Team Deadlock - Test 2 Technical Report

March 2019

1 Algorithm Summary

For our Test 2 submission, we used a light-weight modified **Mask-RCNN** [1] based architecture for detecting the flyable regions because it provides the best trade-off between mAP score and inference time. Our algorithm is tested on both NVIDIA GTX 1080 locally and p2.xlarge Deep Learning AMI instance on AWS. With our network, we obtained an mAP score of 88.90% on the leaderboard and inference time of 0.05s (GTX 1080) and 0.13s (p2x.large). We have also tested our algorithm in our own custom Unity-based simulator with a representative 3D DRL gate model, and it has proven to perform real-time with high accuracy. Mask-RCNN also provides a pixel-by-pixel confidence mask which we intend to use to find a confidence-weighted bounding box center for trajectory planning.

We are confident that the elements of Mask-RCNN is key in robustly identifying the flyable region during racetime because it mimics how a racing pilot would extract the flyable region. First, the general contour of the gate is detected, followed by a confidence-based segmentation of the flyable region to create the region's bounding box. The below figure represents that workflow of our algorithm at each stage.

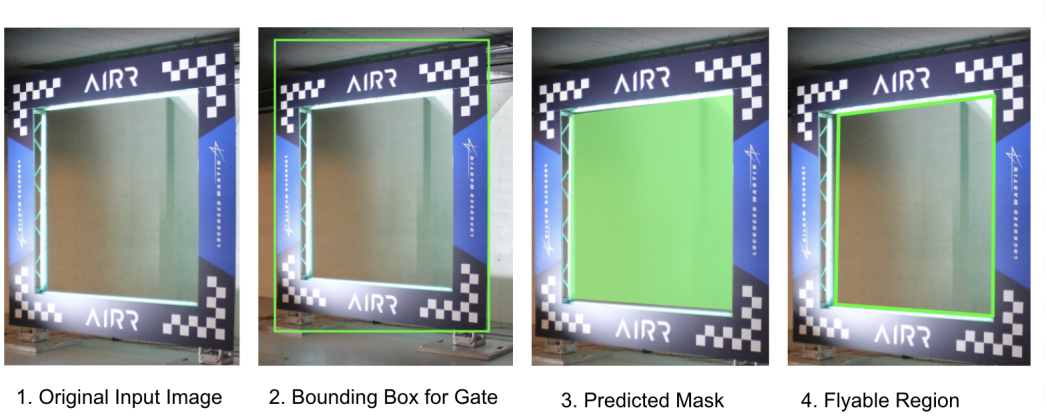


Figure 1: Flow of the detection pipeline. Although not visually depicted here, the predicted mask (3.) is assigned a pixel-by-pixel confidence score on whether it's a flyable region.

2 Implementation Details

2.1 Data Augmentation

On top of regular data augmentation techniques such as rotation, blurring, color augmentation, etc, we have additionally developed a Unity simulator to both generate and test on images in other unseen scenarios that we would define as edge cases. More specifically, we wanted to generalize the learning of the flyable region for angles not available in the training set. Some examples are shown below in Figure 2. Our simulator is also used as a visual sequence to test real-time detection.

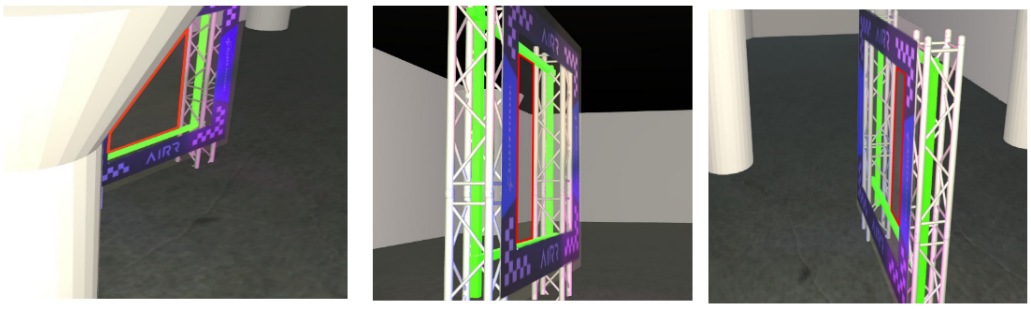


Figure 2: Automatically generated images and bounding boxes of extreme gate poses generated from our simulator.

2.2 Model Training

We utilized the Tensorflow Object Detection API to train our Mask-RCNN model for gate detection. We customize the model structure to better capture gate images' characteristics and speed up inference by (i) Reduce the number of final output detection to get a light weight model; (ii) Increase mask size in the mask head to get more accurate probability mask; and (iii) Increase the weight of mask loss to favor accurate mask pixel assignment versus localization and classification.

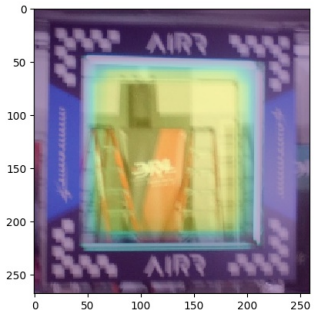
We trained two lightweight architectures as our backbone network for feature extraction process, which were MobileNet [2] and Inception v2 [3]. As we fine-tuned our hyperparameters and training procedure, our models were chosen to be trained for 60,000 steps with a learning rate of 0.0005 which is then decayed every 30,000 with rate of 0.10. A mask size of 105 x 105 produced best results on our validation set. Through thorough design iterations and testing we decided that the InceptionNet backbone provides the best trade-off out of all models, as well as being the safest for real-world implementation because of an extremely low false-positive prediction rate (less than 0.1%).

2.3 Flyable Region Extraction from Pixel-Confidence Segmentation

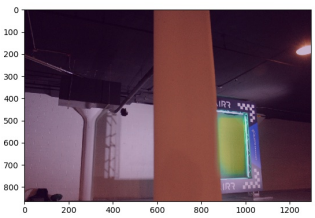
To get the final quadrilateral coordinates, we performed additional post-processing to the probability mask using OpenCV. The probability mask is firstly threshold-ed by setting those pixels having probability larger than 0.2 as 1. Afterwards, the binary mask is resized to match with the detected bounding box size. Contours are extracted from the binary mask, then an approximate polygon estimator is used to estimate the 4 quadrilateral coordinates as the bounding box.

2.4 Edge Case Analysis

We have investigated edge cases in the dataset that have motion blur and occluded gates. Fig.3 below are two examples of the confidence masks on edge cases. As you can see here, majority of the main flyable region is assigned a high confidence score (yellow), and the confidence starts to decrease as it gets closer to the frame edges (color fades to teal).



(i) Blurred Input Image



(ii) Occluded Input Image

Figure 3: Confidence masks on edge cases in the dataset. Yellow segmentation pixels represents high confidence, whereas teal pixels represents low confidence; you can see the gradient going from the middle of the flyable region outward.

2.5 Multiple Gate Detection

Another reason why we pick Mask-RCNN is its native support of multiple gate detection. We can adjust the maximum number of detected objects to balance speed and multiple gate situations. The only required change is a parameter in the Mask-RCNN Inception V2 config.

3 Hardware Deployment

Figure 4: Code generation flow

We are exploring solutions to deploy our Mask-RCNN based gate detection/flyable region detection algorithm on the target board in a drone. Currently, we are adopting TVM for graph optimization, quantization and high quality CUDA code generation. This will make sure our model achieves optimized real time detection and segmentation performance on NVIDIA Jetson.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*.

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*.