

Towards a Fully Autonomous UAV Controller for Moving Platform Detection and Landing

Michalis Piponidis, Panayiotis Aristodemou and Theocharis Theocharides

KIOS Research and Innovation Center of Excellence

Department of Electrical and Computer Engineering

University of Cyprus

Nicosia, Cyprus

{mpipon01,parist02,ttheocharides}@ucy.ac.cy

Abstract—While Unmanned Aerial Vehicles (UAVs) are increasingly deployed in several missions, their inability of reliable and consistent autonomous landing poses a major setback for deploying such systems truly autonomously. In this paper we present an autonomous UAV landing system for landing on a moving platform. In contrast to existing attempts, the proposed system relies only on the camera sensor, and has been designed as lightweight as possible. The proposed system can be deployed on a low power platform as part of the drone payload, whilst being indifferent to any external communication or any other sensors. The system relies on a Neural Network (NN) based controller, for which a target and environment agnostic simulator was created, used in training and testing of the proposed system, via Reinforcement Learning (RL) and Proximal Policy Optimization (PPO) to optimally control and steer the drone towards landing on the target. Through real-world testing, the system was evaluated with an average deviation of 15cm from the center of the target, for 40 landing attempts.

Index Terms—UAVs, drone, autonomous landing, machine learning, neural networks, computer vision, object detection

I. INTRODUCTION

With the continuous growth of Unmanned Aerial Vehicles (UAVs) usage in many applications, their ability to be as autonomous as possible is becoming a primary objective. Multi-rotor UAVs especially are extremely useful due to their high maneuverability in various terrains and are being adopted in numerous missions such as search and rescue [4], military and tracking operations [5] and many more. Autonomous UAV landing, especially on a moving target is one of the most vital and simultaneously challenging operations of a UAV to achieve full autonomy. Research in this area is still lacking and human operators are still necessary which limits the countless possibilities of UAVs, that range from rapid recovery of a fleet [6] to the utilization of mobile recharging stations [7] where UAVs can autonomously recharge during missions.

In the past few years, UAVs are becoming eminently dependent on external communication signals, namely wireless communication and Global Positioning Systems (GPS). While such systems provide UAVs with extremely reliable and safe operational abilities, they become vulnerable to various attacks

such as signal jamming and spoofing, which can cause poor and false readings or even accidents [1]. Additionally, poor GPS signal can cause unacceptable amounts of deviation reaching up to 4m which eliminates the likelihood of a reliable system that can be used repeatably without any problems [3]. Furthermore, in search and rescue operations, and emergency response scenarios, where UAVs have been adopted to provide first responders with extremely valuable and vital information, the availability of communication networks or GPS may not be readily available. Near-field communication (NFC) that has also been proposed as a means of communication between the UAV and the landing platform is impractical in such scenarios, as the UAV still has to detect and move towards the platform for NFC to be effective.

In this paper, we therefore propose, a lightweight, real-time, neural-network based landing controller that relies only on its own sensors and controls the UAV towards a moving target and autonomously lands it. The contribution of this work is a complete system designed and optimized from the very beginning to be deployed on a UAV. The system is evaluated on an NVIDIA® Jetson Xavier™ NX embedded platform and a DJI Mavic Air UAV. In particular, this work advances the state of the art in the following ways:

- The proposed system does not require any external communication signals between the UAV and the target, as it relies only on the on-board UAV camera sensor.
- The controller utilizes an accurate target detector and extracts a positional relationship between the UAV and the target platform.
- We developed a target- and environment-agnostic simulator to generate training data, used to train the controller under different conditions, in a consistent and fast manner.
- The controller is based on a Neural Network (NN), trained using Reinforcement Learning (RL) that uses the positional relationship between the target and the UAV as its input, and accordingly outputs the optimal UAV control signals. Further, we perform design exploration and optimization to reduce its size and latency while keeping the accuracy at desirable levels.
- We evaluated and validated the proposed controller,

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

through a set of real-world experimental scenarios that yielded a 15cm average deviation from the center of the platform for 40 landing experiments. The controller's resource requirements were also validated by deploying it on an embedded platform and extracting performance metrics.

The rest of the paper is structured as follows: Section II discusses the related work that has been done on this subject. Section III explains the modules developed in the proposed framework. Section IV describes the experimental platform that is used for evaluating the system and shows the experimental results. Finally, in Section V we conclude our work and discuss possible future work.

II. RELATED WORK

UAV autonomous landing systems has been a hot topic for researchers, especially since the advancements in the UAV industry that happened in the last decade. Initial works focused on detecting the landing platform, based on traditional digital image processing methods. Xiaoyun et al. [15] designed a color marker platform that gets detected during the landing process by exploiting its colors (red, green and blue), which makes it easy to detect by the combined approach of vanishing point of parallel lines and Levenberg-Marquardt (L-M) optimization method. A similar approach is shown in [35], where the target is detected using adaptive thresholding. Ramos et al. [16] use a landing platform with the Helipad design and using quadrilateral object detection with a Feature Extraction module based on Histogram of Oriented Gradients (HOG) and a Supervised Learning Classifier are able to detect it reliably. Demirhan et al. [22] exploit the contour properties of the Helipad design to detect the platform. Although this approach has very good performance, it can only reliably detect the platform from a maximum altitude of 165cm, which makes it impractical for typical real-world applications. Many research works use the UAV's GPS to assist the landing procedure [32] [15] [33]. The GPS accuracy usually ranges between 2-3m [31], which, depending on the size of the landing platform, may require additional information or human control for reliable landing. The usage of Aruco markers [23] is another popular choice for detecting the landing platform, as seen in [24] [25] [26] [36]. Although they are very accurate and efficient in low altitudes where the marker is clearly visible, when the UAV is higher ($>10\text{m}$), they are harder to detect as we saw from our experiments. This is caused by the combination of their complex design and low resolution. A different approach is presented in [34], where the system is able to ensure the UAV pose estimation at different altitude levels by a coarse-to-fine approach, detecting a different part of the platform at each level. A mathematical approach based on the inclination angle and state of the UAV is also presented in [18], however it was only evaluated by simulation and the platform is assumed to move at a constant velocity.

The use of Machine Learning (ML) in designing landing controllers for multi-rotor UAVs, is also relatively new, as such controllers are typically complex and resource-constrained,

energy-aware NN-based controllers only recently gained popularity through various edge-based optimizations [30]. Thus, RL based controllers, that use Least Square Policy Iteration (LSPI) to learn the optimal UAV control policies have been proposed [24] [27], all of which however use static landing platforms. Moreover, Ramos et al. [17] demonstrate a Deep Deterministic Policy Gradients (DDPG) RL algorithm that can achieve really good accuracy when the positions of the UAV and the moving platform are known. These works prove that RL based controllers are one of the best approaches for solving this problem.

Motivated by the need of a fully autonomous system, in contrast to existing works, we design our controller to use only on-board sensing and computational resources, targeting landing on a moving platform, without constraints on the platform's location or movement patterns (i.e. the platform can move freely with variable direction and/or speed). It is also important to note that the evaluation methods and metrics are not standardized yet - this is due to the fact that these systems are fairly complex, so they can be evaluated using many different metrics. The most descriptive and complete comparison scheme we found is shown in [25], which compares state of the art frameworks in regards of their *Accuracy* (i.e. the average deviation from the center of the platform), the UAV's *Landing Speed*, the *Maximum Altitude* from which it can work and whether it works with a *Moving Platform* and *Outdoors*. We thus use these metrics to evaluate our proposed controller.

III. METHODOLOGY

Our system consists of the following modules:

- *Object detector*: reliable and fast detector in order to detect the position of the landing platform in real-time.
- *Target positioning*: extracts various information providing the positional relationship between the target and UAV.
- *Simulator*: virtual environment where a UAV can be simulated, generating the controller's training and evaluation data.
- *RL based controller*: trained NN for taking decisions about the UAV movement, based on the positional information.

Each of these modules was designed targeting a resource-constrained embedded implementation requirements. We aim for the lowest possible resources such as memory footprint and ultra-low power consumption, while still being able to successfully complete the landing task with the required performance and accuracy.

A visual representation of our system landing process is shown in Fig.1, while the time and data flow between the modules can be seen in Fig.2. We partition the process in discrete time-steps. In each time-step, the process starts with the detector receiving the camera feed. The detector then finds the position of the platform in each frame, for which the information about the positional relationship between the target and UAV is extracted (Fig.1a). That information is sent to the controller, which outputs the optimal UAV movement

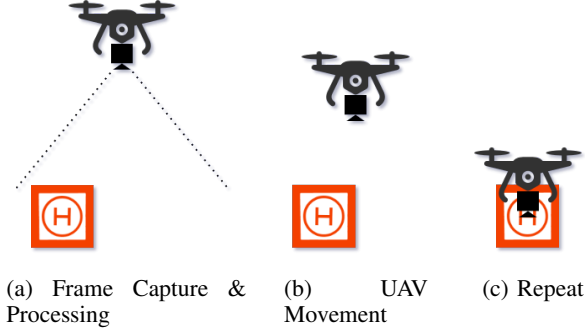


Fig. 1: System visual representation

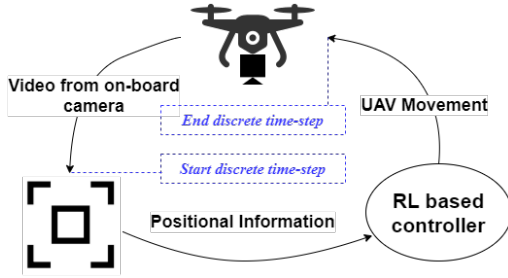


Fig. 2: System communication diagram

commands (Fig.1b). This process is repeated in every time-step, until the UAV lands on the platform (Fig.1c). The duration of each time-step is defined in processed Frames Per Second (FPS), encapsulating all actions as shown in Fig.2.

A. Target Detection and Parameter Estimation

We utilize a red-colored helipad style design as shown in Fig.1 for the targeted platform. The reasoning behind the chosen shape and color, is because it is simple and can be distinguished by the detector easier in higher altitudes. Furthermore, the color red is quite rare in nature meaning that the possibility of a false positive detection is significantly decreased.

Given the abundance of object detectors available, we decided to use a state of the art detector, optimized for UAVs, DroNet [2]. However, our system is vision-agnostic and any detection method (e.g. [19], [20], [21]) can be used, depending on the specific application requirements. Following up the time-step approach explained earlier, we only target 8FPS instead of the 30FPS that the camera provides, further reducing the computational strain. To determine how strict this requirement is, we provide a brief analysis next. Let us consider a hovering altitude of 5m; the UAV camera covers $\approx 9 \times 9$ m of the ground. In order for the moving platform to not be detected in successive frames, it would need to cover 9m in 0.1s. This implies a platform speed of $9 \times 8 = 72 \frac{m}{s}$, which is of course quite unnatural, and means that if needed, we can further reduce the processed FPS to lower the system requirements. Even more, UAVs typically fly at altitudes of

20m and above, which implies a much larger (in terms of surface area) view.

To extract information about the positional relationship between the target and the UAV, the detected bounding box was used. The bounding box is essentially the rectangle that surrounds the detected object as provided by the detector; it consists of four coordinates in the image: $xmin$, $ymin$, $xmax$ and $ymax$. To extract the information, the first step is to find the center of the bounding box as shown in (1).

$$\begin{aligned} x_c &= xmax - \frac{(xmax - xmin)}{2} \\ y_c &= ymax - \frac{(ymax - ymin)}{2} \end{aligned} \quad (1)$$

Where (x_c, y_c) is the center point of the bounding box. Afterwards, the distance of the center point to the image center, in each axis must be found using (2).

$$\begin{aligned} HPDist &= x_c - \frac{imageWidth}{2} \\ VPDist &= y_c - \frac{imageHeight}{2} \end{aligned} \quad (2)$$

Where $HPDist$, $VPDist$ are the horizontal and vertical pixel distances from the center point to the horizontal and vertical axis of the image respectively. $imageWidth$, $imageHeight$ are the image width and height in pixels respectively.

The first variable that is calculated is the Horizontal Angle, also known as Rotation Angle, which is the angle of the target with respect to the heading of the UAV (longitudinal axis). To compute the Horizontal Angle, (3) is used.

$$\theta = \frac{HFOV \cdot HPDist}{imageWidth} \quad (3)$$

Where θ is the Horizontal Angle and $HFOV$ is the Horizontal Field of View of the camera in degrees. The key part of this variable is that depending on the sign, we know meticulously how many degrees the target is positioned right or left from the UAV.

The second variable is the Vertical Angle, which is the angle of the target with respect to the UAV's vertical axis. Calculating this variable involves a two step process. Firstly, we use a similar approach as the Horizontal Angle, as shown in (4).

$$\phi = \frac{VFOV \cdot VPDist}{imageHeight} \quad (4)$$

Where ϕ is the vertical angle from the UAV camera's gimbal heading in relation to the target and $VFOV$ is the Vertical Field of View of the camera in degrees. The sign of ϕ indicates if the target is above or below the gimbal's heading. Given the fact that the gimbal is not static, equation (5) is extracted.

$$\omega = \phi + \alpha \quad (5)$$

Where α is the pitch angle of the gimbal from the UAV's vertical axis and ω is the Vertical Angle. Vertical Angle is

used to dynamically change the gimbal of the UAV to keep the target at the center of the image on the vertical axis.

Another crucial variable is the distance from the UAV to the landing platform. Both Horizontal and Vertical angles are required for accurately computing the distance. By exploiting basic triangulation properties, (6) is derived.

$$d_1 = h \cdot \tan(\omega) \quad (6)$$

Where d_1 is distance from the UAV to the platform on the UAV's longitudinal axis and h is the height of the UAV. The next step for calculating the distance is (7).

$$d_2 = \frac{d_1}{\cos(\theta)} \quad (7)$$

Where d_2 is the direct distance from the UAV to the landing platform.

Finally, using (8) the platform speed can be calculated.

$$V_p = \left(\frac{D_2 - D_1}{T} \right) + V_d \quad (8)$$

Where D_2 is the current distance of the platform, D_1 is the previous distance, T the time between the two measured distances and V_d is the speed of the UAV.

B. Control Neural Network

We used Unity's ML-Agents Toolkit [10] to train the UAV controller NN. ML-Agents is a powerful framework that allows intelligent agents to learn through a combination of Deep RL and Imitation Learning. In particular, we used the ML-Agents RL framework with the Proximal Policy Optimization (PPO) algorithm [14].

The decision making NN receives the information extracted from the detector as the network input, also known as Observations in RL. These include the *Rotation Difference*, *Horizontal Distance* and *Vertical Distance*. These values were normalized because as shown by an ablation study [13], normalized observations contribute to better performance in PPO based RL. The action space has four UAV control outputs: pitch, roll, yaw and throttle. The pseudocode of the reward algorithm is shown in Algorithm 1. Negative rewards are larger than positive rewards in order to prevent unnecessary movement while the values have been chosen empirically after various simulations. Fig.3 shows that the training process progressed smoothly, with the final NN reliably following and landing on the target, using consistent and smooth movements.

The NN was optimized to reduce it from its original size of 78KB to 7KB, for a total of 91% size decrease. The computational complexity of the NN is also notably decreased, resulting in faster inference while keeping the desired performance, making it suitable for the intended on-board implementation. Due to space limitations, we omit the optimization details, however, we used standard pruning and quantization approaches.

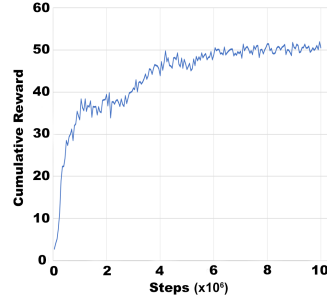


Fig. 3: RL Training Results

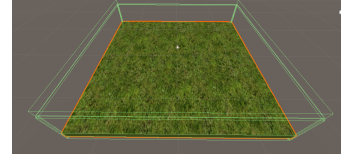


Fig. 4: Simulator

Algorithm 1 RL Rewards Pseudocode

```

if UAV turned  $10^\circ$  closer to the target then
    Give +0.1 Reward
else if UAV turned  $10^\circ$  further from the target then
    Give -0.3 Reward
end if
if UAV is within  $5^\circ$  from the target then
    Give +0.1 Reward
end if
if UAV moved horizontally 0.2m towards the target then
    Give +0.1 Reward
else if UAV moved horizontally 0.2m away from the target then
    Give -0.3 Reward
end if
if UAV moved vertically 0.2m towards the target then
    Give +0.1 Reward
else if UAV moved vertically 0.2m away from the target then
    Give -0.3 Reward
end if
if UAV entered the landing target then
    Give +15 Reward
end if
if UAV stayed in the landing target then
    Give +0.2 Reward
end if

```

C. Training Simulator

The simulator provides a controlled and customizable environment so that any environmental data can be used to train the controller, while also allowing the control of several parameters, such as the UAV and platform positions and velocities. Moreover, it provides a simulated environment where we can perform our training without the need of deploying an actual UAV, which can be time consuming, expensive and can result in accidents. The primary objective of the simulator is to serve as the environment where the RL based controller will be trained, via various scenarios as stated previously. This greatly improves the training process as very large numbers of varied training data can be easily generated, which results in better training and avoids over-fitting. The environment is simulated realistically based on real-world data, thus it can be assured that training data are valid and applicable to real-world situations. The simulator is built using the Unity game engine [9]. The virtual environment (Fig.4) consists of a large hollow rectangular prism as the level where the UAV can fly and a landing platform with adjustable movement parameters.

The virtual UAV [11] that was used provides a realistic model that moves by adding force to each propeller and uses PID controllers for stabilization.

IV. EXPERIMENTAL SETUP AND RESULTS

While the proposed controller is indented to be deployed directly on the UAV, for testing and troubleshooting purposes, it was running on a laptop, communicating with the UAV via a video streaming server and an interface built using Robotic Operating System (ROSTM) [12] for transmitting the controller's output signals back to the UAV. We detail our experimental framework next.

A. Experimental Setup

A labeled dataset consisting of 3419 real-world images in various altitudes, angles and lighting conditions (using a UAV capturing photos of the platform) was used to train the detector. We trained DroNet through Darknet [8], reaching 85.9% mean Average Precision (mAP). We observed extremely high accuracy (more than 95%) detecting the target for altitudes up to 20m, with greater altitudes also resulting in high accuracy as well (more than 80%).

We used the DJI Mavic Air UAV because it provides the necessary features required for the application and an easy to use Software Development Kit (SDK). Furthermore, it has a 3-axis camera gimbal utilized by the detector and positional relationship extraction. We kept the UAV velocity constant at $0.4 \frac{m}{s}$ for evaluation purposes. We also developed an Android application using the Mobile SDK, providing full control over the UAV's movement and gimbal angles, which we used to translate the movement commands issued by the controller NN into UAV movement commands, while the gimbal angle was adjusted accordingly after each time-step. We used ROS to communicate between the modules and a laptop with Intel[®] Core[™] i7-6500U 2.5GHz and 16GB DDR3 RAM running Ubuntu 20.04 LTS. The controller thus receives the UAV's camera stream, performs the target detection and positional extraction, which are then used by the NN controller, which outputs the control signals. These are then transmitted to the UAV using the ROS interface. Further, we built an 100x100cm landing platform consisting of the 60x60cm target in the middle and 20cm padding used as a safety measure, in case the UAV lands on the edge of the platform. The platform was moved using a programmable custom-build robotic platform that enabled us to create various speeds and movements and generate random paths on the ground.

B. Experimental Results

We used the DJI Mavic Air UAV, the controller (as described earlier) and the custom-built moving landing platform, and performed a total of 40 test runs starting from various altitudes and under variable landing platform speeds and directions. The results are shown in Table I. The evaluation was based strictly on the detector outputs, thus any associated limitations and inaccuracies of the vision algorithm are also considered. From the results we can see that 30 out of 40 runs landed inside the

TABLE I: Moving platform testing results

Runs	Average Distance from the center of the platform	Landings inside the target
1-5	20cm	3
6-10	4cm	5
11-15	35cm	2
16-20	10cm	4
21-25	21cm	3
26-30	8cm	5
31-35	15cm	3
36-40	12cm	5

60x60cm target, which yields an accuracy of 75%. Moreover, the average distances were measured from the center of the landing platform to the UAV at its landing position, yielding an average deviation of ≈ 15 cm.

Table II summarizes the results and compares them to related work shown in Section II. We can observe that our work is the only one that works with a moving target outdoors, which in itself is a major step forward towards deployment in practical applications. Also, our evaluation metrics are comparable to similar state of the art works. It should be taken into consideration that these metrics are based on the laptop implementation of the system. We anticipate that on-board deployment, will significantly improve performance due to the absence of the communication overheads between the camera and the controller. To verify this, we implemented the detector and the NN controller on an NVIDIA[®] Jetson Xavier[™] NX embedded platform, which we use as computational payload, to demonstrate its performance. The detector used ≈ 1.3 GB of RAM, while the controller used ≈ 450 MB of RAM, for a total of 25% of the available memory. The detector achieved an average speed of 20FPS (including the positional relationship information extraction) and the NN controller was able to produce the optimal control signals within 1ms on average. Thus, we can achieve the targeted 8FPS as detailed earlier.

V. CONCLUSIONS

We presented an autonomous UAV landing controller that relies only on visual information, to control the UAV towards landing on a moving platform. A target- and environment-agnostic simulator was created and used for training the RL based controller which is responsible for the movements of the UAV. The whole system can achieve speeds of 8FPS, with relatively high accuracy under various environmental conditions, platform speeds and motion patterns. We plan to integrate the controller directly on the UAV by using an embedded board such as NVIDIA[®] Jetson or similar. Further, we plan on introducing adaptive speed control for the UAV, based on the platform's speed and motion, resulting in faster landings. Also, GPS assistive data can be used to improve the accuracy of the algorithm in higher altitudes. The accuracy and trustworthiness of the observations provided by the camera can

TABLE II: Comparison between different frameworks

Framework	Accuracy (m)	Landing Speed (m/s)	Maximum Altitude (m)	Moving Target	Outdoor
Ours	0.15	0.4	20	Yes	Yes
Wubben et al. [25]	0.11	0.3	30	No	Yes
GPS-based	1 – 3	0.6	∞	No	Yes
Chen et al. [35]	n/a	0.23	2.5	Yes	No
Araar et al. [36]	0.13	0.06	0.8	Yes	No
Patruno et al. [34]	0.01	n/a	n/a	No	Yes

also be quantified in order to improve the control algorithm, as shown in [28], [29]. Finally, part of the future work is to expand our experimental deployments of the system in non-ideal conditions as well.

REFERENCES

- [1] J. Aru Saputro et al., "Implementation of GPS Attacks on DJI Phantom 3 Standard Drone as a Security Vulnerability Test," 2020 1st International Conference on Information Technology, Advanced Mechanical and Electrical Engineering (ICITAMEE), 2020, pp. 95-100, doi: 10.1109/ICITAMEE50454.2020.9398386.
- [2] Kyrkou, Christos et al. (2018). DroNet: Efficient convolutional neural network detector for real-time UAV applications. 967-972. 10.23919/DATE.2018.8342149.
- [3] J. Janousek and P. Marcon, "Precision landing options in unmanned aerial vehicles," 2018 International Interdisciplinary PhD Workshop (IIPHDW), 2018, pp. 58-60, doi: 10.1109/IIPHDW.2018.8388325.
- [4] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," 2010 International Conference on Emerging Security Technologies, 2010, pp. 142-147, doi: 10.1109/EST.2010.31.
- [5] M. A. Ma'sum et al., "Simulation of intelligent Unmanned Aerial Vehicle (UAV) For military surveillance," 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2013, pp. 161-166, doi: 10.1109/ICACSIS.2013.6761569.
- [6] G. Kim et al., "Self-recovery scheme using neighbor information for multi-drone ad hoc networks," 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017, pp. 1-5, doi: 10.23919/APCC.2017.8303997.
- [7] L. Angrisani et al., "Autonomous recharge of drones through an induction based power transfer system," 2015 IEEE International Workshop on Measurements & Networking (M&N), 2015, pp. 1-6
- [8] J. Redmon, "Darknet: Open source neural networks in c," <http://pjreddie.com/darknet/>, 2013–2016.
- [9] Unity Technologies. Unity Real-Time Development Platform. [Online]. Available: <https://unity.com>
- [10] Unity Technologies. Unity Machine Learning Agents. [Online]. Available: <https://unity.com/products/machine-learning-agents>
- [11] S. Radivoev, "simeonradivoev/quadcopter-controller," Oct 2018. [Online]. Available: <https://github.com/simeonradivoev/Quadcopter-Controller>
- [12] Open Source Robotics Foundation. About ROS. [Online]. Available: <https://www.ros.org/about-ros/>
- [13] L. Engstrom et al., "Implementation matters in deep rl: A case study on ppo and trpo," in International Conference on Learning Representations, 2020. [Online]. Available: <https://openreview.net/forum?id=r1etN1rtPB>
- [14] OpenAI. Proximal Policy Optimization. [Online]. Available: <https://openai.com/blog/openai-baselines-ppo/>
- [15] Xiaoyun Huang et al., "Vision-based Autonomous Landing of UAV on Moving Platform using a New Marker," 2019 IOP Conf. Ser.: Mater. Sci. Eng. 646 012062.
- [16] Rodríguez Ramos, Alejandro et al., (2017). Towards fully autonomous landing on moving platforms for rotary Unmanned Aerial Vehicles. 170-178. 10.1109/ICUAS.2017.7991438.
- [17] Rodríguez Ramos, Alejandro et al., (2019). A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform. Journal of Intelligent & Robotic Systems. 93. 10.1007/s10846-018-0891-8.
- [18] M. Alijani and A. Osman, "Autonomous Landing of UAV on Moving Platform: A Mathematical Approach," 2020 International Conference on Control, Automation and Diagnosis (ICCAD), 2020, pp. 1-6
- [19] F. Ö. Ünel et al., "The Power of Tiling for Small Object Detection," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 582-591, doi: 10.1109/CVPRW.2019.00084.
- [20] Zicong Jiang et al., & Yanfei Jia. (2020). Real-time object detection method based on improved YOLOv4-tiny.
- [21] M. Zhang et al., "Vision-Assisted Landing Method for Unmanned Powered Parachute Vehicle Based on Lightweight Neural Network," in IEEE Access, vol. 9, pp. 130981-130989, 2021
- [22] M. Demirhan and C. Premachandra, "Development of an Automated Camera-Based Drone Landing System," in IEEE Access, vol. 8, pp. 202111-202121, 2020, doi: 10.1109/ACCESS.2020.3034948.
- [23] S. Garrido-Jurado et al., Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6):2280 – 2292, 2014.
- [24] M. B. Vankadari et al., "A Reinforcement Learning Approach for Autonomous Control and Landing of a Quadrotor," 2018 International Conference on Unmanned Aircraft Systems (ICUAS), 2018, pp. 676-683, doi: 10.1109/ICUAS.2018.8453468.
- [25] J. Wubben et al., "Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition," Electronics, vol. 8, no. 12, p. 1532, Dec. 2019.
- [26] Tianqu Zhao and Hong Jiang, "Landing system for AR.Drone 2.0 using onboard camera and ROS," 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), 2016, pp. 1098-1102
- [27] M. Shaker et al., "Vision-Based Landing of a Simulated Unmanned Aerial Vehicle with Fast Reinforcement Learning," 2010 International Conference on Emerging Security Technologies, 2010 pp. 183-188, doi: 10.1109/EST.2010.14.
- [28] Cheng, Mingxi et al., "A General Trust Framework for Multi-Agent Systems." AAMAS (2021).
- [29] Cheng, Mingxi et al., (2020). There Is Hope After All: Quantifying Opinion and Trustworthiness in Neural Networks. Frontiers in Artificial Intelligence. 3. 54. 10.3389/frai.2020.00054.
- [30] Jingyu He et al., 2021. A Design Methodology for Energy-Aware Processing in Unmanned Aerial Vehicles. ACM Trans. Des. Autom. Electron. Syst. 27, 1, Article 4 (September 2021), 20 pages
- [31] J. Janousek and P. Marcon, "Precision landing options in unmanned aerial vehicles," 2018 International Interdisciplinary PhD Workshop (IIPHDW), 2018, pp. 58-60, doi: 10.1109/IIPHDW.2018.8388325.
- [32] J. Ghommam and M. Saad, "Autonomous Landing of a Quadrotor on a Moving Platform," in IEEE Transactions on Aerospace and Electronic Systems, vol. 53, no. 3, pp. 1504-1519, June 2017
- [33] Guo Liang, Goh et al., (2019). Outdoor Autonomous Landing of a Quadcopter on a Moving Platform using Off-board Computer Vision. Journal of Modeling and Optimization. 11. 86-96. 10.32732/jmo.2019.11.2.86.
- [34] Patruno, Cosimo et al., (2019). A Vision-Based Approach for Unmanned Aerial Vehicle Landing. Journal of Intelligent & Robotic Systems. 95. 10.1007/s10846-018-0933-2.
- [35] X. Chen et al., "System integration of a vision-guided UAV for autonomous landing on moving platform," 2016 12th IEEE International Conference on Control and Automation (ICCA), 2016, pp. 761-766
- [36] Araar, O. et al., I. Vision Based Autonomous Landing of Multirotor UAV on Moving Platform. J Intell Robot Syst 85, 369–384 (2017), <https://doi.org/10.1007/s10846-016-0399-z>