A Deep Reinforcement Learning Visual Servoing Control Strategy for Target Tracking Using a Multirotor UAV

Andreas Mitakidis[1], Sotirios N. Aspragkathos[1], Fotis Panetsos[1], George C. Karras[2,1] and Kostas J. Kyriakopoulos[1]

***Abstract*—In this work, a deep Reinforcement Learning control scheme is developed in order to execute autonomous tracking of an unmanned ground vehicle (UGV) with a multirotor UAV. The UAV is equipped with a downward looking camera and the detection of the target UGV is achieved through a convolutional neural network (CNN). The deep RL control policy is deployed as a cascade position controller, which commands the inner attitude controller of the autopilot, and achieves the following of the UGV despite the aggressive maneuvers of the target. The efficacy of the proposed framework is demonstrated via a set of outdoor experiments using an octocopter flying above the ground vehicle.**

***Index Terms*—target tracking, visual servoing, reinforcement learning**

I. Introduction

Regarding Unmanned Aerial Vehicles (UAVs), in recent years have been gaining more and more attentions while being used in agriculture, transportation, inspection, cinematography, security and many other applications. The motion flexibility and relatively lower cost compared to manned aircraft have made them a very popular platform for various applications. In particular, when autonomous UAVs are combined with artificial intelligence (AI), they gain even more capabilities. Precise monitoring is an essential activity for several purposes, including safety integrity and search rescue. In order to guide a vehicle along a desired path while acquiring data for a subsequent analysis, flying at high altitudes may be sufficient in many of these cases. Certain surveillance applications, however, necessitate flying at lower altitudes (e.g. below $20-m$), as well as effectively incorporating vision data (e.g. features, contours, etc.), leading to a variety of task-specific visual servoing schemes. These applications call for a higher level of image detail as well as precise navigation above the target.

Among the basic vision-aided control approaches that appear in literature, namely Image-based (IBVS), Position-based (PBVS), $2-1/2$, and Direct visual servoing [1]–[5], the IBVS method is considered more effective in target tracking applications, because of the inherited robustness against camera calibration and depth estimation errors, as well as the ability to design the control problem directly in the image space.

There are a number of different control strategies for IBVS that can be used to control UAVs for target tracking applications. These include adaptive sliding mode control [6], prescribed performance control [7], nonlinear model predictive control [8], observer-based IBVS [9], robust control

[1]Control Systems Lab, School of Mechanical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Str. Zografou, Athens 15780, Greece andreas6899@gmail.com `andreas6899@gmail.com`, `saspragkathos`, `fpanetsos`, `karrasg`, `kkyria@mail.ntua.gr`

[2]Dept. of Informatics and Telecommunications, University of Thessaly, 3rd Km Old National Road Lamia-Athens, 35100, Lamia, Greece `gkarras@uth.gr`

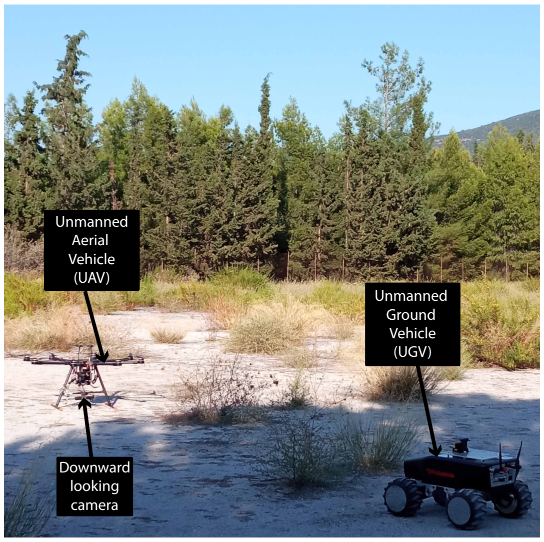


Fig. 1: Experimental setup of the proposed work: Multirotor aerial vehicle and an unmanned ground vehicle (target) on an outdoors experimental setting.

approaches that take uncertainties in the image dynamics [10], deep neural network [11] and Reinforcement Learning [12], [13]. However, the target's motion is defined by either a low or constant velocity profile in all of the aforementioned methods that take IBVS tracking into account.

This work is directed in line with the trends of science utilizing and combining RL and Unmanned Systems. Regarding RL and AI in general, in recent years there has been rapid development and growing interest in this field and its applications. Due to the increased computational power of embedded computers and the amount of available data, artificial intelligence has been incorporated into most applications giving new possibilities and solving problems that were not possible before. For instance, various deep convolutional neural networks have been developped in order to enable image recognition. Interestingly, many of these techniques were inspired by nature and biological systems. Just as convolutional networks were inspired by biological image processing structures, so reinforcement learning relies on basic training principles from the field of psychology. Reinforcement learning enables us to train an agent in a simulated environment and then transfer the learned policy to the real environment.

In this paper, a formulated deep Reinforcement learning control scheme is presented by utilizing the visual feedback of a simple camera in order to follow a ground vehicle (e.g. the

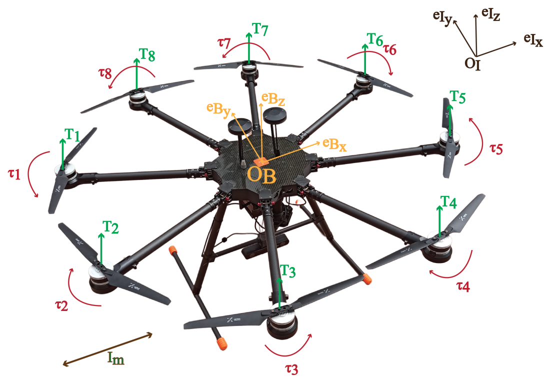


Fig. 2: Reference frames of the multirotor aerial vehicle.

target) which executes an unknown trajectory. More specifically, we leverage reinforcement learning by applying control inputs in the inner attitude loop of the autopilot opposed to the outer velocity one. This allows the vehicle to follow the ground vehicle with better performance compared to classic visual servoing schemes, due to the superior capability of the RL scheme to predict the velocity of the UGV without knowing a priori the trajectory it executes. Also, instead of markers, we deploy a Convolutional Neural Network to identify the position of the ground vehicle, which provides more flexibility and robustness. The training takes place in a simulation with the use of randomness in the creation of the environment as the transition from the simulation to the real world is almost immediate.

The rest of the paper is organized as follows: Section II presents the multirotor equations of motion, the low-level control architecture and the RL Deep Deterministic Policy Gradient that is utilized. The problem formulation is discussed in Section III. Section IV presents the CNN-based ground vehicle detection which feeds the input data in the RL proposed scheme presented in Section V. Section VI presents experimental results, demonstrating the efficacy of the proposed architecture. Finally, Section VII concludes the paper.

II. Preliminaries

A. Multirotor Equations of Motion

The fixed-pitch propellers and an even number of separate motors installed on a rigid cross structure make up the multirotor aerial vehicle. The roll, pitch, yaw, and overall thrust of the vehicle are controlled by differentially regulating the thrust produced by each motor. Because of this, the thrust and torque generated by each individual motor-propeller set govern the system dynamics [14], [15]. Fig. 2 depicts a body-fixed frame $\mathbf{B} = \{e_{B_x}, e_{B_y}, e_{B_z}\}$ attached to the vehicle's center of gravity O_B and an inertial frame $\mathbf{I} = \{e_{I_x}, e_{I_y}, e_{I_z}\}$. Based on [14], the general Newton-Euler motion equations of a 6-DoF rigid body subject to external forces and torques may be used to derive the dynamic model of a multirotor:

$$^{I}\dot{\mathbf{p}}_B = {}^{I}\mathbf{v}_B, \quad m^{I}\dot{\mathbf{v}}_B = {}^{I}\mathbf{R}_B \mathbf{f}_B, \quad \mathbf{J}_B \dot{\omega}_B = \tau_B - \omega_B \times \mathbf{J}_B \omega_B \qquad (1)$$

where $^{I}\mathbf{p}_B = \begin{bmatrix} ^{I}x_B & ^{I}y_B & ^{I}z_B \end{bmatrix}^T$ and $^{I}\mathbf{v}_B = \begin{bmatrix} ^{I}v_{x_B} & ^{I}v_{y_B} & ^{I}v_{z_B} \end{bmatrix}^T$ denote the position and the linear velocity vector of the vehicle w.r.t the inertial frame $\mathbf{I}$ respectively. The angular velocity expressed in $\mathbf{B}$ is given by $\omega_B = \begin{bmatrix} p_B & q_B & r_B \end{bmatrix}^T$. The rotation matrix from frame $\mathbf{B}$ to $\mathbf{I}$ is denoted as $^{I}\mathbf{R}_B$ and is computed by the roll, pitch, yaw angles or ϕ, θ, ψ respectively, $\mathbf{J}_B$ is the inertia matrix expressed in $\mathbf{B}$ and m is the mass. The

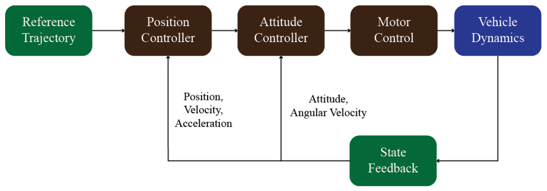


Fig. 3: Ardupilot control architecture

generalized forces and torques applied on the multirotor are defined as follows:

$$\mathbf{f}_B = \mathbf{f}_M + \mathbf{f}_d + \mathbf{f}_g, \quad \tau_B = \tau_M + \tau_d \qquad (2)$$

where $\mathbf{f}_d$ denotes the drag forces, τ_d the drag moments, $\mathbf{f}_g$ is the gravity vector, $\mathbf{f}_M$ and τ_M are the motor thrust and torque vectors, respectively [14], [15].

B. Multirotor Low-Level Control

The open source Ardupilot firmware [16] is used to efficiently control the vehicle's dynamics, which are described in Section II-A. In more detail, a cascaded PID control structure implements the vehicle's low-level control. The outer position loop is in charge of converting the desired position $^{I}\mathbf{p}_{ref}$, velocity $^{I}\mathbf{v}_{ref}$, and heading ψ_{ref} of the vehicle into the goal orientations ϕ_{ref}, θ_{ref}, ψ_{ref}, and climb rate $^{I}v_{z_{ref}}$. Motor's Pulse Width Modulation (PWM) values are finally generated by the inner attitude controller from the commanded thrust and torques. A popular Extended Kalman Filter is used to combine sensor readings, such as those from GPS, compass, and IMU, to produce an usable estimate of the multirotor's actual state. Figure 3 provides a concise summary of the control architecture. It is stated that the outer position controller is omitted in this study while the inner attitude controller is included in the suggested control scheme.

Remark 1: Roll and pitch motions of the vehicle will cause an undesirable flow of the features that may tend to break the field of view restrictions. A virtual camera frame with an origin at the body of the vehicle and an optical axis aligned with the gravity vector are taken into account to mitigate this effect. For more information the reader may refer to [7].

C. Reinforcement Learning - Deep Deterministic Policy Gradient (RL-DDPG)

In the classic Reinforcement Learning theory, an agent interacts with the environment in order to learn an optimal policy which maximizes the accumulated reward. The environment is formulated as a Markov Decision Process (MDP) which is described by the tuple $\left(s_t, a_t, p(s_{t+1}|s_t, a_t), r_t\right)$, where s_t is the current state of the agent, a_t is the action chosen, $p(s_{t+1}|s_t, a_t)$ is the transition probability to the next state s_{t+1} and $r_t = r_t(s_t, s_{t+1}, a_t)$ is the reward function which contributes into learning the best policy.

The obtained policy maps every state $s \in S$ into an action $a \in A$, where S is the state space and A is the action space. The policy π can be deterministic or stochastic and the goal of the agent is to choose the action at each time step that maximizes the accumulated reward over an episode. An episode is terminated either after a finite number of timesteps or when a termination criterion is met. The accumulated reward or return is computed as the discounted sum of rewards obtained during an episode, $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t(s_t, s_{t+1}, a_t)$, where T is the number of timesteps and $\gamma \in (0, 1)$ is a discount factor.

The value function V^{π} is defined, for a given policy π which is the expectation of the accumulated discounted reward for each state s and also the action-value function Q^{π} which represents the value of each action-state pair. The learned

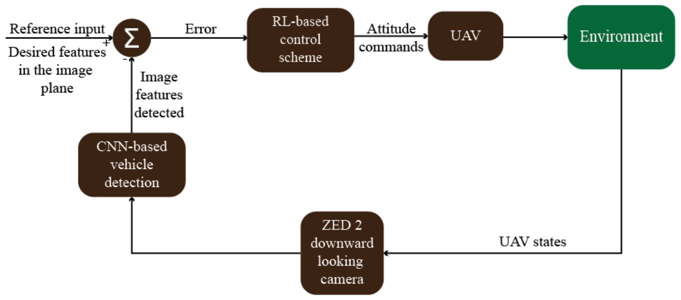


Fig. 4: Block diagram presenting the proposed architecture.

optimal policy is the one that maximizes these functions, i.e., $\pi^* = arg\max_{\pi} Q^*(s,a)$. In an actor-critic scheme, the optimal Q function and the policy are approximated with two neural networks and the learning of these functions relies on the learning of the parameters or weights of these neural networks.

Many robotics applications require the action and state spaces to be continuous. This imposed problem is tackled by state-of-the-art Deep Reinforcement Learning algorithms such as the employed Deep Deterministic Policy Gradient algorithm which is a policy based meaning that it is searching directly the optimal deterministic policy π^* while learning the action-value function Q in an off-policy manner. The most important features that DDPG uses in order to converge are the replay buffer and the target networks. The first is essential to break the correlation between records used for training and the second helps avoid divergence. The DDPG algorithm is proven to be suitable for the application of this work since we have continuous spaces in both action and state space.

III. Problem formulation

In this work a kinematic control problem using RL is addressed, concerning the following of an unmanned ground vehicle by a UAV equipped with a downward looking camera. The kinematic control is applied to the inner loop which implies that an attitude controller exists and, hence, the learned policy produces reference roll, pitch and yaw angles which enable the UAV to directly respond to the motion of the UGV. The detection of the target is achieved through a deep convolutional network (CNN) that performs segmentation of the image obtained by the UAV's camera and identifies the position of the UGV on the image plane. The goal of this application is to enable the UAV to move autonomously following the UGV in a set of different environments and conditions without the need for external intervention. The proposed pipeline is divided in subsystems that coordinate together to achieve the final outcome. The image acquired from the mounted camera on the UAV is fed into a trained Convolutional Neural Network detecting the moving ground vehicle. Utilizing the detection outcome, the distance error of the vehicle from the center of the image plane is formed. According to the error, the state vector is eventually formulated, a very essential variable for the training of the Reinforcement Learning algorithm, accompanied by the action vector and the reward function. Given a trained RL agent the state vector that was computed from the pixel coordinates of the target after the transformation is fed to the RL neural network, called actor, which outputs a set of attitude commands. This actor has been trained through a number of episodes and has learnt an optimal policy and as a result it outputs attitude commands that solve the desired task. Finally, these commands are sent to the flight controller that controls the actual motion of the UAV. The block diagram in Figure 4 depicts the proposed strategy that guarantees autonomous tracking of a ground vehicle from a UAV.

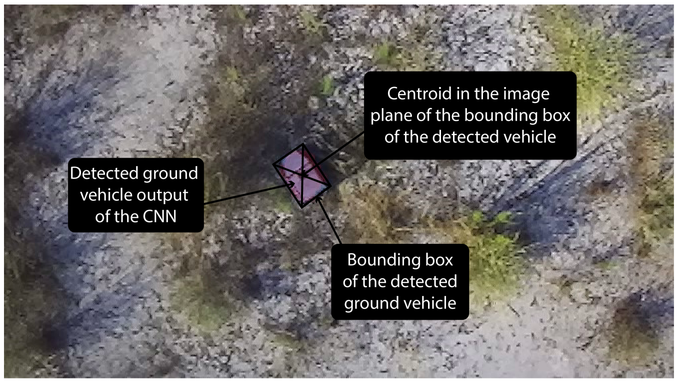

Centroid in the image plane of the bounding box of the detected vehicle
Detected ground vehicle output of the CNN
Bounding box of the detected ground vehicle

Fig. 5: Result of the the trained CNN for an Unmanned Ground Vehicle (UGV) and the bounding box including the detected vehicle (highlighted in red).

IV. CNN-based target detection

To execute accurate contour detection, [17] is done using a pre-trained CNN for image segmentation. The CNN separates the total number of pixels in the image into pertinent object classes. The pre-trained CNN model *mobilenet_segnet* from the Keras image segmentation framework is used to detect the outlines (Base Model: MobileNet and Segmentation Model: Segnet, an encoder network comprising 13 convolutional layers designed for object classification). Data containing frames, obtained from the camera mounted on the multirotor, were utilized to train the chosen CNN. A training and a validation set were created from the data set. The frames have a resolution of 672×376. The following stages were included in the preprocessing procedure:

- Labeling using polygons to indicate the target/object.
- Mask extraction (binary image according to the annotated features from the labeling procedure).
- Resizing of the frames from 672×376 pixels to 224×224 pixels.
- Binary classification (Class 0: Black background that doesn't belong to the target and Class 1: Target/Object).
- Various augmentation tools (i.e. horizontal flip of the image, cropping, addition of noise, tweaking of the saturation etc.) were used for the enhancement of the training and validations sets.

After preprocessing, the CNN was trained on a GPU-powered machine until it converged to the required accuracy after roughly $5-8$ epochs, depending on the application. The output of the trained CNN on a raw image is displayed in Fig. 5. The performance of the trained CNN was verified by making a real-time forecast while flying over the target.

Once the contour has been recognized by the trained DNN, it is assumed that a set of features will be available for specifying a Region Of Interest (ROI) in the image frame that contains the contour (for example, one that is rectangular in shape), which can then be described by a bounding box using conventional image processing methods [18]. In order to accurately match features between successive frames and to reduce the number of features required for vision-based control, a bounding box must be extracted. Fig. 5 shows the bounding box of the identified shape.

V. Reinforcement Learning scheme

RL is utilized to train an aggressive controller required for the target tracking application. Its formulation consists of defining the action and state vector as well as the reward function. These three variables are crucial for the convergence of the algorithm since there are many different possible approaches that would ideally result in the same outcome. However, a proper design for the formulation would accelerate the convergence and minimize the possibility of divergence. The design is considered adequate when the information passed to the Reinforcement Learning agent is compact and useful for the problem that is being solved. In our case, the state vector is defined by:

$$s = [d_x, d_y, \dot{d}_x, \dot{d}_y, {}^{B}v_{x_B}, {}^{B}v_{y_B}] \tag{3}$$

where d_x, d_y are the distances between the center of the detected box and the center of the image plane on x and y axes respectively and $\dot{d}_x$, $\dot{d}_y$ are the derivatives of the aforementioned distances. Since both the target and the UAV are moving vehicles and the only source of information regarding the target's velocity is the onboard camera, the derivative of the distance gives an important measurement that is related to that velocity. Those values combined with ${}^{B}v_{x_B}$ and ${}^{B}v_{y_B}$, which are the velocities of the UAV in x and y axes w.r.t. **B**, provide a better estimation about the relative velocities of the two vehicles. The information about the relative velocity has been proved to be important in order to achieve a better convergence since without it the UAV accelerates towards minimizing the positional error and as a result it oscillates around it. It's important to note that the normalization of all values in the state vector was significant for the convergence of the training. Regarding the action vector, it is described as follows:

$$a = [a_\phi, a_\theta, a_\psi] \tag{4}$$

where a_ϕ is the roll action, a_θ the pitch action and a_ψ is the yaw action which is a value relative to the current yaw angle.

The action space is normalized to [-1,1] and limited to a spectrum of acceptable values by multiplying those normalized values with maximum angles. More specifically for the roll and pitch actions the acceptable values range between [-3,3] degrees and for yaw action range between [-5,5] degrees. The reward function is probably the most important component in the reinforcement learning framework. In our approach the reward function is designed in a way that rewards flying directly over the target while taking smooth actions and, specifically, is defined as:

$$r = -w_1(|d_x| + |d_y|) - w_2(|\dot{d}_x| + |\dot{d}_y|) - w_3(|a_\phi| + |a_\theta| + |a_\psi|) \\ -w_4\left(|{}^{B}v_{y_B}| + clip({}^{B}v_{x_B}, -1, 0)\right) \tag{5}$$

where w_i, $i = 1, \ldots, 4$ are positive gains and clip function keeps only values in the specified range.

The first term is the positional penalty and is defined as the sum of the positional error in x and y axes, meaning the distance between the center of the box and the center of the image plane. The second term is the derivative penalty and is the sum of the derivatives of the error on x and y axes too, while the third term is the action penalty which penalizes big changes in the action commands resulting in a smoother behaviour. Finally, the last term is the velocity penalty which is defined as the sum of the velocity on the y axis and the negative values of velocity in the x axis. This way we penalize backwards movement since we always want to follow the target facing toward the direction of the movement and we penalize velocity on the y axis to enforce forward movement and thus achieve a better alignment with the moving target. The weights w_i are empirically designed in a way that prioritizes the penalties' importance in the training of the reinforcement learning framework and also result in normalizing the reward function to [0,1].

As mentioned before we are utilizing a DDPG agent which is an actor critic scheme. In this framework, the actor and critic networks are feed-forward neural networks with two hidden layers of 256 units each. The activation function of the actor is a hyperbolic tangent function (tanh) and for the critic a Rectified Linear Unit (ReLU). The input and output layer dimensions of the actor network are based on the state and action dimensions (6 and 3 units), respectively. The output layer of the critic network has one unit with a linear activation function in order to provide an estimation of the Q-function. Finally, the learning rate for the actor was decided to be 10^{-4} and for the critic 10^{-3} while the τ parameter for updating the target networks was set to 0.01.

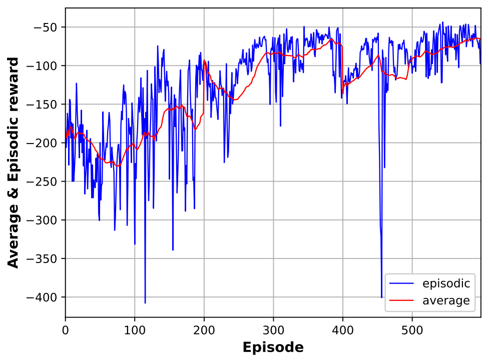


Fig. 6: Average (red line) and Episodic (blue line) reward values fluctuation during all the episodes of the training.

The training procedure of the RL proposed scheme took place in a synthetic environment of a realistic flight simulation of a UAV. The overall development and utilization of simulation environments for autonomous flight applications are due to the following reasons:

- safety reasons $\rightarrow$ increased risk of vehicle crash during the early testing of prototype autonomous flight algorithms and
- logistics problems while rapid prototyping $\rightarrow$ inability to conduct experiments frequently (e.g., every day).

In order to expedite the development process, a synthetic but realistic simulation environment was built using the Robot Operating System (ROS) [19] and Gazebo [20] frameworks, featuring also MAVROS [21] communication protocol as SITL. A 3DR Iris quadrotor [22] is deployed inside the simulation environment equipped with the appropriate sensor suite:

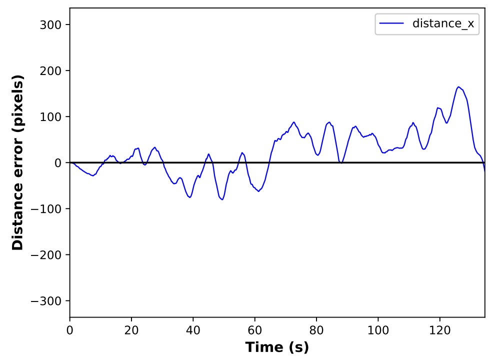
- Navigation sensors (GPS, IMU, altimeter, etc.) and
- Downward-looking stereo camera system (ZED 2), providing frame-based image data.

In this flight simulation environment, the vehicle moves at first in a randomized manner and thus exploring the different states and associated rewards. These sets of states, actions and rewards are stored and used for the training of the agent and after a number of training episodes the agent has been successfully trained as depicted in Fig. 6.

VI. Experimental results

Experimental setup

To test and demonstrate the performance of the proposed strategy, various sets of experiments were conducted. An explanatory video (https://youtu.be/tTx_OR_ivIc) has been filmed for a better understanding of the experimental process. An octocopter flying at relatively low altitude (below 20m above ground-sea level) was used during the experiments, equipped with an onboard computer (NVIDIA Jetson AGX Xavier Developer Kit), a ZED 2 Stereo Camera facing downwards and a Pixhawk Cube Orange employing the ArduPilot



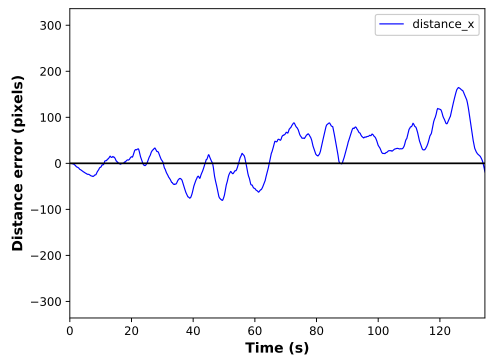


Fig. 7: Distance error of the detected UGV from the center of the camera image plane along x- **(left)** and y-axis **(right)**.

firmware. The CNN-based detection, and the Deep RL scheme presented previously were running in the onboard computer through the use of the Robot Operating System (ROS) [19]. The attitude commands produced by the RL scheme are sent to the octocopter's microcontroller through the MAVROS [21] communication protocol. The attitude commands given to the Pixhawk are manipulated according to the octocopter's low-level control presented in Sec. II-B. The experimental scenario is the following of a Robotnik Summit XL from the deployed flying octocopter.

Results

In this section, experimental ground vehicle tracking scenario is presented. In all experiments, the environmental parameters were considered as exogenous factors that can not be configured. The UGV was in manual control during the experiments executing aggressive maneuvers. Fig. 7 depicts the distance error of the detected UGV from the center of the camera image plane. The error values are large, but without ever the target be outside of the Field of View. The bigger error values and fluctuations are due to the aggressive maneuvering of the UGV during the experiment. Detection was successful as it found the vehicle consistently without false positives or true negatives. As discussed in Section V, the set of points created by the detection (detection) is not sufficient to determine the orientation of the driver and therefore we proceed with the strategy described. The orientation error during an outdoors experimental scenario.

VII. Conclusions

In this paper, we presented a control scheme using Reinforcement Learning for the autonomous detection and tracking of a moving Unmanned Ground Vehicle using an octocopter aerial vehicle flying at low altitudes. The online detection of the UGV was realized from a utilized trained CNN. The outcome of the CNN detection is incorporated into an appropriately formulated Reinforcement Learning attitude control scheme managing to simultaneously retain the ground vehicle inside the camera field of view and as close as possible to center of the image. Experimental demonstration scenarios performed in a set of various environmental locations demonstrated the efficacy of the proposed framework.

References

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

[2] ——, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[3] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[4] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[5] G. Silveira and E. Malis, "Direct visual servoing: Vision-based estimation and control using only nonmetric information," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 974–980, 2012.

[6] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 971–976.

[7] G. C. Karras, C. P. Bechlioulis, G. K. Fourlas, and K. J. Kyriakopoulos, "Target tracking with multi-rotor aerial vehicles based on a robust visual servo controller with prescribed performance," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 480–487.

[8] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2202–2207.

[9] H. Jabbari Asl, G. Oriolo, and H. Bolandi, "Output feedback image-based visual servoing control of an underactuated unmanned aerial vehicle," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 228, no. 7, pp. 435–448, 2014.

[10] H. J. Asl and H. Bolandi, "Robust vision-based control of an underactuated flying robot tracking a moving target," *Transactions of the Institute of Measurement and Control*, vol. 36, no. 3, pp. 411–424, 2014.

[11] M. A. Kassab, A. Maher, F. Elkazzaz, and Z. Baochang, "Uav target tracking by detection via deep neural networks," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 139–144.

[12] 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)*. IEEE, 2018, pp. 979–986.

[13] S. Lee, T. Shim, S. Kim, J. Park, K. Hong, and H. Bang, "Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 108–114.

[14] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[15] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *International conference on simulation, modeling, and programming for autonomous robots*. Springer, 2012, pp. 400–411.

[16] A. P. group. (2016) Ardupilot documentation. [Online]. Available: https://ardupilot.org/ardupilot/

[17] D. Gupta. (2019) A beginner's guide to deep learning based semantic segmentation using keras. [Online]. Available: https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html

[18] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[21] M. M. A. V. Protocol. (2013) Mavlink to ros gateway with proxy for ground control station. [Online]. Available: https://github.com/mavlink/mavros

[22] Iris. (2021) 3dr iris quadrotor. [Online]. Available: http://www.arducopter.co.uk/iris-quadcopter-uav.html