Approach

Given this is an object detection problem we initially looked at the handful of well known object detection algorithms like YOLO, SSD and Faster-RCNN. However because the bounding box we are trying to predict is actually a polygon instead of a rectangle, none of these algorithms would work off the shelf as they are all designed to work with rectangles.

In addition, because we are not detecting multiple classes nor multiples instances of a class these algorithms didn't seem like a good fit. Instead we opted for a simple regression model that would predict 8 numbers, those being the four (x,y) coordinates that describe the polygon that is the flyable region that the drone can safely navigate through.

Data

Given the quality of the ground truth labels provided was quite poor we joined together with several of the other teams to collectively relabel all the gates by hand. Our team then took this set of community labels and touched up another 1000 that were still a little off. In the end we had a very accurate ground truth label set with which to train.

In addition to cleaning up the labels we also developed scripts to reorder any labels that were not in the proper order of starting in the top left and proceeding clockwise. Because our model relied on a consistent ordering of the coordinates, this was essential.

Per the guidance from the organizers, we dropped all images where one or more of the corners was obstructed. We also opted to exclude class detection from the model due to the fact that the leaderboard test set contained no images without a gate, so we also excluded the handful of images in the training set where the gate was completely obstructed.

Model

Our model uses transfer learning and starts out as a resnet50 pre-trained on imagenet. This gives a good built-in understanding of underlying lines and shapes that we can leverage. Because the original model is setup for classification across 1000 categories we replace the final layers with a simple linear layer that outputs 8 continuous values, representing the four (x,y) coordinates that describe the flyable region of the gate. Basically turning the classification model in a regression one.

The loss function also has to change as the cross entropy loss typically used with classification will no longer work. L1 and L2 are common loss functions to use with regression problems and we chose L2 loss (aka mean squared error) as it was observed to work better than L1.

Images are scaled down 2x from their original size of 1296x864 to 648x432 and fed to the model as a rectangle. This is important as it prevents any horizontal distortion that that would result from trying to squish the image into a square.

Training

Training is initially done on only the final layer for about four hours, leaving the weights of the rest of the network frozen. Then the entire network is trained for about 20 hours more. After 24 hours the model begins producing decent results. Additional training beyond this continues to slightly refine predictions.

In addition to calculating loss we also tracked a metric we called mean pixel offset, which used the Pythagorean Theorem to measure the mean number of pixels the predictions were away from the targets. This provided a simple human interpretable measurement of how training was progressing and was much faster to calculate then mAP, which we calculated only occasionally during training.

Libraries

We use the fastai library which is a high level API that sits atop pytorch. Similar to keras, fastai provides several wrappers that reduce the need to write a lot of boilerplate in pytorch. It also implements several useful tricks to speed up training such as: learning rate finder, one cycle learning and discriminative layer training.

Future Work

We have some ideas for improvements that can be made before deploying the model in a real racing competition. It would be better to add single class detection of a gate so the drone doesn't try to erroneously predict a gate where there isn't one. This is quite easily done by adding another node to the output of the model to represent the confidence that the class exists and then modifying the loss function to include the cross entropy loss over this class detection.

The model could also be made more robust and training time could be reduced by giving the model an implicit understanding of geometry. For example, it should be possible to begin with a fixed sized square and have the model predict the following values:

```
center(x, y)
rotate(degrees)
scale(factor)
perspective_transform(x, y)
```

that would transform the fixed square into the four (x,y) coordinates that describe the polygon. This would mean only predicting six numbers instead of eight and also prevent illegal shapes like when the points cross, which is possible with the current model given each point is being predicted independently.

Another useful enhancement would be to add detection of multiple gates as this is something the drone is likely to encounter during competition. Detecting two or three gates should be possible with the current regression setup but beyond that will likely become too unwieldy and a different approach would have to be explored. However it seems unlikely it would need to detect more than two or three gates even in competition.