SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

Self Driving Drone Project Paper

AI LAB: COMPUTER VISION AND NLP

APPLIED COMPUTER SCIENCE AND
ARTIFICIAL INTELLIGENCE

Professor:
Daniele Pannone

Students:
Ruben Ciranni
Gabriele Matiddi
Luigi Pizza

Academic Year 2022/2023

Contents

1 Introduction **2**

2 Materials **2**
2.1 The drone system 2
2.2 The obstacle course 3

3 Obstacle Detection **4**
3.1 Data gathering 4
3.2 First approach: Haar Cascade Classifier 4
3.3 Second approach: Faster R-CNN 5
3.4 Third approach: SSDLite 5

4 Automatic Control **5**
4.1 General approach 5
4.2 Setting the PID controllers 6
4.3 Visualization 7

5 Conclusions **7**

References **8**

1 Introduction

This project aims to develop an autonomous obstacle navigation system for Unmanned Aerial Vehicles (UAVs) harnessing different computer vision techniques. The objective is to enable a drone to autonomously fly through a predetermined set of obstacles while avoiding collisions. The drone will be capable of analyzing its surroundings, detecting obstacles, and making real-time decisions enabling safe and reliable navigation.
Throughout this project, several key challenges will be addressed, such as enabling communication between the drone and the host computer, gathering data for training the models and implementing an automatic control system.

2 Materials

2.1 The drone system

This project uses a small commercial quadricopter manufactured by DJI, the Tello, shown in **Figure 1**. The size of the DJI Tello with dimensions of 98×92.5×41 mm and a weight of 80 grams makes the device ideal for a small sized obstacle course. The DJI Tello is equipped with a propeller guard to protect the propeller when it hits an object, a 3-axis gyroscope and accelerometer sensor that is used to obtain the angular position and speed of the vehicle, a barometer sensor and a downward-facing infrared sensor to maintain a precise hovering. The onboard camera contained in this mini drone captures 5-megapixel photos and 720p live video streams, which is ideal for real-time communication with the computer that will give commands to the drone. The maximum flight time is 13 minutes, with a maximum distance of about 100m.

Figure 1: The drone.

DJI released a Software Development Kit (SDK) with various read and write commands over UDP communication. The *djitellopy* library is used for sending commands to the drone via UDP. There are two basic network streams: a dual-ended full-duplex connection for sending and receiving commands between the laptop and drone and a one-way half-duplex connection for streaming video from the drone to a

laptop connection. Once the connection is established, the frames will be processed on the computer side, and the commands will be sent back to the drone in real time. The complete proposed system is composed as follows (also shown in **Figure 2**):

- A DJI Tello drone, which is connected to a laptop via a Wi-Fi network.
- The camera from the DJI Tello, which captures frames and transmits them to the laptop.
- The laptop, which executes computer vision models and sends commands to the drone.

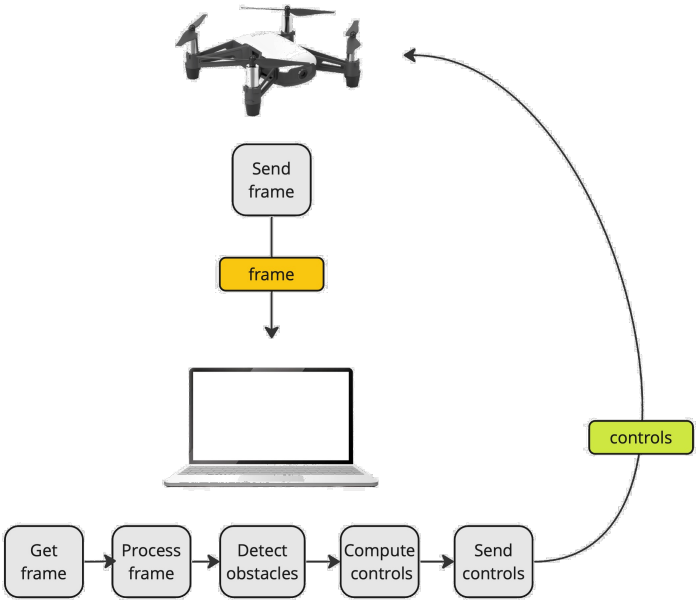

Figure 2: The pipeline of the drone system.

2.2 The obstacle course

The obstacle course is made of cardboard rectangles of size 33×24 cm similar to the one shown in **Figure 3**. The goal of the drone is to pass through all the obstacles without colliding with them. The obstacles are positioned at different heights and widths so that the drone needs to use the implemented automatic control to reposition itself and correctly fly across the course.

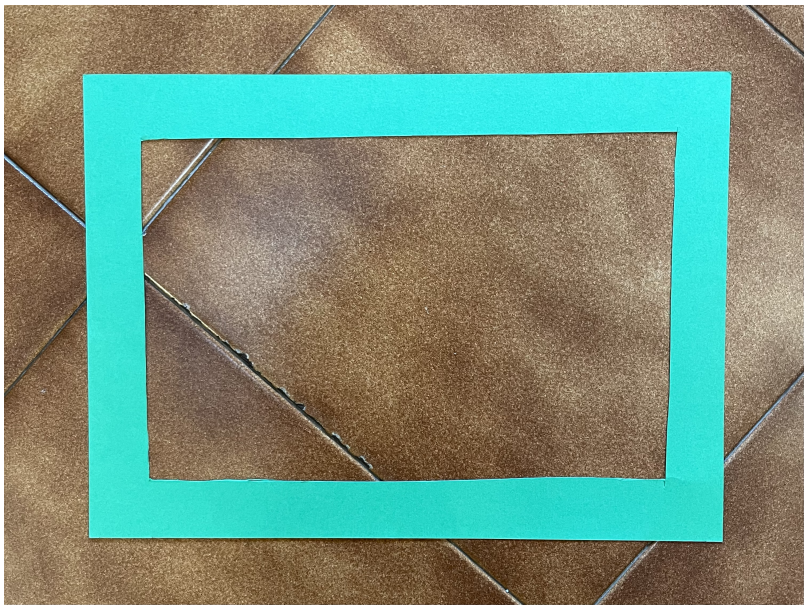

Figure 3: Obstacle example.

3 Obstacle Detection

3.1 Data gathering

The first operation was to gather data for training the models. The data is comprised of two sets of images, positives and negatives. The negatives are a set of images where the obstacle is not present, and they are used exclusively for the Haar Cascade Classifier training, while the positives are a set of images that display the obstacles, and they are annotated using the OpenCV annotation tool. The same annotations are used for the training of all the models.

3.2 First approach: Haar Cascade Classifier

The Haar cascade classifier [3] is a machine learning-based object detection algorithm that is particularly effective in detecting objects within images or video frames. The Haar cascade classifier works by training a cascade of weak classifiers to identify specific patterns or features within an image, known as Haar-like features, which are simple rectangular areas with specific intensity variations.
The training of this model was achieved using the OpenCV tools for Haar Cascade Classifier Training on our dataset of positive and negative images.
It is a good classifier in detecting objects in real-time, due to its great capability of separating object-containing regions quickly, however it is not as precise as more recent deep learning models.

3.3 Second approach: Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) [2] is a popular object detection model used in computer vision tasks. The Faster R-CNN model consists of two main components: a region proposal network (RPN) and a detection network. The RPN generates potential bounding box proposals in an image, while the detection network classifies these proposals into different object categories and refines their bounding box coordinates. Faster R-CNN is known for its accuracy and efficiency in object detection tasks, thanks to the shared computation between the RPN and the detection network.
For this project we fine-tuned a pre-trained model on our custom training data. The colab notebook at the following link shows how the fine-tuning was achieved:

`https://colab.research.google.com/drive/17pVZ1IO6O_dyIhsBiWCW-P-r0Sdjxuac?usp=sharing`

Despite being the most precise among the three models, there is a huge trade-off in performance. This model cannot provide the real-time feedback for the drone system without a powerful GPU.

3.4 Third approach: SSDLite

SSDLite (Single Shot MultiBox Detector Lite) [1] is a lightweight variant of the Single Shot MultiBox Detector (SSD) model, which is widely used for real-time object detection. SSDLite was introduced to address the need for efficient and faster object detection on resource-constrained devices like mobile phones or embedded systems. Similarly to the previous approach, we fine-tuned a pre-trained model on our custom training data.
The following colab notebook shows how the fine-tuning was achieved:

`https://colab.research.google.com/drive/1zxGPGy8ccqepxvUqhS7pVVzn45K5Mazf?usp=sharing`

This last model is the best compromise between efficiency and precision; for this reason, it was chosen as default option for our project.

4 Automatic Control

4.1 General approach

The computer vision model predicts a bounding box for every obstacle in a frame, if any. This information is then used to compute the commands to send to the drone in order for it to correctly fly across the course. In particular, at each received frame, the forward-backward, up-down and left-right instantaneous velocity of the drone is updated. This is achieved by means of three PID controllers, one for each velocity.

4.2 Setting the PID controllers

The proportional–integral–derivative (PID) controller is a widely used control loop mechanism that employs continuous feedback to update some output variables, in our case the three instantaneous velocities. At each iteration of the loop the difference between a desired setpoint and a measured input variable is computed and a correction on the output variable is applied based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.
Suppose that in a given frame of size (w, h) the position of the biggest area obstacle (i.e. the closer one) in the image is (x_0, y_0) and the area of its bounding box is A_{box}. Our goal is that the drone centers the obstacle in the frame (with some correction on the y-axis since the Tello camera points downwards), and gets close enough to it before traversing, i.e.:

1. $(x_0, y_0) = \left(\frac{w}{2}, \frac{h}{2} - k_1\right)$
2. $A_{box} = \frac{w \times h}{k_2}$

where k_1 and k_2 are some empirically set constants.
Once the previous two conditions are met, the drone is ready to move forward and pass through the obstacle. The PID controller of the left-right velocity and up-down velocity will then receive as an input respectively x_0 and y_0 and will have as setpoint respectively $\frac{w}{2}$ and $\frac{h}{2} - k_1$. The PID controller for the forward-backward will receive as an input the area of the obstacle bounding box and have as setpoint $\frac{w \times h}{k_2}$. Note also that the PID controller for the forward-backward velocity is enabled only when the obstacle has been approximately centered; this is to avoid that the object moves out the range of vision of the drone.
The following table indicates the input variables, setpoints and the empiriaclly set parameters k_p, k_i, k_d for each PID controller, assunimg the frame size is (360, 240):

Output	Input	Setpoint	k_p	k_i	k_d
v_{lr}	x_0	$360/2 = 180$	0.23	0.0018	0.067
v_{ud}	y_0	$120 - 60 = 60$	0.4	0.002	0.1
v_{fb}	A_{box}	$360 \times 240/4 = 21600$	0.0005	0.00003	0.000016

Table 1: Parameters of each PID for frames of size (360, 240).

where v_{lr}, v_{ud}, v_{fb} are respectively left-right, up-down and forward-backward velocity.

4.3 Visualization

To analyse the action of the PID controllers, 2 real-time windows are initiated. These windows serve the purpose of visualizing:

1. The live video feed captured by the drone, with bounding boxes drawn around any obstacle present in the scene.

2. A graphical representation, comprised of 3 `matplotlib.subplots`, that illustrates the difference between each input variable and its corresponding setpoint over time (see Table 1).

5 Conclusions

Applying different models will yield different results, some doing better than the others in particular situations, but the drone system has good performance nevertheless thanks to the robustness of the whole structure. An example of a successfully completed obstacle course can be found here.

References

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, 2001.