suited for tasks that require long-term planning and decision-making.

Significant progress has been made in recent years in applying DRL for UAV control. A wide range of DRL policies exist, examples are Deep-Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), Dueling-DDQN [13]. The following section will investigate different approaches for the automation of UAV. [14] uses the DQN policy to train UAVs for tracking and navigation in virtual environments, with the reward function being based on the Euclidean distance between the image and the camera center point. [15] instead employs a grid-based navigation approach with DDQN networks and a reward function discouraging negative and emphasizing positive actions to guide the UAV. Whereas [16] uses DDPG

\*This work was not supported by any organization

<sup>1</sup> Department of Energy, Aalborg University, 6700 Esbjerg, Denmark

<sup>2</sup> Cornell University, Ithaca, 14850 New York, United States

methods to navigate a UAV in simulation to a target location without visual tracking, with the reward shaped by the distance from obstacles to avoid a collision. In [17] they present a tool that extends the creation of realistic simulated worlds using the Unreal Engine with OpenCV simulated integration. The tool provides a set of OpenCV-like cameras to generate data for computer vision tasks. [18] and [19] focus on using deep reinforcement learning (DRL) and object detection to enable collision-free UAV navigation with a single monocular camera. They both train object detection by incorporating it into DRL policy, thus using raw pixel images in an unreduced state form. Lastly, [20] presents using the Proximal Policy Optimization (PPO) and ADAM optimizer for DRL policy training in simulation and validation/implementation in the real world for autonomous multi-view navigation. The proposed system uses multi-camera sensing and a trained object detection network to reduce state dimensionality. In addition to this, the system exhibits robustness in sim2real transfer as it was trained only with simulated scenarios but performed well in real-world tests.

In many cases the objectives of UAVs are subject to dynamic changes, rendering a fixed reward function insufficient in achieving optimal performance. In such scenarios, it is possible to implement a dynamic reward function that can adjust to changes in the objective. The following section discusses recent work and advances in the utilization of dynamic reward functions for UAVs. In [21] a dynamic adjustment of the reward function takes into account three distinct perspectives, where each of them changes dynamically during operation. These perspectives include comfort, with the aim of avoiding risky maneuvers; efficiency, aimed at reducing travel time; and safety, with the goal of avoiding collisions. The total reward function is the sum of each of these parameters. [22] proposes the utilization of the DDPO-IL algorithm for dynamic adjustment of the reward function. This approach adjusts the behavior of the agent through dynamic modification, incorporating an intrinsic reward calculated based on the state of the agent into the conventional external reward. The result is a refined reward function that facilitates improved performance and stability. [23] introduces Hybrid Reward architecture (HRA) which decomposes the reward function into sub-intervals and the agent learns about each reward function separately and then aggregates it into one, optimizing the reward based on multiple aspects. [24] proposes Deep MaxPain as an improvement of HRA where it minimizes the expected sum of negative rewards and maximizes the sum of positive rewards. This architecture utilizes knowledge of future punishment at states for higher probability of avoiding them.

## II. SIMULATION ENVIRONMENT

Simulation environments are increasingly important in the field of robotics, particularly for training and testing autonomous systems that operate in complex and hazardous environments. In this paper, a realistic simulation environment is used to enable the training of UAVs for navigation in wind turbine inspection using DRL.

### A. Unreal Engine

Unreal Engine is a software development platform that is widely used in the video game industry but its advanced graphics and physics simulation capabilities make it an ideal tool for creating realistic virtual environments for scientific research, especially in DRL training and validation.

### B. AirSim

The interaction with the UAV inside the simulation is performed with AirSim [25] which acts as a middleware bridge, connecting UAV's model to a C++ or Python API. One of the key features is its full kinematic, dynamic, and graphical model of the UAV, which creates the possibility to design closed-loop control systems. Its main advantage is using programming scripts outside the Unreal Editor, where proprietary blueprints must be used.

In addition to this, it allows for different modeling and control approaches. In the case of this paper, a GPS-controlled drone is assumed, which is equipped with an internal low-level controller to counteract external inputs and disturbances (such as wind) and maintain the same position. The user's high-level can then be supplied on top of that, which determines the new position's set point.

### C. Object Detection

In this project, as YOLOv5s is one of the most popular single-shot detection network[26], that can easily be retrained and still provide relatively high performance, it is used to detect wind turbine blades to find the optimal path to a wind turbine for a UAV using reinforcement learning. The network was trained using 1655 images of wind turbine blades in a simulation environment. Key point here is that even though the object detection model has been trained on different rotational positions of the wind turbine's rotor to ensure high accuracy, during actual training and implementation the rotor is stationary, as this is the usual procedure for inspections. Transfer learning was used, where the first ten convolutional layers of the YOLOv5s network, responsible for feature extraction, were frozen and the last layers retrained to detect wind turbines. Fig. 1 shows a few examples of the object detection model's performance on the testing set. The model detects all of the wind turbine blades with an mAP ranging between 0.81 and 0.97.



Fig. 1. Confidence scores of some test images

### III. DRL IMPLEMENTATION

#### A. Deep Q-Network

The DQN (Deep Q-Network) algorithm is a well-known reinforcement learning (RL) technique [12]. DQN uses neural networks to estimate the optimal quality function  $Q$ , where the neural network is trained to approximate the optimal  $Q$ -value for a given state-action pair. This process can be modeled as a Markov process since the next state (i.e., the neural network's approximation of  $Q$ -value) depends only on the current state (i.e., the given state-action pair) and not on any previous states. The quality function  $Q$  is defined as [14]:

$$Q'(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

Here,  $Q'(s_t, a_t)$  represents the updated value for  $Q$  and  $Q(s_t, a_t)$  denotes the current value for  $Q$  corresponding to the state-action pair. The learning rate  $\alpha$  controls the rate at which the network updates the  $Q$ -values, while  $r_t$  represents the immediate reward at time  $t$ . The discount factor  $\gamma$  determines the weight given to future rewards, and  $\max_a Q(s_{t+1}, a)$  is the maximum expected  $Q$ -value for the next state.

To minimize the loss function shown in Equation (2), the quality function  $Q$  is parameterized with neural network weight  $\theta$ , such that  $Q(s_t, a_t) \approx Q(s_t, a_t, \theta)$ . By optimizing over the  $\theta$  value, the network learns to minimize the expectation of the square of the temporal difference error. This results in the neural network approximating the optimal  $Q$ -values for a given state-action pair.

$$L_i = \mathbb{E}[(r_t + \gamma \max_a Q(s_{t+1}, a, \theta) - Q(s_t, a_t, \theta))^2] \quad (2)$$

#### B. Action space

The action space is a set of all valid actions and choices available to an agent as it interacts with the environment. As in a real-world situation, when examining a UAV, the possible movement can be counted in 6 different degrees of freedom. The real-world action space has been limited to a finite set so that the UAV can only perform 7 different actions, that last a predetermined amount of time, with fixed angular and linear velocities. The main advantage of utilizing a discrete action space is the simplification of training. As this research acts as a proof of concept, minimizing the action space allows for easier and faster computation, as well as creates a higher probability of convergence of the policy. After the simplification is implemented, a graphical explanation of possible movements of the UAV can be seen in Fig. 2.

#### C. Observation Space

Observation space is a set of observations that can be made in a given environment. The feedback is constructed from three sources:

- 1) YOLOv5 Object Detection
- 2) Distance measurement from the depth camera

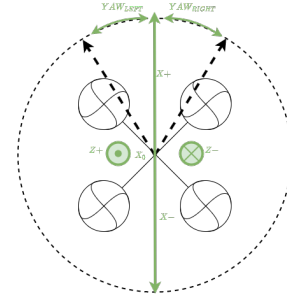


Fig. 2. Visualized Discrete action space of the UAV in this project

#### 3) Kinematics of the UAV

Both color and depth cameras are aligned with the  $X$  axis of the UAV and facing towards the direction of flight. Using the depth image, the edge detection is performed, as described in Section III-D. In addition to this, the observation space is monitoring the kinematics of the UAV, such as its position, velocity, linear and angular acceleration, as well as collision detection.

YOLOv5 model, described in Section II-C, is used for camera feedback processing. With its aid, the wind turbine's rotor can be marked with a bounding box, and the outputs are the coordinates of its corners, which dictate the rewards for the action of the UAV.

The observation space can be divided into 5 tasks:

- 1) Obtain kinematic states of the UAV
- 2) Reset the position of the camera, re-aligning it in the correct orientation
- 3) Parse the color camera image and perform object detection using YOLOv5
- 4) Parse the depth camera image, interpolate, crop to bounding box size, and measure the distance to the closest point.
- 5) Using the depth camera image, perform edge detection for inspection purposes

Using AirSim built-in functions [25], all kinematic state values are obtained, along with the position, orientation, and velocity in both global and body-frame world coordinates, which are necessary for later computation and evaluation. The position stored in memory from the previous iteration (time-step) is saved as well. In addition, the observation space monitors collisions between the UAV and its surroundings.

After yielding the color image from the Unreal Engine, through AirSim in a correct format, the YOLOv5 Object Detection Function can be called, which outputs the coordinates of the two opposing corners of the bounding box, the width, and height of the image, and a boolean value that corresponds to wind turbine detection

The second image is the depth camera output. Its resolution is greatly smaller than the one of the color image, therefore for this reason the image is being interpolated, based on a bi-linear function, which creates a new pixel using a weighted average of the two neighboring pixels before interpolation. The goal of this approach is to have the same resolution for both color and depth camera images, which allows for

easy manipulation later when measuring the distance to the wind turbine. The next step is to crop the interpolated depth image to the size and location of the bounding box. For this purpose, coordinates  $x_{min}$ ,  $y_{min}$ ,  $x_{max}$ , and  $y_{max}$ , which were retrieved earlier from YOLOv5 model, are utilized. At this point, the output looks as shown in Fig. 3.

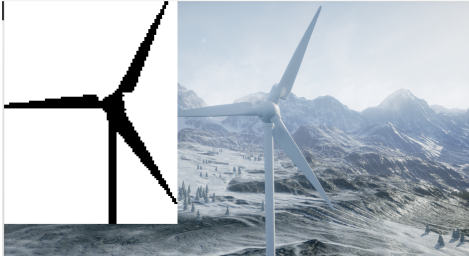


Fig. 3. Cropped depth image and full-size color image

The goal is for the cropped depth image to bind only the rotor of the wind turbine so that the closest point that the depth camera can measure a distance to is the wind turbine itself. The darkest pixel represents the closest object to the camera, it corresponds to *meter* unit in the simulation, and it can be saved directly and used for further evaluation.

Finally, using the uncropped depth image, as described before, edge detection is performed, which outputs coordinates of the beginning and end points of the longest detected line.

#### D. Dynamic Reward function

As the UAV aims to fulfill two tasks sequentially, the reward evaluation and grading have to be constructed dynamically. The dynamic reward function consists of a weighted sum of two sub-functions, as seen in Eq. 3. First is the  $R_{navigation}$  reward function, which evaluates the centering of the wind turbine's rotor in the camera frame. For this purpose, the piece-wise linear centering function is used, called *center()* in the equation, which output can be seen in Fig. 4). The input to the function are the  $x$  and  $y$  coordinates of the corners of the bounding box that overlays the rotor in the camera view. The sub-function also monitors the Euclidean distance  $D$  between the UAV's position  $pos_{UAV}$  and the wind turbine's position  $pos_{WT}$ , to determine whether the UAV is moving in the correct direction. Its simplified version can be seen in the Eq. 4.

The second part ( $R_{inspection}$ ), as seen in Eq. 5, concerns inspection in terms of focusing on one of the leading edges of the rotor's blade in the camera view. This is done by replacing the input to the centering function with the detected leading edge and continuing to track the change in the Euclidean distance.

$$R_{dynamic} = w_1 R_{navigation} + w_2 R_{inspection} \quad (3)$$

$$R_{navigation} = center(rotor) - \Delta D(pos_{UAV}, pos_{WT}) \quad (4)$$

$$R_{inspection} = center(edge) - \Delta D(pos_{UAV}, pos_{WT}) \quad (5)$$

The factor that determines the values of the weights  $w_1$  and  $w_2$  is based on the distance between the wind turbine and the UAV and is adjusted using *tanh* function to smoothly change focus between current objectives.

$$w_1 = \frac{1 + \tanh(2(distance - 30))}{2} \quad (6)$$

$$w_2 = 1 - w_1 \quad (7)$$

Before the design of the final reward function, the available feedback for both sub-functions has to be determined. In this project, such feedback is provided by these variables:

- Coordinates of the bounding box opposing corners
- Distance to the wind turbine
- Coordinates of the detected edge that falls on the leading edge of the wind turbine's blade

The evaluation is then based on these determining factors, which should yield specific, ideal results, based on empiric data from human actions:

- The UAV is moving towards the wind turbine in each step
- It is learning the environment and making sure not to collide with it
- Works its way to keep the wind turbine centered in the camera view
- When the distance to the wind turbine is less than 30m, instead of focusing on the rotor, try to maximize the blade in the camera frame

The centering evaluation is based on a piece-wise linear function, which applies a floating integer value of a reward, based on the distance between the center of the camera frame and the center of the bounding box or the edge. An optimal region was defined where values are positive, linearly increasing to a reward of 1 when the center of the bounding box aligns with the center of the region, and decreasing to 0 when the object/edge center is at the border. Negative rewards are applied, when the center of the detected object/edge is outside of the optimal region, with a minimum value of -1, if the center lands on the border of the camera view. Those rewards can be visualized by applying the function to the image's  $x$  and  $y$  axis, and the result can be seen in Fig. 4. The image can be interpreted as a heat map, where the *RGB* value of the red component is derived based on the output from the reward function. *RGV* value of  $[0, 0, 255]$ , which represents *red*, corresponds to a reward of +2, whereas a value of  $[0, 0, 255]$ , which represents *blue*, corresponds to a reward of -2. Value of  $[255, 0, 255]$ , visualized as *pink* is a reward of 0. If applied to a single direction, the inside of the green box would contain positive rewards, the outside negative values, and the border itself is a 0. Additionally, the object or the edge has to be detected at all times during the flight, and if such a requirement is not completed, then the episode is terminated.

The edge detection is performed on the depth camera image, as thanks to the relative change in the distance measurement



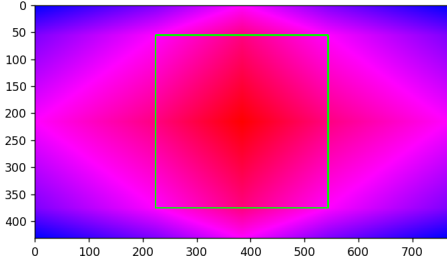


Fig. 4. Visualized piece-wise linear reward function for centering, where the reward of 1 is pure red, -1 is pure blue, and 0 is pink

between the background and the blade, it is comparatively less challenging to detect the leading edge, opposite to using only a color camera. It allows performing a simple segmentation based on grouping objects by distance, thus verifying that the correct line is being detected and decreasing the chance of false positive results.

The process consists of blurring the image, using the *canny* function and probabilistic Hough Transform for line detection. The output has been filtered and sorted to output the longest line that falls on the leading edge of the blade.

The inspection reward function consists of two aspects upon which the reward is given:

- 1) Evaluating centering of the line based on its midpoint location in the camera frame, utilizing the same centering function as for navigation
- 2) Evaluate maximization in the camera frame using a percentage of the detected line's length in terms of the maximum possible line length in the camera frame based on its slope.

#### IV. RESULTS

Quantitative improvement in the results have been achieved after performing several training batches and adjusting the hyper-parameters of the policy. Fig. 5 shows the mean length of an episode in terms of time steps. Noticing that the maximum length is predefined in the reward function III-D to a value of 30. During the exploration period (until around 7500 *timesteps*), the policy is looking for the most optimal approach, constantly looking for the global minimum. After the exploration rate is close equal to 0.0025, the model's training period takes place and illustrates the repeatability of the policy's performance.

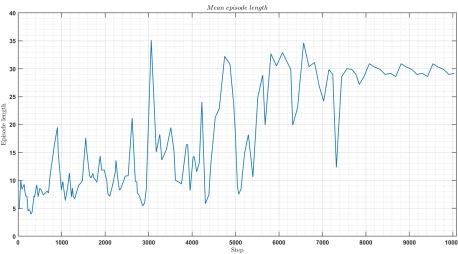


Fig. 5. The mean episode length curve during the training

Furthermore, as depicted in Figure 6, the policy aims to optimize the sequence of actions in order to attain the highest possible global reward. This is achieved by leveraging observations of the UAV's movement and selected actions, aligning with the anticipated outcome resulting from the implementation of the Bellman equation.

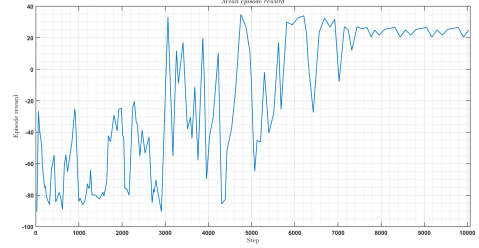


Fig. 6. The mean episode reward curve during the training

The exploration rate is the ratio between exploration and exploitation of the environment, and it has been set to start with a value of 1, which ensures exploration at the beginning of training and a slow decay to full exploitation at the 7500<sup>th</sup> timestep. Thanks to this approach, the policy has a substantial amount of time and a number of possible approaches to experiment with.

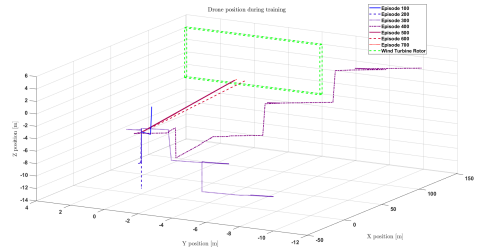


Fig. 7. Position of the drone during the training

This can be seen especially in Fig 7, where trajectories flown by the drone can be seen. The plot shows paths taken every 100<sup>th</sup> episode with a significant improvement after 500<sup>th</sup> episode, which is approximately the point where full exploitation has been reached. Last 300 episodes show the most optimized path, where the drone moved in a nearly straight line from its starting position and aligns itself in front of the wind turbine, which is marked with a green, dashed wireframe box.

#### V. CONCLUSION

This paper presents our work on utilizing a DQN policy for navigation optimization of a UAV in a virtual simulation environment, implemented with the Unreal engine and AirSim a realistic physics environment. A YOLOv5 network for object detection was trained using transfer learning and fine-tuned with a series of virtual images of wind turbines. UAV's performance in the environment has been evaluated by a dynamic reward function, which is a weighted sum of two sub-function that evaluated different objectives. Integration of both systems

showcased the possibility of training the UAV through the bounding box pixels obtained by the object detection network. Our results showed improvements in the average length of the episode and the total reward. Additionally, plotting the position of the UAV inside the simulation environment showed that with a trained network, the optimal trajectory between the start point and the wind turbine's rotor has been found. Future work will be focused on optimizing and improving the current approach, while implementing more realistic conditions in the reward function, such as energy consumption and including object detection confidence in relation to the visibility level inside the environment.

#### REFERENCE LIST

- [1] Anwar Sahbel, Ayman Abbas, and Tariq Sattar. "System design and implementation of wall climbing robot for wind turbine blade inspection". In: *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*. IEEE. 2019, pp. 242–247.
- [2] Haowen Guo et al. "Detecting and positioning of wind turbine blade tips for uav-based automatic inspection". In: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2019, pp. 1374–1377.
- [3] Andrius Kulsinskas, Petar Durdevic, and Daniel Ortiz-Arroyo. "Internal wind turbine blade inspections using UAVs: Analysis and design issues". In: *Energies* 14.2 (2021), p. 294.
- [4] GE Renewable Energy. *The onshore wind power industry*. 2022.
- [5] Ben DuBose. *New Rotor Blade Inspection Methods for Offshore Wind Turbine*. 2020.
- [6] Next Era Energy Resources. *Wind energy and safety*.
- [7] FlyAbility. *Wind Turbine Inspection: A guide*.
- [8] Michael Blösch et al. "Vision based MAV navigation in unknown and unstructured environments". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 21–28.
- [9] HU Zijian et al. "Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments". In: *Chinese Journal of Aeronautics* 34.12 (2021), pp. 187–204.
- [10] Daniel Ortiz-Arroyo. "A hybrid 3D path planning method for UAVs". In: *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2015, pp. 123–132.
- [11] Daeil Lee et al. "Comparison of deep reinforcement learning and PID controllers for automatic cold shut-down operation". In: *Energies* (2022).
- [12] Ahmad Taher Azar et al. "Drone deep reinforcement learning: A review". In: *Electronics* 10.9 (2021), p. 999.
- [13] Ziyu Wang et al. "Dueling network architectures for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1995–2003.
- [14] Jin-Hyeok Park et al. "Deep reinforcement learning-based DQN agent algorithm for visual object tracking in a virtual environmental simulation". In: *Applied Sciences* 12.7 (2022), p. 3220.
- [15] Mirco Theile et al. "UAV path planning using global and local map information with deep reinforcement learning". In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE. 2021, pp. 539–546.
- [16] Yibing Li et al. "A UAV path planning method based on deep reinforcement learning". In: *2020 IEEE USNC-CNC-URSI North American Radio Science Meeting (Joint with AP-S Symposium)*. IEEE. 2020, pp. 93–94.
- [17] Weichao Qiu et al. "Unrealcv: Virtual worlds for computer vision". In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 1221–1224.
- [18] Xueqin Huang et al. "Autonomous Multi-View Navigation via Deep Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13798–13804.
- [19] Yun Chen, Nuria González-Prelcic, and Robert W Heath. "Collision-free UAV navigation with a monocular camera using deep reinforcement learning". In: *30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2020, pp. 1–6.
- [20] A Tzimas, Nikolaos Passalis, and Anastasios Tefas. "Leveraging deep reinforcement learning for active shooting under open-world setting". In: *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2020, pp. 1–6.
- [21] Jialin Hao, Rola Naja, and Djamal Zeghlache. "Drone-Assisted Lane Change Maneuver using Reinforcement Learning with Dynamic Reward Function". In: *2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2022, pp. 314–320.
- [22] Mohamad Albilani and Amel Bouzeghoub. "Dynamic Adjustment of Reward Function for Proximal Policy Optimization with Imitation Learning: Application to Automated Parking Systems". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 1400–1408.
- [23] Harm Van Seijen et al. "Hybrid reward architecture for reinforcement learning". In: *Advances in Neural Information Processing Systems* 30 (2017).
- [24] Jiexin Wang, Stefan Elfving, and Eiji Uchibe. "Deep reinforcement learning by parallelizing reward and punishment using the maxpain architecture". In: *2018 Joint IEEE 8th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE. 2018, pp. 175–180.
- [25] Microsoft. *AirSim's Documentation*.
- [26] Marko Horvat and Gordan Gledec. "A comparative study of YOLOv5 models performance for image localization and classification". In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2022, pp. 349–356.