Big Data Project Report

Mattia Samuel Mancosu matr.65120
Federico Maria Cau matr.65131

April 12, 2019

1 The project

The goal of the project is to create a real-time detection program for racing drone gates, including the visualization of distance and best approaching trajectory.
Gates(1) are particular semicircle shaped 'flags' placed on the ground under which drones pass during races. They act as checkpoints and a series of gates defines a track.

Figure 1: Gate examples.

Racing drone pilots use specific goggles capable of giving the point of view of the drone in real-time thanks to a FPV (first person view) camera. The target users of this project are novice pilots: this tools shows not only a box on the gates but also the optimal trajectory for going through the checkpoint considering its position and distance.
The real-time video feed is provided by a 5.8GHz OTG receiver, which is considered as a webcam by the computer and can be used by libraries like OpenCV. Every frame is analyzed by the program and a video stream is realized adding the detection information to the original feed.

2 Used techniques

The first step was training the model for gate recognition. We decided to use Python and Tensorflow machine learning library for the training, starting from the custom object detection tutorial[1] and adding a few modifications.
Since a complete labeled dataset containing gate images does not exist and since creating a new dataset from scratch specific for one class would take too long, we decided to retrain a preexisting model with 90 classes taken from *Tensorflow detection model zoo*[2]. In particular, the chosen model is *ssd-mobilenet-v2-coco*, which uses the *Single Shot MultiBox* detector since it is very fast and performs real-time detection[3].
Unfortunately, retraining leads to a higher number of false positives because the new model is influenced by the first training. Moreover, at certain distance, perspective and noise condition, the detection becomes difficult, since gates are near-monodimensional objects and their bounding box contains background pixels for the most part.
We labeled about 2000 images containing many types of gates, divided them in training set and test set and created different csv files and tf_records that will be processed by the model. We ran the code on a server built for GPU computation through ssh since the training part is costly in terms of resources. This task has been achieved by using *Docker*[4] for the creation of a container with all the object detection dependencies for our application.
We created a web application using *Jupyter Notebook*[5] in order to manage the training process easily. For this phase we used a GeForce RTX 2080Ti GPU.
Many attempt of training has been made to obtain a good result, each of them with different parameters of the model settings and different separations for training and test sets.

Model evaluation results are shown in the next section.
Once the training was completed we used the following code[9] for detecting gates in real-time using the webcam with appropriate modifications for showing also the trajectory and the distance.
The receiver for the video feed sent by the drone is a ROTG02 by Eachine. It is capable of scanning for 5.8GHz analog video signals and send them by USB with a latency close to the latency detected in FPV goggles.
The detection is made on this video stream since the ROTG02 module is considered as a second webcam by the computer. Gates are shown by green box with the label "GATE" followed by a number which is the distance to the drone. This measurement is not universal since it is calculated as proportion between the dimension of the screen and dimension of the gate, and the optical sensor mounted on the drone can differ leading to different values for the same real distance. An ideal curve for the correct approach to the gate is also shown to the use. In case of multiple detection the trajectory appears only for the closest one since it is supposed to be the next of the track.

3 Results

The evaluation gave the following results:

- classification_loss: 1.390427;
- localization_loss: 0.292649;
- Pascal_Boxes_PerformanceByCategory-AP@0.5IOU(gate): 0.927998;
- Pascal_Boxes_Precision-mAP@0.5IOU: 0.927998.

More information about the first and the second parameter can be found in this website[6] and for the last two in those websites[7][8]. Briefly, the loss values tell how predicted bounding boxes differ from real ones. We obtain about 93% on the precision/performance value for this model. Since we obtain loss values close to zero and an high detection value we can affirm that our model is effective and efficient.
When applying our model to this webcam detection program[9], we have achieved very good results, considering also:

- The object detection process is complicated by the presence of noise and light variation of the image. The noise is always present since the video feed is analog and transmitted by ether. This is necessary in order to have the minimum possible latency. Of course the consequence is the appearance of false positives, although they lasts for one frame only usually.
- The overall latency is improvable but sufficient for learning flights. The response time from the stick movement of the radio controller and the perception of the movement of the drone is about 0.2 seconds (using a laptop with a GPU in our case). It is an excellent time since it is the addition of latency from radio to drone, movement of the drone, video signal latency, module reception, detection and output drawing.

Here we can see some examples of detection:

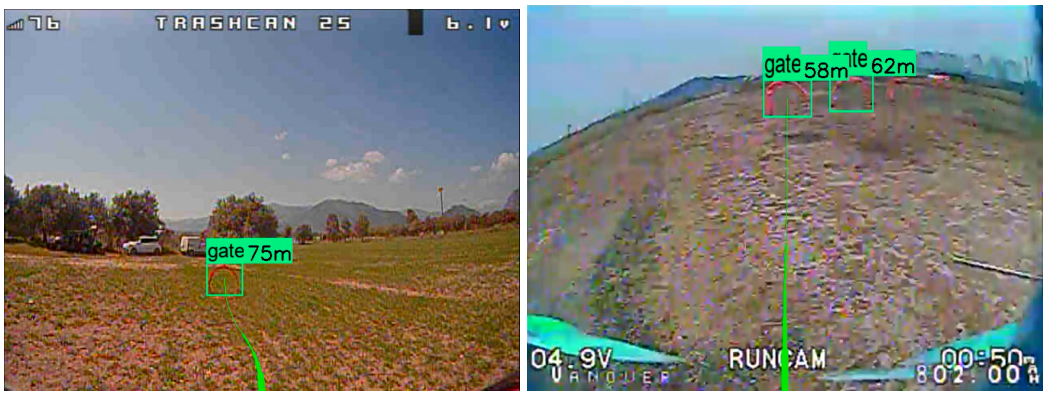

Figure 2: The image on the left shows the detection of a gate 75 meters away from the drone and the suggested trajectory. On the right image we can see the detection of two gates, their distance and the trajectory of the closer one.

References

[1] Custom object detection with TensorFlow: go to site

[2] Tensorflow detection model zoo: go to site

[3] Single Shot Multibox Detector go to site

[4] Docker go to site

[5] Jupyter Notebook go to site

[6] Localization and classification loss go to site

[7] mAP (mean Average Precision) for Object Detection go to site

[8] Understanding the mAP Evaluation Metric for Object Detection go to site

[9] Detect Objects Using Your Webcam go to site