A Deep-learning-aided Automatic Vision-based Control Approach for Autonomous Drone Racing in Game of Drones Competition

Donghwi Kim[1,*] dw95kim@kaist.ac.kr
Hyunjee Ryu[2,*] localryu@kaist.ac.kr
Jedsadakorn Yonchorhor[1,*] j.yonchor@kaist.ac.kr
David Hyunchul Shim[1] hcschim@kaist.ac.kr
[1] *Unmanned System Research Group (USRG), School of Electrical Engineering, KAIST, South Korea*
[2] *Unmanned System Research Group (USRG), Division of Future Vehicle, KAIST, South Korea*
[*] *Denotes equal contribution*

Editors: Hugo Jair Escalante and Raia Hadsell

Abstract

In Game of Drones - Competition at NeurIPS 2019, this autonomous drone racing requires the drone to maneuver through the series of the gates without crashing. To complete the track, the drone has to be able to perceive the gates in the challenging environment from the FPV image in real-time and adjust its attitude accordingly. By utilizing deep-learning-aided detection and vision-based control approach, Team USRG completed the tier 2 challenge track passing the whole 21 gates in 81.19 seconds, and complete the tier 3 challenge track passing the whole 22 gates in 110.73 seconds.

Keywords: Drone Racing, Deep Learning, Vision-based Control

1. Introduction

The drone racing, which requires agile maneuvering of the drone through the series of the gate, has skyrocketed as a popular attraction that challenges the robotics enthusiasts. Using the high-fidelity simulation, *AirSim*, (Shah et al., 2017), Game of Drones - Competition at NeurIPS 2019 (Madaan et al., 2020), which encourages the racing with intelligent algorithms, has drawn much attention from both roboticists and artificial intelligence researchers.

Our team participates in two tiers of the challenges. In the tier2 and tier3 challenges, given the first-person view (FPV) camera images from the RGB facing forward camera, the ground-truth odometry of the drone and the ground-truth pose of the gate with perturbations, the drone needs to complete the track by flying through the series of the gates as fast as possible without crashing to the environment. Moreover, in tier3 challenges, the opponent drone is present in the racing, therefore, the position of the opponent drone from the FPV image also needs to be considered so that our drone is not disqualified by crashing with the opponent drone.

In these challenges, the problems could be divided into two parts which are the perception problem and the control problem. First, the perception problem is how to reliably detect the gate and the opponent drone when our drone is moving at the high speed. Second, the control problem questions on how to properly control the drone to pass through the gates. Although the problems are twofold, these problems could not be decoupled because the drone needs to adjust its pose according to the perception of the gates. Hence, our team utilizes the deep-learning-aided automatic vision-based control approach to solve these problems.

2. Related Work

2.1. Gate detection method

The detection of the gates in the *AirSim* environment is a challenging task due to many factors such as severely differing lighting conditions, occlusion of the gates, nonconformity of the gates, etc., as shown in Figure 2. Therefore, the pixel-based algorithm, such as color or shape detection (Cho and Shim, 2017), would not be suitable for this environment.

Thanks to the breakthrough of Convolution Neural Network (CNN), deep-learning-based detector shows promising performance in terms of both accuracy and speed of the inference in the aforementioned challenging environments (Jung et al., 2018b).

2.2. Vision-based control for drone

In IROS 2018 Autonomous Drone Racing (ADR) competition, Jung et al. (2018b) has proposed the framework which combines the deep-learning-aided gate detection with Line-Of-Sight (LOS) guidance to maneuver the drone through the center of the gate.

In our work, we improve the framework by combining the gate depth estimation, which will be explained in Section 3.3.2. By knowing how far the drone is from the targeted gate, the drone heading speed could be adjusted resulting in faster flight and smoother trajectory, hence, improving the performance overall.

3. Approaches

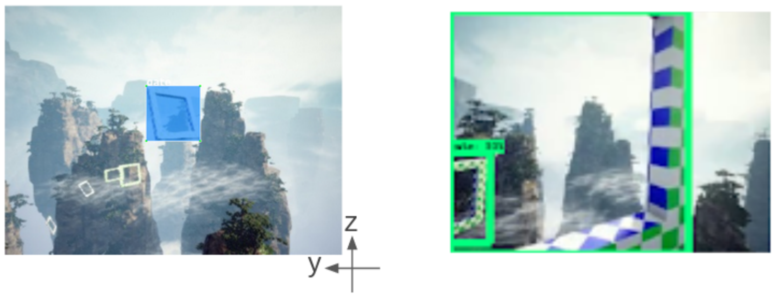
3.1. Frame Conventions

In the following part, there are two important conventions: the drone frame and the image frame. The drone reference coordinate frame is defined as North-West-Up (NWU). In other word, X-axis aligns with the front of the drone, Y-axis aligns with the left side of the drone and Z-axis align with the top of the drone. For the FPV image frame, the vertical axis is defined as Z-axis and the horizontal axis is defined as Y-axis, as depicted in (a) in Figure 1, with the origin at the center of the FPV image. This is to coincides both references frame for simplicity.

3.2. Perception Part

In this racing, *MobileNetSSD*, because of its fast inference capability, is employed to robustly detect the gates in real-time. *MobileNetSSD* is a combination of convolutional neural network (MobileNet, Howard et al. (2017)) and objects detection network (SSD, Liu et al. (2016)). Therefore, it needs training data: the RGB images of the gates in different environments, which are collected from FPV image of several flights in the simulator.

Assuming that the drone could see the gate in the sequential order it needed to navigate through, we label the training data by creating the ground truth bounding box only at the nearest gate in the image as shown in (a) in Figure 1.



(a) Labeling data (b) Detection Result

Figure 1: FPV image: the image frame is defined to be coincided with the drone frame.

To make our gate detector more robust, the collected training data are augmented in several ways such as adjusting brightness, flipping horizontally, random cropping, etc. Then, those augmented data are used to train the *MobileNetSSD*. The data augmentation and network training are done in the *Tensorflow* framework (Huang et al., 2017).

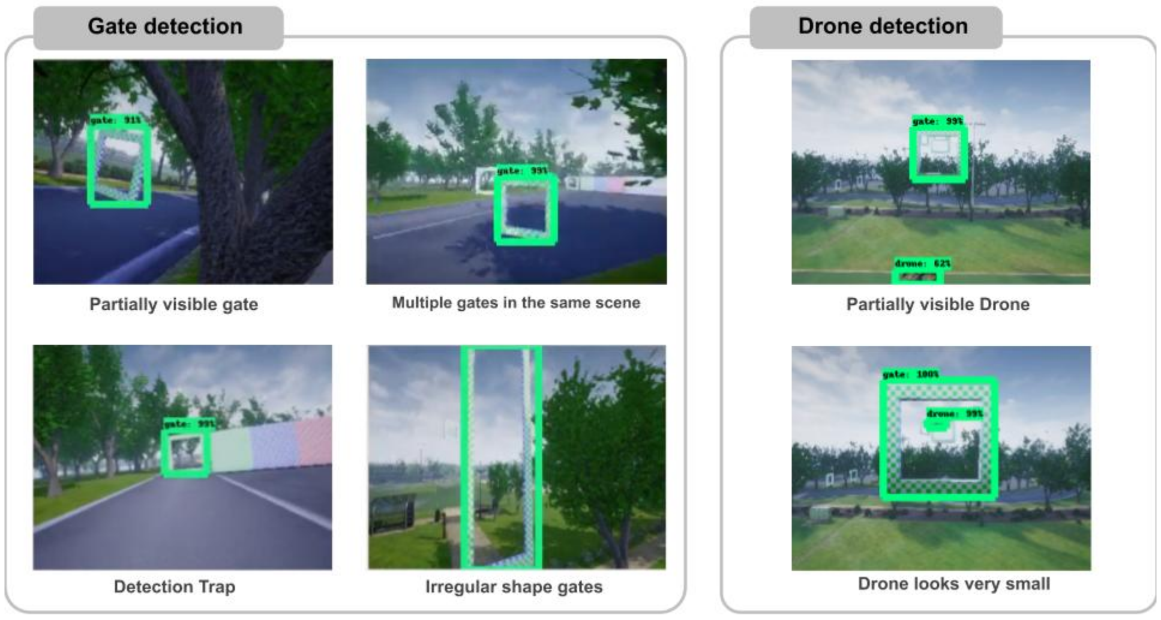


Figure 2: Gate and drone detection result using two separated *MobileNetSSD*.

The gate detector outputs the location of the gate by creating the bounding box around it as shown in Figure 2. Ideally, the bounding box should tightly fit with the gate such that the center of the gate coincides with the center of the bounding box, which is later used as the reference point to the controller. In our approach, the bounding box is created at the gate only when it is detected with more than 90% confidence.

Although the employed neural network model is capable of marvelously detecting the gates throughout the race, in rare cases, it could detect more than one gate at a time (false positive), as shown in (b) in Figure 1, for instance. In this case, the drone could not properly track the next target gate. Therefore, our algorithm is designed such that the drone should navigate through the gates in sequence such that it always tracked the nearest gate first in the case of multiple detections of the gates. This is archived by choosing the biggest bounding as the target in the presence of multiple bounding boxes. Algorithm 1 summarizes the gate detection procedure.

Algorithm 1 Gate Detection Algorithm

```
1. While receiving a FPV image
    2. Feed the image to the trained MobileNetSSD (gate detector)
    3. Create the bounding box at the gate detected with more than 90% confidence
    4. If there is more than one bounding box
           Calculate the area of each bounding boxes
    5. Calculate the center of the bounding box with the largest area using vertices
    6. Return the center of the bounding box
```

3.3. Control Part

In the control part, we design a hybrid control algorithm: position control and velocity-yaw control. Given the noisy ground truth of the gate poses, the approximate locations of the gates with respect to the world frame are known. Therefore, in the scenario that the drone could not detect the gate, we use position control to maneuver the drone to the approximate location of the gate so that the drone could detect the next target gate. On the other hand, if the drone could detect the gate, then it follows the velocity-yaw control algorithm.

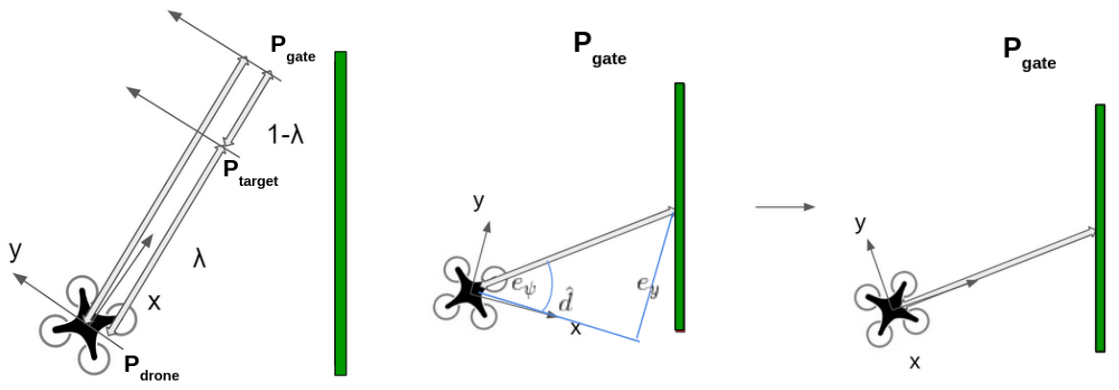
3.3.1. Position Control

The position control is the command to fly the drone directly to the target position with respect to the world frame using airsimneurips API. This mode of controller is only used when the drone could not detect the next target gate. In such cases, the position control flies the drone to the appropriate position which allows the drone to the detect the next target gate.

In the racing, given the noisy ground-truth pose of the gates, the drone knows roughly about the positions of each of the target gates. Therefore, the drone needs to keep track of the number of gates the drone has passed through to determine the proper next position it should go to look for the next target gate. However, instead of going directly to the perturbed position of the next target gate, the target position is calculated by the weighted average of the current drone position and the noisy ground-truth position as illustrated in Figure 3(a) to ensure that the drone would be in the position that it could detect the next target gate using the perception algorithm described in Section 3.2.

$$P_{target} = \lambda * P_{drone} + (1 - \lambda) * P_{gate}, \tag{1}$$

where P_{target}, P_{drone} and P_{gate} are the target, drone and (noisy) gate position, respectively and $\lambda \in [0, 1]$ is the weight parameter.



(a) Position control (b) Yaw Angle Control

Figure 3: Control Method (top-view): a) the target point is calculated using the current position of the drone and the noisy ground-truth pose, where P_{gate} is the noisy ground truth pose of the gate and m is the weight parameter. b) X-axis of the drone is the camera direction and also drone heading vector

3.3.2. Velocity-yaw Control

Inspired by the work of Jung et al. (2018b), Jung et al. (2018a), Jung et al. (2018c) our team makes use of the classical Proportional-Derivative (PD) controller (Ang et al., 2005) to command the linear velocities in 2 axes, namely, Y, Z with respect to the drone frame, and yaw angle of the drone. Then, the velocity of the X-axis is set proportionally to the distance of the drone to the target gate obtained from the gate depth estimation algorithm explained in the next topic. The velocity and yaw commands from the controller are then inputted to the airsimneurips API to maneuver through the center of the gates.

Gate Depth Estimation To improve the speed of the flight, we want to set the X-axis linear velocity to be proportional to the distance from the target gate. Therefore, knowing the distance of the drone from the gate is crucial to our approach.

From the output of *MobileNetSSD*, we could obtain the gate's pixel width and pixel height in the image frame. We model the relationship between the distance from the drone to the detected gate, called depth, $\hat{d}$, and the width of the detected gate using the exponential model as shown in Equation (2):

$$\hat{d} = K_1 * e^{K_2 * x}, \tag{2}$$

where K_1 and K_2 are the parameter to be obtained from the collected data, x is the measured width of the gate (in pixel), $\hat{d}$ is the estimated depth (in meters).

To obtain the parameter in Equation (2), we collected various of the gate width data with the distance by flying the drone heading straight toward the gate. Then, we use the regression technique to fit the model to the collected data to obtain the parameter K_1 and K_2 that best fit the model with the collected data.

Velocity Control The controller obtains the coordinate of the center of the bounding box in the image frame, which is assumed to be very closed to the center of the detected gate in the image frame, from the gate detector stated in Section 3.2. To align the center point of the gate to the center point of the image frame, the controller tries to minimize the position error $e_{position}$, calculated from the pixel distance between the two points in the image frame.

$$\begin{aligned} e_{position} &= (f(e_y), f(e_z)), && (3)\\ &= (f(\frac{C_{By} - C_{Iy}}{W_g}), f(\frac{C_{Bz} - C_{Iz}}{H_g})), && (4) \end{aligned}$$

where (C_{By}, C_{Bz}) are the center of the bounding box in Y-axis and Z-axis, (C_{Iy}, C_{Iz}) are the center of the FPV image in Y-axis and Z-axis, and (W_B, H_B) are the width and the height of the bounding box of the detected gate, respectively. The error is scaled by the size of the bounding box to reduce the sensitivity of the movement of the center of the bounding box, as the drone flies toward the gate. Moreover, f is a function to impose Dead Zone concept which is defined as

$$f(x) = \begin{cases} x, & |x| < \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

The Dead Zone is defined near the center of the FPV image to avoid overshooting and to decrease sensitivities. In the image frame, if the center of the detected gate has entered the Dead Zone, the position error will be nullified. This reduces the oscillation of the flight toward the gate.

Desirably, the displacement error of the two points should approach to zero to ensure that the drone flies through the center of the gate. PD controller plays an important role to determine the velocity command in the Y-axis, u_y, and Z-axis, u_z, with respect to the drone frame to minimize such an error

$$\begin{aligned} u_y &= K_{Py} * e_y + K_{Dy} * \dot{e_y}, && (5)\\ u_z &= K_{Pz} * e_z + K_{Dz} * \dot{e_z}, && (6) \end{aligned}$$

where (K_{Py}, K_{Pz}) and (K_{Dy}, K_{Dz}) are the proportional gains and derivative gains in the Y-axis and Z-axis that need to be tuned. In our approach, the velocity commands are updated with 50 Hz frequency.

The command of the heading speed, u_x, is set proportionally to the estimated distance from the drone to the gate, $\hat{d}$, defined in Equation (2),

$$u_x = K_3 * \hat{d} + K_4, \tag{7}$$

where K_3 and K_4 are parameters to be tuned.

Yaw Angle Control For the drone to fly through the center of the gate, the drone heading vector needs to align with the vector from the drone to the center of the gate as shown in Figure 3(*b*). Since the ground truth pose of the drone is given, we could compute the error of yaw angle, e_ψ, using the estimated depth, $\hat{d}$, defined in Equation (2), and the displacement of the center of the gate and the image frame in the Y-axis, e_y, defined in Equation (3),

$$e_\psi = \arctan \frac{e_y}{\hat{d}} \tag{8}$$

In the same manner with position error, the controller tries to minimize the angle of drone heading vector and drone-gate vector by executing the yaw command

$$u_\psi = K_{Pe_\psi} * e_\psi + K_{De_\psi} * \dot{e_\psi} \tag{9}$$

where K_{Pe_ψ} and K_{De_ψ} are the proportional gains and derivative gains which require tuning, respectively.

3.4. Drone Overtaken Part

In tier 3, we also need to consider the presence of the opponent drone and try to outrace it without crashing.

3.4.1. Drone Detection

The detection of the drone is done in the same manner with the gate detection explained in Section 3.2. However, using the same network model to detect both the opponent drone and the gates cause imbalanced data. Therefore, two separate networks are used for detection: one for drone detection and one for gate detection.

3.4.2. Strategy

We considered three scenarios from the RGB input image.

First, only the drone was detected in the image. In this case, we separated the image into two areas (altitude separation) and set the collision risk area. Then, if the drone detected in the considered area, do the altitude separation. If the opponent drone is detected higher than the center of the image, our drone lowers the altitude. Otherwise, the drone will raise the altitude.

Second, only the gate was detected in the image. In this case, we could consider this challenge as the tier 2 challenge, hence, the same control algorithm is employed.

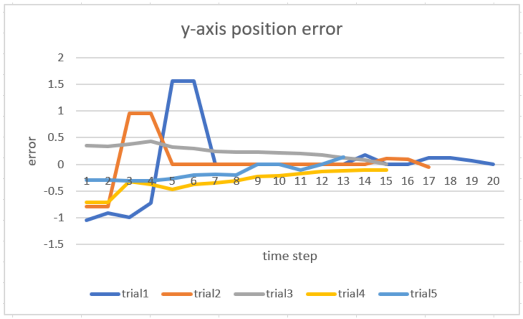
Last, both the drone and the gate are detected, we re-select the area that drone would fly through and reset the drone's goal point. Calculating the area separated by the opponent drone in the gate, we select the largest area and set the center point of the selected area as a new goal point of the drone.

4. Result

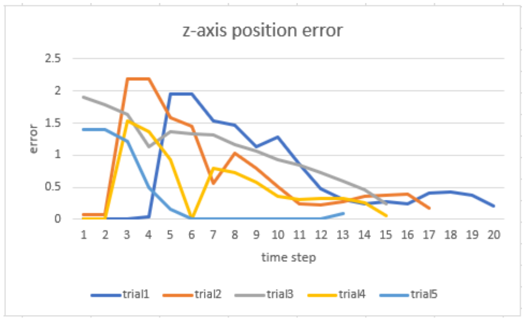
To verify the performance of our approach in maneuvering the drone through the targeted gate, the position error and yaw error from the flight between gate 7 and gate 8 in challenge 2 in 5 different race trials are collected. Note that the error is only recorded when the gate is detected by the detection model. The results are depicted in Figure 4.

The result illustrates that the proposed approach shows a promising performance by nullifying the position and yaw error such that the drone could pass through the target gate.

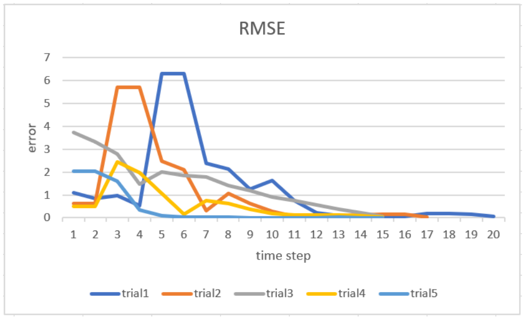
Next, the RMSE of position error, yaw error and the heading velocity of the drone during the flight of the first ten gates in tier 2 challenge is illustrated in Figure 5. This shows that our approach successfully reduces the positional error and yaw error before passing through each gate. As a result of setting the heading velocity proportionally to the gate depth, it is automatically adjusted such that the drone does not overshoot when passing through the gate resulting in zigzag trajectory while flying at the high speed.



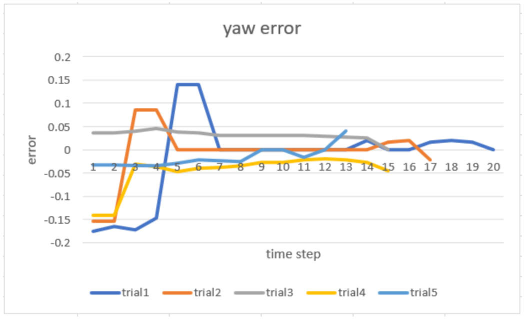
(a) error in Y-axis



(b) error in Z-axis



(c) RMSE of position error



(d) error in yaw angle

Figure 4: Error data of the flight from gate 7 to gate 8 in challenge 2 in 5 different trials

With the proposed approach, our team manages to complete both tier 2, passing through 21 gates in 81.19 seconds, and tier 3 challenges, passing through 22 gates in 110.73 seconds. For tier 3 challenge. The videos of the result could be accessed from `https://youtu.be/uBwf0NA7eTE` and `https://youtu.be/VEuHEvct-_4`.

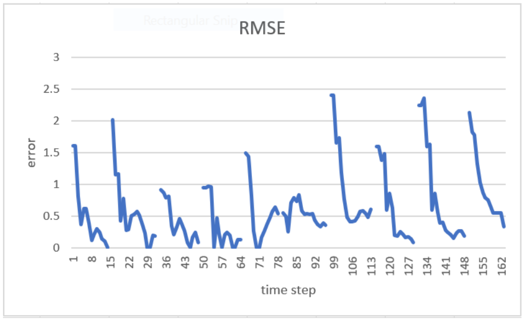
5. Conclusion

The deep-learning-aided automatic vision-based control appraoch shows promising performance in this Autonomous Drone Racing in Game of Drones Competition. By labeling only the closest gate to the drone and choosing the biggest bounding box, the gate detector, *MobileNetSSD*, could always locate the correct targeted gate.

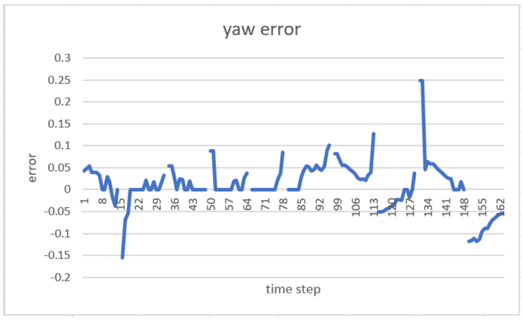
Once the location of the targeted gate is identified in the FPV image, the controller commands the drone to minimize the position error between the center of the target gate and the center of the FPV image in order to fly the drone to the center of the gate.

At the same time, the gate depth, which is the distance of the gate from the drone, determines the heading speed of the drone.

Our proposed appraoch is capable of competing in the challenging drone racing track. This method is used to win second place in both tier 2 and tier 3 challenge in Game of Drones - Competition at NeurIPS 2019.

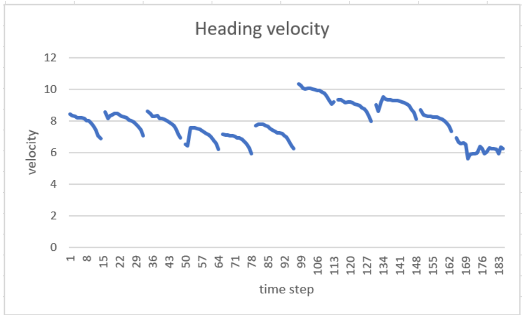


(*a*) RMSE of position error



(*b*) error in yaw angle

velocity.PNG



(*c*) heading velocity

Figure 5: The flight data from the first ten gates of tier 2 challenge

References

Kiam Heong Ang, Gregory Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, 13(4):559–576, 2005.

Sungwook Cho and David Hyunchul Shim. Development of a vision-enabled aerial manipulator using a parallel robot. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 15(APISAT-2016):a27–a36, 2017.

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. *arXiv preprint arXiv:1704.04861*, 2017.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, 35(1):146–166, 2018a.

Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018b.

Sunggoo Jung, Hanseob Lee, Sunyou Hwang, and David Hyunchul Shim. Real time embedded system framework for autonomous drone racing using deep learning techniques. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, page 2138. 2018c.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

Ratnesh Madaan, Nicholas Gyde, Sai Vemprala, Matthew Brown, Keiko Nagami, Tim Taubner, Eric Cristofalo, Davide Scaramuzza, Mac Schwager, and Ashish Kapoor. Airsim drone racing lab. *arXiv preprint arXiv:2003.05654*, 2020.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.