Received: 15 February 2017 | Revised: 15 May 2017 | Accepted: 3 July 2017

DOI: 10.1002/rob.21743

FIELD REPORT

WILEY

A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge

Sunggoo Jung | **Sungwook Cho** | **Dasol Lee** | **Hanseob Lee** | **David Hyunchul Shim**

Unmanned Systems Research Group, Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

Correspondence
David Hyunchul Shim, Unmanned Systems Research Group, Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 34141, Republic of Korea.
Email: hcshim@kaist.ac.kr

Abstract

This paper presents a framework for navigating in obstacle-dense environments as posed in the 2016 International Conference on Intelligent Robots and Systems (IROS) Autonomous Drone Racing Challenge. Our framework is based on direct visual servoing and leg-by-leg planning to navigate in a complex environment filled with many similar frame-shaped obstacles to fly through. Our indoor navigation method relies on the velocity measurement by an optical flow sensor since the position measurements from GPS or external cameras are not available. For precision navigation through a sequence of obstacles, a center point–matching method is used with the depth information from the onboard stereo camera. The guidance points is directly generated in three-dimensional space using the two-dimensional image data to avoid accumulating the error from the sensor drift. The proposed framework is implemented on a quadrotor-based aerial vehicle, which carries an onboard vision-processing computer for self-contained operation. Using the proposed method, our drone was able to finished in first place in the world-premier IROS Autonomous Drone Racing Challenge.

KEYWORDS

Aerial robotics, autonomous drone racing, collision avoidance, stereo vision, visual servoing

1 | INTRODUCTION

Drone racing is a new-generation sport where small unmanned aerial vehicles (UAVs) fly through a series of obstacles by human pilots (hereinafter, the terms drone and UAV will be used interchangeably). Unlike traditional radio control airplane races, drone-racing pilots use first-person-view displays, which are mounted on the head of the pilot and deliver a live video stream sent from the camera mounted on the vehicle wirelessly in real time. Since drone racing showcases impressive human capabilities to control highly agile UAVs only using the onboard video stream, it has become a very popular sport.

Recently, various vision algorithms have been applied to the UAV flight control systems. The most notable success is the application of motion capture systems, which provides incredibly accurate position measurements using a set of special wall-mounted cameras.[1] These systems enabled many impressive research results that were not previously possible owing to limitations such as the weight, accuracy, or the dynamic range of the onboard sensor.[2–9] However, such delicate and expensive systems cannot be assumed always available in real world, to deploy UAVs into various real-world applications, one needs to employ a self-contained navigation method. Among a few solutions, the vision-based navigation is quite promising: indeed, a variety of vision-based approaches ranging from target recognition to visual servoing[10–13] have been successfully applied to UAVs. As the hardware for onboard vision system becomes more powerful yet lighter, self-contained vision-based navigation has become a viable solution for precision indoor navigation.

Motivated by the aforementioned advances in the vision-processing area, there have been discussions about creating autonomous drone racing events, where UAVs navigate through obstacles without any help from pilots. Moreover, the drones rely only on their onboard sensors, not external sensors such as motion capture cameras or high precision GPS. Such events are expected to promote not only the development of onboard navigation algorithms using vision sensors but also the drone industry as a whole.

The Autonomous Drone Racing Challenge was held at the 2016 International Conference on Intelligent Robots and Systems (IROS) in Daejeon, South Korea.[14] In this event, the participating teams were required to compete with drones, which can navigate through a series of frame-shaped obstacles, followed by a frame with a rotating fan at the end. The drones are expected to pass through the obstacles in the correct order to earn scores. The color, geometry, and locations of the obstacles are announced prior to the event, allowing participating teams to utilize this information for better localization.

In this world-premiere event, five international teams, including the team from Korea Advanced Institute of Science and Technology

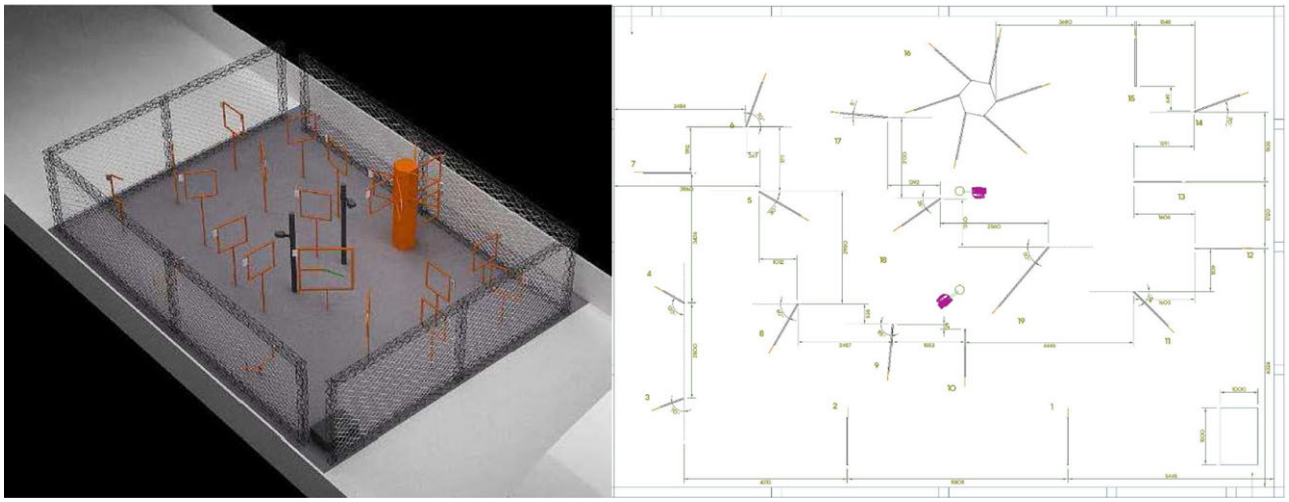

FIGURE 1 IROS 2016 autonomous drone racing track (left: bird's eyeview of the arena, courtesy of RISE Lab, Sungkyunkwan University; right: detailed course drawings)

(KAIST), competed on October 12, 2016. Each team operated their vehicle using distinct algorithms for indoor navigation. Our system utilized direct vision-based servoing and optical flow for indoor navigation. In this challenge, our team came in the first place by passing 10 obstacles in 86 s, demonstrating the viability of our direct visual servoing-based approach for small drones with limited computation capability.

In this paper, we present our indoor navigation framework developed for the 2016 IROS Autonomous Drone Racing Challenge. The remainder of this paper is organized as follows: First, we provide a brief introduction to the challenge then we review the latest technological trends in vision-based navigation for UAVs. Then we describe our strategy for indoor navigation, the flight control computer (FCC), and the vision system with stereo camera–based direct image-processing algorithm used for the challenge. The flight data logged during the main event are presented with discussions on the outcomes from the finished sections. The paper is concluded with the discussions on our approach and future directions to further improve our framework so that our drone can fly through the entire race.

2 | INTRODUCTION TO THE IROS 2016 AUTONOMOUS DRONE RACING CHALLENGE

The 2016 IROS Autonomous Drone Racing Challenge was organized to promote the development of indoor navigation technology for small drones. Participants were required to build a drone that can fly through 26 obstacles in a sequential manner. The color, geometry, and installation position/angle were announced on a web site prior to the event (Fig. 1). The actual obstacles were installed within a prescribed error margin of 10 cm. No limit was given on the type of UAV, but the size was limited to $1 \times 1 \times 1$ m. Human intervention was not allowed after the referee started each run.

The course was divided into 10 sections, labeled from A to J, as shown in Figure 2 and Table 1. Figure 3 gives accurate dimensions of all the obstacles in the challenge. As one can see, some of them are an open U-shaped frame whereas most of others are a square frame of 1.3 m × 1.3 m. For convenience, each obstacle in the racing course is called a "gate" in this paper. Each team was required to start at a designated location. Once the drone took off, it could not be piloted by the team members and had to fly through the obstacles using its own sensors and computers.

The team whose vehicle passes through all of the gates sequentially in the shortest time would win the grand prize. If no team's vehicle passed through all of the gates, the winner would be decided using the score, which is the number of the obstacle that the drone passed through before failing, drifting away, or the time ran out. If the drone deviated from the course and could not pass through the next gate, the number of the last gate the drone successfully passed through would be considered the score. If two or more teams reached the same gate, the team with fastest time would win the event. Each team was given 30 min and could attempt up to two runs, and the best score is used for judging.

Any type of sensor is allowed to use in the challenge as long as they are onboard. In other words, no external sensors can be used at all times. Premapping is allowed if it can be done using only the onboard sensors during flight. On the other hand, computing-heavy algorithms such as vision processing can be performed on a ground computer, which receives the video stream using a wireless video transmitter. This rule is intended to allow participants to run algorithms that can be heavy for today's (onboard) computers, but not for those in near future. In the end, no participants used off-board computing, possibly due to concerns regarding the quality of wireless communications in the arena.

3 | RELATED WORK

3.1 | Visual servoing for unmanned aerial systems

Computer vision has been found very useful for many applications of UAVs such as target detection and tracking. Recently, visual servoing, a

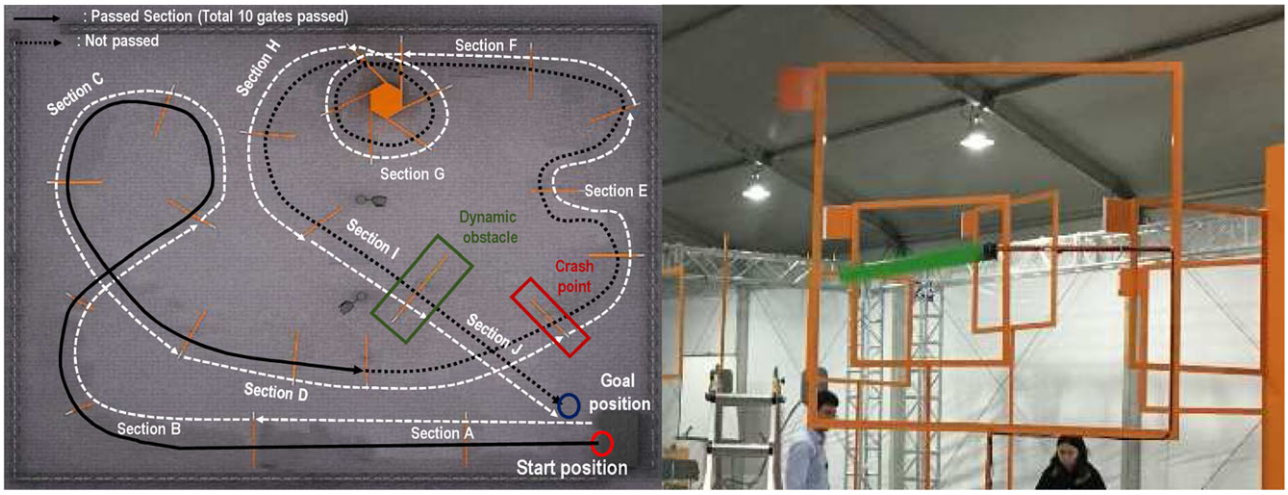

FIGURE 2 Course breakdown (left): course have 10 sections with various characteristics. The dynamic obstacle (right): the green bar rotates around 0.5–1 rpm

TABLE 1 Detailed section descriptions

Section	Course	Gate characteristic	Gate
A	Straight	Two open top gates	1,2
B	Smooth turn	Two side-open gates and one square gate	3–5
C	Sharp turn	180° turn, three square gates	6–8
D	Smooth turn	Three square gates	9–11
E	Horizontal zigzag	Three square gates and zigzag course with short gate distance	12–14
F	Sharp turn	Sharp 90° turn	15
G	Upward spiral	Spiral stairway with eight gates installed at 60 °	16–23
H	Smooth turn	Two square gates	24, 25
I	Dynamic obstacle	Avoid bars rotating at 30 rpm	26
J	Straight	Final section and finish line	Finish

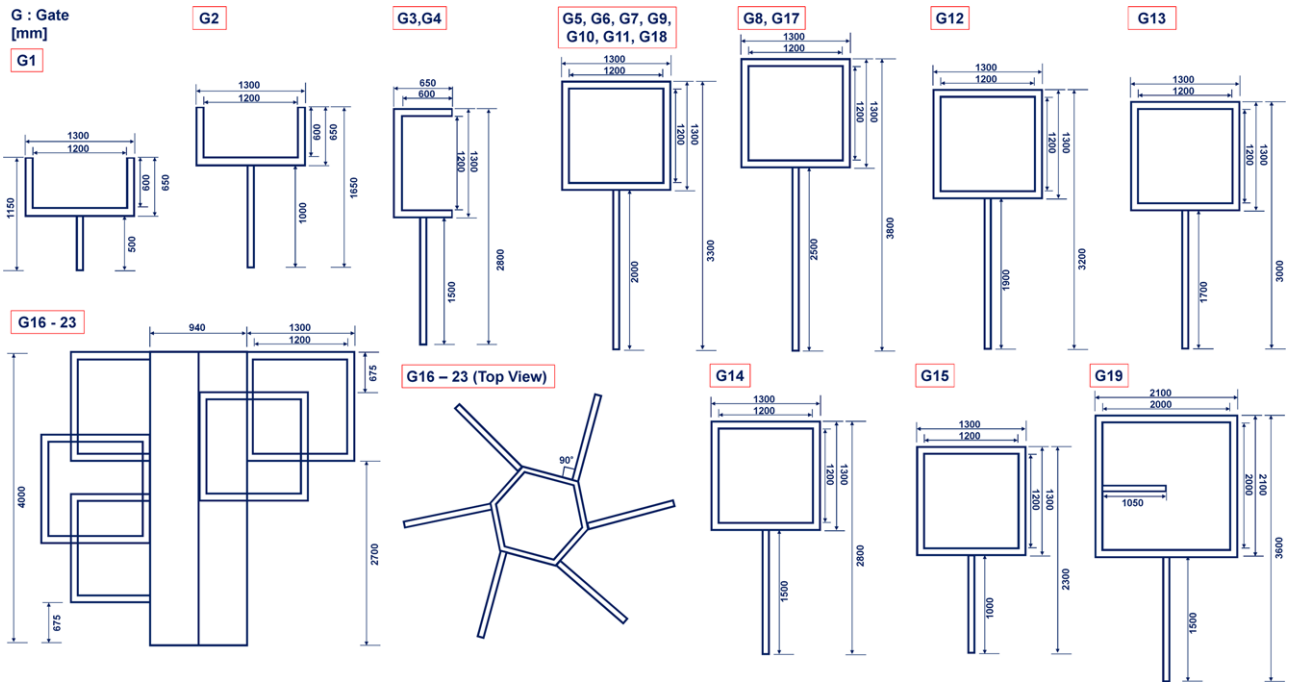

FIGURE 3 Specifications of the gates showing that have various height and shapes. All gates frame size are equal except for gates 1 and 2. Gates 1 and 2 are half sized than the other gates, and it shows open top characteristic

process to generate a three-dimensional (3D) relative motion based on two-dimensional (2D) image data, has become a very promising alternative for UAV navigation.[15]

Visual servoing can be categorized into four types as follows: position-based visual servoing (PBVS), image-based visual servoing (IBVS), hybrid strategy visual servoing, and direct visual servoing.[15–19] The PBVS method requires an accurate calibration of intrinsic and extrinsic image sensor parameters to reconstruct 3D environments. Therefore, the uncertainties in these parameters are difficult to overcome. In contrast, the IBVS method does not require these parameters or 3D relative pose estimation. However, the relative scale should be computed to calculate the Jacobians of images. Furthermore, this approach cannot be used in highly complex environments. Although hybrid visual servoing can compensate for the disadvantages of the IBVS method, it may not be usable in underactuated systems. Compared to PBVS, IBVS, and hybrid visual servoing methods, direct visual servoing is relatively simple and performs well. However, it requires a large number of fault-prevention algorithms. In addition, schemes that only use image data (e.g., IBVS and direct visual servoing) require additional algorithms to prevent abnormal results caused by singularities or local minima.[17,19–22] Several advanced IBVS approaches were proposed to overcome the drawbacks of classical IBVS methods. These approaches can be divided into three categories: geometry-based schemes based on homography or epipolar geometry, path planning-based schemes using the camera frame, and schemes using joint space or the image frame.[23] Although each scheme has significant advantages, problems still exist because all of the methods use point-like features. Therefore, a new approach intended to improve the performance of the IBVS method by using image momentum and intensity-based photometric features was investigated.[24] This method can be employed to help gather target data focusing on the center of the moment without corner points. However, the method can only be used if a feature does not have an empty space. Accordingly, a direct visual servoing algorithm was employed to solve this problem. The term "*direct*" indicates that the estimation of the 3D environment or the calculation of the image Jacobian is removed, such that the control law must stabilize the system by using visual sensing results that are directly fed directly into the FCC.[18,24,25] The reference velocity is generally used in direct visual servoing to allow the current image to reach the desired image instead of using any point-like or image features. An optimization scheme should be applied for the direct usage of the images. However, the images contained so much data that additional effort was required to reduce the processing time.[24]

3.2 | High-speed collision avoidance in complex environments

Shim et al pioneered the "in situ" obstacle avoidance problem by using an onboard laser scanners for a helicopter UAV in complex urban environment.[26] For a motion capture system–based research, UAVs were able to avoid the obstacles by precisely following the precomputed collision-free paths.[7] However, applying this method in a real environment with imperfect mapping, such as the IROS Challenge arena, would not be safe. Barry and Tedrake [27] presented an interesting depth-based high-speed collision avoidance method in which a stereo vision system was employed. Their method involved performing onboard image processing in high-resolution environments and enabling rapid and accurate obstacle detection via depth searching. However, the experimental environment was relatively simple (i.e., a wide outdoor environment), and applying their method in more complicated or indoor environments would not be suitable. Similarly, some studies used obstacles in environments that were more complex than those employed in Barry and Tedrake's research. In such cases, depth sensors were employed to avoid obstacles and real-time path planning was performed to avoid collisions.[28]

3.3 | Applicable strategies

For the 2016 IROS Drone Racing Challenge, the organizers published the layout of the arena prior to the event. Hence, it was possible to use a simultaneous localization and mapping (SLAM) method by using a LiDAR or a vision sensor.[29–33] Another possible method would be to use sensor fusion based on the vision and inertial sensors.[34–37] Traveling along a preplanned path that avoided the obstacles using the localization results based on a map and the detected obstacles becomes possible when SLAM or visual-inertial sensor fusion-based localization is properly implemented. However, there was a possibility that the SLAM or visual-inertial fusion solution would easily diverge because the venue had multiple groups of similar objects without clearly distinguishing features or landmarks. It is true that each obstacle was clearly marked with a QR tag (it reads the serial number of the obstacle), but the tag was visible only from the front of the obstacle. LiDAR-based SLAM solution may be promising, but a typical scanning laser sensor is somewhat heavy and large to be fit on a drone for the racing course.

Second, one may attempt to navigate through the obstacles using accurate time-based control by exploiting the information on the sizes and positions of the obstacles announced in advance. This technique would enable control commands to be scheduled and the time to complete the course to be set in advance. This method was not strictly autonomous, but was not against the challenge regulations. However, this method was strongly influenced by the initial position error and control accuracy so it is not really practical.

Third, a visual servoing-based method can be used for the challenge. After the UAV fly from one gate to the next using a non-GPS based position sensors, it passes through the gate using a direct visual servoing. This algorithm is computationally not as heavy as the SLAM algorithm mentioned above, making it more suitable for the challenge. However, an additional algorithm would be needed to reduce the error between the visual servoing in the 2D image plane and the 3D real-world environment effectively.

In contrast to typical obstacle avoidance problems where the drones simply fly away from the obstacles, flying through the obstacles with small margins as posed in the IROS Challenge is quite different from other cases, therefore requires a whole new approach. After reviewing state-of-the-art research on vision-based navigation, considering many restricting factors from the challenge rules, direct visual servoing using a stereo vision system was found preferable to standard visual servoing, SLAM, or other schemes due to its relatively

FIGURE 4 UAV configuration for an autonomous indoor flight. Sensors such as optic flow and one-dimensional (1D) LiDAR, stereo camera, embedded board, and in-house FCC are highlighted by the enlarged image

lower computing load with acceptable performance. In this research, we selected a stereo vision–based obstacle avoidance approach. The position control of the UAV relative to the gate in heading, and pitch axes are achieved within acceptable error margin by directly using the heading and pitch angle deviations from the horizontal bars. A stereo camera obtains the depth image, which is used to compute a reference body velocity command fed into the FCC. Our proposed approach is intended to be light enough for onboard computation on a small drone yet reasonably robust to handle the complex indoor navigation problems in the Drone Racing Challenge.

4 | SYSTEM DESCRIPTION

Our system for the drone challenge consists of a quadrotor-type platform, which carries a FCC, a vision-processing computer, and various sensors such as a stereo camera and a laser range finder (Fig. 4). The following sections describe the key features of these components and our indoor flight strategies.

4.1 | UAV platform

We chose a quadrotor as the vehicle platform owing to its compact size and flight characteristics that are suitable for a dense indoor environment. Our drone is based on the popular DJI F450 frame, which has four 28 cm (11 in.) propellers driven by four 690 kV motors and four 40 A electronic speed controllers. The propellers are protected with four prop-guards, which are quite useful when bumping into obstacles. The platform is powered by a 5200-mAh lithium polymer battery and is capable of flying for 10 min. The drone is controlled by our in-house flight control system, which provides greater flexibility and more precise control to meet the stringent requirements of tight indoor navigation. The FCC is based on an embedded processor and a three-axis gyro sensor, a three-axis acceleration sensor, and a three-axis magnetometer. The in-house FCC stabilizes and controls the quadrotor, which is known to be highly unstable. The FCC estimates the linear velocity and attitude by filtering the raw sensor input and computes the control output for individual motor control in real time. For demanding vision processing, the UAV was equipped with a NVIDIA Jetson TK1 single-board computer, which offers GPU-powered computing in a compact form factor. The TK1 computer, using a visual servoing algorithm, generates velocity commands, which are sent to the FCC via USB and RS232. On the TK1 computer, ROS indigo runs on Ubuntu 14.04 LTS for image processing.

TABLE 2 UAV specifications for IROS Autonomous Drone Racing

Size	Width: 0.58 m; height: 0.26 m
Weight	Empty weight: 2.26 kg (with battery and FCC system)
	Takeoff gross weight: 2.8 kg (with payload)
Powerplant	DYS D4215-650kV (1.62 kg per 1 each)
	Maximum 12 min (battery 5200 mAh, 14.8 V, 30 C)
Flight control system	TM4C129XNCZAD TI microcontroller
	PWM generation board,
	BNO55 IMU + HMC5983 compass
	2.4 GHz Wi-Fi access point network
	1D LiDAR
Position accuracy	Horizontal: 0.3 m; vertical: 0.1 m

4.2 | Sensor system

First, we selected a stereo camera to implement the relative-distance-based method for collision avoidance. This approach, which will be described in Section 5.3, worked very efficiently during the contest. Figure 5 shows the stereo camera that provided various data to the user using FPGA and an internal library, thereby allowing the users to easily obtain a depth map and point cloud data in a single line of library functions in their code. The RGB and depth map image were used for gate detection and obstacle avoidance. An optical flow sensor and a 1D LiDAR sensor were installed at the bottom of the vehicle to enable basic hovering and autonomous flight in indoor environments, where GPS signals cannot be received. The optical flow sensors were capable of obtaining vehicle speeds at frequencies of approximately 400 Hz. However, a complementary filter was used in combination with an acceleration sensor to solve the problem in our system because the sensor noise and speed to be measured varied.[38] This technique will be described in more detail in Section 4.3. Furthermore, we applied two

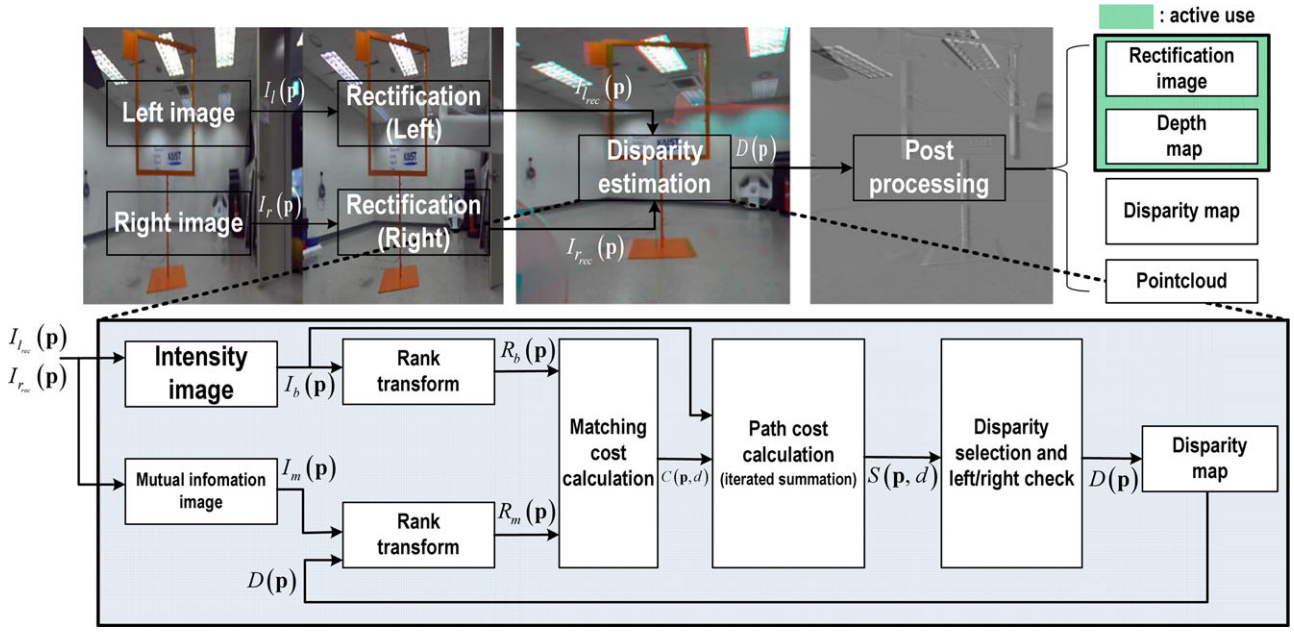

FIGURE 5 Stereo camera image-processing flow using the left/right input image.[41,42] RGB and depth map image were used for gate detection and obstacle avoidance

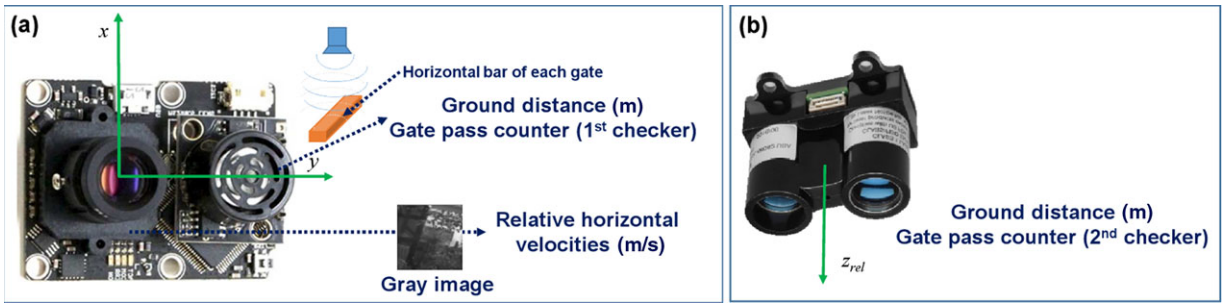

FIGURE 6 Additional sensors for the challenge: (a) optical flow sensor (PX4Flow, 3D Robotics) for velocity estimation and (b) 1D LiDAR (LiDAR Lite, Garmin) for height sensing

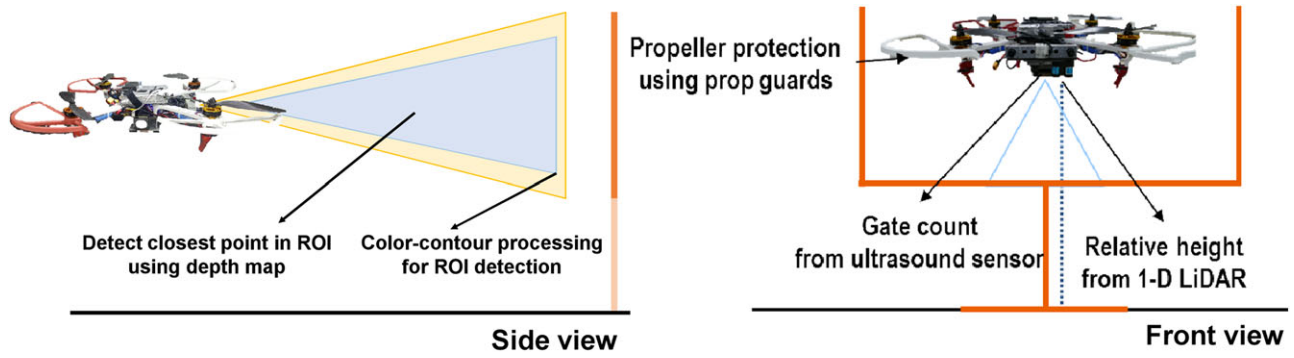

FIGURE 7 Conceptual sketch of sensors usage for gate and pass through detection (left: stereo camera's; right: 1D LiDAR and optical flow sensor)

additional sensors (Fig. 6). PX4Flow is one of the most famous optical flow sensors made by ETH Zurich, Switzerland.[39] Its detailed specifications can be found in Meier.[40] The horizontal relative velocities obtained from the optical flow sensor and the relative ground distance obtained from the ultrasonic sensor were used for 3D relative navigation and the gate pass counter to implement our strategy. The relative horizontal velocities were used to replace the GPS velocities in the navigation filter. The relative altitude was measured using the 1D LiDAR because it had less noise and was faster and more accurate than the ultrasonic altimeter. However, since the LiDAR can fail to detect the horizontal bar below due to its small detection area, the ultrasonic sensor is used as a complementary manner. Therefore, we used both sensors as the redundancy concept to count or check each gate. Figure 7 described the role of sensors, which are represented in this section.

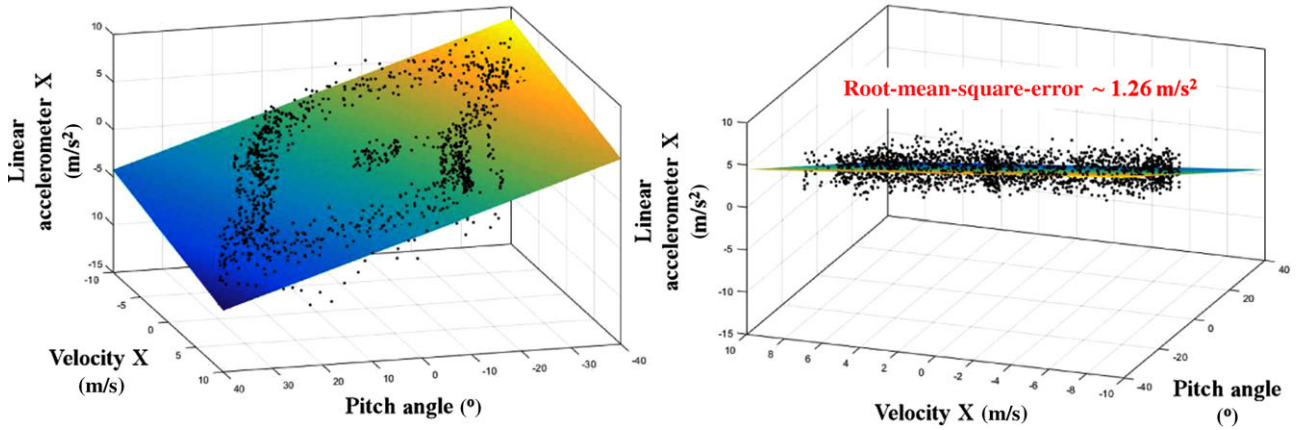

FIGURE 8 Flight data according to Eq. (3). c_1 and c_2 in Eq. (3) can be obtained through the correlation between velocity, Euler angles, and acceleration, so that acceleration term can be estimated only by the information of velocity and Euler angle

4.3 | Indoor configuration

As the challenge was held in an indoor environment with no external positioning sensors (e.g., GPS or a motion capture system), we employed visual servoing as the main strategy. To stabilize the vehicle, the controller needs to know the accurate velocity. A complementary filter with an acceleration sensor (MPU9250) and an optical flow sensor (PX4Flow) attached to the lower part of the vehicle (Fig. 4) was constructed to achieve a more robust velocity estimation. The accuracy of velocity estimation was further improved by filtering the linear acceleration using a Luenberger observer, as explained in the following subsection.

4.3.1 | Design of linear acceleration observer

A Luenberger observer[43] is applied to estimate the noise-free angular velocity and linear acceleration components along the body axis, as the noise generated by these components is the greatest during flight. In general, the biggest criterion for applying the Luenberger observer and the Kalman filter depends on whether the system is deterministic or probabilistic. In the case of typical multirotor drones, the stochastic modeling approach, such as the nonlinear Kalman filter, is adopted to estimate state variables with the process noise covariance matrix, Q and measurement noise covariance matrix, R. However, in our case, both matrices are manually tuned with respect to the changing flight conditions because the PX4Flow sensor does not offer measurement noise covariance. To overcome these difficulties, a set of attitude control flight tests were analyzed to determine the gain of the Luenberger observer as shown in Figure 8. Therefore, the proposed observer is designed with respect to the angular velocity and linear acceleration components along the body axis and the proper estimated states can be obtained. It is described by Eq. (1):

$$\dot{\hat{X}} = A\hat{X} + BU + L(X_{\text{raw}} - \hat{X}) \tag{1}$$

The linear acceleration has a relationship with the trigonometric function of the attitude, and the square of the linear velocity (Eq. (2)). Where A_{x_1} is the linear acceleration, θ_{cur} is the pitch angle and V_{x_1} is the linear velocity. In this study, however, there is only a relatively slow linear velocity (the drag can be approximated by a proportional relation to the speed at a relatively low speed) and a small attitude change (the sine function can be linearly approximated to a small angle θ).

$$A_{x_1} = g\sin\theta + 0.5\rho V^2 C_d \tag{2}$$

Thus, the linear acceleration dynamic model of the UAV can be approximated as shown in Eq. (3) for the x-axis (of the $x_1 - y_1 - z_1$ coordinate system). The two constants in Eq. (3) used the results obtained from the curve fitting based on actual flight-test data (Fig. 8); it reflects the actual flight characteristics.

$$A_{x_1} = C_1\theta_{\text{cur}} + C_2 V_{x_1} \tag{3}$$

The derivative of Eq. (3) is as follows:

$$\dot{A}_{x_1} = C_2 A_{x_1} + C_1\dot{\theta}_{\text{cur}} \tag{4}$$

The attitude control response, which is similar to the first-order transfer function can be expressed as Eq. (5), where θ_{cmd} is the pitch angle command, and b is a constant:

$$\dot{\theta}_{\text{cur}} = b\theta_{\text{cmd}} - a\theta_{\text{cur}} \tag{5}$$

The Luenberger observer can be obtained as follows by substituting Eqs. (4) and (5) into Eq. (1):

$$\dot{\hat{A}}_{x_1} = C_2\hat{A}_{x_1} + C_1 b\theta_{\text{cmd}} - C_1 a\theta_{\text{cur}} + L\left(A_{x_1,\text{raw}} - \hat{A}_{x_1}\right) \tag{6}$$

The designed Luenberger observer for linear acceleration adequately reduces noise and can therefore enhance linear speed control. Figure 9 presents the experimental data obtained using the designed Luenberger observer. The estimated linear acceleration exhibited much less noise than the raw data.

4.3.2 | Velocity estimation algorithm

A complementary filter was used to estimate the velocity of the vehicle. Complementary filters have many advantages when implemented in microcontrollers because their structures are simpler than other navigation filters, thereby making them easier to implement and their

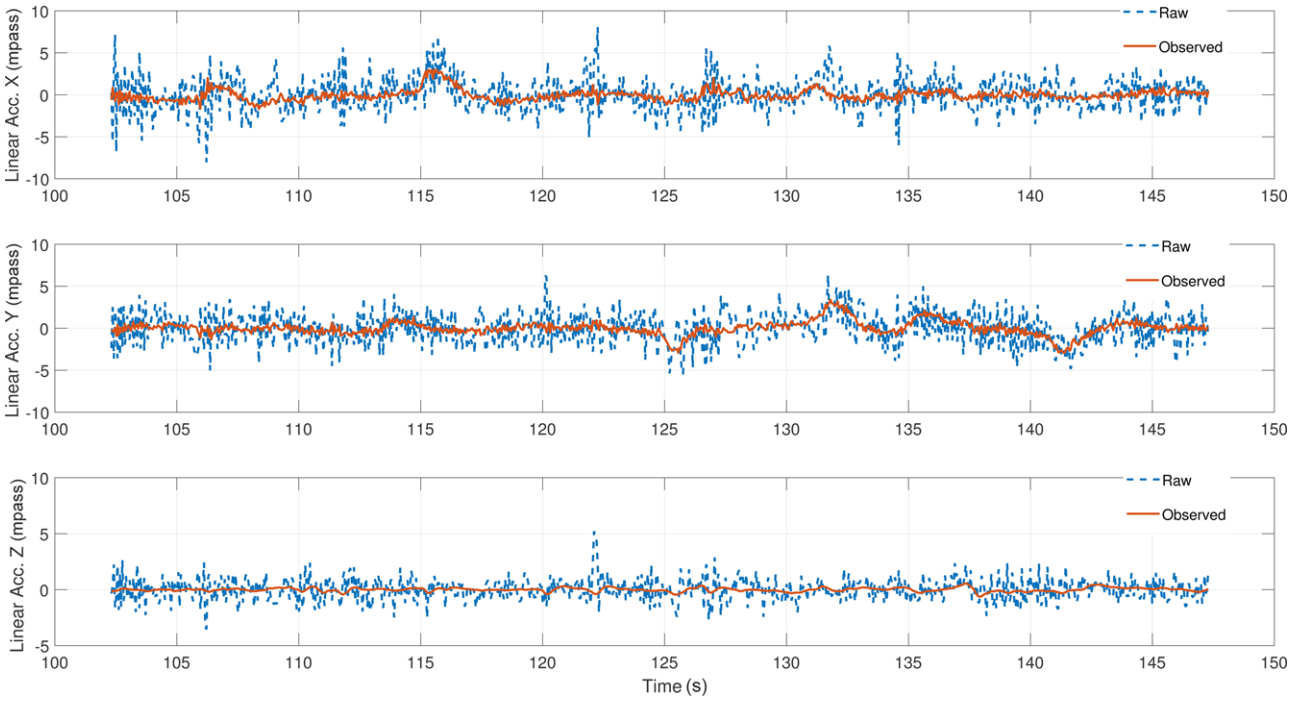

FIGURE 9 Luenberger observer for linear acceleration filtering. Noise in the raw data of acceleration measurement is effectively reduced without phase lag

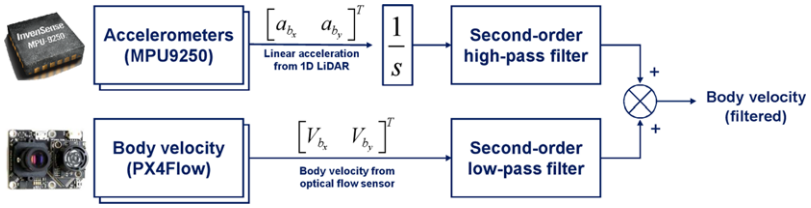

FIGURE 10 Complementary filter for the velocity estimation with acceleration compensation. Acceleration data are comes from Luenberger observer for noise reduction purpose

computational loads are easily reduced.[44,45] Acceleration can be more accurately measured using a high-pass filter to eliminate bias and its integration. For velocity measurements, a low-pass filter is used to filter out any high-frequency noise in the PX4Flow measurement. Finally, both values that passed through each filter are combined to estimate the velocity. The structure of this complementary filter is depicted in Figure 10.

5 | OBSTACLE DETECTION AND GUIDANCE LAW

This section describes the creation of a guidance point for visual servoing. The guidance point includes a center point for the UAV to pass through at the center of the gate, a depth value at both ends of the gate for heading correction, and a depth map–based minimum distance point for collision avoidance. The challenge web site* provided information on the obstacles, such as colors (bright orange), shapes (square), and dimensions (1.2 × 1.2 m). All of this knowledge was crucial for detecting the gates using our system. When an edge of gate in the image frame is detected based on the color information and the edge detector, it is expanded using dilation to easily detect the contours of the gate. The center point of the gate can be obtained if the contours of the gate are detected by this method (Fig. 11). The center point was used as the guidance point through which the vehicle intended to pass. The detailed descriptions of the algorithms are provided in the subsections that follow.

5.1 | Center point matching

The challenge required the vehicle to pass through obstacles, rather than simply avoiding them; thus, a center point-matching technique was developed. In this method, the horizontal center of the gate was made to coincide with the center of the projected image in a 2D image plane, thereby directing the vehicle toward the center of the gate and enabling it to pass through the gate by using this center point for guidance.

* http://ris.skku.edu/home/iros2016racing.html.

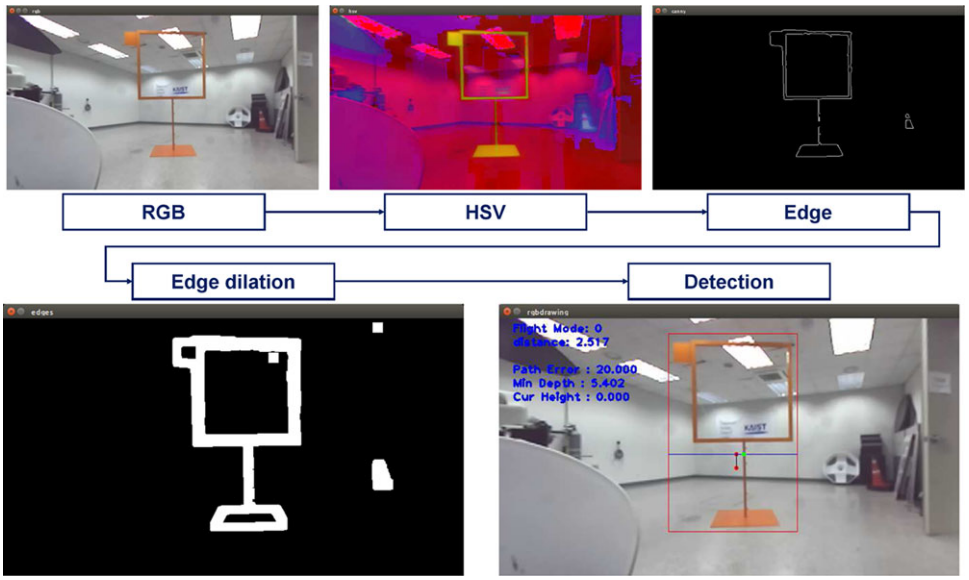

FIGURE 11 Gate detection process based on color information. The RGB image is converted to a hue-saturation-value (HSV) frame to capture orange color. The Canny edge detection method is used, and detected edges are dilated to have a thick frame shape

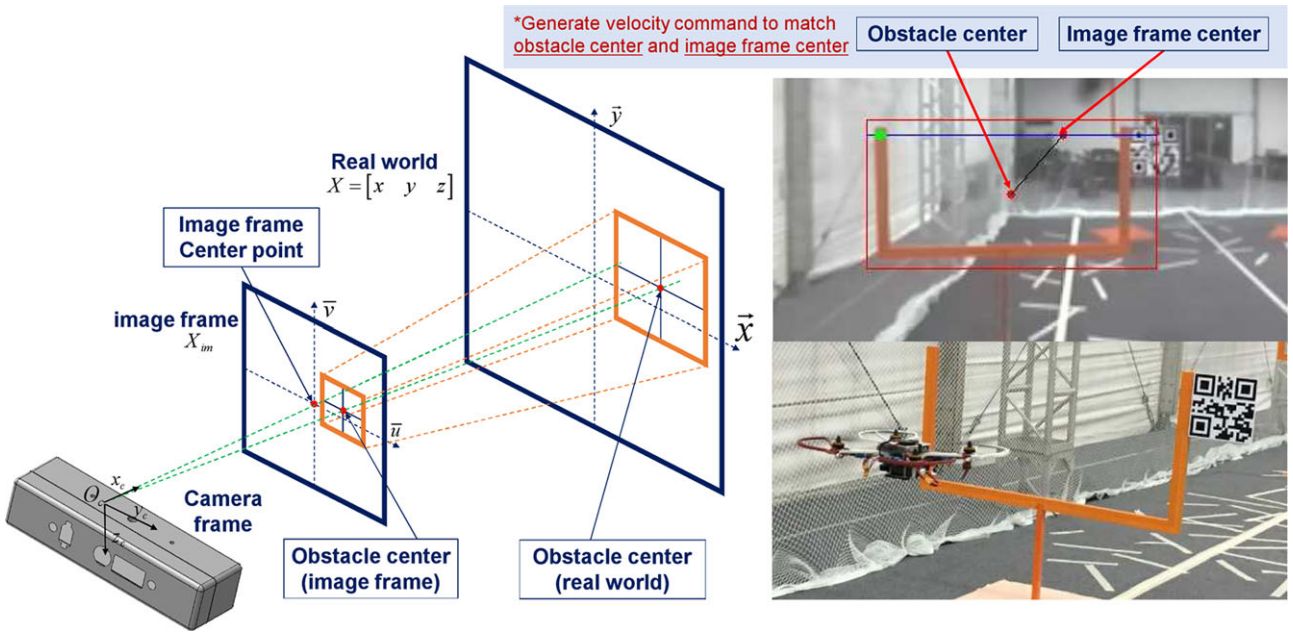

FIGURE 12 Geometric relationship between the image frame and the real world; the horizontal center points of the image frame and the detected gate are used for reference in our visual servoing technique

First, a binary mask was generated with a certain threshold (i.e., a threshold appropriate for bright orange image data was employed in this challenge) to find the gate center. Moreover, the edges were extracted from the images by using the OpenCV* open source library. A red square box was drawn around the detected gate, and a center point was found based on the width and height of the box. Figure 12 shows the generation of the UAV body velocity control command; the horizontal coordinate of the gate center point coincided with the *x*-coordinate of the center point of the current vehicle image frame.

For the vertical center, we use the map information, which contains the geometry of each gate. We use the 1D LiDAR reading to accurately determine the altitude. This approach is quite effective when no global localization is available owing to limited computation power.

5.2 | Heading angle compensation using depth map data

In order for the drone to pass through the gate without colliding, it is desirable that the vehicle flies through the gate while aligned with the normal vector of the gate, as shown in Figure 13 without skew.

* opencv.org.

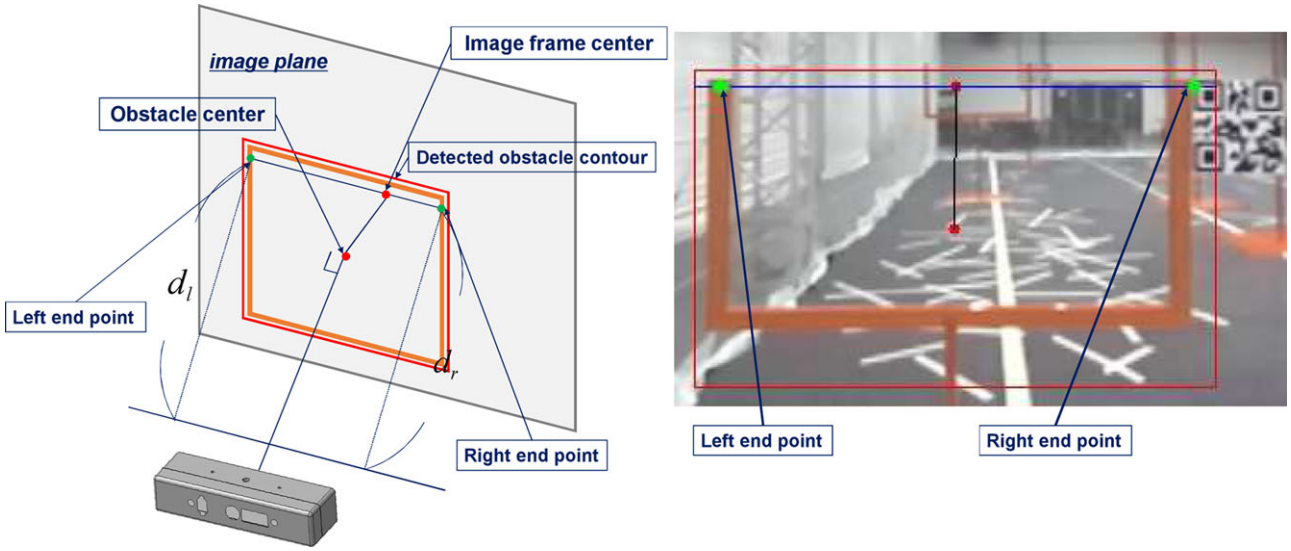

FIGURE 13 Heading errors are corrected based on the distance differences between the left and right ends of the detected gate. This method makes the gate is in a plane parallel to the vehicle's image plane

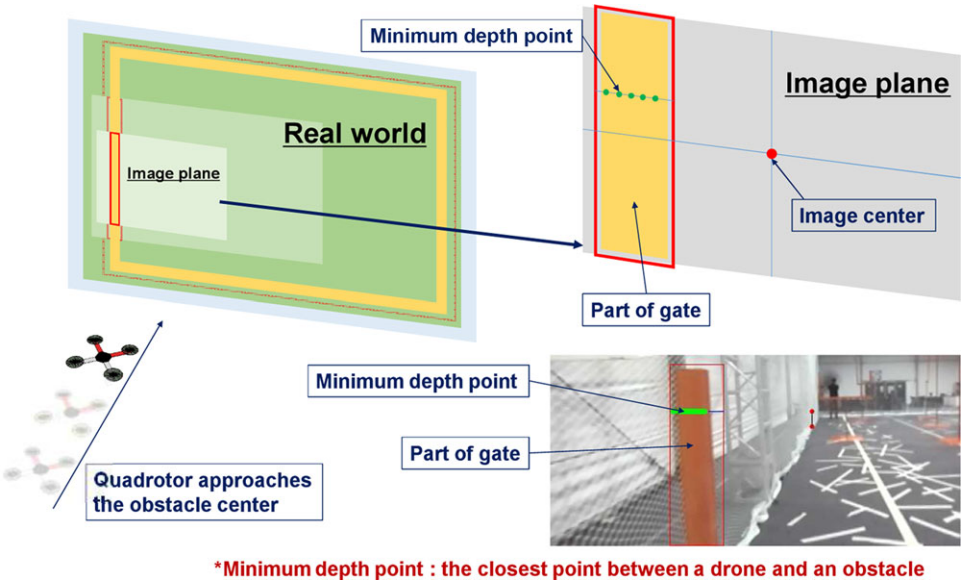

FIGURE 14 Only a portion of the gate is visible as the drone approaches; therefore, the center point is not detected. Thus, the minimum depth point of the detected gate region is used as the guidance point for collision avoidance instead

The nominal angle of each gate was provided in the official map. However, a gate is always installed with a certain amount of error. This mismatch cannot be known prior to the flight. Therefore, the heading angle between the gate and the vehicle was corrected according to the depth map data provided by the onboard stereo camera. Once an obstacle is detected, a line is drawn through the image frame center (Fig. 13). The locations of the two points at the ends of the obstacle, at which the line and the contour of the detected obstacle meet, were then determined. In this part, we only considered the horizontal mismatch of the alignment problem as the obstacles had very small vertical errors (i.e., they were always installed upright). The heading error can be obtained using Eq. (7) based on the difference between the distances d_l and d_r between the drone and the left and right ends, respectively, of the obstacle. These errors generate the yaw rate command input to correct heading mismatches between a drone and an obstacle, as shown in Eq. (8):

$$\psi_{\text{error}} = (d_l - d_r) \begin{cases} \text{turn left}, & (d_l - d_r) > \varepsilon_{\text{thres}} \\ \text{turn right}, & (d_l - d_r) < -\varepsilon_{\text{thres}} \end{cases} \tag{7}$$

$$r_{\text{cmd}} = K_p \cdot \psi_{\text{error}} \tag{8}$$

5.3 | Depth-based collision avoidance

As the drone approaches a gate using the center-point matching method, the gate gradually escapes the field of view of the lens and cannot be captured entirely by the camera (Fig. 14). At this point, the center-point matching method cannot be applied. Instead, a collision

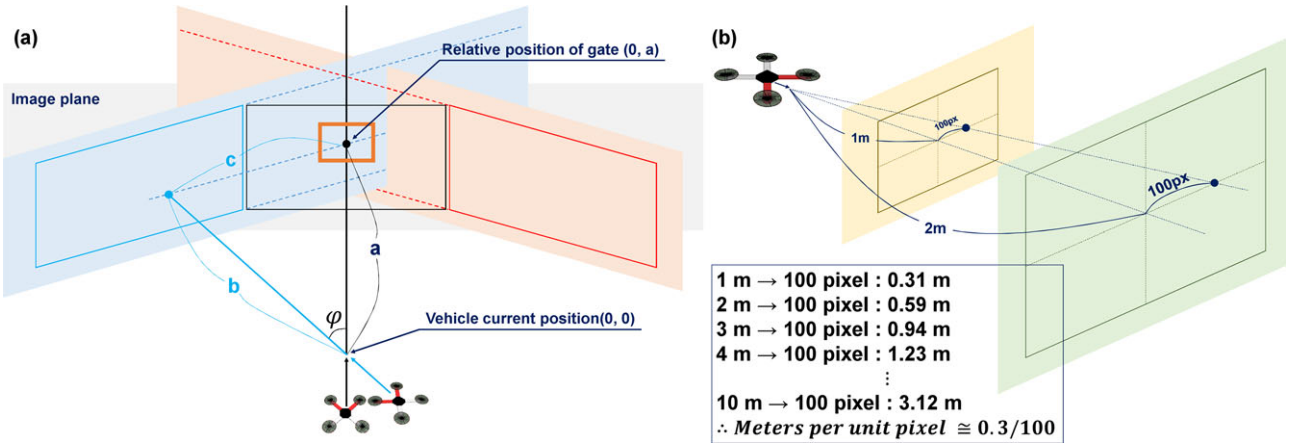

FIGURE 15 Relative position acquiring procedure according to (a) heading angle and (b) pixel distance calibration to real-world distance for depth-based collision avoidance. When the projected minimum depth point from image frame to real world is within 1 m in the vertical direction and 0.4 m in the horizontal direction the collision avoidance command are generated

avoidance method based on the depth information is used to prevent such guidance point loss problems from occurring. In this method, the minimum depth points are used. The minimum depth point is the closest point from a drone to an obstacle. When this point is detected and the depth map value of this point is within 1 m, the lateral velocity command is generated to avoid from collision by moving in the direction opposite to the gate.

To use this point for collision avoidance, the coordinates of this point from the image frame should reflect the current attitude information of the vehicle. Figure 15(a) describes that the minimum depth point was recognized at a location of (c, b) with respect to the drone if a gate at $(0, a)$ is viewed at an angle ψ (the heading angle, ψ between the current obstacle and the drone was obtained by calibrating the difference of d_l and d_r to the heading angle). The obstacle's relative position, irrespective of the attitude of the drone, can be obtained by applying Eq. (9), where b is the distance between the gate and the UAV and c is the distance between the calibrated lateral axis and the pixel distance (Fig. 15(b)). The lateral current position X_{cur} was determined by applying Eq. (10), whereas the longitudinal current position Y_{cur} was measured from the depth map.

$$\begin{bmatrix} X_{rel} \\ Y_{rel} \end{bmatrix} = \begin{bmatrix} 1 & -\cos\psi \tan\psi \\ 0 & (1+\cos^{-1}\psi) \end{bmatrix} \begin{bmatrix} X_{cur} \\ Y_{cur} \end{bmatrix} \tag{9}$$

$$X_{cur} = Y_{cur} \cdot (\text{Number of pixels on image}) \cdot (\text{Meters per unit pixel}) \tag{10}$$

The obstacle position was given by the corrected relative coordinates. Consequently, collisions when the vehicle was within 1 m off the front left or right of the gate can be avoided by moving in the direction opposite to the gate. Algorithms 1 and 2 represent the overall algorithm flows explained in this section.

5.4 | Mission planner

Since the arena is small yet densely populated with the similar gates, it is true that many existing vision-based localization method would fail in this environment. So, it is hard to expect robust positioning solution for path or mission planning. Therefore, localization techniques such as visual-odometry or SLAM are not applied to our strategy. Instead, for mission planning, logic-based three preplanned commands are calculated for each gate: heading, altitude, and forward speed. Heading and altitude commands are calculated based on the global map (Fig. 1) provided by the competition organization. Headings are preprogrammed to pass through the gate orthogonal, and altitudes are calculated to pass the vertical center of the square frame. In addition, forward speed was chosen from a series of flight tests prior to the competition by considering the image-processing speed and gate-checking accuracy. Whenever the gate checking is detected from sonar and LiDAR, which are mounted under the vehicle, a control command is updated to go over to the next gate. The mission commands are updated as follows:

$$\begin{cases} \psi_{cmd}^{k+1} = \left(\psi_{cmd}^{k} + \psi_{wp}^{k+1}\right) \\ h_{cmd}^{k+1} = \left(h_{cmd}^{k} + h_{wp}^{k+1}\right) \end{cases} \tag{11}$$

where ψ represents the heading, h represents the altitude, and the subscripts cmd and wp represent control command and precalculated value from the global map, respectively.

6 | EXPERIMENTAL RESULTS

The proposed visual servoing method was implemented on an embedded NVIDIA Jetson TK1 computer. The main algorithms used during the challenge were the center point matching, heading angle error compensation, and depth-based collision avoidance algorithms described in Section 5. These algorithms were all based on ROS* and transmitted the calculated speed command to the FCC via USB to the serial interface at 20 Hz.

The gate location and the heading info for the venue were entered according to the map data announced before the challenge. We corrected the misalignment between the map and the actual venue

* https://www.ros.org.

TABLE 3 Pseudocode for obstacle detection and mission control command generation algorithm

Algorithm 1. Obstacle Detection

Inputs : Stereo image data P_{edge}, P_{depth}

while ( )
  Contour ← findContours(P_{edge})
  if ***Contour*** is not empty **then**
    for *i* = 1 to *i* = ***Contour***.size( )
      BoundRect[*i*] ← BoundingContour(***Contour***)
      if ***BoundRect***[*i*].area > BoundMaxArea **then**
        BoundMaxIndex ← *I*
        BoundMaxArea ← ***BoundRect***[*i*].area
      end if
    end for
    BoundCenter = ***BoundRect***[BoundMaxArea].center
  end if
  for *j* = ***BoundRect***.TopLeft to *j* = ***BoundRect***.BottomRight
    Distance[*j*] ← P_{depth}[height/2, j]
    if ***Distance***[*j*] < MinDistance **then**
      MinDistanceIndex ← *j*
      MinDistance ← ***Distance***[*j*]
    end if
  end for
end while

Algorithm 2. Mission Command

Inputs : BoundCenter, MinDistance

InitialSetting ( )
while ( )
  if flight mode is mission **then**
    HeadingCmd ← ***WayPointHeading*** [GateNum]
    AltitudeCmd ← ***WayPointAltitude*** [GateNum]
    if (CurrentAltitude − SonarOffset) > GateThreshold
      GateNum ← GateNum + 1
    end if
    if HeadingError < 20.0 [deg] **then**
      VelocityCmd ← FeedbackPDcontroller(HeadingCmd, AltitudeCmd)
      if (MinDistance > 0.5)&&(MinDistance > 0.95)
        VelocityCmd.Aileron = ***VelocityCmd***.Aileron +Avoidance
      end if
    end if
    SerialSend(VelocityCmd)
  end if
end while

through the proposed visual servoing method. The passage through each gate was checked using the LiDAR, and the ultrasonic sensor installed at the bottom of the UAV. The present position was estimated according to the number of gates checked. The altitude was controlled based on the previously obtained map data according to the estimated current position. The 1D LiDAR sensor mounted at the lower end was used for altitude measurement. Our flight strategy was to pass through the gates one by one without collision by using roll control and the proposed visual servoing guidance law while flying at a precalculated forward velocity and adjusting the course heading by sensing the nearby gates. All computations were performed by onboard computers, and no communication with a ground station was utilized. Figure 16 shows the overall algorithm flow. The data logged during the challenge are shown in Figures 17–22. The velocity, attitude, and altitude commands and vehicle responses from the actual challenge in each section are shown in Figure 16.

6.1 | Section A (straight)

Figure 19(c) shows a course that consists of two open top gates with different heights, which were indicated by "Gate check" in the 1D LiDAR time history. Figure 18 represents the legend of the overlay texts in Figures 19–22. The lateral velocity command sent using the proposed visual servoing guidance law was generated as well and is shown in Figures 19(d) and 19(e). Furthermore, the longitudinal velocity, pitch angle and heading were controlled accurately by incrementally adjusting the predefined reference commands based on the environment in Figures 19(f)–19(h). The relative altitude increased [Fig. 19(c)] because of the predesigned guidance law based on the global map.

6.2 | Section B (smooth turn)

Section B consisted of two right-open gates and one square gate. All three gates in this section were installed at different heights. The time history of the flight states indicated that, overall, the vehicle passed through this section successfully by following the preplanned reference commands, except for collision avoidance between the fourth and fifth gates [Fig. 20(e)]. The preplanned course heading and forward velocity commands in this section were expected to be very difficult for the underactuated platform to simultaneously follow. A collision avoidance command based on the proposed visual servoing approach was generated to enable the vehicle to pass through the gate safely. The UAV followed the command well (Fig. 20).

6.3 | Section C (sharp turn)

Section C consisted of three closed-type gates at the same height [Fig. 21(d)]. Overall, the vehicle passed through the section well by following the preplanned reference commands, except for collision

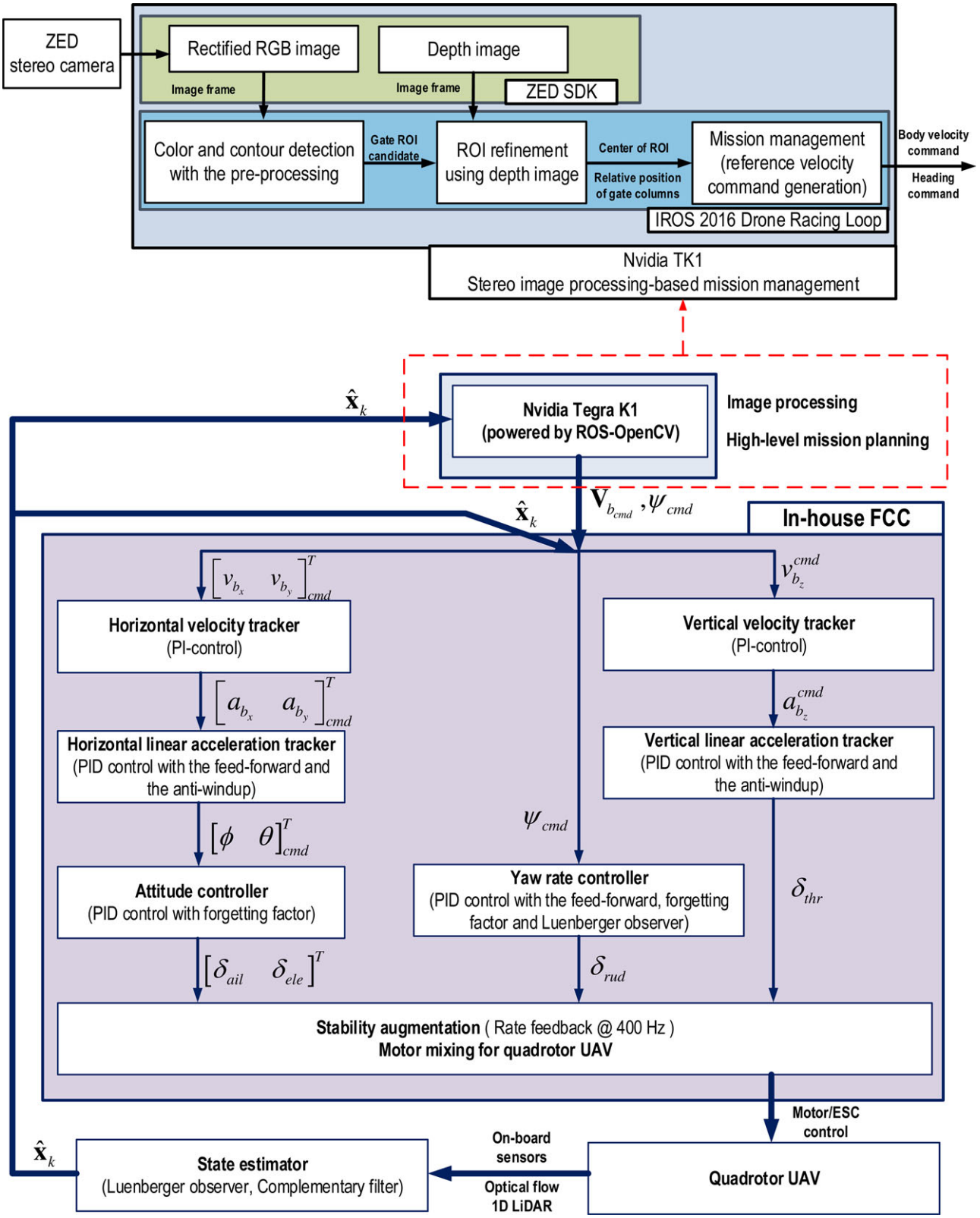

FIGURE 16 Overall algorithm flow of the UAV, for visual servoing–based obstacle detection and avoidance

avoidance near the seventh gate. Unlike in the previous sections, the UAV passed through the gates sequentially while detecting only the right side of each gate. The preplanned course heading changed significantly while the vehicle followed the forward velocity command in this section of the course. However, the UAV had difficulty in quickly following the preplanned reference command because the moment of inertia on the *z*-axis was larger than the moments of inertia on the other axes. In addition, the UAV passed through each gate

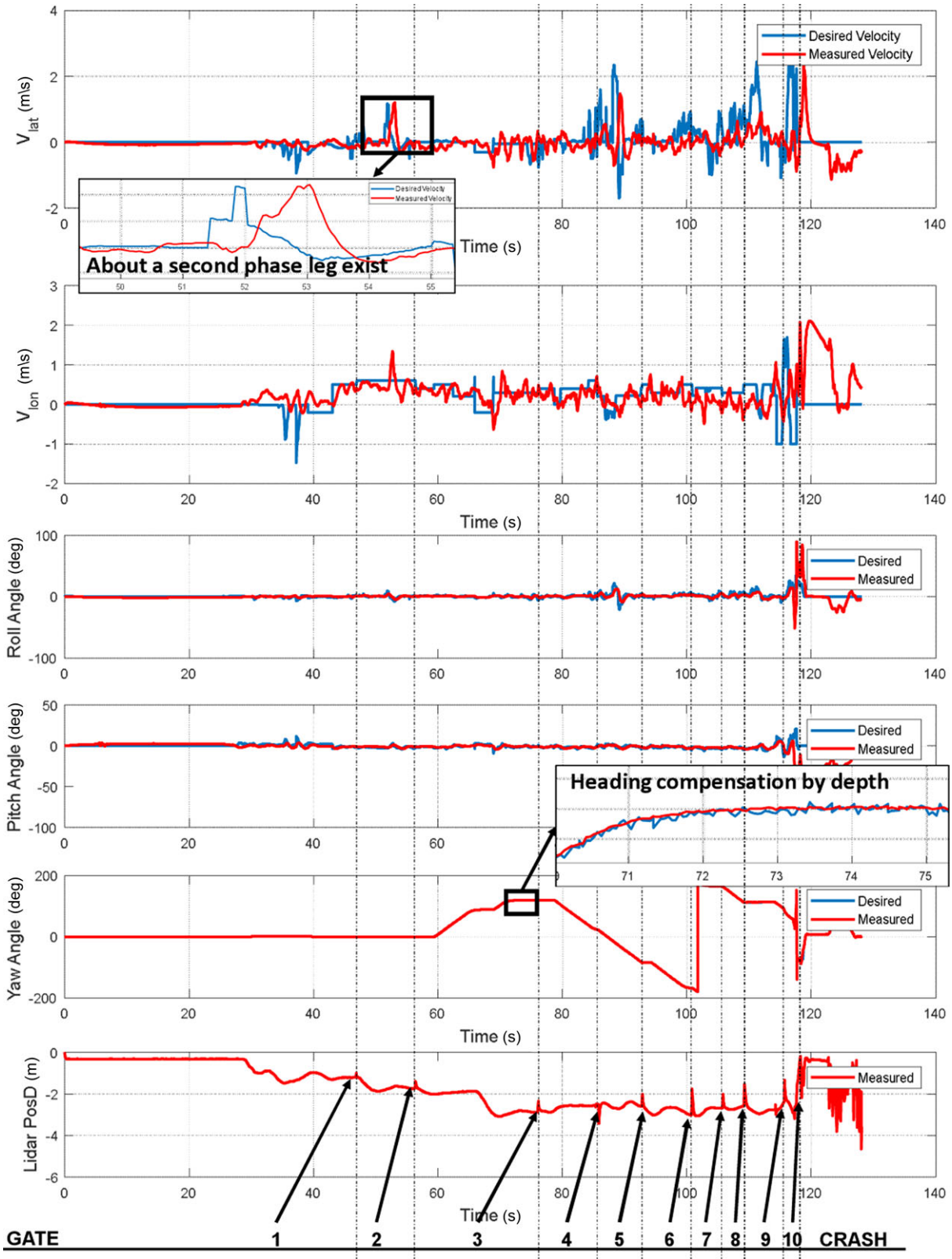

FIGURE 17 Flight data obtained during the challenge. The underactuated system produced a phase delay of approximately 1 s. Passing through the 10 gates is clearly visible as a series of spikes in the LiDAR data, and the results of the depth-based heading correction are also shown in the yaw angle graph

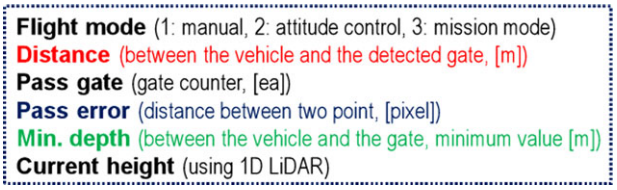

FIGURE 18 Legend of overlay text in the left top of image

without seeing its left column because is simultaneously followed the forward velocity and the course heading commands. Although the UAV could not see the left columns in these cases, our proposed visual servoing approach effectively generated the collision avoidance commands, which were also followed (Fig. 21).

6.4 | Section D (smooth turn)

Section D consisted of four closed-type gates at the same height. Our UAV passed through two gates in this section. The ninth gate was easily navigated, but the UAV could not achieve the desired velocity after passing it and the error occurred simultaneously in the course heading data (Fig. 22). For this reason, after passing the 10th gate, the UAV collided with the right column of the 11th gate and crashed. A more detailed analysis of the cause of the collision is given in the section that follows.

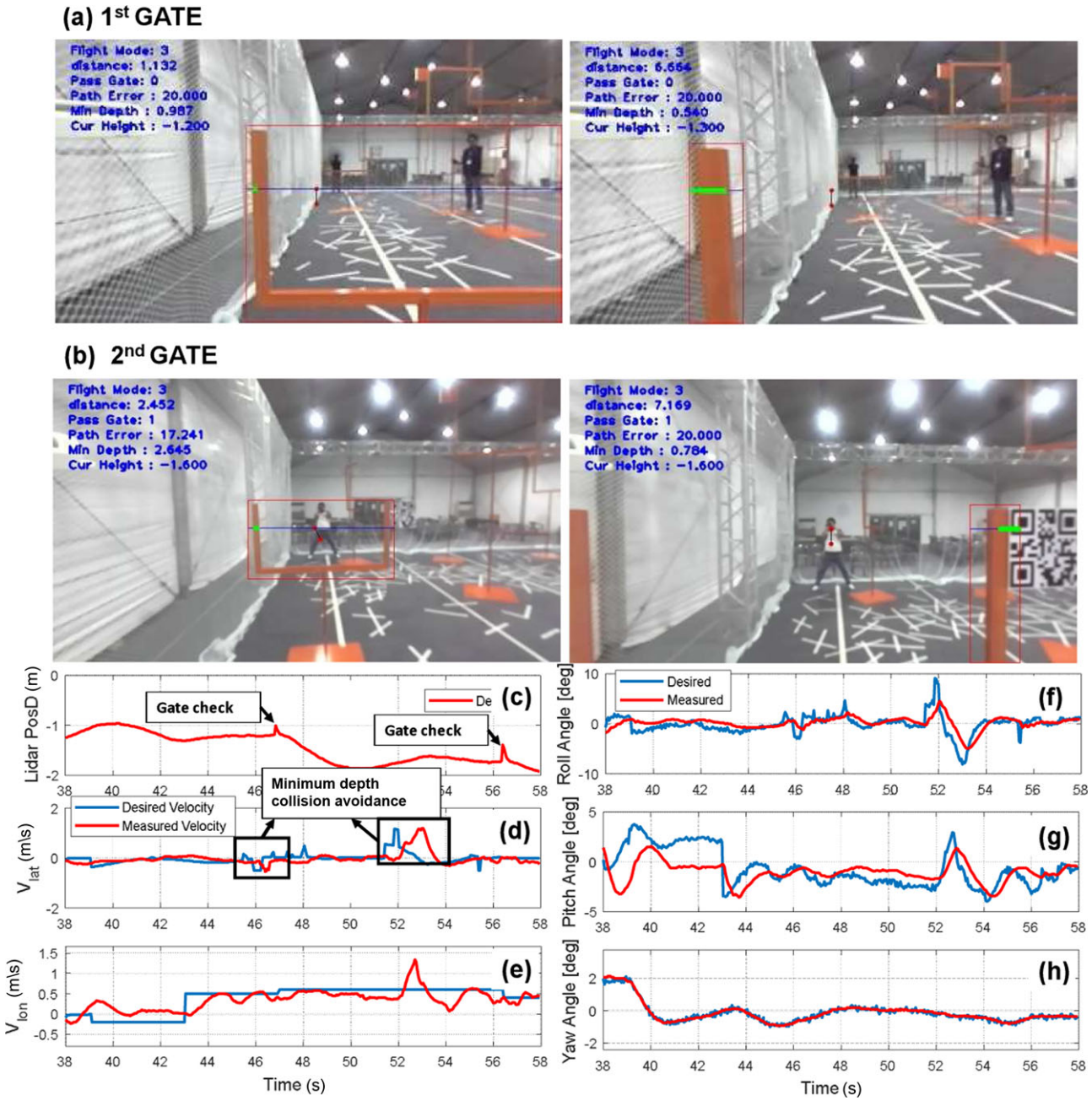

FIGURE 19 Processed images (upper) and the flight state time histories (lower) while the vehicle passes through the Section A (straight); the green dots of the right side of (a) represent the minimum depth point to collision avoidance; (c) shows that the gates were successfully detected; and (d) indicates the minimum depth collision avoidance visual servoing commands and responses

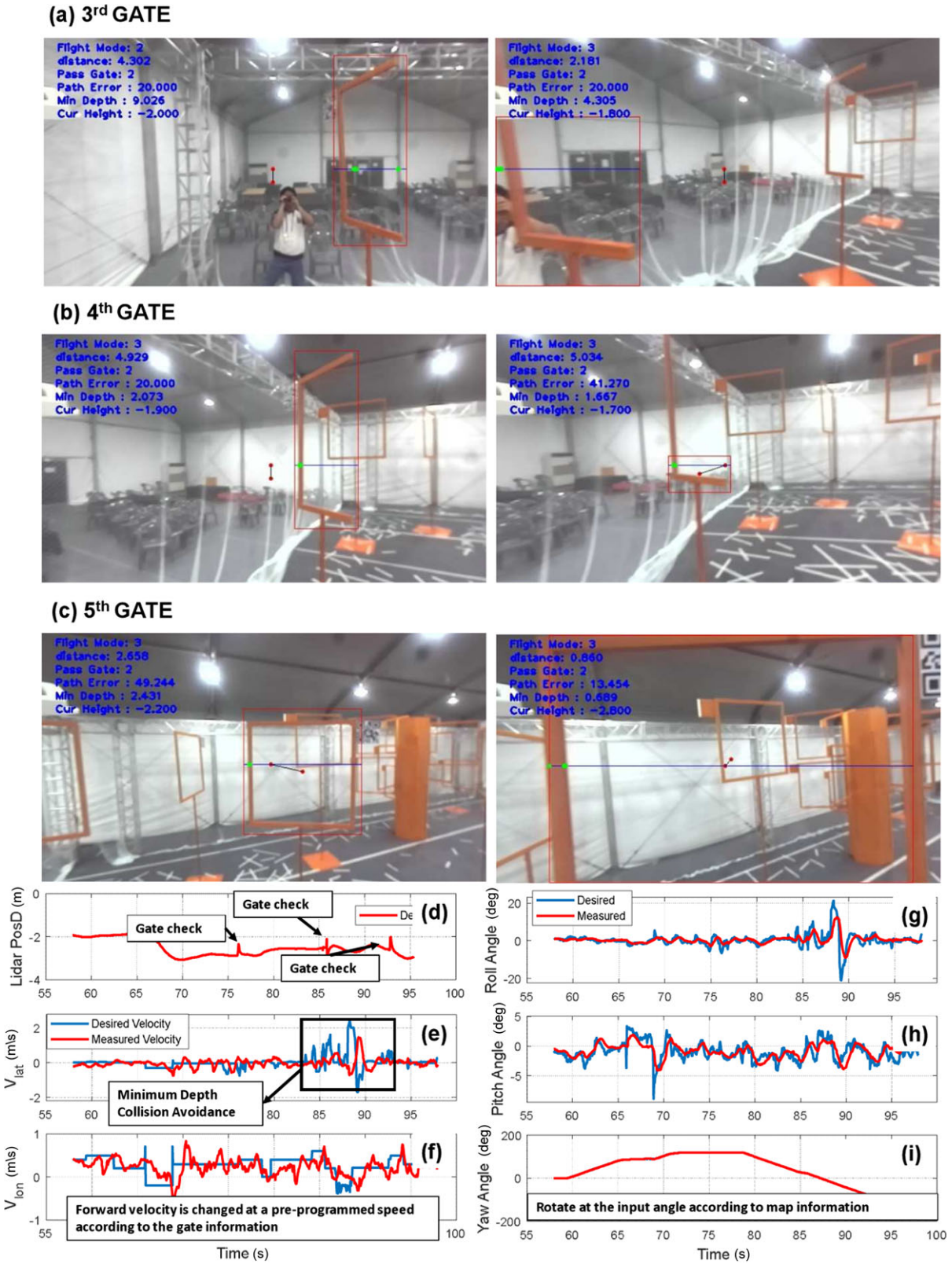

FIGURE 20 Processed images (upper) and the flight state time histories (lower) obtained while the vehicle passes through Section B (smooth turn); (d) shows that the gates were clearly detected, whereas (e) indicates the minimum depth collision avoidance visual servoing commands and responses; the vehicle heading and forward velocity in this course were followed by preprogrammed data ((f) and (i))

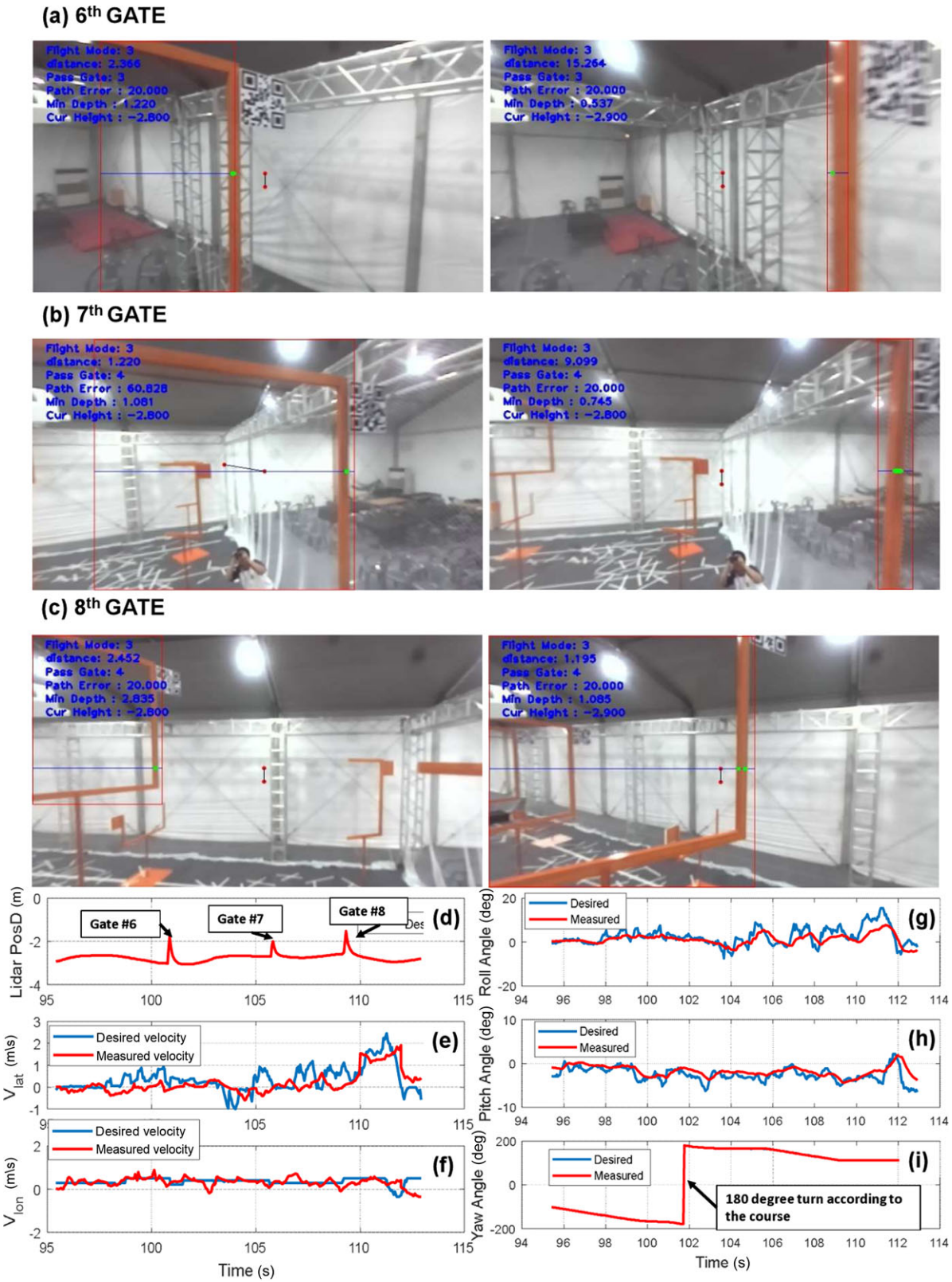

FIGURE 21 Processed images (upper) and the flight state time histories (lower) obtained while the vehicle passes through Section C (sharp turn); (d) shows that the gates were clearly detected, whereas (i) indicates that a 180° turn occurred

6.5 | Discussion

Our approach for this challenge was to correct errors between the given map (e.g., heading angles and height of the gates) and the actual obstacles using the center points and the minimum depth points of the detected gates. This approach allowed the drone to pass safely through the gates without any collisions. As this approach required a low computation load, it was suitable for small drones operating in a complex indoor environment. However, this method alone does not guarantee

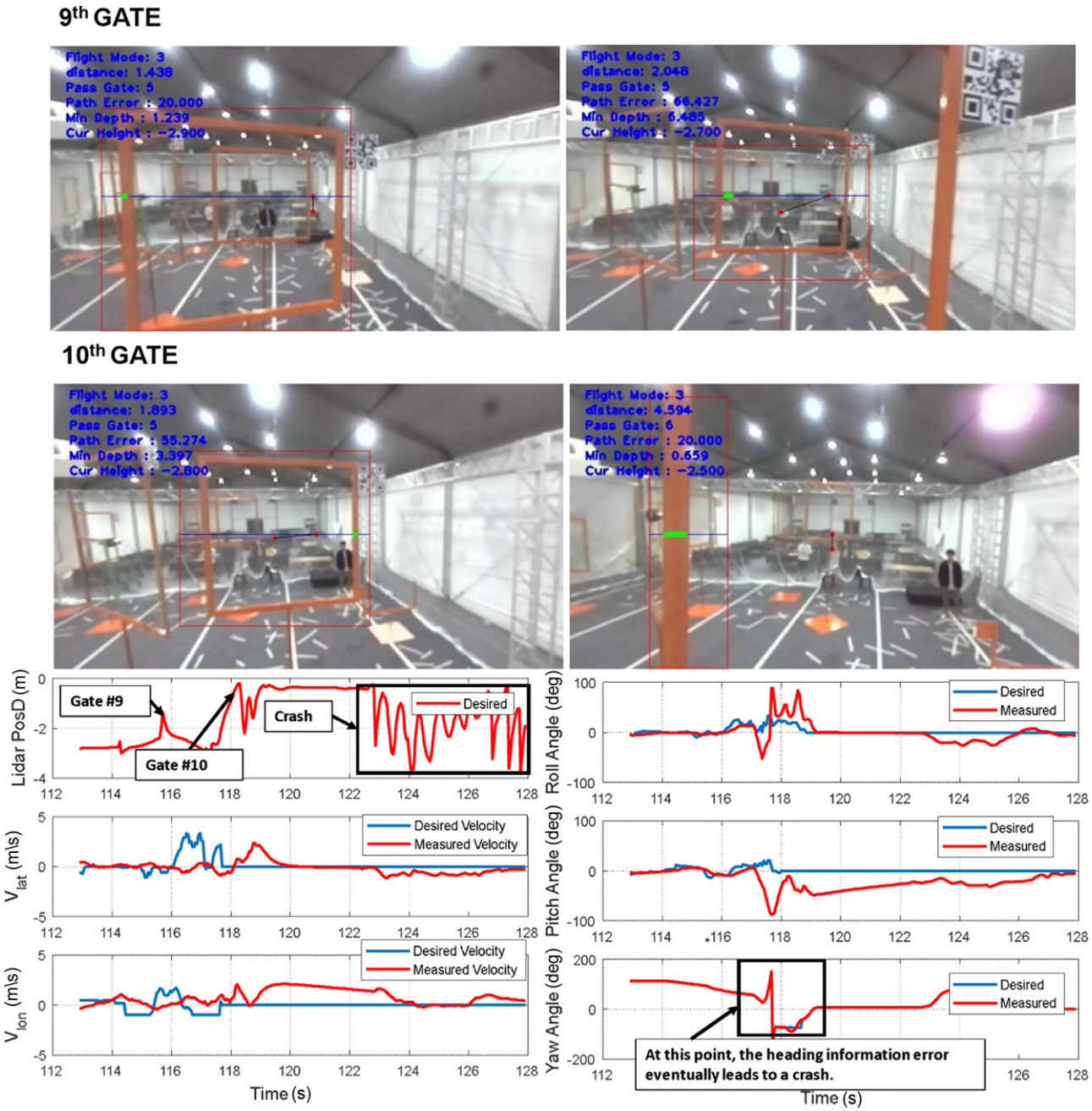

FIGURE 22 Processed images (upper) and the flight state time histories (lower) while the vehicle passes through Section D (smooth turn); (c) shows the clearly detected gates and crash point, whereas (h) indicates heading errors which prompted it to lose the visual servoing guidance point that eventually led to a crash

the drone will find its way back onto the correct course if it fails to follow the preset heading and altitude commands. In our case, we passed through Gate 10 but collide with Gate 11 owing to heading error. It seems to be occurred that the heading angle errors accumulated during Sections B and C caused the drone to deviate from the original course. Therefore, it generated the wrong commands which caused the drone to deviate from its predetermined heading and altitude commands. The list below describes in detail the advantages and disadvantages of our approach, and some lesson learned.

- *Direct visual servoing and depth-based collision avoidance method:* In the challenge, these methods worked very effectively and eventually we won with this method. Our proposed approach correcting errors including the heading and altitude between the predefined global map and the local sensing result has shown well performance. However, it is vulnerable to the current sensing error and its accumulation. Furthermore, it does not guarantee the redundancy or the contingency planning. After the challenge, interesting papers passing narrow gaps with aggressive maneuvering through image recognition or trajectory planning have been published [46,47] and this approach is also worth considering.
- *Color-based gate detection*: The color detection method based on the HSV color model has advantages of fast and robust image-processing performance. Also, once the parameters are set, HSV

model–based color filtering can return the proper solution under the same light conditions. Therefore, it can be said that it is suitable for the indoor environment such as the venue where the light condition is unchanged. However, if multiple obstacles are overlap each other with a depth difference, an error may occur in finding the center point of the obstacle by judging two obstacles as one because this algorithm is not considering the scale problem.

To overcome this lack of information problem, the depth-based algorithm is added. However, this information fusion algorithm cannot complement all aspects such as the light source change, color tuning failure, or depth information error, etc. Therefore, we would like to use a deep-learning based object detection algorithm such as a single shot multibox detector. Deep learning algorithms are still burdensome to implement in the embedded level. However, with the development of high-performance embedded processors, it is gradually becoming possible. If we try to detect the gate by learning in various light conditions and situations, it will be closer to a fully autonomous drone where human intervention is reduced.

- *Logic-based preprogrammed planning*: In this challenge, we did not use any indoor localization technique because the venue is surrounded by the net with no intense feature and unicolor with no texture in the ground. Furthermore, some key features to localize the platform can move by following the movement of people during the flight. Thus, it can lead to the failure of the localization algorithm.

  Therefore, we have implemented the preprogrammed logic-based mission planning method. Each gate was set to a waypoint and when a gate is detected on the LiDAR and ultrasonic sensor mounted on the bottom of the drone, the command is updated to proceed to the next waypoint. This method has an advantage that it can follow the path without error by following only the command to go to the next waypoint if there are no errors including the navigation or visual sensing part. However, as a result of this challenge, if the vehicle state coming from the sensor goes wrong, it cannot be recovered because our strategy does not guarantee the redundancy or the contingency planning.

  In fact, among the competitors, one team was trying to use a direct extended Kalman filter SLAM approach.[48] Using this method, complex obstacles such as spiral stairways can be avoided if the map is thoroughly built and used for accurate localization. However, the venue was densely filled with many similar frame-shaped obstacles with few distinct features. Therefore, localization solutions could easily diverge. This is probably the reason why the team cannot pass many obstacles through the challenge.

  To overcome such problems while minimizing disadvantages, the global path planning based on the sensor fusion–based localization is inevitably required. To reduce this position error drift and make the vehicle's state estimate robust, we will apply the UKF-based 6DOFs state estimation with the visual odometry, IMU, and altitude sensor. Through this state estimation method the heading following will be more robust then our proposed direct visual servoing method can enable us to fly further than Gate 10. For negotiating with the dynamic obstacle at Gate 26, since the bar rotates slowly, the motion can be measured and used to predict a safe four-dimensional path (position with time). This method can also be regarded as direct visual servoing using the center point of the detected bar.
- *QR code detection*: The QR code is widely used, and the recognition is not too difficult. However, since the global map as provided by the organization and since our mission planner uses this information to follow the gate sequentially using the map, the QR code was not really needed for the competition. We did not use the code partially due to the added computation load. Also, our camera has limited resolution (VGA), the QR code becomes decipherable in 2 m, which is somewhat limiting. However, if we consider the possibility to lose the track of the progress due to any kind of error, the QR code will be very helpful to get the drone back on the course. In our future work, we would like to use the QR code to make our system perform robustly even when losing track of the course.

As we discussed herein, our proposed direct visual servoing method had similar limitations due to heading and altitude errors. Overall, the combination of robust visual odometry, vision-based state estimation, and direct visual servoing is expected to deliver positive results for negotiating through a complex environments with a lower computation capacity.

7 | CONCLUSION

In this paper, we present a visual navigation strategy to fly through various obstacles in complex environments, as those created for the IROS 2016 Autonomous Drone Racing Challenge. The advantage of our proposed approach was its capability to perform reasonably well in the challenge using limited computational power. We implemented a visual servoing method as well as a depth-based collision avoidance method on a UAV with stereo cameras. Using the proposed system, the UAV could be made fully autonomous in GPS-denied environments and did not require any external positioning systems. We applied the proposed algorithm during the IROS 2016 Autonomous Drone Racing Challenge and were able to successfully navigate through 10 gates in 86 s. After a reviewing the flight data logged during the challenge, we were confirmed that our method worked quite well, making the drone to fly through 10 obstacles in the correct order. Our approach will perform notably better and enable us to achieve significantly higher score in future challenges if augmented with robust state estimation and deep learning algorithms more suitable for the challenge environment.

ACKNOWLEDGEMENTS

This work was supported by the ICT R&D program of MSIP/IITP. [R-20150223-000167, Development of High Reliable Communications and Security SW for Various Unmanned Vehicles]. This work was also supported by a grant (no. NRF-2012M1A3A3A02033464) from the National Research Foundation of Korea (NRF) funded by the Korean Ministry of Science, ICT, and Future Planning.

ORCID

Sunggoo Jung http://orcid.org/0000-0001-7672-6885

REFERENCES

1. VICON. What is motion capture? Available at https://www.vicon.com/what-is-motion-capture. Accessed January 31, 2017.
2. Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011:2520–2525.
3. Lupashin S, Schöllig A, Sherback M, D'Andrea R. A simple learning strategy for high-speed quadrocopter multi-flips. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)* 2010:1642–1648.
4. Müller M, Lupashin S, D'Andrea R. Quadrocopter ball juggling. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011:5113–5120.
5. Lindsey Q, Mellinger D, Kumar V. Construction of cubic structures with quadrotor teams. Paper presented at *Proceedings of Robotics: Science & Systems VII*, *2011*.
6. Thomas J, Loianno G, Pope M, Hawkes EW, Estrada MA, Jiang H. Planning and control of aggressive maneuvers for perching on inclined and vertical surfaces. In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2015:V05CT08A012.
7. Mellinger D, Michael N, Kumar V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int J Robot Res*. 2012;31(5):664–674.
8. Hofer M, Muehlebach M, D'Andrea R. Application of an approximate model predictive control scheme on an unmanned aerial vehicle. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016:2952–2957.
9. Muehlebach M, D'Andrea R. Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints. In: *American Control Conference (acc), 2016*. 2016:2669–2674.
10. Mebarki R, Lippiello V. Image moments-based velocity estimation of UAVs in GPS denied environments. In: *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2014:1–6.
11. Mebarki R, Lippiello V, Siciliano B. Nonlinear visual control of unmanned aerial vehicles in gps-denied environments. *IEEE Trans Robot*. 2015;31(4):1004–1017.
12. Ozawa R, Chaumette F. Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011:5670–5676.
13. Mahony R, Corke P, Hamel T. Dynamic image-based visual servo control using centroid and optic flow features. *J Dyn Syst Meas Control*. 2008;130(1):011005.
14. Moon H. IROS — 2016 Autonomous Drone Racing in MOS ESPA Daejeon arena. http://ris.skku.edu/home/iros2016racing.html. Accessed January 31, 2017.
15. Corke P. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics, Vol. 73. Berlin: Springer; 2011.
16. Chesi G, Hashimoto K. *Visual servoing via advanced numerical methods*. Lecture Notes in Control and Information Sciences, Vol 401. Berlin: Springer; 2010.
17. Hamel T, Mahony R. Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach. *IEEE Trans Robot Autom*. 2002;18(2):187–198
18. Huh S, Shim DH. A vision-based landing system for small unmanned aerial vehicles using an airbag. *Control Eng Pract*. 2010;(7):812–823.
19. Lee D, Lim H, Kim HJ, Kim Y, Seong KJ. Adaptive image-based visual servoing for an underactuated quadrotor system. *J Guid Control Dyn*. 2012;35(4):1335–1353.
20. Cho S, Lee D, Shim DH. Image-based visual servoing framework for a multirotor UAV using sampling-based path planning. In: *AIAA Guidance, Navigation, and Control Conference*. Reston, VA: AIAA; 2015:0846.
21. Cho S, Shim DH, Kim J. Gaussian process-based visual servoing framework for an aerial parallel manipulator. Dallas, TX presented at AIAA SciTech: Information Systems, Dallas, TX, 2017.
22. Thomas J, Loianno G, Sreenath K, Kumar V. Toward image based visual servoing for aerial grasping and perching. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014:2113–2118.
23. Kazemi M, Gupta K, Mehrandezh M. Path-planning for visual servoing: A review and issues. In: Chesi G, Hashimoto K, eds. *Visual Servoing via Advanced Numerical Methods*. London: Springer London; 2010:189–207.
24. Dame A. *A unified direct approach for visual servoing and visual tracking using mutual information*. PhD Thesis, Université Rennes 1.
25. Lee H, Jung S, Shim DH. Vision-based UAV landing on the moving vehicle. In: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2016:1–7.
26. Shim D, Chung H, Kim HJ, Sastry S. Autonomous exploration in unknown urban environments for unmanned aerial vehicles. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit* Reston, VA: AIAA; 2005:6478.
27. Barry AJ, Tedrake R. Pushbroom stereo for high-speed navigation in cluttered environments. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015:3046–3052.
28. Liu S, Watterson M, Tang S, Kumar V. High speed navigation for quadrotors with limited onboard sensing. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016:1484–1491.
29. Weiss S, Scaramuzza D, Siegwart R. Monocular-slam–based navigation for autonomous micro helicopters in GPS-denied environments. *J Field Robot*. 2011;28(6):854–874.
30. Faessler M, Fontana F, Forster C, Mueggler E, Pizzoli M, Scaramuzza D. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. *J Field Robot*. 2016;33(4):431–450.
31. Mueggler E, Rebecq H, Gallego G, Delbruck T, Scaramuzza D. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. arXiv preprint arXiv:1610.08336, 2016.
32. Kohlbrecher S, Meyer J, Stryk O. von Klingauf U. A flexible and scalable slam system with full 3d motion estimation. In: *Paper presented at Proceedings of IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011.
33. Zhang J, Singh S. Loam: LiDAR odometry and mapping in real-time. In: *Robotics: Science and Systems* (Vol. 2). 2014.
34. Bloesch M, Omari S, Hutter M, Siegwart R. (2015). Robust visual inertial odometry using a direct ekf-based approach. In: *2015 IEEE/RSJ International Conference on* Intelligent Robots and Systems (IROS). 2015:298–304.
35. Forster C, Carlone L, Dellaert F, Scaramuzza D. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. 2015.
36. Forster C, Zhang Z, Gassner M, Werlberger M, Scaramuzza D. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans Robot*. 2016.

37. Forster C, Carlone L, Dellaert F, Scaramuzza D. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Trans Robot*. 2016.
38. Marantos P, Koveos Y, Kyriakopoulos KJ. UAV state estimation using adaptive complementary filters. *IEEE Trans Control Syst Technol*. 2016;24(4):1214–1226.
39. Honegger D, Meier L, Tanskanen P, Pollefeys M. An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013:1736–1741.
40. Meier L. PX4Flow smart camera. http://ris.skku.edu/home/iros2016racing.html. Accessed January 31, 2017.
41. Hirschmuller H. Stereo processing by semiglobal matching and mutual information. *IEEE Trans Pattern Anal Mach Intell*. 2008;30(2):328–341.
42. Banz C, Hesselbarth S, Flatt H, Blume H, Pirsch P. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. In: *2010 International Conference on Embedded Computer Systems (SAMOS)*. 2010:93–101.
43. Luenberger DG. Observing the state of a linear system. *IEEE Trans Military Electron*. 1964;8(2):74–80.
44. Higgins WT. A comparision of complementary and kalman filtering. *IEEE Trans Aerosp Electron Syst*. 2007;11:321–325
45. Vasconcelos J, Calvário J, Oliveira P, Silvestre C. GPS aided IMU for unmanned air vehicles 1. 2004.
46. Loianno G, Brunner C, McGrath G, Kumar V. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU. *IEEE Robot Autom Lett*. 2017;2(2):404–411.
47. Falanga D, Mueggler E, Faessler M, Scaramuzza D. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing. arXiv preprint arXiv:1612.00291, 2016.
48. Michael B, Michael B, Helen O, Juan N, Roland S. Robust state estimation and replanning for fast uav navigation. Paper presented at Conference Workshop, 2016. https://www.dropbox.com/s/2ey1lnwsugpu4qv/iros2016_workshop_burri_compressed.pdf?dl=0

SUPPORTING INFORMATION

Additional Supporting Information may be found online in the supporting information tab for this article.

How to cite this article: Jung S, Cho S, Lee D, Lee H, Shim DH. A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge. *J Field Robotics*. 2018;35:146–166. https://doi.org/10.1002/rob.21743