Performance Analysis of Deep Reinforcement Learning based Object Detection and Tracking from a UAV

Aliasgher Mohammed* Muhammad Ibrahim*, Wajih Hassan Raza*, Muhammad Moiz*, Hafsa Iqbal*
*School of Electrical Engineering and Computer Science (SEECS),
National University of Sciences & Technology (NUST), Islamabad, Pakistan
Email: {amohammed.bee20seecs, miibrahim.bee20seecs, wraza.bee20seecs, mmoiz.bee20seecs, hafsa.iqbal }@seecs.edu.pk

***Abstract*—This research conducts a performance analysis of multiple Deep Reinforcement Learning (DRL) algorithms for image based visual servoing (IBVS) controller for unmanned aerial vehicles (UAVs). Our porposed methods compares that performance of each model on the computed bounding-box error and linear-velocity between the UAV's position and the target object. For object detection, Yolov2 algorithm is utilized to identify the target. The computer bounding box is then fed to the selected DRL algorithm for the visual servoing task. We perform our simulation on the Gazebo environment, a robust platform for testing and development of robotic applications. Our analysis show that Prioritized Experience Replay-Deep Deterministic Policy Gradient (PER-DDPG) provides the fastest convergence on the errors in the considered environment. Our findings contribute to the advancement of DRL-based IBVS systems, highlighting the potential for improved real-time control and accuracy in UAV operations. The results underscore the importance of selecting optimal DRL algorithms to enhance UAV performance in dynamic and complex environments.**

***Index Terms*—Object Detection, Object Tracking, Deep Reinforcement Learning (DRL), Unmanned Aerial Vehicle (UAV)**

I. Introduction

Recent advancements in the field of vision sensor has played a key role in its usage with Unmanned Aerial Vehicles (UAVs). The data acquired through the sensors integrated with high computational power available using advanced processing techniques leads to a variety of applications such as object detection, localization, tracking and surveillance etc. Among these applications, visual servoing is an active and emerging resarch area in this field.

Visual servoing is a technique that uses visual data from the sensor to control the motion of a robot in a real-time environment. It is classified in two main categories i.e., image based visual servoing (IBVS) and position based visual servoing (PBVS). In IBVS systems, the acquired image features are used to control robot movement, whereas, in PBVS systems, the 3D position of target object is feed to control the robot actions. The main thing in both the approaches is to first detect target object in the environment. The primary difficulties in object detection within dynamic environments include: 1) intricate movements of the target in three-dimensional space, 2) varied shapes appearing against either the ground or sky, which results in complicated backgrounds, 3) rapid movement that leads to blurring and variations in lighting, among others.

The dynamic object detection and object tracking in video streaming studied in [1] highlights various factors brightness variations, blockages, shape deformations, rapid movements of either the object or camera, and resemblance to the background complicate the detection-based tracking.

The existing research on IBVS is mainly based on the classical visual servoing in which the interaction matrix is determined [2]. This approach suffers from estimation errors and lack of image features in unstable environments that leads to difficulty in visual servoing.

The emphasis of our system model is on detecting and tracking humans through visual servoing to maintain the targeted individual at the center of the image plane. This method is used to track the target human in real-time video streaming. For the initial object detection phase, Yolov2 is selected and trained because it is computationally efficient. After object detection, the next phase is to track the human in real-time. A reinforcement learning agent is used to perform IBVS on aerial drones for tracking purposes. The controller of drone is trained using deep deterministic policy gradient algorithm. The communication link between the script and drone controller is made possible by Mavlink software. This system model is tested on a quadrotor drone in Gazebo.

The organization of the paper is as follows, section II discusses the related work. Section III provides preliminary data related to this topic. In section IV, three deep reinforcement learning models that are implemented in this paper are explained. Section V describes the combined system architecture of this article. Section VI provides the simulation result that are obtained using Gazebo simulation environment.

The main contributions of our work are:

- We performed the analysis of three deep reinforcement learning models on the system architecture to determine the best model out of them.
- The system model is tested by creating a simulation environment using Gazebo software.
- Our model successfully detects the target object and hovers the UAV through control commands for tracking.

II. Related Work

In recent years, RL based controllers have gained a lot of attention, due to the multiple advantages of utilizing the RL algorithms in dynamic environments. The RL algorithms do not require any human intervention and are optimal for solving any complicated targets in such environments.

Few of the recent research has focused on the use of RL based system models for drone applications, for applications like landing on platforms [3]. A lot of machine learning approaches have been utilized in locating useful features and classification for drone detection using radars [4], and vision sensors [5]. There are multiple drones that are being used for classification for drones against birds [6]. Multiple proposed methods like Faster RCNN with MDNET have been utilized as well [7]. Yolov2 classification for spatio-temporal image cube, obtained through the motion stabilization from the CNN-based regressors [8] and SURF based ordered minimum distance Bag-of-Words using an extended histogram [9]. It is important to note that training a deep learning model requires a large amount of training data, but the drone's vision data is quite different from the available ImageNet and COCO datasets. However, in recent years, many different datasets have become available with drone captured images, which have eased the applications [7].

There are other learning based approaches that have been proposed as well. For example, a Fuzzy Q-learning based Image Based Visual Servoing (IBVS) was proposed in [10], which utilizes servoing gain techniques with the help of fuzzing Q-learning algorithms, for both linear and angular motion of the drone. Similarly, another research in [11] proposed a solution which uses a VGG convolutional neural network for extracting visual features from an image, and them applied a Q iteration for comparing the relevant features that were obtained from the CNN model. Another research in [12] utilized the popular FlowNet CNN architecture to determine the transformation between the initial and the current camera frame, by utilizing the Position Based Visual Servoing (PBVS) architecture to track the target.

However, the most relevant work to the implementation that we are pursuing is the research conducted in [13], [14]. In these work, the authors employ a deep deterministic policy gradient (DDPG) algorithm for learning environment policies in the Gazebo simulation environment through ROS (Robot Operating System). The authors show that the model can learn the optimal policy through the simulations, and perform the visual servoing tasks aimed. All the learned policies were implemented on the available intelligent quadcopter robot in the environment setting.

III. Preliminaries

A. Object Detection

We chose the computationally efficient model, YOLOv2, to locate and detect a person due to resource constraints. YOLOv2 comprises 24 convolutional layers and 2 fully connected layers. The drone camera's input image of the person undergoes processing through these convolutional layers to extract image features. Subsequently, leveraging these extracted features, the model predicts the person's location and provides us with the bounding box coordinates.

B. Reinforcement Learning

Reinforcement Learning (RL) is a technique of machine learning that learns the optimal behavior by interacting and getting feedback from the environment to obtain maximum reward. Unlike supervised learning, it does not require a large amount of data to train, it can interact with the environment and optimize by analyzing the feedback response to get the desired results. In RL an agent explores an unfamiliar environment to obtain the desired results. It operates on the idea of maximizing cumulative reward. The agent learns to observe and behave according to the environment's state to maximize reward. This approach is rooted in the control problem of Markov Decision Process (MDP). The major components of RL include action, states, reward, and environment.

In our case, the drone controller acts as an agent that performs control actions a_t and receives the scalar reward r_t. The agent performs action based on policy π: $s_t \rightarrow a_t$. The main goal of a drone will be to maximize reward which means minimizing the error between the image center and object center. In many reinforcement learning approaches, it is accomplished using the action value function $Q(s_t, a_t)$. This function estimates how good it is to take a certain action when in a certain situation.

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim P, a_t \sim \pi}[R_t | s_t, a_t] \tag{1}$$

We obtain the action value function using Bellman recursive equation as stated below:

$$\begin{aligned} Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim P}[r(s_t, a_t) \\ + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]] \end{aligned} \tag{2}$$

IV. DRL Approaches

This section presents the frontiers of the applied DRL algorithms and discusses the process of stating a problem statement within the DRL framework

1) *Deep Deterministic Polic Gradient (DDPG):* DDPG algorithm determines optimal actions *a* in states *s* by maximizing the action-value function *Q(s,a)* as *a*(s)* = argmax *Q(s,a)*. Unlike Q-learning and SARSA, which use tabulated values, DDPG employs neural networks for prediction. Suited for continuous action spaces, DDPG's actor-critic technique, along with a replay buffer, enhances agent training by randomly selecting past experiences. The deterministic policy μ can be written as

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim P}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, a_{t+1})] \tag{3}$$

To obtain optimal policy and action value function, DDPG uses an off policy learning approach which includes actor-critic architecture. The actor and critic are embodied by neural networks and show deterministic policy and action value functions respectively. The loss function used to update the critic network is as follows:

$$L(\theta^Q) = \mathbb{E}_{s_t, a_t \sim \beta, r_t}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \quad (4)$$

where the term $(Q(s_t, a_t|\theta^Q)$ denotes the approximation of the value function $Q(s_t, a_t)$ with weights θ^Q that can be adjusted based on the environment and y_t is the target value which is in our case the center point of the person whom we are detecting using YOLOv2. It can be defined as

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q) \quad (5)$$

2) *Prioritized Experience Replay (PER)-DDPG:* The PER algorithm, initially associated with the Deep Q-network (DQN) algorithm, has gained substantial traction due to its superior efficacy compared to traditional replay buffers. Now commonly paired with the DDPG algorithm, its innovation lies in assigning priorities based on temporal difference (TD) errors for each experience. This prioritization boosts the likelihood of sampling crucial instances, accelerating training and enhancing overall performance of the conventional DDPG algorithm.
Traditionally, experiences are randomly selected from the replay memory and equal importance is given to each memory. However, in PER we can adjust the priority of the experiences as some memories might be more important than others. Consequently, the significance of transitions is evaluated for each experience. During the replay phase, experiences are sampled from the replay memory using a specific strategy. However, it is crucial to prevent the risk of losing diversity and introducing bias towards certain transitions.
To overcome this issue [] introduced stochastic prioritization. It measures the transitions based on the magnitude of the TD-error, denoted as TD-err$_i$. Each transition i is then sampled with a probability

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad (6)$$

where α determines the level of prioritization, and in our experiment, we set $\alpha = 0$. When a new transition is added to memory with maximum priority, it encourages training with fresh transitions. Subsequently, for the sampled transitions, their priorities are updated based on the TD error during the training period.

3) *Twin Delayed DDPG (TD3):* The TD3 algorithm is an advanced version of the DDPG, strategically designed to address overestimation biases in Q-values for improved optimization. The algorithm iteratively updates both policy and Q-function, employing twin Q-networks. The idea of using two critical Q-networks and taking the minimum of their predicted Q-values during the learning process helps reduce the potential for overestimation and improves the stability of the algorithm.
Two Q-functions Q_{ϕ_1} and Q_{ϕ_2} are concurrently. It learns by minimizing the mean square Bellman error similar to DDPG. In TD3, the actions used to construct the Q-learning target are derived from the target policy, $\mu_{\theta_{\text{targ}}}$, but with clipped noise introduced to each dimension of the action. Following the addition of clipped noise, the target action is bounded to fall within the valid action range (where all valid actions, a, satisfy $a_{\text{Low}} \leq a \leq a_{\text{High}}$). Consequently, the target actions are as follows:

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right) \quad (7)$$

V. System Architecture

In this section, we discuss the deep learning-based vision servoing drone architecture for detecting and tracking a target.

A. Object Detection Algorithm:

The drone's camera captures images of the survey area, which are processed by the YOLOv2. While it may predict multiple target objects in the area, we will prioritize the target object with the highest probability. The output is in the form $(Z_k^i = (x_k^i, y_k^i), w_k^i, h_k^i, p_k^i)$, representing the bounding box's top-left corner coordinates $((x_k^i, y_k^i))$, width (w_k^i), height (h_k^i), and detection probability (p_k^i). These parameters will be provided to the DRL controller for further processing as shown in Fig. 2.

B. DRL Controller

The center point of the target object's bounding box is computed and compared with the center point of the image. Errors in the x and y directions are obtained by calculating the differences between these points. These errors constitute the state space of the DRL controller, as described below.

$$s_t = [e_x, e_y, e_a, \dot{e}_x, \dot{e}_y, \dot{e}_a]^T \quad (8)$$

where e_x and e_y represent the errors computed in the pixel coordinates along the x and y axes as shown in Fig. 1, and e_a denotes the error magnitude between the target point and the image point. The corresponding rate terms are obtained by computing the backward difference of the error terms between the current state and the previous state.

The reward function is designed in a way that reduces the above defined error and keeps the center point of the object aligned with the center point of the image plane. The reward function is given as follows

$$\begin{aligned} r(s_t, a_t) = 1 - w_1 \left\| \left[e_x, e_y\right]_t \right\| + s_2 \left\| \left[\dot{e}_x, \dot{e}_y\right]_t \right\| \\ - w_3 \left|e_a\right|_t + s_4 \left|\dot{e}_a\right|_t - w_5 \left\| \left[a_x, a_y\right]_t \right\| \end{aligned} \quad (9)$$

where,

$$s_2 = \text{sign}\left(\left\| \left[e_x, e_y\right]_{t-1} \right\| - \left\| \left[e_x, e_y\right]_t \right\| \right) w_2$$

$$s_4 = \text{sign}\left(|e_a|_{t-1} - |e_a|_t \right) w_4$$

where w represents the scalar weights used to scale the corresponding error terms. To prevent the drone from keeping the object outside the surveying area, a reward of -100 was given.

It is very difficult to make a mathematical model of all the physical behavior of the environment. Moreover, in most of the drones, the inner loop of the controller is not often accessed. Therefore, we design an outer loop to cover all the physical parameters of the environment. We will train all the reinforcement learning algorithms as defined in the previous section using a straightforward point mass kinematic model, which provides reference trajectories for the drone's inner loop controller.

$$Y_{t+1} = Y_t + V_t dt \tag{10}$$

where Y_t represents the position at time instant t and V_t represents the input linear velocity given to the drone controller. Using these linear velocities, the drone's movement will be controlled in Gazebo.

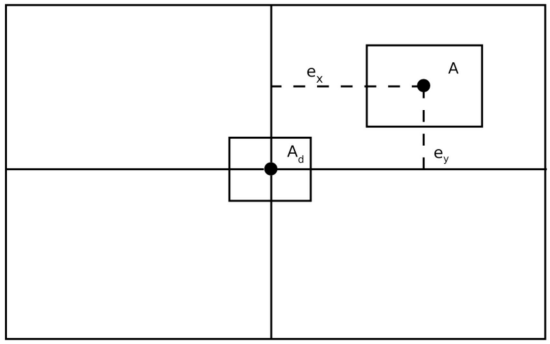

Fig. 1: Error Computation

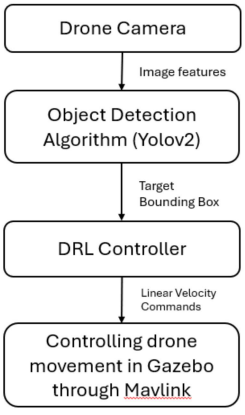

Fig. 2: System Architecture

VI. Results and Discussions

In this section, we evaluate the performance of our system architecture and compare the performances of the various DRL algorithms deployed. The trained actor model is utilized for

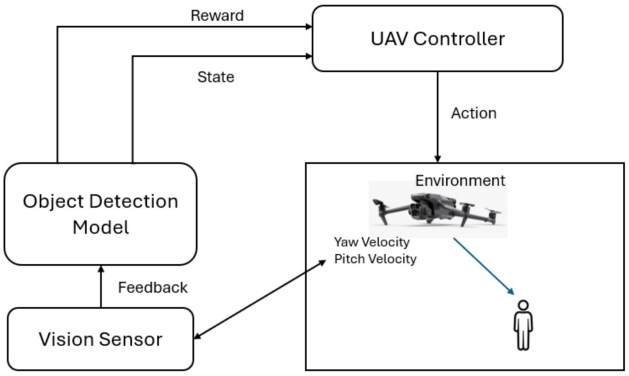

Fig. 3: DRL IBVS Controller

driving the drone in the Gazebo environment. The simulation was performed considering that the target was kept stationary at a fixed point. The tracker drone starts from a specific pixel location, and the bounding box locates the target while calculating the error in both the x and y directions. The normalised error with respect to the x position is $e_x := e_x/320$ and with respect to y position is $e_y := e_y/240$

Fig. 3. illustrates the variation of the normalized error for the UAV object tracking, when a trained DDPG algorithm is employed to actively track the human. In the result, the normalised error in the x-direction follows a sinusoidal approach with large variations at initial time. After time $t = 6s$, the normalised error starts to converge to a low value, and finally converges to a considerable value after $t = 9s$. The normalised error in the y-direction has a different convergence as it has a very high normalised error value (0.5) and has even slower convergence approach. As shown in the result, the error for y-axis starts to converge in a sinosoidal form after $t = 6s$ and has not converged to a considerable value even till $t = 9s$.

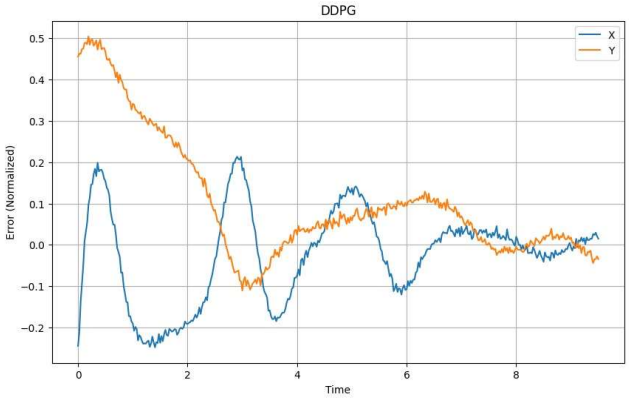

Fig. 4: Normalised Error rate vs time for DDPG algorithm

Fig. 4. illustrates the variation of the normalized error for the UAV object tracking, when the trained TD3 algorithm is employed to actively track the human. In the result, the normalised error in the x-direction follows a sinosoidal approach with more variations, but at a smaller value than DDPG, at the initial time. After time $t = 7s$, the normalised

error starts to converge to a low value and converges to an optimal value after $t = 9s$. The normalised error in the y-direction has a similar convergence to that of DDPG, but with much lower errors (starting error = 0.4) and has faster convergence. Considering the result, the graph converges to a low value of 0.02 after $t = 9s$.

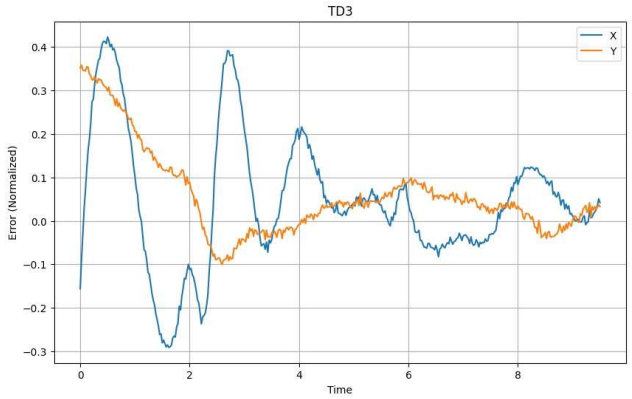

Fig. 5: Normalised Error rate vs time for TD3 algorithm

Fig. 5. illustrates the variation of the normalized error for the UAV object tracking, when the trained PER-DDPG algorithm is employed to actively track the human. The PER-DDPG shows much different result as after initial high fluctuations in normalized error in the x-direction till $t = 2s$, the algorithm shows fast convergence. At $t = 4s$, the normalized convergence has already converged to an optimal value, and continues to fluctuate around this value. Similarly, the normalised error in the y-direction also has fluctuations till $t = 3s$, after which it also converges to an optimal values, while having minor fluctuations after this time.

Considering, the results, we can easily conclude that PER-DDPG has the best performance among the all considered DRL algorithms for our analysis. PER-DDPG shows the fastest convergence rates and optimal performance in quick object detection and tracking, as compared to the other two algorithms. After PER-DDPG, TD3, and DDPG have similar performance, however, TD3 has slightly better performance considering that it has lower peak fluctuations and quicker convergence as compared to the basic DDPG performance model.

VII. Conclusion

In this research article, we measured the performance of multiple deep-reinforcement learning algorithms in visual servoing a target human. We successfully tested our system model by creating a simulation file using Gazebo software. The results showed that PER-DDPG has the best performance because it achieves the lowest normalized error rate in the least time. Our analysis also concludes that DDPPG has the slowest convergence rate, while TD3 DDPG shows some improvement. In the future, we can extend this work potentially to more complex environments, or perform analysis by conducting a real test flight on a drone.

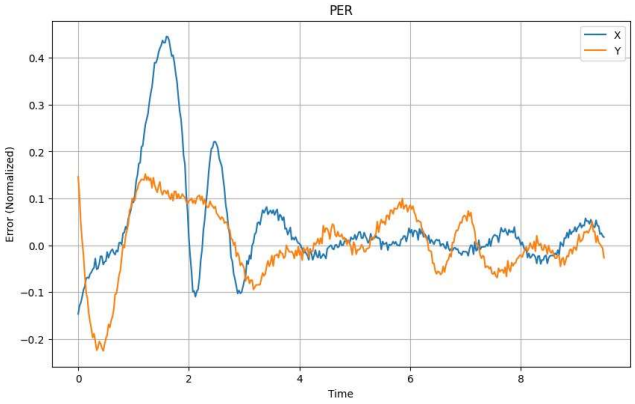

Fig. 6: Normalised Error rate vs time for PER-DDPG algorithm

References

[1] M. Yazdi and T. Bouwmans, "New trends on moving object detection in video images captured by a moving camera: A survey," *Computer science review*, vol. 28, pp. 157–177, 2018.

[2] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[3] A. Rodriguez-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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1010–1017.

[4] J. Kim, C. Park, J. Ahn, Y. Ko, J. Park, and J. C. Gallagher, "Real-time uav sound detection and analysis system," in *2017 IEEE Sensors Applications Symposium (SAS)*, 2017, pp. 1–5.

[5] G. J. Mendis, J. Wei, and A. Madanayake, "Deep learning cognitive radar for micro uas detection and classification," in *2017 Cognitive Communications for Aerospace Applications Workshop (CCAA)*, 2017, pp. 1–5.

[6] "Drones-vs-birds challenge with the international workshop on small-drone surveillance, detection and counteraction techniques," in *https://wosdetc.wordpress.com/challenge/,*, AVSS, 2017.

[7] Y. Chen, P. Aggarwal, J. Choi, and C.-C. J. Kuo, "A deep learning approach to drone monitoring," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017, pp. 686–691.

[8] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2017.

[9] E. Unlu, E. Zenou, and N. Riviere, "Ordered minimum distance bag-of-words approach for aerial object identification," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[10] H. Shi, X. Li, K.-S. Hwang, W. Pan, and G. Xu, "Decoupled visual servoing with fuzzy q-learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 241–252, 2018.

[11] A. X. Lee, S. Levine, and P. Abbeel, "Learning visual servoing with deep features and fitted q-iteration," *arXiv preprint arXiv:1703.11000*, 2017.

[12] A. Saxena, H. Pandya, G. Kumar, A. Gaud, and K. M. Krishna, "Exploring convolutional networks for end-to-end visual servoing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3817–3823.

[13] A. Rodriguez-Ramos, C. Sampedro, H. Bavle, P. De La Puente, and P. Campoy, "A deep reinforcement learning strategy for uav autonomous landing on a moving platform," *Journal of Intelligent & Robotic Systems*, vol. 93, pp. 351–366, 2019.

[14] C. Sampedro, A. Rodriguez-Ramos, I. Gil, L. Mejias, and P. Campoy, "Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 979–986.