ISSN 2007-9737

Autonomous Drone Racing with an Opponent: A First Approach

L. Oyuki Rojas Perez[1], J. Martinez Carranza[1,2]

[1] Instituto Nacional de Astrofisica, Optica y Electronica, Mexico

[2] University of Bristol, United Kingdom

{oyukirojas, carranza}@inaoep.mx

Abstract. Drone racing is a popular sport where human pilots control their drones via radio frequency to fly at high speed through complex race tracks. The latter has motivated to pose the question: could an autonomous drone beat a human in a drone race. Thus, some works have dealt with the autonomous drone racing challenge; however, very few works have dealt with the case of a race when an opponent is present in the track. In this work, we present an initial approach to address the problem of autonomous navigation while considering the presence of an opponent in the race track. To address this problem, we present a compact Convolutional Neural Network that predicts flight commands from a set of camera images captured on the fly. This network has been trained by imitation; this is, the network learns from examples generated by a human pilot. It is important to highlight that there is no explicit gate detection or trajectory planning/tracking in our approach. We have carried out experiments in the Gazebo simulator in a race track where the autonomous drone will face an opponent on its way to the gate. Our results show that the network manages to pilot the drone to evade the opponent, and after the evasion, the drone gets on-track towards the gate.

Keywords. Autonomous drone racing, visual perception, autonomous navigation.

1 Introduction

Autonomous Drone Racing (ADR) is a scientific topic that has gained broad interest in a relatively short amount of time. Interestingly, research on the ADR was not only motivated by the scientific challenges but also strongly driven by the annual competitions. Since its inception in 2016 in the IROS ADR competition (which has been organised since then every year in IROS) [14], other competitions have also sought to push the scientific limits behind this challenge. Take, for example, two representative competitions such as the Alpha Pilot competition, organised by Lockheed Martin [5], and the Game of Drones, organised by Microsoft and Stanford University [13], both in 2019.

Across the years, different approaches have been presented, but in any of these competitions, the typical scenario is for the contestant autonomous drone to fly through the race track alone, with no opponent in the race. The Game of Drones competition posed a couple of tasks where an opponent was mentioned. However, competitors focused only on racing as fast as possible, and in fact, the opponent was always left behind from the beginning. Outside of these competitions, a couple of works have addressed the problem of flying against an opponent, although not in the ADR context.

In contrast, in this work, we present a novel approach considering a first scenario where the drone has to face an opponent on its way to the gate. Our approach is based on a compact Convolutional Neural Network (CNN) that regresses 2 flight commands, namely, roll and pitch flight commands. These flight commands are angular positions of the drone's body frame. These angular positions are passed and interpreted by the drone's inner controller, producing translation

motion in the roll and pitch angles respectively. As an initial approach, we assume that the gates are at the same height and parallel to the drone's camera plane. Thus, we use a Proportional Integral Derivative (PID) to control height and heading.

Our strategy is based on the work presented in [18]. In the latter, we used PoseNet [11], a CNN for the camera relocalisation problem. PoseNet receives a single image and predicts the 6-D camera pose from where the image was taken, in a world coordinate system. We re-trained and adapted PoseNet to predict the same 2 flight commands, roll and pitch and showed that it is possible to use a CNN-based approach to pilot the drone for these 2 flight commands. We also showed that *temporality* works better than using a single image. Temporality refers to the use of consecutive of certain frames in a line time as input frames to the CNN.

Rather than using stacking [20] as a way of combining these frames, we proposed to use a mosaic image composed of these input frames. We prefered this option over stacking because we did not want to increase the number of parameters within the network nor we wanted to increase the processing time. In our experiments, we showed that a mosaic of 6 images is enough to capture the motion trend towards the gate, thus enabling better flight command predictions than when using a single image.

From the above, in the current work, we present a more compact network, with much less convolutional layers and inception modules. In addition, we have created a new training dataset, including cases in which the opponent shows up in the images. Note that in this work, neither we carry out explicit gate detection nor trajectory planning/tracking. Our goal is to produce an artificial pilot that imitates the decision making performed by a human, this is, deciding on the corresponding flight commands only by looking at the camera images.

In order to present our work, this paper has been organised as follows: Section 2 discusses the progress regarding the ADR; Section 3 describes our methodology; Section 4 presents our experimental framework, and Section 5 presents our conclusions and future work.

2 Related Work

In the 2016 IROS ADR [7, 15, 14], the participant teams relied on gate detection to drive their drone's controller. The drones in this competition flew at slow speed in a cluttered environment where the gates were placed very near to each other. Teams used colour segmentation [6] and QR identification to detect the gates. The QR were provided by the organisers.

Next year, in the 2017 ADR competition [14], most of the teams included visual Simultaneous Localisation and Mapping (SLAM) techniques for drone localisation and flight planning based on waypoints [17]. Gate detection, based on conventional computer vision techniques, was also used by some of the teams, following the idea of controlling the drone based on where the gate appear on the camera image [4]. Gate detection has also been leveraged by using CNNs [8, 1], more robust to changes in illumination and gate overlapping. In these cases, the controller seeks to align the drone's camera optic centre w.r.t. the gate's centre detected on the image.

Other works have proposed to use the gate detection to infer the 3-D position of the drone w.r.t. the gate [10, 9, 2, 3, 5]. In these cases, the controller works in the 3D world, either using trajectory planning/tracking or waypoints to fly the drone towards the gate. Other CNN-based works have proposed the combination of CNNs. For instance, in [16], the authors use a first CNN that receives an input image to predict a 5-point trajectory; the second network uses this trajectory as much as the orientation and the velocity of the drone as inputs. The CNN's output will be a prediction of speeds in throttle, roll, pitch and yaw.

Regarding the case of when a drone races against another opponent, we have identified two relevant works based on game theory [21, 19]. In a set of strategies, one of them consists of blocking the passage of the drone behind. To carry out these strategies, the authors use a motion capture system to identify their drone's position and that of the opponent.

In this work, we do not rely on external localisation systems, explicit gate detection, trajectory planning/tracking or any other waypoint-based controller. Our goal is to develop an artificial pilot that resembles the human pilot's decision making. This is, deciding the flight commands based on the camera images.

3 Proposed Framework

In this section, we describe our proposed CNN architecture to predict 2 flight commands, roll and pitch. We also describe our data acquisition method, whose data was used to train our network. We will also describe the whole control architecture consisting of the CNN and 2 PID controllers for height and heading.

3.1 CNN-Pilot

Based on our previous work [18], we propose a more compact network architecture aiming at extracting features from the input image. The data flows through some convolution layers and three inception modules, to be finally evaluated with a regressor layer for the prediction of the flight commands, this is, for roll and pitch. We refer to this network as CNN-Pilot.

Figure 1 depicts our proposed architecture. In sum, it has 4 convolutional layers with 3 inception modules, 1 fully connected layer and a regressor layer. The loss function used for each branch is shown in the equation 1. Table 1 shows the parameters used to train our CNN-Pilot:

$$loss(I) = ||\hat{x} - x||_2, \tag{1}$$

where *x* corresponds to the flight command values for each image (I), recorded when piloting the drone manually, and $\hat{x}$ is the flight command predicted by the model. The loss function is evaluated for both flight commands: roll and pitch. Thus, our CNN-Pilot predicts roll and pitch whose values fall in the rage of $[-1, 1]$.

Table 1. Parameters used to train our proposed CNN-Pilot to learn 2 flight commands: roll and pitch

Parameters	Value
Optimiser	Adam
Epoch	500
Batch size	32
Activation function	ReLu
Learning rate	0.001

3.2 Data Acquisition

To create our image dataset, we manually flew the drone in a race track with several gates and with opponents placed in between some gates. For these race tracks, the positions of the gates were chosen randomly. The goal was to provide different examples of what the appearance of gates from onboard camera's viewpoint. This included overlapping gates, nearby and faraway gates, etc. Note that these tracks were different to those used in the experiments section.

We acquired images to demonstrate 2 flight commands in the roll and pitch angles. These commands produce flight translation in those angles. Thus, for each one of these images, a pair of flight commands (roll,pitch) was recorded. These flight commands will be used as labels for the training stage. In the collected images, the gate was always kept in the camera's viewing area. After the recording, a manual adjustment is made to the flight commands to identify the dominant drone's motion in roll and pitch. Once identified, the flight command corresponding to the dominant motion will be kept as recorded, where as the other value is set to zero. This avoided ambiguous commands.

In total, we recorded 9,500 single images with their corresponding roll and pitch flight commands. However, to train our CNN architecture, we used these images to generate a dataset of mosaic images, composed of 6 frames. In total, we generated 7,298 mosaic images for training and 654 mosaic images for the network validation during the training stage. Examples of mosaic images, in the training dataset, can be seen in Figure 2.

ISSN 2007-9737

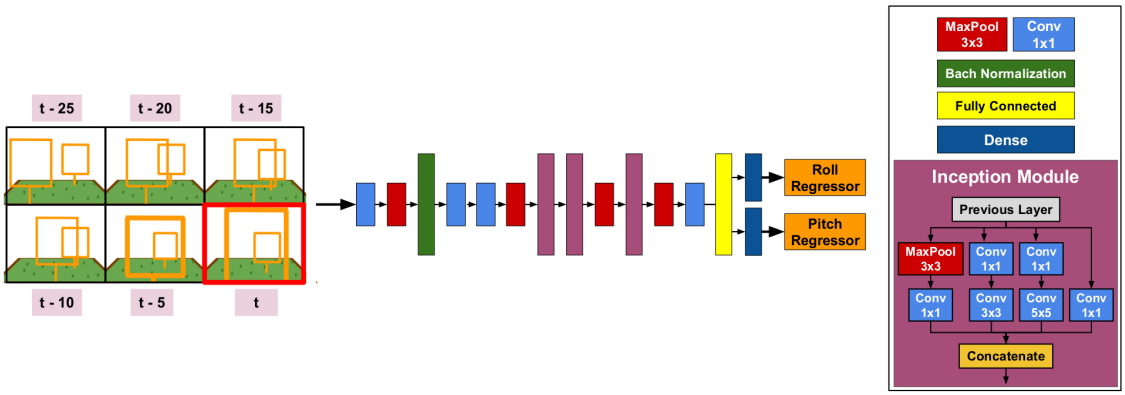

Fig. 1. Proposed CNN architecture. This network receives a single mosaic image composed of 6 camera frames and predicts 2 flight commands, roll and pitch. The size of the kernels in this network is indicated in the coloured boxes to the right of the image

Each mosaic image is associated to 2 flight commands, roll and pitch, with values ranging from $[-1, 1]$. These values represent scaled values of the angular positions of the drone's body frame.

3.3 Control System

The control system runs on the Ubuntu 16.04 LTS operating system, using the Kinetic Kame version of the Robotic Operating System (ROS) as the communication architecture. The process is performed in a laptop with the following specifications:

— Intel Core i7 processor,

— 16 GB of RAM,

— NVIDIA GEFORCE GTX 1070 graphics card.

The control system is divided into two main modules, which work within the ROS system. In the first module, the roll and pitch control commands are obtained using the CNN-Pilot. In the second module, a PID controller is implemented to maintain the height and heading of the drone, which is fed-back using the drone altimeter and the Inertial Measurement Unit (IMU).

The reference for the height and yaw angle are set manually since all the gates have the same height and angle. Figure 3 shows the general diagram of the control architecture.

4 Experimental Framework

In this section, we present our experimental framework to evaluate the performance of our proposed CNN-Pilot approach. We executed the experiments in the Gazebo simulator [12] to show the efficacy of CNN-Pilot; we created a Zig-Zag race track. The track spans over a surface of 60m. $\times$ 7m in length. There are 11m in of space in between gates and the track is composed of 5 gates of 2.5 m height.

Rather than having an opponent drone flying in the race track, we forced the situation in which the drone faces an opponent on its way to the next gate to be crossed. For this reason, we placed 4 static opponents between the 5 gates, see Figure 4. In this manner, we can observe the behavior of the control system when the drone flies near by the opponent, above or below it, or even when the opponent is in front of it.

We placed the opponents at different heights intentionally because we wanted to observe the decision taken by the CNN-Pilot in concrete scenarios: 1) the opponent is nearby but it is not blocking the way; 2) the opponent in above or below, but is not blocking the way; 3) the opponent is block the way, therefore an evasion has to be carried out via the corresponding flight commands and then, once evaded, the drone has to get on-track to its way to the next gate.

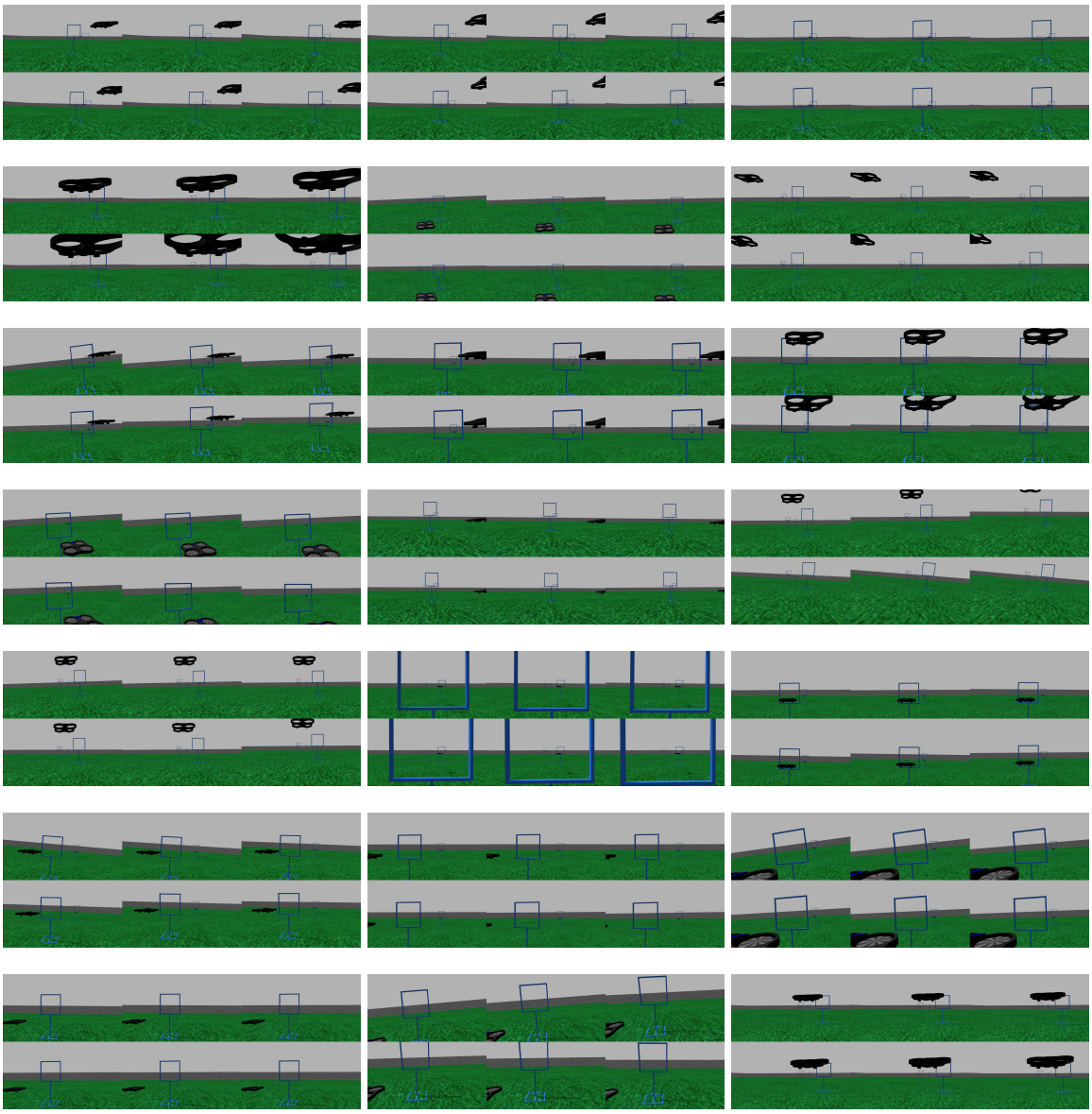

Fig. 2. Examples of the dataset built for this work. This consist of mosaic images composed of 6 camera frames. The frames used in this dataset were recorded during a manual flight

We perform 10 runs to assess the repeatability of the control system. In total, 8 runs were successfully completed. Figure 5(a) shows a top view of the trajectories obtained by the drone, controlled by our CNN-Pilot, in these runs. The opponents are depicted as black circles.

The first opponent is not totally blocking the way, and after the drone crosses the first gate, it is easier for the CNN-Pilot to keep the course towards the second gate. The second opponent is partially blocking the way, but the CNN-Pilot manages to perform a slight evasion to the right, managing to

ISSN 2007-9737

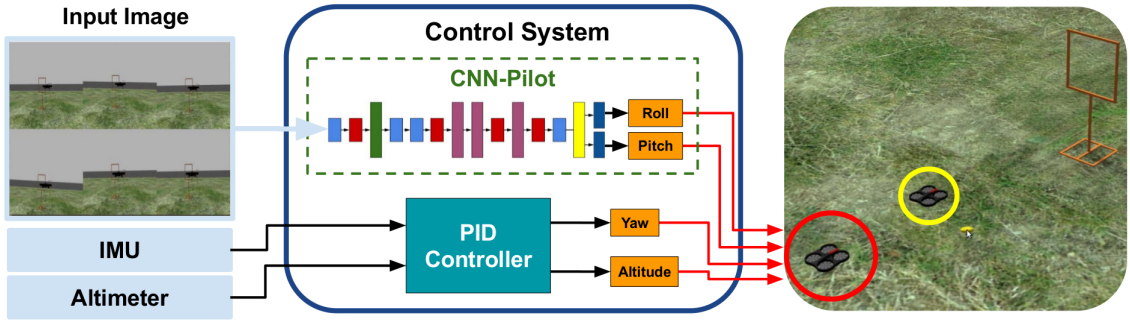

Fig. 3. Proposed control architecture. The CNN-Pilot receives a mosaic image, composed of 6 camera frames, and predicts 2 flight commands, roll and pitch. Height and heading (yaw angle) are controlled by a PID controller

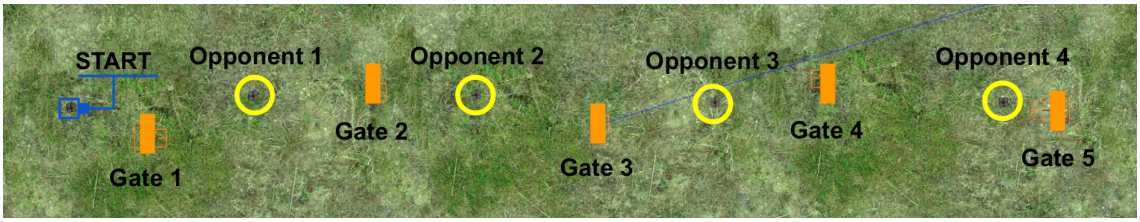

Fig. 4. Race track used in our experiments, simulated with the Gazebo simulator. The opponent was simulated by a set of drones placed in between gates and highlighted in yellow circles. The idea behind this scenario is that, when the drone, controlled by our CNN-Pilot, flies towards the next gate, it will face the opponent

fly away from the opponent and then to get on track to cross the third gate. The third opponent blocks the way, however, the CNN-Pilot manages to stop and evade successfully by moving to the left and then it navigates towards the fourth gate to cross it. Note that for this segment of the track, in two runs CNN-Pilot decided to move the drone to the right in order to evade the opponent. For this reason, the fourth gate got out of view, hence the drone navigated towards to fifth gate, skipping the fourth gate.

We concluded that this was due to the speed and trajectory that had been resulted by crossing the third gate, which leaned slightly more to the right than in the other runs. This caused the opponent to appear with an image position more prone to the left, hence CNN-Pilot decided to evade by the right. Finally, the fourth opponent was located below, with a lower height. Thus, CNN-Pilot correctly predicted the flight commands for the drone to fly over the opponent, since it did not block the way, enabling the drone to cross the fifth gate without problem.

The trajectories can also be better appreciated in the side view, shown in Figure 5(b). A video with 3 illustrative runs can be found here[1].

Figure 6 shows illustrative examples of the drone's navigation and evasion along the race track. The red circle indicates the drone piloted by the control system and the yellow circle indicates the opponent drones, which are placed in between the gates and at different heights (2.3m, 2m, 2m and 1.5m). Each image shows the external view of the drone (Gazebo), keyboard and 2 additional windows. The keyboard is only used to take off and landing of the drone, as well as to start or cancel the autonomous flight.

The first window corresponds to the mosaic image, composed of 6 camera frames, and used as input for the CNN-Pilot. The second window shows the current frame obtained by the front camera onboard the drone. This image also indicates the altitude and heading (yaw angle) reference, roll and pitch flight commands obtained by the CNN-Pilot,

[1] https://youtu.be/OKJg4Jy71K4

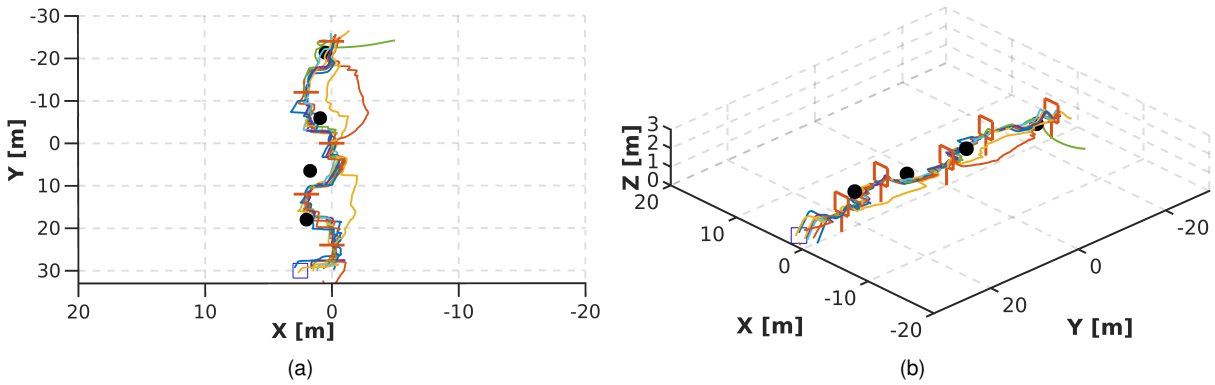

Fig. 5. Top view (a) and side view (b) of the trajectories flew by the drone controlled with our CNN-Pilot. This network has been trained to predict 2 flight commands: roll and pitch. The plots show 10 runs in a zig-zag track, opponents are marked as a filled black circle. A video illustrating of our approach can be found at `https://youtu.be/OKJg4Jy71K4`

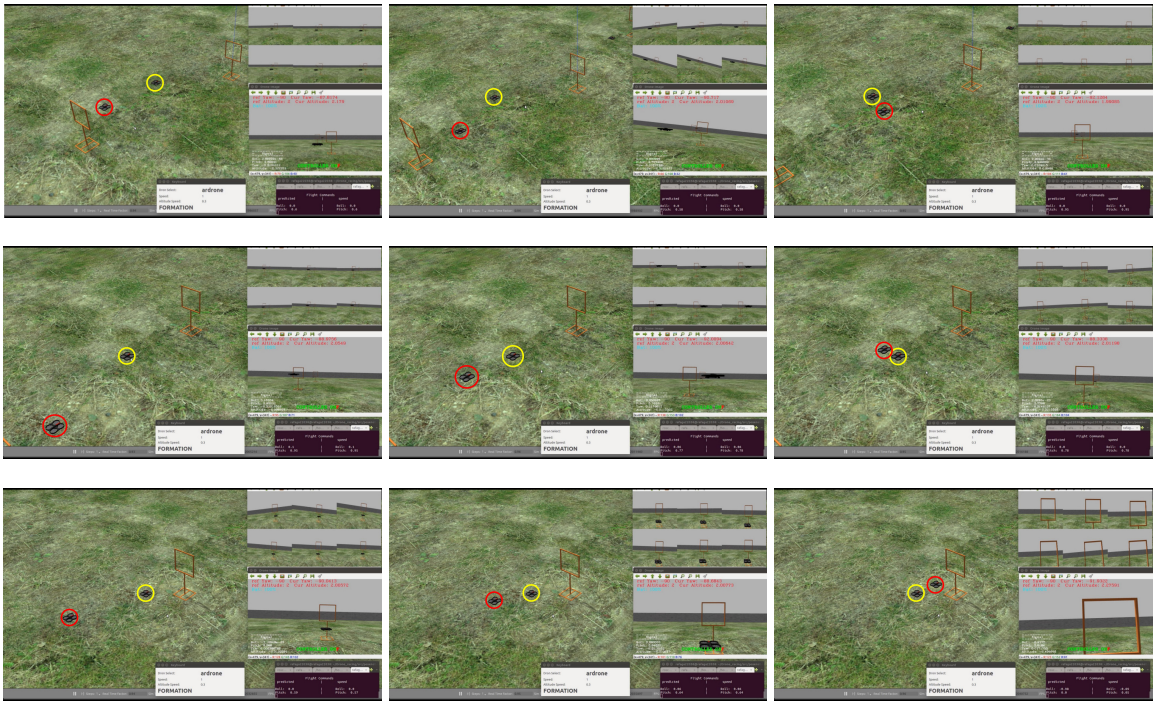

Fig. 6. Illustrative examples of our initial approach for ADR with an opponent. Each row shows the performance of our control system, which controls the drone to fly towards the gate. However, an opponent is blocking the way. Hence, the CNN-Pilot predicts the corresponding commands in roll and ptich to evade and later to get on-track towards the next crossing gate. The opponent drone is indicated in a yellow circle. A video illustrating of our approach can be found at `https://youtu.be/OKJg4Jy71K4`

as much as the PID control flight commands for altitude and heading. In average, the flight speed was of $0.61m/s$.

Finally, we highlight that the reduction of the inception modules and the convolutional layers of the CNN presented in [18] increased the prediction time from 21 fps to 30 fps. This contributes significantly to the performance of the drone to fly across the race track since the average time of the 8 runs was 1 minute 42 seconds.

5 Conclusions

In this work, we have presented the initial results of a CNN-based approach for Autonomous Drone Racing (ADR), considering an opponent. Our proposed CNN is a compact architecture that predicts 2 flight commands, roll and pitch, from images captured with the drone's onboard camera. These flight commands represent scaled angular positions of the drone's body frame, which are passed to the drone's inner controller. As a whole, our control system uses our proposed CNN-Pilot for roll and pitch and 2 PID controllers for height and heading.

We should highlight that our approach does not require explicit gate detection, trajectory planning/tracking, or waypoint-based control. Our approach can be seen as an imitation approach, meaning that the CNN has learned from manual pilot commands executed in a race track. The training dataset includes examples with and without opponent visible in the camera images. The goal was to show that our CNN-Pilot was capable of controlling the drone to fly autonomously toward the next gate and, when facing an opponent on its way, to predict the corresponding flight commands to evade the opponent. After the evasion, the control system was capable of getting the drone on-track to navigate towards the next gate to be crossed.

We carried out experiments in the Gazebo simulator, performing several runs to demonstrate the repeatability of our approach. In each run, the drone flown by our control system managed to evade the opponent and finished the race track. In average, control system flew the drone at a speed of $0.61m/s$ and predicted the flight commands at 30 fps.

In our future work, we will port our CNN-Pilot to be run onboard a physical drone. Porting over will be a seamless task since we will use a drone compatible with the simulated model. In addition, we will extend our study to consider scenarios with a more complex opponent's behavior.

References

1. **Cabrera-Ponce, A. A., Rojas-Perez, L. O., Carrasco-Ochoa, J. A., Martinez-Trinidad, J. F., & Martinez-Carranza, J. (2019).** Gate detection for micro aerial vehicles using a single shot detector. *IEEE Latin America Transactions*, Vol. 17, No. 12, pp. 2045–2052.
2. **Cocoma-Ortega, J. A. & Martinez-Carranza, J. (2019).** A cnn based drone localisation approach for autonomous drone racing. *11th International Micro Air Vehicle Competition and Conference*, Madrid, Spain.
3. **Cocoma-Ortega, J. A. & Martínez-Carranza, J. (2019).** Towards high-speed localisation for autonomous drone racing. *Mexican International Conference on Artificial Intelligence*, Springer, pp. 740–751.
4. **de Croon, G. C., De Wagter, C., Remes, B. D., & Ruijsink, R. (2012).** Sub-sampling: Real-time vision for micro air vehicles. *Robotics and Autonomous Systems*, Vol. 60, No. 2, pp. 167–181.
5. **Foehn, P., Brescianini, D., Kaufmann, E., Cieslewski, T., Gehrig, M., Muglikar, M., & Scaramuzza, D. (2020).** Alphapilot: Autonomous drone racing. *arXiv preprint arXiv:2005.12813*.
6. **Illingworth, J. & Kittler, J. (1988).** A survey of the hough transform. *Comput. Vision Graph. Image Process.*, Vol. 44, No. 1, pp. 87–116.
7. **Jung, S., Cho, S., Lee, D., Lee, H., & Shim, D. H. (2018).** A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, Vol. 35, No. 1, pp. 146–166.
8. **Jung, S., Hwang, S., Shin, H., & Shim, D. H. (2018).** Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, pp. 2539–2544.

9. **Kaufmann, E., Gehrig, M., Foehn, P., Ranftl, R., Dosovitskiy, A., Koltun, V., & Scaramuzza, D. (2018).** Beauty and the beast: Optimal methods meet learning for drone racing. *arXiv preprint arXiv:1810.06224*.

10. **Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., & Scaramuzza, D. (2018).** Deep drone racing: Learning agile flight in dynamic environments. *arXiv preprint arXiv:1806.08548*.

11. **Kendall, A., Grimes, M., & Cipolla, R. (2015).** Posenet: A convolutional network for real-time 6-dof camera relocalization. *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946.

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

13. **Madaan, R., Gyde, N., Vemprala, S., Brown, M., Nagami, K., Taubner, T., Cristofalo, E., Scaramuzza, D., Schwager, M., & Kapoor, A. (2020).** Airsim drone racing lab. *arXiv preprint arXiv:2003.05654*.

14. **Moon, H., Martinez-Carranza, J., Cieslewski, T., Faessler, M., Falanga, D., Simovic, A., Scaramuzza, D., Li, S., Ozo, M., De Wagter, C., et al. (2019).** Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, Vol. 12, No. 2, pp. 137–148.

15. **Moon, H., Sun, Y., Baltes, J., & Kim, S. J. (2017).** The iros 2016 competitions [competitions]. *IEEE Robotics Automation Magazine*, Vol. 24, No. 1, pp. 20–29.

16. **Muller, M., Li, G., Casser, V., Smith, N., Michels, D. L., & Ghanem, B. (2019).** Learning a controller fusion network by online trajectory filtering for vision-based uav racing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0.

17. **Rojas-Perez, L. O. & Martinez-Carranza, J. (2017).** Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight. *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pp. 234–239.

18. **Rojas-Perez, L. O. & Martinez-Carranza, J. (2019).** A temporal cnn-based approach for autonomous drone racing. *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pp. 70–77.

19. **Spica, R., Falanga, D., Cristofalo, E., Montijano, E., Scaramuzza, D., & Schwager, M. (2018).** A real-time game theoretic planner for autonomous two-player drone racing. *arXiv preprint arXiv:1801.02302*.

20. **Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016).** Temporal segment networks: Towards good practices for deep action recognition. *European conference on computer vision*, Springer, pp. 20–36.

21. **Wang, Z., Spica, R., & Schwager, M. (2019).** Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*. Springer, pp. 225–238.

Article received on 17/06/2020; accepted on 21/07/2020.
Corresponding author is L. Oyuki Rojas Perez.