# Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning

Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim

***Abstract*—In autonomous drone racing, a drone is required to fly through the gates quickly without any collision. Therefore, it is important to detect the gates reliably using computer vision. However, due to the complications such as varying lighting conditions and gates seen overlapped, traditional image processing algorithms based on color and geometry of the gates tend to fail during the actual racing. In this letter, we introduce a convolutional neural network to estimate the center of a gate robustly. Using the detection results, we apply a line-of-sight guidance algorithm. The proposed algorithm is implemented using low cost, off-the-shelf hardware for validation. All vision processing is performed in real time on the onboard NVIDIA Jetson TX2 embedded computer. In a number of tests our proposed framework successfully exhibited fast and reliable detection and navigation performance in indoor environment.**

***Index Terms*—Autonomous drone racing, search and rescue, deep learning based object detection.**

## I. Introduction

RECENTLY, drone racing, which uses an agile drone controlled by a remote pilot using a first-person-view goggle, has become one of the most popular e-sports. While this showcases the remarkable human capability to control a highly dynamic vehicle using only the visual cue, it is desirable to develop a drone that can do the same task autonomously. With this motivation, the first autonomous drone racing competition in the world was held during IROS 2016 [1]. In this competition, a drone was required to fly through a series of 26 gates with 1.3 m $\times$ 1.3 m square as fast as possible. Due to the unavailability of GPS signals along the track, it was required to develop an indoor navigation algorithm with high speed vision processing for the gate detection. Participating teams attempted to solve the problem by applying various methods such as Simultaneous Localization And Mapping (SLAM) [2], logic-base approach and visual-servoing [3]. In this competition, our team won by passing through 10 gates in 86 seconds, but the performance was poorer than that of a skilled human pilot, who could pass through all 26 gates in 90 seconds or shorter. Our team's approach to the competition was based on color-based gate detection and indoor navigation using optical flow. However, the color-based gate detection was sensitive to the indoor lighting condition and had a tendency to get easily confused when two or more gates are seen overlapped. Therefore, the parameters of the algorithm had to be hand-tuned prior to each run. Even then, the drone found it difficult to pass through the gate when multiple gates were seen overlapped on one another. The optical flow sensor also complicated the indoor navigation problem due to its velocity estimation error.

In order to improve the detection performance, we presents a deep-learning model for real-time gate recognition on embedded platforms. The proposed learning model improves inference speed with better speed-accuracy trade-off by modifying the network structure and adjusting the parameters using Single-Shot Detection (SSD) [4], [5]. We also propose to use use a Line-Of-Sight (LOS) guidance algorithm [6] for more precise flight control. The contributions of this letter are as follows.

- Development of ADRNet: Default box-size tuning, improvement of the accuracy of gate detection, and minimized performance degradation due to network structure change.
- Application of the guidance method for fixed-wing aircraft with ADRNet: Modification of guidance law for quadrotor dynamics using the coordinates of the gate center estimated by ADRNet. This result is extended to the case of moving gates, which is made possible by real-time detection and guidance by the framework proposed in this letter.
- Provision of a custom data set for drone racing: By using the dataset of the gates captured in various environments, it is easy to reproduce the result of this research to make it useful for other drone-racing studies.

The rest of this letter is organized as follows: In Section II, we present an overview of related works. Section III describes the hardware and software of the proposed system. Details of our perception method are presented in Section IV. The guidance algorithm is presented in Section V. The practical implementation and evaluation are presented in Section VI. Section VII presents the conclusions and future works.

Manuscript received September 10, 2017; accepted January 27, 2018. Date of publication February 21, 2018; date of current version April 3, 2018. This letter was recommended for publication by Associate Editor J. C. Coo and Editor W. K. Chung upon evaluation of the reviewers' comments. This work was supported by the Ministry of Trade, Industry, and Energy, South Korea, under the Industrial Technology Innovation Program 10067202, "Development of robot system for indoor reconnaissance mission in complex disaster situations." *(Corresponding author: David Hyunchul Shim.)*

S. Jung, S. Hwang, and D. H. Shim are with the Unmanned Systems Research Group, KAIST, Daejeon 34141, South Korea (e-mail: sunggoo@kaist.ac.kr; sunyouh35@kaist.ac.kr; hcshim@kaist.ac.kr).

H. Shin is with the Aeronautics Research and Development Head Office, Korea Aerospace Research Institute, Daejeon 34133, South Korea (e-mail: hmshin@kari.re.kr).

Digital Object Identifier 10.1109/LRA.2018.2808368

## II. Related Work

### A. Object Detection by Unmanned Aerial Vehicle (UAV)

Image-based object detection for UAVs can be achieved using a number of techniques, including classical color and shape-based detection [7], ground vehicle detection [8], [9], Lucas-Kanade optical flow-based visual surveillance [10], and techniques that use the Scale-Invariant Feature Transform (SIFT) and Extreme Learning Machine (ELM) to obtain good results in agriculture [11]. However, these pixel-based image processing techniques are severely affected by environmental changes. Recently, the object-detection technique based on the deep-learning method was found to be highly capable of solving these problems. In particular, through the development of Convolutional Neural Networks (CNN) [12], the field of object-detection has achieved remarkable growth. For embedded computing, the deployed algorithms are desired to achieve better performance while consuming less energy due to the limited payload, computing power, and energy source. AlexNet and GoogLeNet [13], [14] showed excellent speed and performance in the ImageNet competition, and algorithms such as SSD and YOLO [4], [15] showed good performance in the detection part. However, these algorithms were still inadequate in terms of the computing power requirement imposed on an embedded system.

### B. Guidance and Control

As for the guidance of UAVs, the pursuit guidance using the LOS vector from the current position to the target point, proportional navigation guidance law controlling the LOS rate, and Command to LOS (CLOS) guidance were studied [16], [17]. In this study, we propose a velocity and heading command generation method using the LOS guidance law based on the vertical error and the rate of change of the reference trajectory. This LOS guidance law is widely used because of its intuitive and simple structure [18], [19], and is generally applied to fixed-wing UAVs, which have coupled dynamics between roll and yaw-axes. In this letter, the LOS-based guidance law is modified to handle the decoupled quadrotor dynamics.

In this letter, we propose to combine a CNN-based deep-learning and the LOS guidance for the IROS-style indoor drone racing [1], which only allows onboard computing. The proposed algorithms were validated by repeated indoor flight tests using various racing course setup. We also extended to the case with moving gates, which is possible owing to the real time detection, not the fixed way-point navigation with indoor localization methods only.

## III. System Overview

In order to meet the requirements of the IROS indoor drone racing [1], the platform needs to be: 1) small enough to pass through the gate, 2) agile to finish the course as quickly as possible, and 3) perform the vision processing onboard. We chose a commercially available quadrotor platform designed for racing. This platform can carry all the sensor systems required for indoor navigation and fly for five minutes, which is a sufficient time to finish the course.

Fig. 1. A snapshot of the drone passing the gate using the proposed learning guidance algorithm.

For vision processing, we used an NVIDIA TX2 module for its fast processing speed and compact footprint. The TX2 module is mounted on a third-party carrier board, which has 32 GB of eMMC (embedded Multi Media Card) storage, 8 GB RAM and a USB 3.0 interface. The UAV was equipped with a Terabee TeraRanger-One one-dimensional Lidar for altitude measurement, a PX4Flow for ground speed measurement, and a ZED stereo camera for odometry measurements. Especially, the ZED camera computes the odometry in real time using Semi-direct Visual Odometry (SVO) algorithm [20]. The velocity and odometry information were sent to the Unscented Kalman Filter (UKF) running on the NVIDIA Jetson TX2 to obtain more accurate velocity and position information through sensor fusion with the Inertial Measurement Unit (IMU). The sensor fusion process will be explained in detail in the next section. Fig. 2 illustrates our software architecture. The operating system is based on Ubuntu 16.04 and works in a ROS-Kinetic environment. Vision processing for object detection was performed at 30 fps (frames per second) in VGA mode with a fisheye lens on the PointGrey Firefly camera. We trained our ADRNet using the Caffe library. When a racing gate was detected, it extracted the center point and executed the guidance algorithm to make the UAV follow this center point. The guidance algorithm generated forward, left/right, and heading velocity commands, which were sent to the Flight Control Computer (FCC) via the RS-232 serial communication connector. For flight control, we used an in-house FCC called DS board [21].

## IV. Perception Using ADRNet

The Single Shot Detector (SSD) is a deep convolutional neural network architecture for multi-box object detection [4]. The SSD model offers high accuracy and a fast speed of over 40 fps on a typical desktop computer. However, the inference time on the TX2 embedded board was 462.04 ms, which is not suitable for autonomous drone racing. To improve the speed of the detection model, we initially applied AlexNet [13] as a

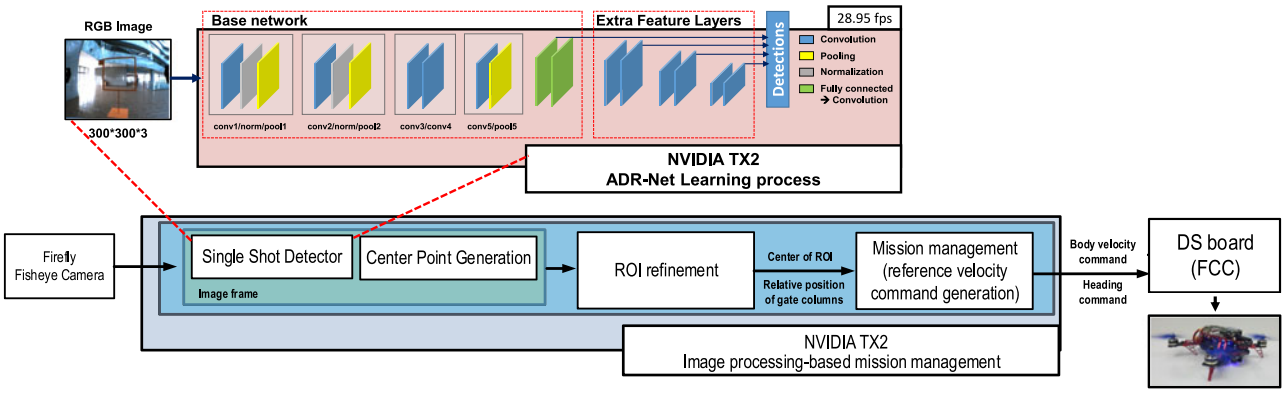


Fig. 2. Software architecture of our system for vision processing.

base network instead of the VGG-16 [22] used in the original SSD model. As in the case of VGG-16 in the SSD paper [4], fc6 and fc7 were replaced with a convolutional layer among three fully connected layers of AlexNet, and fc8 was deleted. As a result, we achieved an inference time of 84 ms, which is faster than the VGG-16-based SSD model. However, in actual flight, it operated at a speed slower than the inference rate because it executed concurrently with other processes that used the GPU. As this network could not achieve the speed needed for racing, we modified the structure of the network for further speed improvements. The base network used was AlexNet. As our primary objective was a single gate detection, we reduced the number of unnecessary high-level feature layers. Additionally, since the gates are placed no farther than five meters, a regression operation to find a small object was eliminated. The speed-accuracy tradeoff is an inevitable problem of deep neural networks [23]. The desired speed of about 30 fps was achieved by deleting the layers. The average precision was reduced by 0.07 compared to the original SSD model. To compensate for the reduced accuracy, we tuned the network parameters. In the SSD model, the sizes of the default boxes used to match object positions were set according to the PASCAL VOC dataset [24], which could degrade the performance in real-world situations. We improved the performance by changing the previous box sizes of the model for better gate detection. The box sizes were determined by referring to the image data obtained from the experiment.

As the gate was not of a very small size, the minimum default box size was made larger.

## V. Guidance and Control

The guidance algorithm we considered for this research was based on the LOS vector guidance used for automatic landing of fixed wing UAVs [6]. However, since this algorithm is for fixed-wing guidance, the following changes were made to apply to our quadrotor UAV. As the lateral axis and the yaw axis dynamics of a quadrotor can be decoupled unlike fixed-wing aircraft, each axis is controlled separately. In addition, as the in-house FCC accepts roll and yaw rate commands only, the command equations in [6] were modified to (4) and (5). To

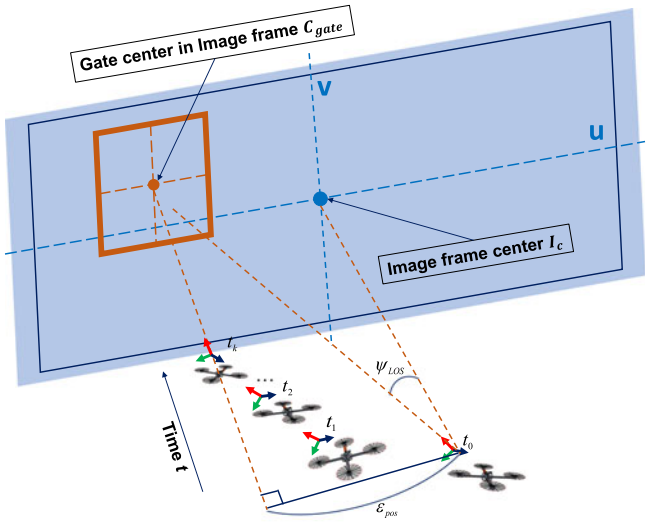


Fig. 3. By applying LOS guidance, $\epsilon_{\text{pos}}$ and $\psi_{LOS}$ is controlled to be zero when passing the gate.

obtain the heading and roll rate commands using the LOS vector, the distance between the center point $C_{\text{gate}} \in \mathbb{R}^2$ of the gate and the center point $I_c \in \mathbb{R}^2$ of the current quadrotor image frame could be expressed as a distance in the world coordinate system as shown in (1), where $C_1$, $C_2$, and $C_3$ are constants calibrated using the actual distance and the object of the image pixel (Fig. 3).

$$\varepsilon_{\text{pixel}} = C_{\text{gate}} - I_c$$
$$f : \varepsilon_{\text{pixel}} \mid\!\rightarrow \varepsilon_{\text{meter}}$$
$$\varepsilon_{\text{pos}} = f\left(\varepsilon_{\text{pixel}}\right) = C_1 \cdot \left(\varepsilon_{\text{pixel}}\right)^2 + C_2 \cdot \varepsilon_{\text{pixel}} + C_3 \quad (1)$$

The heading and roll rate commands were generated as follows. $V_{\text{lat}}$ was the roll-axis velocity obtained from the PX4Flow optical-flow measurement and $\psi_{\text{path}}$ was the angle between the starting line and the gate (the heading angle of the starting line was set to $0^{\circ}$ as the measurement were conducted indoors) and $\lambda_{\text{app}}$ is the scale factor.

$$\dot{\varepsilon}_{\text{pos}} = V_{\text{lat}} \quad (2)$$

TABLE I
COMPARISON OF SSD-BASED ARCHITECTURES FOR GATE DETECTION

| | VGG-16-based SSD | AlexNet-based SSD | ADRNet |
|---|---|---|---|
| Desktop (GTX 1080 ti) benchmark | 24.16 ms (41.39 fps) | 4.86 ms (205.76 fps) | 3.97 ms (251.89 fps) |
| TX2 board benchmark | 462.04 ms (2.16 fps) | 84.21 ms (11.86 fps) | 34.54 ms (28.95 fps) |
| Average Precision | 0.824 | 0.774 | 0.755 |

$$\psi_{LOS} = \tan^{-1}\left(\frac{K_p \cdot \varepsilon_{\text{pos}} + K_d \cdot \dot{\varepsilon}_{\text{pos}}}{\lambda_{\text{app}}}\right) \tag{3}$$

$$\dot{\psi}_{\text{cmd}} = \psi_{\text{path}} + \psi_{LOS} \tag{4}$$

The decoupled roll command was:

$$\dot{\phi}_{\text{cmd}} = K_{\text{lat}} \cdot (K_p \cdot \varepsilon_{\text{pos}} + K_d \cdot \dot{\varepsilon}_{\text{pos}}) \tag{5}$$

Using the roll and heading command proposed above, the quadrotor can be guided to fly through the center of the gate along the line perpendicular to the gate.

## VI. EXPERIMENTAL RESULT

In this section, we evaluate the performance of our proposed autonomous drone racing strategy with emphasis on the convolutional neural network and the LOS guidance algorithm.

### A. Detection Performance Evaluation

Training data were collected using the rosbag function of the ROS using a fisheye camera mounted on the front of the quadrotor. A camera with a Field-of-View (FOV) of approximately 170° was able to capture almost all gates in the frontal side of the quadrotor with the VGA quality at 30 Hz.

The performances of the original VGG-16-based SSD, AlexNet-based SSD, and our proposed ADRNet are compared in Table I. All three models used the same training set consisting of 1544 images and were trained for 12,000 iterations. Training was conducted on a desktop computer with an Intel i7-6700 CPU 3.40 GHz, 16G RAM, and a NVIDIA GTX 1080 Ti, on Ubuntu Linux 16.04 using Caffe Framework with CUDA 8.0 and CuDNN 5. The test data used various background images composed of 7380 frames. As shown in Fig. 4, the original VGG-16-based SSD model showed the best AP (0.824) and the worst inference speed (2.16 fps). In the case of AlexNet-based SSD, the inference speed was improved to 11.86 fps and AP was decreased slightly (0.774). On the other hand, our proposed ADRNet operated at 29 fps on the TX2 board with 0.755 AP, which meets the required real-time performance. Although ADRNet had a lower detection rate than AlexNet based SSD, false alarms decreased. The reason for the low detection rate was that ADRNet was designed not to consider cases where the gate was at a long distance from the UAV, considering the

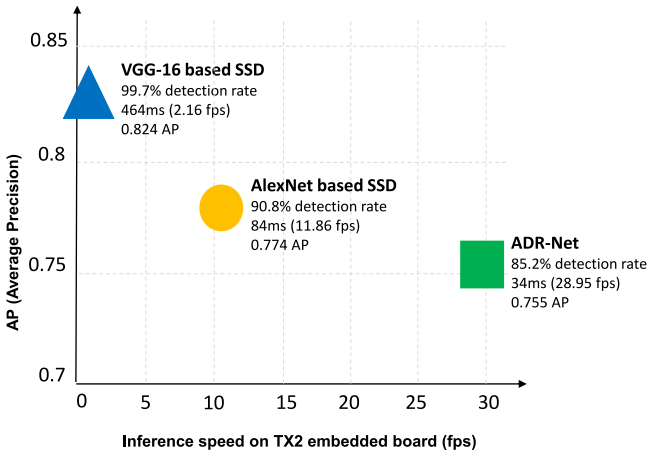


Fig. 4. AP and inference speed comparison for three network models. Our proposed ADRNet shows applicable performance in both accuracy and speed criteria.

Fig. 5. Gates detected by proposed ADRNet running onboard the UAV. Whole snapshots are not included in the training data set and are used only for performance evaluation.

racing arena. When the distance between the drones and the gate was approximately 10 meters or more, the detection rate decreased. These results met our expectations for reliable gate detection using the onboard computer. In particular, as shown in Fig. 5, overlapped gates that could not be detected by the existing color-based detection could be recognized as individual gates by applying our proposed ADRNet.

We also compared ADRNet with the color-based detection method used in IROS2016 [3]. Using the color-based method, the detection rate under one case, shown Fig. 7(a), was 89.8%, but for another case, shown Fig. 7(b), was 7.6%. In summary, the color-based method showed about 50% detection rate. This is because the color-based method works well only for the specific situation in which the parameters were tuned. On the other hand, our learning-based method achieved on average of 90% detection rate for the same tests. Furthermore, the color-based detection method recognized the whole gate, including base plate and support, unlike ADRNet, which looks for square frames only.

In actual racing, distinguishing the closest gate from other gates seen overlapped is an important issue. Our training data was designed so that the nearest gate has the highest probability. Therefore, the algorithm chooses the most probable object, which is the closest gate. Furthermore, if there were multiple

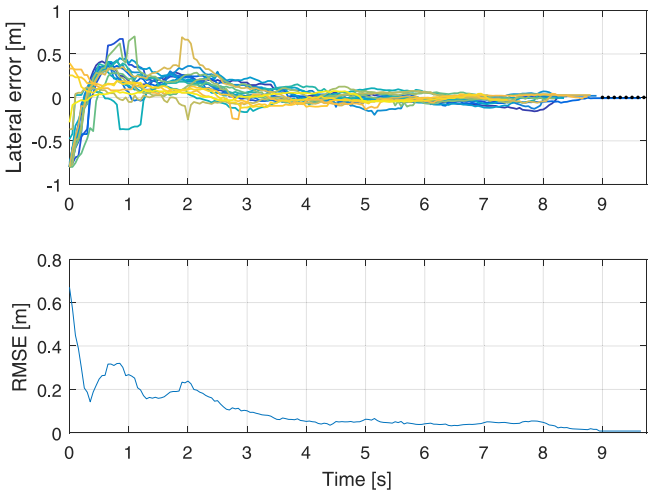


Fig. 6. Errors between the gate center and the quadrotor for 20 repeated experiments. At the end of the passage through the gate, the error converges to less than 0.1 m.

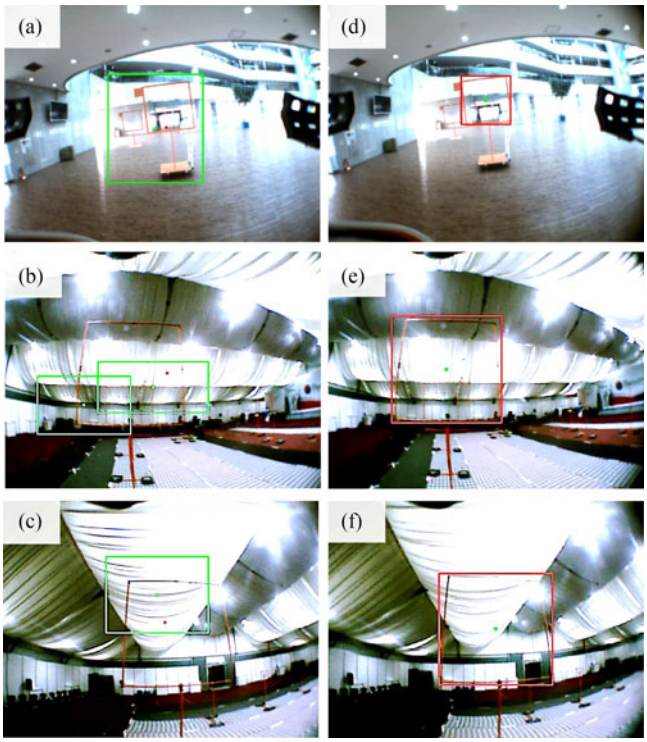


Fig. 7. Detection results for two different environments. (a)–(c) are color-based results, while (d)–(f) are ADRNet-based results.

gates, our method would detect the closest gate among them. As a result, our proposed framework can be applied to various drone racing environments such as an S-course or a straight course.

### *B. LOS Guidance*

The LOS guidance algorithm is formulated to lead the drone to fly through the gate center detected by the proposed deep learning network. For the evaluation, we performed twenty


Fig. 8. Snapshot of the course with nine randomly placed gates. Using this setup, we evaluated the performance of passing through multiple gates.

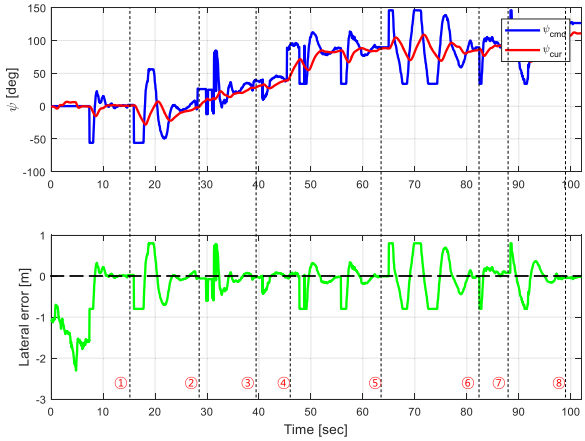


Fig. 9. Evaluation result of the race course consisting of nine randomly placed gates. The dotted vertical line represents the moment when the gate passes. In the vicinity of the vertical line, the center point error is close to zero.

separate runs to fly through a gate. The first row of Fig. 6 shows the distance between the gate center and the quadrotor. In all twenty experiments, it was confirmed that the tracking error converged to zero with errors no larger than 0.1 m. To clarify the convergence of the lateral error between the gate center and the quadrotor, we provide a root mean square error (RMSE) graph in second row of Fig. 6. The RMSE plot clearly shows precise tracking performance of the gate center point. The initial overshoot in the lateral error graph is attributed to the phase-lag caused by the under-actuated dynamics of a quadrotor.

As for the next test, we constructed a test track by randomly placing nine gates as shown in Fig. 8. The first row of Fig. 9 shows the heading command and the tracking performance. In addition, the center-point error graphs during the nine-gate test is provided in the second row of the graph. The black dotted line shows the moment the gate is passed. The drone passed all nine gates successfully and the lateral center point error was close to zero. This shows that the proposed algorithm works very well, even when passing through multiple gates.

Finally, in order to validate the real time performance of the deep learning-based detection and the guidance command generation, another test was conducted using a moving gate.


Fig. 10. Time lapse of moving gate passing. We tried our algorithms on moving gates and showed that the proposed algorithm is applicable to various dynamic environments.

With the gate moving, pre-determined way-point following methods with indoor localization will not work.

Fig. 10 shows the time-lapse image of a drone passing through a moving gate. It can be seen that our proposed deep learning assisted LOS guidance framework can be extended to a more dynamic environment with moving gates.

## VII. Conclusion

Using a deep-learning-based perception, we developed real-time gate detection networks on an embedded computer. We confirmed that we could successfully pass through the gate by applying the proposed guidance algorithm to a quadrotor. We showed the drone can reliably fly through multiple gates in a randomly arranged nine-gate race-course. Finally, we validated the superiority of the real-time recognition performance of the proposed method and the tracking performance of the guidance algorithm by passing through a moving gate. While we presented the results of gate detection for drone racing in this letter, our algorithm can be extended to various scenarios such as search and rescue missions that require flying through windows, or avoiding falling debris, and other dynamic obstacles. The dataset and codes were released and made available for future autonomous drone racing competitors.[1]

## References

[1] H. Moon, "IROS—2016 autonomous drone racing in MOS ESPA daejeon arena," 2016. [Online]. Available: http://ris.skku.edu/home/iros2016racing.html. Accessed on: 31 Jan, 2017.

[2] B. Michael, B. Michael, O. Helen, N. Juan, and S. Roland, "Robust state estimation and replanning for fast UAV navigation," in *Proc. Conf. Workshop*, 2016.

[3] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, "A direct visual servoing-based framework for the 2016 IROS autonomous drone racing challenge," *J. Field Robot.*, 2017. [Online]. Available: http://dx.doi.org/10.1002/rob.21743

[4] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Porc. Eur. Conf. Comput. Vis*, 2016, pp. 21–37.

[5] S. Jung, H. Lee, S. Hwang, and D. H. Shim, "Real time embedded system framework for autonomous drone racing using deep learning techniques," in *Proc. 2018 AIAA Inf. Syst.-AIAA Infotech@ Aerosp.*, 2018, Art. no. 2138.

[6] D. I. You, Y. D. Jung, S. W. Cho, H. M. Shin, S. H. Lee, and D. H. Shim, "A guidance and control law design for precision automatic take-off and landing of fixed-wing UAVS," in *Proc. AIAA Guid., Navig., Control Conf.*, 2012, Art. no. 4674.

[7] S. CHO and D. H. SHIM, "Development of a vision-enabled aerial manipulator using a parallel robot," *Trans. Jpn Soc. Aeron. Space Sci., Aerosp. Technol. Japan*, vol. 15, no. APISAT-2016, pp. a27–a36, 2017.

[8] T. Moranduzzo and F. Melgani, "Detecting cars in UAV images with a catalog-based approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6356–6367, Oct. 2014.

[9] K. Liu and G. Mattyus, "Fast multiclass vehicle detection on aerial images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 9, pp. 1938–1942, Sep. 2015.

[10] S. Kamate and N. Yilmazer, "Application of object detection and tracking techniques for unmanned aerial vehicles," *Procedia Comput. Sci.*, vol. 61, pp. 436–441, 2015.

[11] S. Malek, Y. Bazi, N. Alajlan, H. AlHichri, and F. Melgani, "Efficient framework for palm tree detection in UAV images," *IEEE J. Sel. Topics. Appl. Earth Observ. Remote Sens.*, vol. 7, no. 12, pp. 4692–4703, Dec. 2014.

[12] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Process. Lett.*, vol. 21, no. 9, pp. 1120–1124, Sep. 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[14] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[16] C.-L. Lin and H.-W. Su, "Intelligent control theory in guidance and control system design: An overview," *Proc Nat. Sci. Council Republic China A Phys. Sci. Eng.*, vol. 24, no. 1, pp. 15–30, 2000.

[17] I.-J. Ha and S. Chong, "Design of a clos guidance law via feedback linearization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 28, no. 1, pp. 51–63, Jan. 1992.

[18] T. I. Fossen, *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics, 2002.

[19] S. Park, J. Deyst, and J. P. How, "Performance and lyapunov stability of a nonlinear path-following guidance method," *J. Guid. Control Dyn.*, vol. 30, no. 6, pp. 1718–1728, 2007.

[20] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. 2014 IEEE Int. Conf. Robot. Autom.*, 2014, pp. 15–22.

[21] D. Lee, H. Lee, J. Lee, and D. H. Shim, "Design, implementation, and flight tests of a feedback linearization controller for multirotor uavs," *Int. J. Aeron. Space Sci.*, vol. 18, no. 4, pp. 112–121, 2017.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[23] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *arXiv:1611.10012*, 2016.

[24] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[1]Code and dataset are available at our project page: https://sunyouh.github.io/projects/adr2017