RISE-Q 6DoF Gate Pose Estimation

Algorithm Submission for Flyable Area Detection for Test 2 Alpha Pilot AI Challenge

I. Introduction

Human sight affords us estimation of any object's position, orientation and relative scale in space. These functions are fundamental in daily life and are leveraged in drone racing by human pilots. Drone pilots fly drones through -partially visible, obscure- gates, performing high-speed high-accuracy maneuvers against time, in accordance to the track layout, and avoiding obstacles and collisions.

For an autonomous drone to have a fair race against human pilots, it is logical to think that it must display similar perceptual skills, namely being capable of robust, fast gate and obstacles detection, and estimation of their 6DoF pose and scale. With these ideas, we propose an algorithm that is robust to extreme lighting variations, blurring and obstructions, does not depend on fiducials, performs in real time, can be extended to multi-object detection (drones, obstacles for collision avoidance) and can be trained to achieve high accuracy. Added to this, it is trained fully on synthetic images.

Our team got a Test 2 score of 82.06% @ 34.89fps on a GeForce GTX1050Ti GPU. By reducing the input resolution, we achieved up to 37fps in our tests, at the expense of some accuracy. Although for Test 2, algorithms that rely on fiducials might provide higher scores, in the long-term we believe a general, robust, fast and scalable approach as our is indispensable for success both in drone racing and general drone navigation contexts.

II. High Level Decisions and Reasoning

A. Why 6DoF gate pose estimation?

In the context of Test 2, the objective is to recognize the flyable area in images containing the ADRL gate. However a broader objective is to estimate gates' pose to actually traverse them successfully. For this reason, we decided to implement an algorithm that detects gates in images and estimates its 6DoF pose. This idea was proven in [1]. For the purpose of Test 2, the projection of the flyable area into the camera plane is calculated and the 4 points defining the bounding box are reported.

B. Robustness of our Algorithm

In general, objects do not have clear fiducials, and other drone racing contexts [2] do not provide them. Furthermore, other drones and obstacles must be avoided even if they are not provided with special markings. In this sense, we opted for an algorithm capable of making gate pose estimation without need for fiducials, clear lighting or full visibility. State of the art Convolutional Neural Network (CNN) have shown good generalization and speed, and thus we leverage well known architectures for this task [3],[5]. We discuss details in Section III.

C. Generalization of our Algorithm

Training of the CNN was approached using only domain randomized and photorealistic synthetic images. Good generalization is achieved by making real examples appear to the CNN as another randomization. This is demonstrated by our CNN, which obtained a high score on the leaderboard dataset, even though it never saw a single real image during training. This approach to training and domain-transfer has been proven in [4].

III. Technical Description

We made use of singleshot6Dpose [3]. This is a -CNN-based algorithm for general 6DoF pose estimation. It is based on the well-known YOLOv2 [5] architecture which achieves state-of-the-art precision and real time speed. The algorithm takes a single RGB image as input and predicts an object's 6DoF pose as its output. To predict a 6DoF pose, the algorithm works in 2 steps: First, the network predicts the 2D projection on the camera plane of the 3D bounding cube of the object. Second, it uses the predicted points to estimate 6DoF pose using a PnP algorithm and camera calibration parameters. The implementation is done in Python 3.5.2 and using pyTorch 0.4.

A. Formulation

The problem of predicting the 2D projection of the 3D bounding cube of an object is modelled as regression problem from image pixels to bounding cube coordinates and class probabilities. The 6DoF gate pose estimation problem is formulated as the estimation of the 2D projection (over the camera plane) of the 3D bounding cube corners and centroid associated with the gate. A total of 9 control points. Given these 2D projections, a PnP algorithm estimates the 6DoF pose of the gate.

B. Network Architecture

The input image is divided as an S x S grid (S = 13). It is processed with a CNN, and the output is a 13x13xD (D = 9*2 + 1 + C) tensor, which contains the estimation of the 2D projections of the 9 control points (9x2), one object confidence value and C class probabilities. The network architecture is presented in Figure 1. The object confidence score should reflect the presence or absence of an object. This is, higher values if the gate is present and smaller ones if it is not. In practice, the confidence function $c(x)$ (1) is applied to each one of the estimated 2D control points, and the mean value of the 9

points is used as final confidence score. $D_T(x)$ represents the Euclidean distance between the predicted 2D projections and ground truth, d_{th} = 30 pixel is a cutoff value for the distance, after which the confidence is set to zero (meaning no object present).

$$c(x) = \begin{cases} e^{\alpha\left(1-\frac{D_T(x)}{d_{th}}\right)}, & if\ D_T(x)<d_{th} \\ 0 & otherwise \end{cases} \quad (1)$$

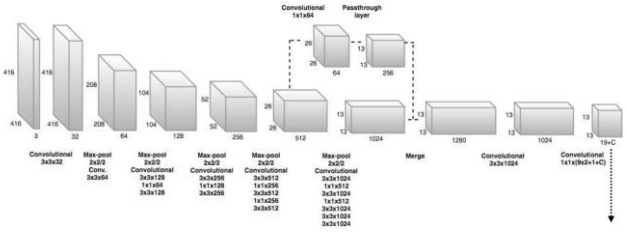

Figure 1. Network architecture. Extracted from [3].

C. Training

To learn the parameters of the network, the loss function L_θ (2) is used. Mean-squared error is used for both coordinate and confidence losses, and cross entropy for classification loss.

$$L_\theta = \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2] + \lambda_{conf}\sum_{i=0}^{S^2}\sum_{j=0}^{B}(c_i-\hat{c_i}) + \lambda_{class}\sum_{i=0}^{S^2}\sum_{j=0}^{B}(y_i\log(\hat{y_i})+(1-y_i)\log(1-\hat{y_i})) \quad (2)$$

D. Dataset Generation

Ground truth 2D bounding box data is not enough for our network. Ground truth rotation and translation of the object is also necessary. Since we do not have a real physical gate and if we had the labeling process would be long and tedious, we decided to create a synthetic dataset using the Nvidia Deep Learning Dataset Synthesizer and Unreal Engine 4, as described in [4]. It is important to note that, to the best of our knowledge, this would be the first time singleshot6dpose[3] is trained purely on synthetic images. In total, 25,000 domain-randomized and photorealistic images of the gate were created. The dataset was augmented with varying jitter, shape, saturation, hue, noise and exposure during training. Ground truth rotation, translation and 2D projection of the 3D bounding cube of our simulated gate is obtained from Unreal Engine 4. Some samples images are presented in Figure 2.

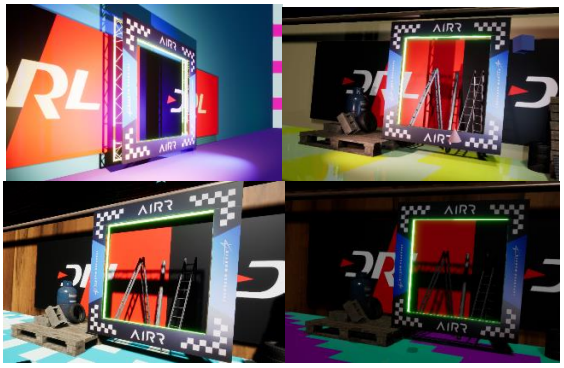

Figure 2. Sample synthetic images generated

E. Pose Prediction

At test time, the class specific confidence value is calculated by multiplying the class probabilities with the object confidence. This is done for each cell in the SxS grid. Since several cells can predict the presence of the same object (albeit with difference class probabilities), and since the object can in fact occupy various cells, the 3x3 neighborhood to each cell is accounted for. The weighted average of the 2D predictions in the neighborhood is taken and reported as result. The weights are the confidence scores from each cell. These final 2D predictions are then passed to a PnP algorithm which estimates their 3D position, and finally the object's 6DoF pose.

IV. Results

We got 82.06% score @ 34.89fps in Test 2 Leaderboard, using a GeForce GTX1050 Ti GPU. Figure 3 presents some samples of the bounding cube prediction (green lines) and flyable area calculation (red lines) on the Leaderboard dataset performed by our algorithm. Note the algorithm is robust to lighting, blur and considerable occlusion.

Figure 3. Bounding cube and flyable area prediction results.

V. Future Development

This is just the beginning of our perception algorithm. In terms of improving gate's pose estimation accuracy, we will leverage pose estimations obtained by using the checkerboard patterns (when available) and fuse them with results from our network. We follow this paradigm because we need the robust, fast gate pose estimations even when fiducials are not visible or useful, and leverage them when they are. Furthermore, we will reduce the number of layers while keeping high accuracy by using residual blocks [6] as in [7],[8] and decreasing input image size. This will increase both speed and accuracy of our network, while reducing requirements for later processing on a Nvidia Xavier platform. Additionally, we will train our network to detect other drones together with their 6DoF pose. This will be critical for collision-free, fast navigation. Finally, to account for real-world effects that synthetic images cannot represent, we will include real images in our training dataset, using our algorithm's prediction as their ground truth labels.

VI. References

[1] 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. Retrieved from http://arxiv.org/abs/1810.06224

[2] Moon, H., et. al. (2018). Challenges and implemented technologies used in autonomous drone racing. Intelligent Service Robotics. Springer-Verlag GmbH. https://doi.org/10.1007/s11370-018-00271-6

[3] Tekin, B., Sinha, S. N., & Fua, P. (2017). Real-Time Seamless Single Shot 6D Object Pose Prediction. *Revue Du Marche Commun et de l'Union Europeenne*, (407), 229-x1. https://doi.org/10.1109/CVPR.2018.00038

[4] Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., & Birchfield, S. (2018). Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects, (CoRL), 1–11. Retrieved from http://arxiv.org/abs/1809.10790

[5] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017–Janua*, 6517–6525. https://doi.org/10.1109/CVPR.2017.69

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.1109/CVPR.2016.9

[7] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://doi.org/10.1109/CVPR.2017.690

[8] Loquercio, A., Maqueda, A. I., del-Blanco, C. R., & Scaramuzza, D. (2018). DroNet: Learning to Fly by Driving. IEEE Robotics and Automation Letters, 3(2), 1088–1095. https://doi.org/10.1109/LRA.2018.2795643