Special Issue: The Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2020

Systems Article

Team JSK at MBZIRC 2020: Interception of fast flying target using multilinked aerial robot

Moju Zhao, Fan Shi, Tomoki Anzai, Takuzumi Nishio, Toshiya Maki, Keita Ito, Naoki Kuromiya, Kei Okada and Masayuki Inaba

Department of Mechano-Infomatics, The University of Tokyo, Tokyo, Japan

Abstract: The JSK (Jouhou System Kougaku) team from The University of Tokyo was selected as one of the 32 finalist teams out of 134 applicants to participate in the second edition of the Mohamed Bin Zayed International Robotic Challenge (MBZIRC) held in 2020. One of the challenges (Challenge 1) required an aerial robot to detect, follow and catch a yellow ball hanging underneath a fast-moving drone. In this challenge, we developed an articulated aerial robot that could intercept the target. This study presents reliable vision-based detection, target-trajectory estimation, fast interception-planning, and -control. Furthermore, we discuss outdoor experiments for evaluating our hardware and software systems, as well as the competition results in Abu Dhabi. With our developed system, we achieved third place in Challenge 1, as one of only four teams in the entire competition to successfully intercept the moving target.

Keywords: MBZIRC2020, aerial robotics, motion planning, visual servoing

1. Introduction

Operating aerial robot systems without human interaction in outdoor environments is demanding. Particularly, these systems should enable autonomous flight and must be robust against failures, crashes, and environmental changes (e.g., varying lighting conditions). Robotic competitions such as the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) facilitate the development of robot platforms that can quickly adapt to new situations and work robustly in the field.

In comparison to the first edition of MBZIRC, held in 2017 (Dias et al., 2019), the second edition took place on February 23rd to 25th, 2020, in Abu Dhabi and presented outdoor tasks of advanced difficulty: three individual challenges and a Grand Challenge that integrated all three. Although the individual challenges at MBZIRC were still moderately complex, their combination, time constraints, and fully-autonomous system requirements posed high demands on the participating teams. As

Moju Zhao and Fan Shi have contributed equally to this work.

Received: 29 September 2020; revised: 11 February 2021; accepted: 19 March 2021; published: 01 November 2021.

Correspondence: Moju Zhao, Department of Mechano-Infomatics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan, Email: chou@jsk.t.u-tokyo.ac.jp

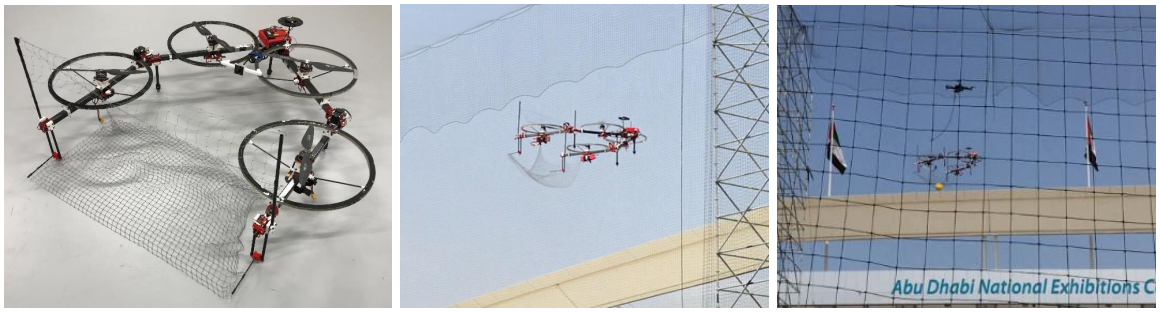

Figure 1. (A) Deformable multilinked aerial robot equipped with nets for intercepting a flying target. (B) Autonomous flight in the outdoor environment. (C) Autonomous interception with a yellow ball suspended from a fast-moving quadrotor in MBZIRC 2020 Challenge 1.

one of the 32 finalists, we extended a versatile multilinked aerial robot presented in M. Zhao et al. (2021) as the basic robot platform to participate in all challenges. Our new system demonstrated advanced capabilities in aerial grasping and manipulation, necessary for all challenges. Furthermore, Challenge 1 required fast maneuvering to catch a flying target, which ability aligns with current work to address drone safety issues and autonomous aerial interception problems. In this article, we describe our designs, in both hardware and software to achieve the autonomous interception in Challenge 1. As shown in Figure 1, our system includes customized hardware, robust, vision-based target detection and target-trajectory estimation, and incremental intercept planning and control.

1.1. Related work

1.1.1. Autonomous aerial interception for drone safety

Challenge 1 requires fast maneuvering for catching a flying target, which is motivated by drone safety. Intercepting and catching a flying target are gaining increased attention with the growing concern on the malicious use of drones, especially in complex scenarios such as urban environments where it is necessary to neutralize the drone without exposing the people around to danger (Kang et al., 2020). Regarding the detection of intruder drones, a stereo-camera-based detection and localization method was proposed by Vrba et al. (2019), and a net cannon was deployed to catch the target. A cage structure was proposed by García et al. (2020) to actively capture flying targets based on non-cooperative aerial physical interaction.

1.1.2. Comparison with Challenge 1 in the first edition of MBZIRC

As shown in Figure 2, the interception target in Challenge 1 is a small yellow ball, 12.5 cm diameter and magnetically attached to a suspending cable from a back quadrotor. Observing from the top, the quadrotor follows an eight-shape trajectory inside an arena with a size of 100 m × 60 m; however, the height continuously varies from 5 m to 15 m, and the horizontal orientation of this orbital trajectory also varies. During the 15 min challenge, the speed of the quadrotor is maintained at 8 m/s for the first 8 min and then slows down to 5 m/s for the rest of the time. Besides, Challenge 1 also contains a sub-task which requires aerial robots to pierce six static balloons with a height of 2 m from the ground. The interception and balloon piercing tasks can be operated individually. However, the main interception task is considerably difficult to complete because it is the only task requiring fast maneuvering in all challenges. Only three teams, including our team, succeeded to autonomously drop or catch the yellow ball[1] in Challenge 1, which deserved higher scores than the teams that only completed the balloon piercing task.

[1] https://www.mbzirc.com/winning-teams/2020

Figure 2. (A) Competition arena of Challenge 1, of which the size is 100 m × 40 m × 20 m. (B) Quadrotor suspending a magnetically attached yellow ball (target). (C) Eight-shaped path of quadrotor looking from the top. The height varies from 5 m to 15 m, and the speed is constant.

This challenge can be considered as an advanced task compared with Challenge 1 in the last MBZIRC, which only required an aerial robot to land on a moving truck following an eight-shape trajectory on the ground. Comparing with the vision-based detection based on the line or circle extraction from the simple landing heliport in the last MBZIRC (Bähnemann et al., 2019; Beul et al., 2019; Jin et al., 2019; Z. Li et al., 2019; Štěpán et al., 2019; Tzoumanikas et al., 2019), a more robust vision processing is necessary, because it is difficult to distinguish the ball-quadrotor target as shown in Figure 2b from the background as shown in Figure 2a. Chasing a moving target in a 3D manner is much more difficult than following a target moving on the ground. Besides, the varying lighting conditions influence the image captured from the front-view camera. Thus, a simple tracking motion is not proper, and smart interception planning is required. Furthermore, a proper hardware mechanism to block or catch the ball is also important. Otherwise, rotating propellers would be tangled in the detached ball or suspending cable, inducing a crash.

1.1.3. Multilinked aerial robot

To block or capture the target ball, most of the competition teams designed cage-like attachments in front of the robot to catch the ball. However, a foldable cage structure can further improve the catching performance. Instead of employing additional actuators for a cage, we focus on the deformability of the robot to enable grasping. In terms of deformability for grasping or the gripper, various mechanical designs are proposed (Falanga et al., 2019; Gabrich et al., 2018; N. Zhao et al., 2018). However, in order to create a synergy among all challenges in MBZIRC, as well as among previous projects in our group, a multilinked aerial robot with tilting propellers as presented in M. Zhao et al. (2021) was employed, which can further enhance the controllability compared to the original model presented in M. Zhao et al. (2018). Given that the two-dimensional multilink structure of this robot facilitates grasping from the side, a special foldable cage composed of nets tied between two end links as shown Figure 1a is developed in this work. A significant advantage of the multilinked structure in competitions is that all links are highly modularized, allowing an easy and fast replacement of certain links when the robot suffers damage.

1.1.4. Flying target detection and tracking

Flying target detection and tracking by onboard sensors can be categorized into two groups: depth-based method and vision-based method. The depth-based method like Vrba et al. (2019) uses a depth sensor such as a ToF (time-of-flight) sensor or stereo camera to obtain a depth map, and extracts an isolated depth cluster from the map for target detection. The main advantages of this method are the direct measurement of the target distance using depth map, and the long range of distance estimation (∼100 m). The vision-based method uses a monocular camera, which is lightweight and easy to equip compared with most of the depth sensors, and then extracts color, geometry or deep features from

images to classify the target. The ball suspended by the flying drone in Challenge 1 consists strong features (e.g., circle shape and single color), making it well suited for the vision-based method. Thus, we choose vision-based target detection and tracking for our aerial interception system.

Autonomous vision-based target detection and tracking can be divided into two stages. The first stage involves a detector to find the target and produce a bounding box. In the second stage, an individual tracker uses the initial bounding box from the detector as the template to find the target in the following frames. In such a two-stage framework, the accuracy of the detector and speed of the tracker are critical. Most state-of-the-art detectors, such as Faster-RCNN (Ren et al., 2017), RetinaNet (T.-Y. Lin et al., 2017), and CenterNet (Zhou et al., 2019), take advantage of deep neural networks (DNN) to achieve high detection accuracy for multiple object classes. However, these detectors highly demand the computational resources of GPUs. The tracker can be further divided into two types. The first type is based on the correlation filter (CF) (Bolme et al., 2010; Chaudhary et al., 2017; Henriques et al., 2015) which can perform online learning for the target classification. Most CFs such as KCF (Henriques et al., 2015) are less computationally demanding and can even perform in real-time in a single CPU. However, most of them lack the ability to estimate the shape of the bounding box (i.e., bounding box regression), which is important for tracking a deformable target such as the drone with a suspended ball in this challenge. The second type of tracker called Siamese-network-based tracker (Bertinetto et al., 2016) has been developed rapidly recently. The key idea of this type of tracker is the cross-correlation operation between the extracted features from the target template and search region using an offline trained shared convolutional neural network (CNN). SiamRPN (B. Li et al., 2018) and SiamRPN++ (B. Li et al., 2019) further use the output of cross-correlation operation to predict the shape of the bounding box as well as target classification, which can outperform the first type of tracker. However, there is also the computational resource problem to perform CNN as stated in the detector.

Given that we only need to detect one object class (i.e., ball-drone target) in this task, it is possible to use a lightweight detection model that is less accurate but much faster than the above state-of-the-art methods, and design a single-stage detection and tracking framework only requiring this lightweight detector. Regarding the lightweight detection model, MAVNet (Nguyen et al., 2019) achieved a promising detecting speed for aerial tasks with a compact computer with embedded GPU such as NVIDIA Jetson Xavier. The YOLO series (Redmon et al., 2016; Redmon & Farhadi, 2017, 2018) as one of the pioneers of a one-stage detector can run in a single CPU in real-time. As another early one-stage detector, SSD (W. Liu et al., 2016) has a simple architecture that uses multi-scale feature maps for classification and bounding box regression. Recently, M. Liu et al. (2019) and Zhu and Liu (2018) integrate SSD with LSTM (Hochreiter & Schmidhuber, 1997) to jointly perform detection and tracking for multiple object classes in videos. We choose SSD for our task and further apply the quantization process (Jacob et al., 2018) to produce a significantly small model (< 5 MB). The quantized inference model can be deployed in a USB inference device performing real-time detection independently with less than 5 ms per frame. We also develop a pipeline from bounding box detection to task-tailoring ball tracking for continuous frames.

1.1.5. Target trajectory estimation

Target trajectory estimation is important for interception planning and control. In the last MBZIRC, either an extended Kalman filter (Tzoumanikas et al., 2019) or a particle filter (Bähnemann et al., 2019) was designed to estimate the eight-shape path of the target truck and heliport. The 2D motion on the ground is much easier to estimate because the bird-view observation from a downward camera can provide sufficient and reliable information on the target position; the height of the target is constant. Unlike a stereo camera or a ToF sensor that can measure the distance to the target, the estimation of a 3D trajectory using the vision-based method demands a proper depth estimation. Thus, in this work, a robust depth estimation with a monocular camera based on both the ball radius and width of the drone is developed to achieve the maximum depth of up to 30 m. Furthermore, given that the sunlight reduces the visibility on a target, it is difficult to estimate an entire 3D trajectory

in the air. Thus, we only focus on the straight segment parts of the eight-shape trajectory and apply a RANSAC (Fischler & Bolles, 1981) based filter to formulate the target motion.

1.1.6. Interception planning and control

Intercepting a moving target in the air is much difficult than following, because the precise tracking performance is required at an extremely close range. Although various landing planning and control methods (Bähnemann et al., 2019; Beul et al., 2019) have demonstrated the effectiveness in the last MBZIRC, the landing task is still easier than capturing a small object in the air, because the heliport for landing is larger than the aerial robot implying that more error tolerance is allowed at the moment of landing (i.e., interception). Furthermore, if the entire trajectory of the target is unable to be comprehended due to factors such as sunlight as discussed above, interception would be more demanding because of the delay in feed-back control. Thus, a special incremental interception planning is developed in our work, which attempts to intercept the target in the straight segment of the eight-shape trajectory with several iterations. Although it is difficult to intercept the target at the first time, the standby position of the robot is updated after every trial and finally converges to a proper interception point. To maximize the sighting duration at each interception trial, a position close to the end of the straight segment is selected as the initial standby point. A critical assumption of this strategy is that the moving target should follow an orbital motion. However, most of the flying targets in a finite space follow relatively orbital trajectories, indicating that our strategy is effective for the interception task in a finite space.

1.1.7. Comparison with other teams

In the competition, only four teams successfully intercepted the yellow ball: our team, an allied team from Czech Technical University in Prague, University of Pennsylvania and NYU (Baca et al., 2021), and another allied team from Polytechnic University of Madrfourid, University Pablo Olvide, Poznan University of Technology and CNRS (in Grand Challenge), succeeded to block and drop the ball from the quadrotor, whereas the team from the Beijing Institute of Technology successfully captured the ball and delivered it to a goal point. The allied teams employed a quadrotor attached with an unfoldable net which is specialized in blocking. Their interception strategy is similar to ours which lets the robot wait at the cross point of the eight-shape trajectory. The team from the Beijing Institute of Technology employed a quadrotor equipped with an unfoldable cage and a gimbal camera that can actively visually track the target. Their interception strategy is also different from other teams, which is active chasing. Nevertheless, their strategy also focuses on the straight segment part: the robot waits around the end of the segment at the beginning, and starts chasing after the target enters the segment part. The main limitation of the active chasing strategy is that it is necessary to catch the ball before the target enters the arc part.

1.2. Main contributions

To the best of our knowledge, ours is the first study to employ a deformable multilinked aerial robot to achieve fast maneuvering in outdoor environments. In short, the main contributions of this study, which can benefit the field robotics community are summarized as follows:

- we develop a multilinked aerial robot with a foldable net for interception and capture and a hardware design allowing fast processing on the deep neural network.
- we develop a computationally light framework for detecting and tracking the target with single-shot detection, color filtering, and robust depth estimation.
- we develop a robust, random-sampling technique for estimating the target's linear trajectory, and a motion planner that incrementally updates the interception point.
- we evaluate the feasibility of our system by field experiments and demonstrate good results in challenge competition.

1.3. Organization

The remainder of this paper is organized as follows: the multilinked aerial robot platform customized for Challenge 1 is presented in Section 2. Then the vision-based target detection and tracking are presented in Section 3, followed by the trajectory estimation and interception planning and control method in Section 4. Finally, we demonstrate a representative on-site experiment as well as the results in the real competition in Section 5 before concluding in Section 6.

2. Platform

We develop a robot platform to achieve the synergy between all challenges by originality, namely aerial deformability. A versatile multilinked aerial robot is then designed as presented in M. Zhao et al. (2021). To complete Challenge 1 autonomously, we adopt customization to enable the interception and capturing of a flying target.

2.1. Hardware

2.1.1. Multilinked structure

The basic architecture of our robot is a three-link model as shown in Figure 3. The propulsion system is built based on T-motor products (rotors: T-motor MN3510 KV360[2]; ESC: T-motor Air40A[3]; propeller: T-motor P14×4.8 Prop[4]), and the maximum thrust generated by this propulsion system is 16 N. As shown in Figure 3b, the propeller is tilted at an angle of 10 deg to enhance the controllability on yaw rotational motion. There are two joints connecting links with a length of 0.6 m. As shown in Figure 3c, these joints are actuated by servo motors (Dynamixel XH430-W350R)[5] and rotated in parallel, resulting in two-dimensional deformation like a gripper.

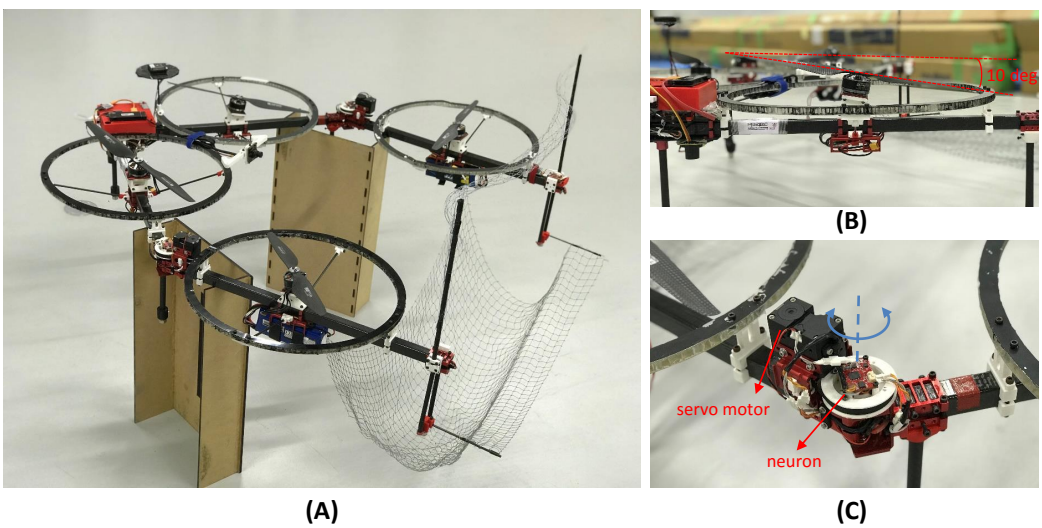

Figure 3. (A) Multilinked aerial robot with two joints. Link length is 0.6 m. (B) Tilting propeller (10 deg) in each link. (C) Servo motor (Dynamixel XH430-W350R) actuating the joint. The Neuron is an original MCU (micro controller unit) which enables the communication between the servos and onboard computer as shown in Figure 6

[2] https://store-en.tmotor.com/goods.php?id=337
[3] https://store-en.tmotor.com/goods.php?id=368
[4] https://store-en.tmotor.com/goods.php?id=380
[5] http://www.robotis.us/dynamixel-xh430-w350-r

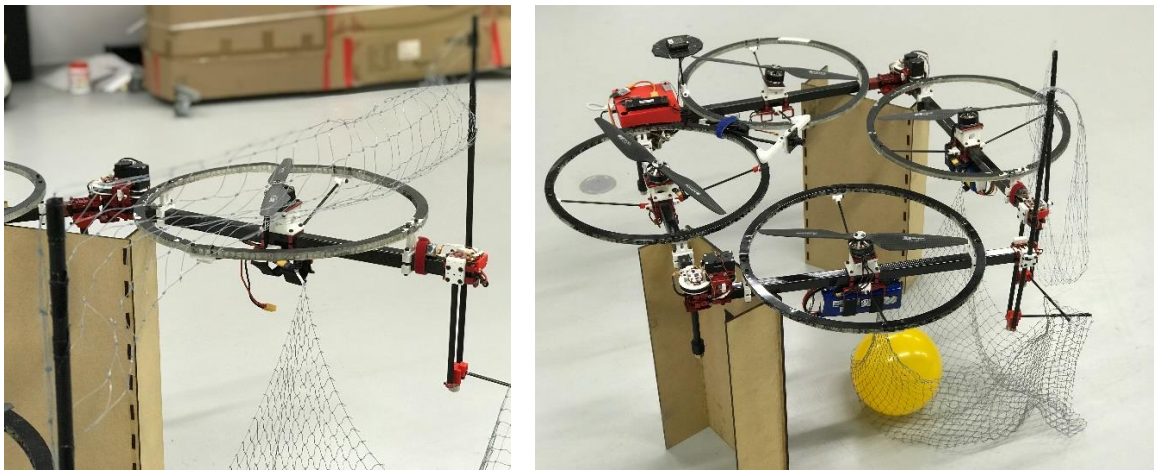

Figure 4. (A) Top net made of nylon to block an approaching ball. (B) Bottom net make of polyester to hold the target ball while joints are closed.

2.1.2. Net for interception

A super light net is tied between two poles attached on the top of the link ends as shown in Figure 4a. This net can be expanded by deformation for blocking the target ball. We opt nylon material because it can relatively keep its form while folding, thus can avoid being wrapped in the rotating propeller as shown in Figure 4b. Another net made from polyester is tied at four points for capturing the ball from below while closing the joints as shown in Figure 4b.

2.1.3. Onboard sensors

The sensors for localization include an embedded IMU (Inertial Measurement Unit) & magnetometer, a GPS (Global Positioning System) receiver[6], a downward-facing LiDAR (Light Detection and Ranging)[7], and a downward-facing light VIO (Visual Inertial Odometry) module[8] as shown in Figure 5b,c. It is notable that we do not employ RTK-GPS (Real Time Kinematic GPS) in our system. Similar to the motion capture system in the indoor environments, dependency on this localization device can not be considered as a fully autonomous system. The challenge also regulated a score penalty on the usage of the RTK-GPS device.

For vision-based detection, a front view camera (ELP-SUSB1080P01-LC1100[9]) is employed as shown in Figure 5c. The resolution of this camera is 1920×1080 and the angle of view is $60\,\text{deg} \times 40\,\text{deg}$. The mounting position is decided such that the target ball can be tracked at extremely close range and also has less occlusion caused by link ends as shown in the sketch of Figure 5c.

2.1.4. Onboard computer

Given that vision-based detection is required in this work, a relatively sufficient computation resource provided by GPU is desirable. Basically, a GPU embedded computer such as NVIDIA Jetson series[10] is an effective solution. However, the bottleneck of such a compact computer is the relatively weak CPU power, and limited memory. Thus, a second computer with a more powerful CPU is usually employed (Y. Lin et al., 2018) to handle processes heavily depending on the CPU (e.g., flight control,

[6] https://www.u-blox.com/en/product/neo-m8-series
[7] https://leddartech.com/lidar/leddarone/
[8] https://www.intelrealsense.com/tracking-camera-t265/
[9] http://www.webcamerausb.com/
[10] https://developer.nvidia.com/buy-jetson

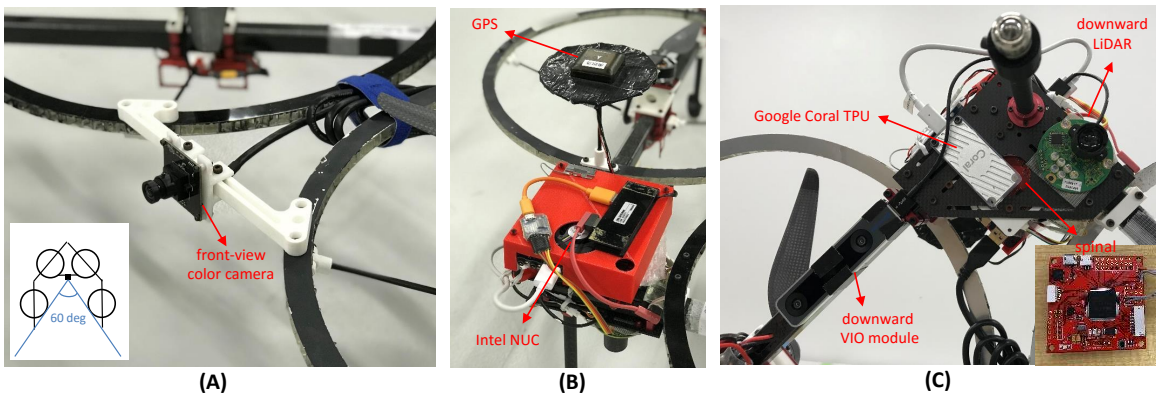

Figure 5. (A) Front view camera (ELP-SUSB1080P01-LC1100) for target detection. The resolution of this camera is 1920 × 1080 and the angle of view is 60 deg × 40 deg. (B) Top of the connection part between link2 and link3, where the onboard computer (Intel NUC with Core i7) is mounted. (C) Bottom of the connection part between link2 and link3, where Google Coral TPU for inference with the neural network is mounted.

motion planning). However, this coupled computer system not only increases the payload, but also induces difficulty on the compact design for multilinked model. Thus, we employ a USB accelerator device called Google Coral[11] which is an edge TPU (tensor processing unit) to perform inference with the neural network. Using the edge device can significantly save the resource and memory of an onboard computer. Furthermore, the weight of Google Coral is significantly light (i.e., 20 g), which is suitable for an aerial platform. The usage of the edge device is one of the advantages of our hardware decision, which allows us to use computational resources for other processes. We then employ Intel NUC BLKNUC7I7DNHE[12] instead of other lighter computers presented in M. Zhao et al. (2021) with the aim of handling heavy processes on target trajectory estimation, motion planning, and data logging.

2.1.5. Battery

Given that the sensor and computer are different from the default platform as presented in M. Zhao et al. (2021), the battery arrangement is modified. Two 6 s 3000 mA h batteries are attached in two end links, respectively as shown in Figure 3a. The weight of the robot is 4.14 kg, which is relatively heavier than the basic platform causing a short flight time of 10 min. Thus, it requires a battery replacement for a 15 min challenge. However, a parallel connection design allows the hot-plug during the challenge which does not need to restart the system.

2.2. Software

The onboard computer (i.e., Intel NUC) runs Ubuntu and the robot operating system (ROS)[13] as middleware to execute the state estimation, flight control, target detection and motion planning for autonomous flight and interception as shown in Figure 6. The original printed circuit board (PCB) called *Spinal* and *Neuron* are MCU (micro controller units) which enable the internal communication system between links. In Figure 6, the blocks corresponding to the main contribution of this work are colored in gray, whereas other white blocks are common in all challenges. The details of these white blocks, especially the flight control and state estimation can be found in M. Zhao et al. (2021).

[11] https://coral.ai/products/accelerator
[12] https://www.intel.sg/content/www/xa/en/products/boards-kits/nuc/kits/nuc7i7dnhe.html
[13] http://www.ros.org/

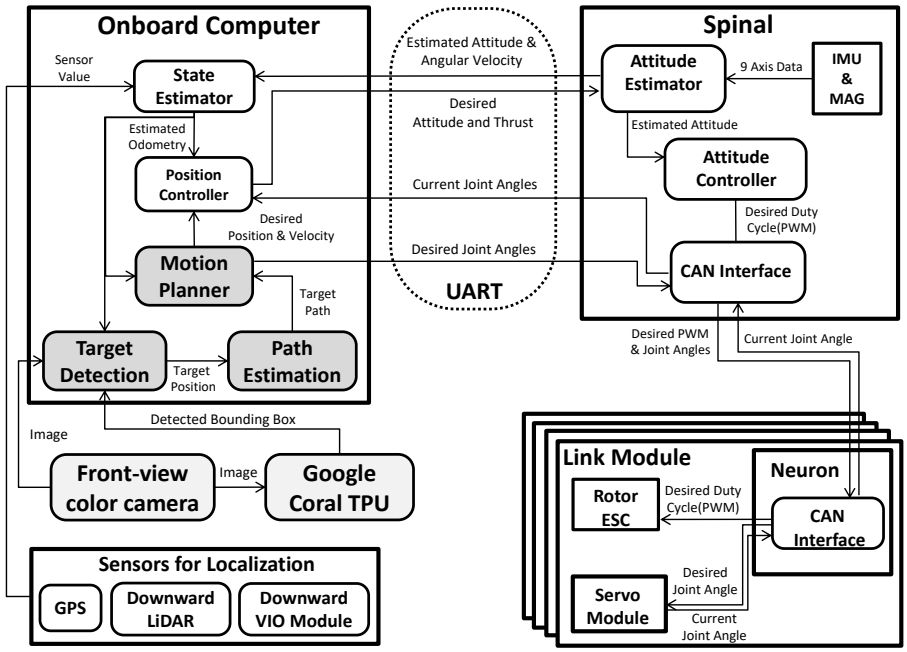

Figure 6. System architecture to achieve fully autonomous flight and interception. The gray blocks are the main contributions in this work, whereas other white blocks are common in all challenges presented in M. Zhao et al. (2021).

2.3. Flight control

Here, we briefly introduce the flight control method for the multilinked model, which is presented in M. Zhao et al. (2021). As stated in our previous work (M. Zhao et al., 2018), we assume the joint motion is sufficiently slow. In other words, we ignore the velocity and acceleration of the joint angles $\boldsymbol{q} \in \mathcal{R}^2$. The multilinked model can be regarded as a time-variant rigid body, whereas the dynamic model for flight control is approximated as a single rigid body model. However, the change in joint angles $\boldsymbol{q}$ still influences not only the entire inertial tensor I_Σ, but also the allocation matrix $Q(\boldsymbol{q})$, converting thrust force u to wrench $(\boldsymbol{f}, \boldsymbol{\tau})$. Besides, Euler angles $\boldsymbol{\alpha} \in \mathcal{R}^3$ are used to express the robot attitude.

Given that the control inputs are four thrust forces $\boldsymbol{u} \in \mathcal{R}^4$, we apply a cascaded control flow for position and attitude control as shown in Figure 7. In the position controller as the outer loop, a general PID control is applied to calculate the desired thrust $\boldsymbol{u}_{\text{pos}}^{\text{des}} \in \mathcal{R}^4$ for the z motion and the desired roll and pitch angles $\alpha_x^{\text{des}}, \alpha_y^{\text{des}}$ from the desired position $\boldsymbol{r}^{\text{des}}$ and desired velocity $\dot{\boldsymbol{r}}^{\text{des}}$. In the attitude control as the inner loop, Linear Quadratic with Integral (LQI) control approach (Young & Willems, 1972) is applied, because we use the cost function in this optimal control framework to suppress the lateral force generated by the tilted propeller. The input of the attitude controller is the desired Euler angles of which the roll and pitch elements are from the position controller, whereas the output is the desired thrust forces $\boldsymbol{u}_{\text{att}}^{\text{des}} \in \mathcal{R}^4$. Eventually, the final desired thrust force for each rotor should be the sum of the control input from both the attitude controller and position control: $\boldsymbol{u}^{\text{des}} = \boldsymbol{u}_{\text{att}}^{\text{des}} + \boldsymbol{u}_{\text{pos}}^{\text{des}}$.

3. Vision-based target detection and tracking

The flying target in Challenge 1 is a yellow ball hanging from a quadrotor following an eight-shaped path of which the height varies and the speed in constant as shown in Figure 2. The target ball is

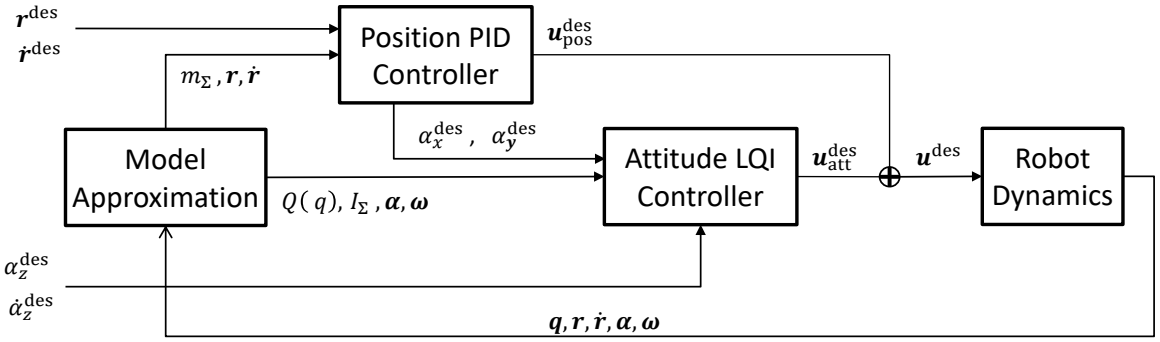

Figure 7. Flight control framework for multilinked aerial robot, which involves the dynamic model approximation and a cascaded control flow including the position and attitude controller. References of this cascaded control flow are the desired 3D position $\boldsymbol{r}^{des}$, the desired linear velocity $\dot{\boldsymbol{r}}^{des}$, the desired yaw angle α_z^{des}, and the desired yaw angular velocity $\dot{\alpha}_z^{des}$. The input includes robot position $\boldsymbol{r}$, velocity $\boldsymbol{r}$, Euler angles $\boldsymbol{\alpha}$, angular velocity $\boldsymbol{\omega}$, inertial parameters $m_\sum$, $I_\sum$, and allocation matrix $Q(\boldsymbol{q})$, whereas the output is the desired thrust $\boldsymbol{u}^{des}$.

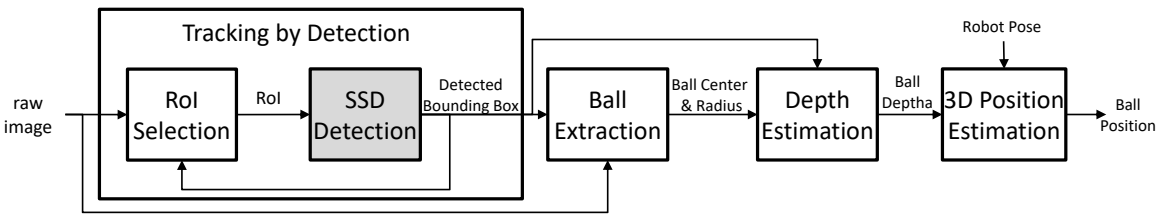

Figure 8. Pipeline involving vision-based ball detection and estimation of the 3D position. The white blocks are run on an onboard computer, whereas the gray block is run in the USB device (i.e., Google Coral TPU).

magnetically attached to the suspending cable. For such a target, we develop a vision-based detection and tracking method as well as the estimation of the 3D position as shown in Figure 8.

3.1. Tracking by detection

In this work, a computationally light framework for tracking by detection which only requires few resources on the CPU owing to an external USB device (i.e., Google Coral TPU) for detection based on a deep neural network is developed as shown in Figure 8.

3.1.1. SSD detection

We use SSD as the detector in our vision tracking system. The main advantages of SSD are the possibility to select different backbone classification networks and transfer learning using a pretrained classification network. For the backbone network, a light convolutional neural network, namely MoblieNet V2 (Sandler et al., 2018) is applied to reduce the computation consumption. Furthermore, the network is quantized (Jacob et al., 2018) to further reduce the network size and enable inference with 8 bit integers in the USB device Google Coral TPU. With the quantized SSD, inference on an image input of the size of 256×256 only spends approximately 4 ms to 5 ms.

Regarding the choice of the target in this challenge, only selecting a quadrotor can induce many false-positives because there are many similar structures with the drone frame in the background environment. Only selecting a ball is also ambiguous in a complex environment. Thus, we select both the quadrotor and suspending ball as an integrated training model. This can significantly increase the detection reliability in a complex background due to the extraction of features from both parts. As a result, the trained SSD can provide reliable detection at a long range ($\sim$30 m) as shown in

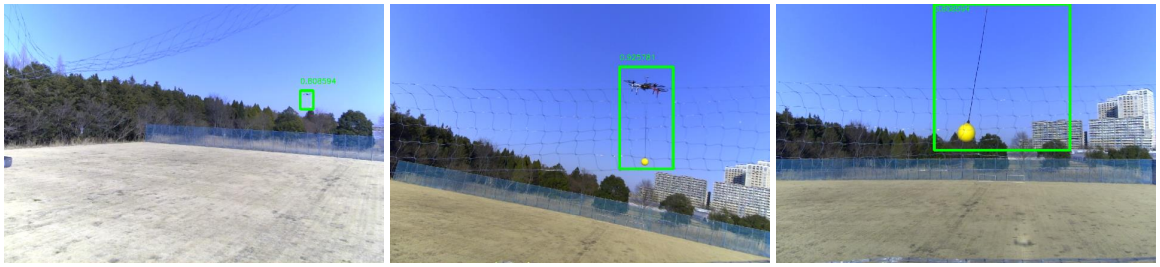

Figure 9. Detection results at (A) long range (> 15 m); (B) medium range (5 m to 15 m); (C) close range (< 5 m). The case of the only ball at extremely close range can also be detected, indicating the robustness of target detection.

Figure 9a, as well as robust detection at an extremely close range where only the ball is within an image as shown in Figure 9c. This wide range detection provides sufficient target data in time sequence, which benefits later target path formulation and early tracking motion. Details of the offline training procedure are presented in Section 5.1.

3.1.2. RoI selection

SSD, which only contains a single class (i.e., the target hanging ball and quadrotor) can serve as a tracker; however, the false-positive candidate cannot be effectively avoided, indicating that reliable visual tracking can not be achieved only by detection. Thus, a simple tracking strategy is introduced before SSD detection as shown in Figure 8 to limit the region of interest (RoI) for detection, which can improve tracking on a continuous motion on a target. There are three phases designed to achieve reliable tracking from initial detection to on-going tracking:

- **initial detection phase**: we perform the SSD detection on the entire image input as well as five half-size images to enable detection on a small target at a long range as shown in Figure 10a. All input is resized to 256×256 for SSD inference. The continuous motion of a candidate is defined as a small motion deviation between two neighboring frames (i.e., $\|\boldsymbol{p}_k - \boldsymbol{p}_{k-1}\| < \delta$, where $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k-1}$ are the pixel position of the candidate bounding box center at the current and last frames). Once the continuous motion of a detected candidate is confirmed with certain continuous frames (e.g., 10 frames), the phase is shifted to the tracking phase. If there are several candidates satisfying the continuous motion, we chose the candidate with a height detection score.
- **tracking phase**: A RoI, which is an expanded region around the last detection bounding box is selected as the input to the SSD as shown in Figure 10b, which obtains the detection on a continuous moving target. The expanding rate is chosen between 2 and 3. Detection only on the selected RoI instead of the entire image can significantly increase the detection, especially at long range because the resizing rate is low. If there is no target detected at the current frame, the RoI would not change at the next frame. This is effective for the temporary target lost, because such detection failure is due to the difficulty in distinguishing between the target and background. Furthermore, if the target cannot be detected in certain continuous frames (i.e., 5), the phase is shifted to the lost target phase.
- **lost target phase**: the same detection procedure in the initial detection phase is performed. However, the frames to confirm a continuous motion of the target is half of the frames in the initial detection for quick recovery.

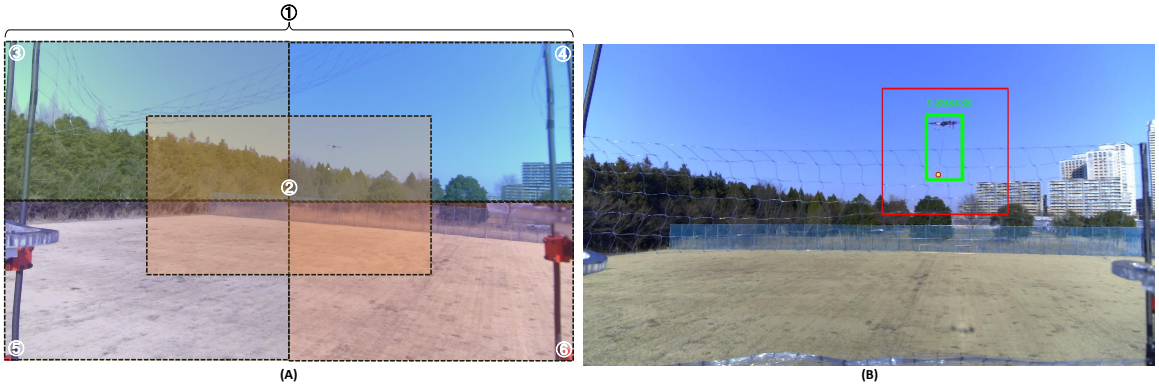

Figure 10. (A) Six input for SSD per frame in initial detection phase: the image ① and five half-size images ②–⑥; (B) Expanded region (red frame) around the last detection bounding box (green frame) is selected as the input to the SSD to obtain the detection on a continuous moving target.

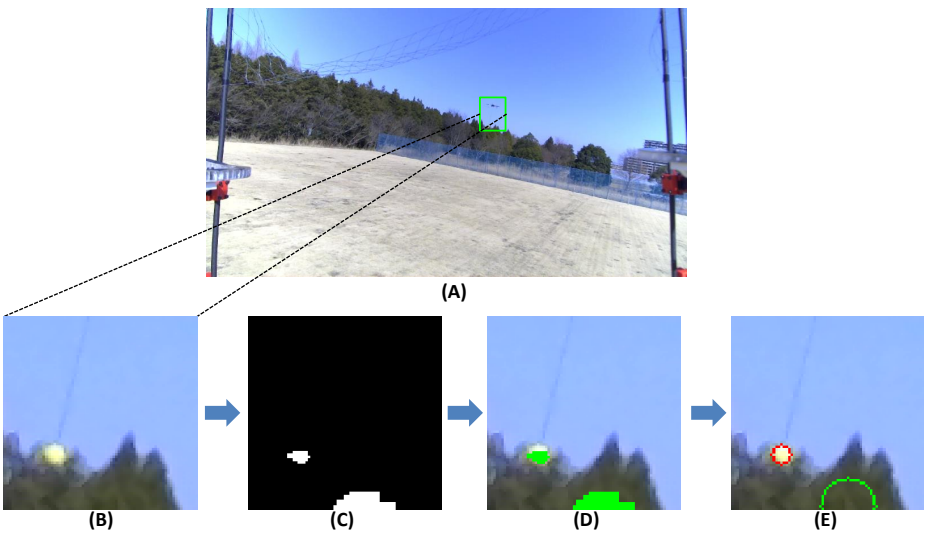

Figure 11. (A) Raw image with the detected bounding box from SSD. (B) Lower half part of the bounding box for ball extraction. (C) Result of HSV color filter. (D) Result of contour clustering. (E) Minimum enclosing circle for each contour. The circle protruding from the bounding box is considered as an invalid candidate (green circle), whereas the valid candidate with the maximum size is the final target result (red circle).

3.2. Ball extraction

The location of the ball is important to estimate the 3D position. The detected bounding box about the coupled model as shown in Figure 9 provides a reliable region to extract the ball part. We developed a simple pipeline for ball extraction focusing on the bottom half of the bounding box with a certain expanding rate (i.e, 1.1) as shown in Figure 11b. Then, a Hue-Lightness-Saturation (HLS) color filter is applied as shown in Figure 11c. The HLS color filter is more robust compared to the RGB color filter and the Hue-Saturation-Value (HSV) color filter in the outdoor environments, since the lightness value in HSL can handle the influence of sunlight. Subsequently, contour extraction is further applied on the filtered output as shown in Figure 11d to perform clustering, which is followed by the calculation of the minimum enclosing circle for each contour as shown Figure 11e. The enclosing circles protruding from the valid region are considered as outlier. Finally, the inlier

enclosing circle with the maximum size is considered as the target ball, and the center point and radius in the pixel can be obtained, which is further used to estimate the 3D position. In most of the cases, the HSL color filter and the associated contour clustering can only extract the portion of the ball as shown in Figure 11c,d; however, the calculation of the minimum enclosing circle can improve the robustness against the insufficient area as well as the outlier rejection.

3.3. Distance estimation

The depth of the ball can be straightforwardly calculated using the ball radius as follows:

$$d_{\text{ball}} = f_{\text{cam}} \frac{r_{\text{ball}}}{r_{\text{pixel}}} \tag{1}$$

where, f_{cam}, r_{ball} and r_{pixel} are the focal length of the camera, the ball radius in metric and the radius in pixel respectively. We assume that the ball metric radius is known.

Although the outlier rejection as shown Figure 11e can filter out most of the invalid candidates, the color filter in ball extraction can still induce error regarding r_{pixel} especially at long range. It is also possible to roughly estimate the depth of the quadrotor from the width of the bounding box of the SSD:

$$d_{\text{drone}} = f_{\text{cam}} \frac{w_{\text{drone}}}{w_{\text{bb}}} \tag{2}$$

where, w_{drone} and w_{bb} are the known width of the quadrotor in metric and the width of the bounding box in pixel, respectively. It is notable that when the edges of the bounding box are close to the boundary of the raw image, the quadrotor might protrude from the camera frame as shown in Figure 9c making the width of the bounding box unreliable. The definition of the valid bounding box is then introduced by:

$$\mathcal{I} := \{bb = (u_{\min}, v_{\min}, u_{\max}, v_{\max})|, u_{\min} > \epsilon, v_{\min} > \epsilon, u_{\max} < w_{\text{img}} - \epsilon, v_{\max} < h_{\text{img}} - \epsilon\} \tag{3}$$

where, $u_{\min}$, $v_{\min}$, $u_{\max}$ and $v_{\max}$ are four corners of the bounding box, w_{img} and h_{img} are the width and height of the raw image, and ϵ is a small positive threshold.

In the case of the long range, given that the ball is relatively small, there are no sufficient pixels to represent the ball (i.e., 1 to 2 pixels). Therefore, the depth of the quadrotor calculated from Equation 2 is more reliable. In the case of the close range, the depth calculated from Equation 2 is more reliable since the quadrotor might protrude from the image.

Then the estimation of the ball depth and center, which combines the information of the bounding box and ball extraction is developed as shown in Figure 12. This estimation flowchart has three main cases according to two depth threshold: d_{middle} for medium range and d_{close} for close range.

Furthermore, a simple low pass filter is introduced to merge the depth calculated from both the ball radius and quadrotor width as follows:

$$\bar{d}_{\text{ball}}(t) := (1-\gamma)\bar{d}_{\text{ball}} + \gamma d_{\text{ball}} \tag{4}$$

$$\bar{d}_{\text{ball}}(t) := (1-\gamma)\bar{d}_{\text{ball}} + \gamma d_{\text{drone}} \tag{5}$$

where, $\bar{d}_{\text{ball}}$ is the filtered depth of the ball, and γ is a filtering rate. It is decided whether $\bar{d}_{\text{ball}}$ is updated by either Equation 4 or Equation 5 according to the current ball depth $\bar{d}_{\text{ball}}$ as shown in Figure 12.

3.4. 3D position estimation

The estimation of the ball center $(u_{\text{ball}}, v_{\text{ball}})$ is updated according to the ball depth. In the case of close range, the estimated center is equal to the result from the ball extraction $(u_{\text{cricle}}, v_{\text{cricle}})$ as

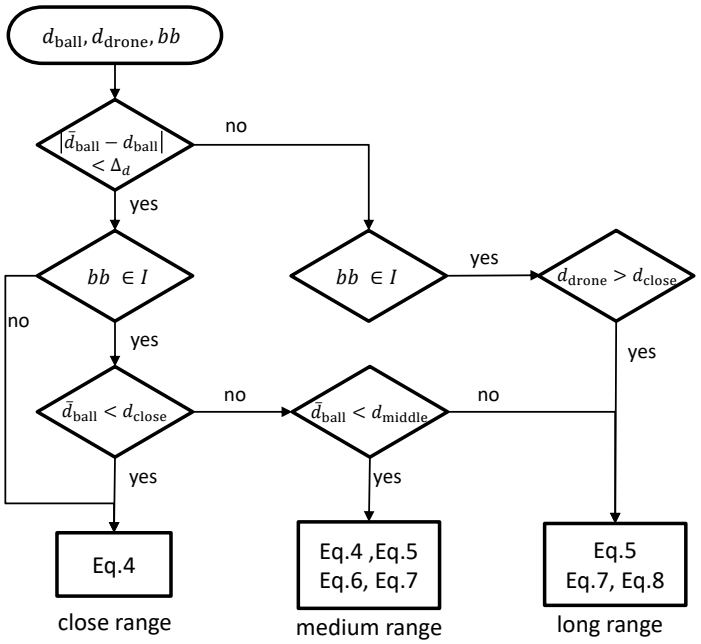

Figure 12. Flowchart to estimate the ball depth and center, which combines the depth calculated from the ball radius and depth calculated from the width of the bounding box.

shown in Figure 11e, whereas in the case of medium distance, both the ball position and bounding box are used as follows:

$$u_{\text{ball}} = \frac{\frac{bb(0)+bb(2)}{2} + u_{\text{cricle}}}{2} \tag{6}$$

$$v_{\text{ball}} = \frac{bb(3) + v_{\text{cricle}}}{2} \tag{7}$$

where, bb is the bounding box array as defined in Equation 3. It is notable that, we assume the bottom edge of the bounding box contacts with the ball circle. Thus, the height of the ball center can be approximated as the height of the bottom edge.

In the case of long range, only the bounding box is used as follows:

$$u_{\text{ball}} = \frac{bb(0) + bb(2)}{2} \tag{8}$$

$$v_{\text{ball}} = bb(3) \tag{9}$$

In addition, outlier rejection is introduced to improve the validity of the ball depth (i.e., $|\bar{d}_{\text{ball}} - d_{\text{ball}}| < \delta_d$) and bounding box (i.e., $bb \in \mathcal{I}$) as shown in Figure 12.

Using Equations 4–9, the 3D position of the target ball with reference to (w.r.t.) the camera frame can be calculated based on the camera intrinsic parameters:

$${}^{\{\text{Cam}\}}\boldsymbol{p}_{\text{ball}} = \bar{d}_{\text{ball}} \begin{bmatrix} f_{\text{cam}} & 0 & c_u \\ 0 & f_{\text{cam}} & c_c \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_{\text{ball}} \\ v_{\text{ball}} \\ 1 \end{bmatrix} \tag{10}$$

where, $[c_u\ c_v]$ is the point in the camera frame where the optical axis passes. Note that, the z axis of the camera frame {Cam} coincides with the optical axis.

Finally, the ball position w.r.t the world frame ${}^{\{W\}}\boldsymbol{p}_{\text{ball}}$ can be transformed using the robot pose (i.e., ${}^{\{W\}}R_{\text{robot}}$, ${}^{\{W\}}\boldsymbol{p}_{\text{robot}}$).

4. Interception planning and control

There are three main difficulties in planning and control for the interception task in Challenge 1.

- First, the flying arena in Challenge 1 is large with more than [100 m, 60 m]. Since the target drone is small with a 6.25 cm radius and shaking ball, the estimated 3D position has a large uncertainty when being distant.
- Second, the flying speed of the target is very fast (two mode: 8 m/s and 5 m/s) and challenging for a normal multirotor to intercept.
- Third, the background view of the competition arena is very crowded with many dark-colored buildings. Comparing to the open court, the detection quality is lower, and the maximum detection distance is shorter, which causes very little reaction time (≤ 4 s).

To tackle these difficulties, the approach should be robust and flexible to cope with the large uncertainties in the task level. Consequently, we develop an incremental interception approach to iteratively adjust to a final grasping point with heuristic uncertainty-aware target trajectory estimation and robot motion control.

4.1. Target trajectory estimation

Based on the rules in Challenge 1, the target follows an eight-shape route with two different constant speeds for the first 8 min and the next 7 min, respectively. The motion in the z axis is random with unknown speed and height (the height region is roughly given in competition: 5 m to 15 m). The horizontal orientation can also be random.

In summary, the target trajectory has many random settings with limited prior knowledge to fully reconstruct the global route map, especially for the robot with a limited visual view and perceptive region. Instead of fully reconstructing the global route, we estimate a very local route of target motion with sufficient prior knowledge.

In detail, we take advantage of the local route near the center of the eight-shape map, which is nearly straight in the x-y plane with constant speed and random motion in the z axis. The target trajectory estimation is naturally divided into two components in our approach, which include the straight line estimation in the horizontal plane using random sample consensus (RANSAC) and point estimation in the vertical axis using a low-pass filter (LPF).

4.1.1. Horizontal trajectory estimation

The input of trajectory estimation is from target detection results, which are further recovered into the 3D position based on the robot localization results and the target depth as presented in Figure 8. Therefore, the input data is highly noisy.

A robust and computationally efficient approach is required against these uncertainties. We apply the 2D RANSAC approach for line trajectory estimation as Figure 13, which is robust against outliers comparing to classical line fitting methods such as least square fitting and Deming regression (Adcock, 1878), and efficient comparing to robust regression approaches such as H. Li (2009) and Olsson et al. (2008).

We divide the entire estimation procedure into three phases as shown in Figure 14, which are the initial, tracking, and final phases, according to the distance between the target and robot. In each phase, the number of data points would be different. The reason is:

- in the initial phase, the target is distant from the robot and the estimated 3D position has significant uncertainties. The latest information with more accuracy is preferred, so that the old data is abandoned quickly to avoid generating misleading results.
- in the tracking phase, the detection results from Figure 8 are more reliable, so that more history data is preferred to lead to a more robust estimation.

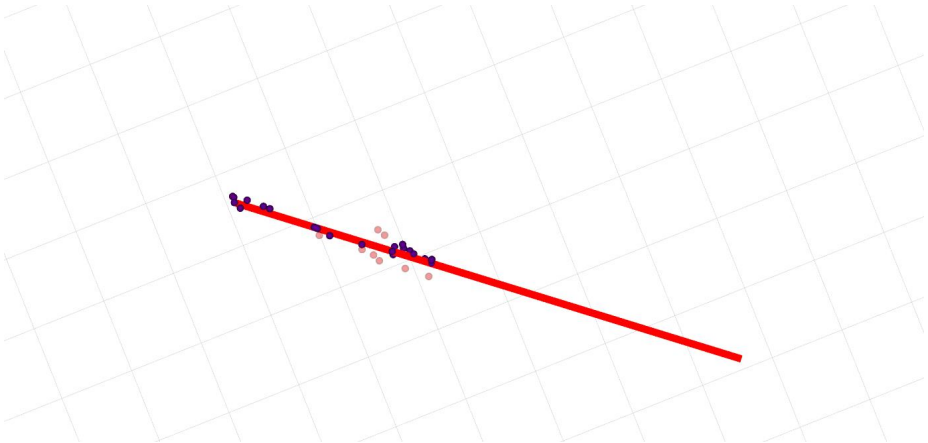

Figure 13. Result of line estimation by RANSAC. Both blue and pink points are reconstructed 3D position of the target from Figure 8, where the blue ones are classified as inliers, and the pink ones are outliers. The red line is the estimated straight line.

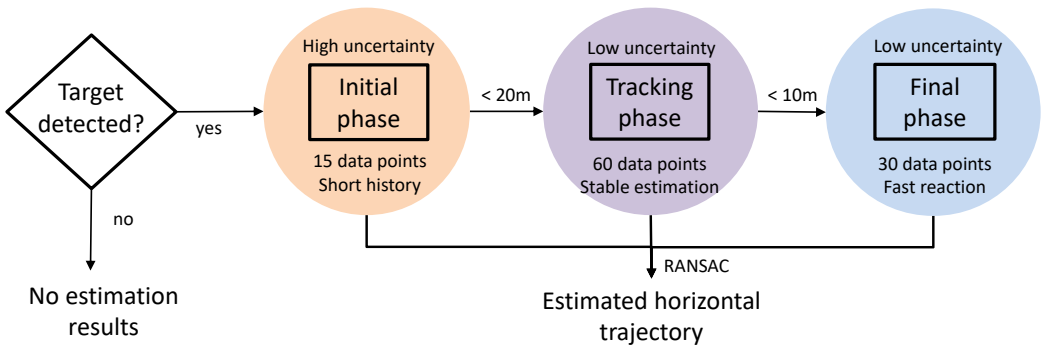

Figure 14. Pipeline of trajectory estimation in the horizontal plane. Three phases are designed according to the distance between the target and robot. The input is a target 3D position obtained from Figure 8.

- in the final phase, the target is close and the number of history data is reduced to half to encourage rapid response to the latest target motion while guaranteeing robust estimation results.

In detail, the object detection pipeline as shown in Figure 8 generates results as input at 30 Hz in general. Thus, we use 15 data (i.e., 0.5 s) for the initial phase, 60 data (i.e., 2 s) for the tracking phase, and 30 data (i.e., 1 s) for the final phase, respectively.

4.1.2. Vertical trajectory estimation

The vertical trajectory is difficult to predict because the maximum and minimum height values vary in the real competition and the speed is also unknown. In the real competition, we found the gap between the maximum and minimum height could at most vary nearly two times in different trials.

The only information we could obtain is that the target would not suddenly change its vertical speed. To solve this problem, we develop a straightforward low-pass filter (LPF) to refine the latest height of the target:

$$z_{\mathrm{lpf}} := z_{\mathrm{lpf}}\alpha_{\mathrm{lpf}} + (1-\alpha_{\mathrm{lpf}})z_{\mathrm{detection}} \tag{11}$$

where we balance between the smoothness and time delay by adjusting the LPF gain α_{lpf}. In the real competition, it is set as 0.8.

4.2. Interception position planning

4.2.1. Target moving state classification

Our strategy is to block the moving target in the straight segment part. However, given the eight-shape route, there are two straight segment parts, in which the target moves away from the robot or gets close. Thus, we need to classify the two states for different reactive motions. From the trajectory estimation result, our heuristic method is to make the classification based on the average of five latest and five oldest estimation positions in the estimator buffer of the target.

The target motion (i.e., getting close or moving away) can be classified according to the angle between the robot orientation and target trajectory:

$$\begin{cases} (\overline{P}_{\text{head}} - \overline{P}_{\text{tail}}) \cdot \vec{n}_{\text{robot}} < v_{\text{thre}}, & \text{target getting close} \\ (\overline{P}_{\text{head}} - \overline{P}_{\text{tail}}) \cdot \vec{n}_{\text{robot}} > v_{\text{thre}}, & \text{target moving away} \end{cases} \tag{12}$$

where $\overline{P}_{\text{head}}$ and $\overline{P}_{\text{tail}}$ are the average points of the five latest and oldest detection results, respectively, $\vec{n}_{\text{robot}}$ is the robot unit orientation vector, v_{thre} is the threshold value from experiments, and all vectors are in the horizontal 2D plane. As shown in Figure 15, the dot product is expected to be negative when the target gets close to the robot or otherwise positive when the target leaves away. In the real experiments, although there are noises from vision-based detection results, we found a small negative value could be good enough for the classification threshold v_{thre}.

It is necessary to indicate that there are corner cases when a target is in the arc part of the eight-shape route instead of the straight segment part. However, the corner cases seldom appear in the real challenge because in the arc part, the target is too distant (∼30 m) to be detected by the robot. However, to be complete, the presented method should be constrained by the target distance based on the task details.

4.2.2. Moving-away tracking scenario

When the target motion is classified as moving away from the robot, there is not much useful information on the interception point in the horizontal plane. In the vertical plane, the robot is commanded to track the height of the target to keep the target inside the visual view.

4.2.3. Getting-close tracking scenario

When the target motion is classified as getting close to the robot, the basic idea is to find the desired interception position and send the feasible control commands to the robot.

To avoid the unfeasible interception position, first, the quality of trajectory estimation is evaluated and the estimation results with a large error are abandoned. By assuming that the orientation margin of the eight-shape route is known in the challenge, the trajectory estimation results could be

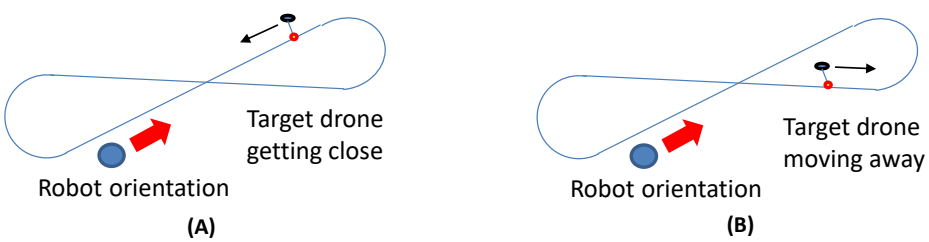

Figure 15. Examples of the target getting close and moving away. The robot is waiting close to the end of the straight segment.

compared with the trusted margin for the horizontal slope of line parts. In practice, the estimation results with a large difference to the reference slope margin would be evaluated as the low-quality estimation and abandoned.

Given trajectory estimation results with high quality, the nearest point from the robot to the estimated straight trajectory in the horizontal plane is selected as the interception position for the shortest response time, and could be directly computed by the perpendicular line towards the estimated straight trajectory. Furthermore, the interception point keeps updating when a new trajectory estimation result arrives.

To track the desired interception point, we send a velocity command proportional to the position tracking error to the robot:

$$\dot{\boldsymbol{r}}_{\mathrm{des}} = k_{\mathrm{track}}(\boldsymbol{r}_{\mathrm{des}} - \boldsymbol{r}_{\mathrm{robot}}) \tag{13}$$

where, $\boldsymbol{r}_{\mathrm{des}}$ and $\boldsymbol{r}_{\mathrm{robot}}$ are the desired and current robot positions, and k_{track} is the velocity gain.

Because of the uncertainties in the estimation results, the tracking procedure is further divided into three phases, including conservative tracking, active tracking, and open-loop interception:

- in the conservative tracking, the target is distant from the robot and the estimated trajectory has large uncertainties; thus, the velocity gain k_{track} is set as small to avoid the overshooting motion caused by the unreliable estimation results.
- in the active tracking, the estimated results are more reliable; thus, the velocity gain k_{track} is set to be larger to encourage the robot to make the rapid reaction to the latest desired interception position.
- in the open-loop interception, the target is at an extremely close-range and likely to disappear in the camera view, where the detection and estimation results are unreliable; thus, the desired interception is no longer updated, and the robot tracks the last interception point and close joints to capture the target.

The phases are switched heuristically according to the estimated distance between the robot and target as shown in Figure 16. The conservative and active tracking phases depend on the estimation results as shown in Figure 14, and Equation 11, whereas the open-loop interception phase does not

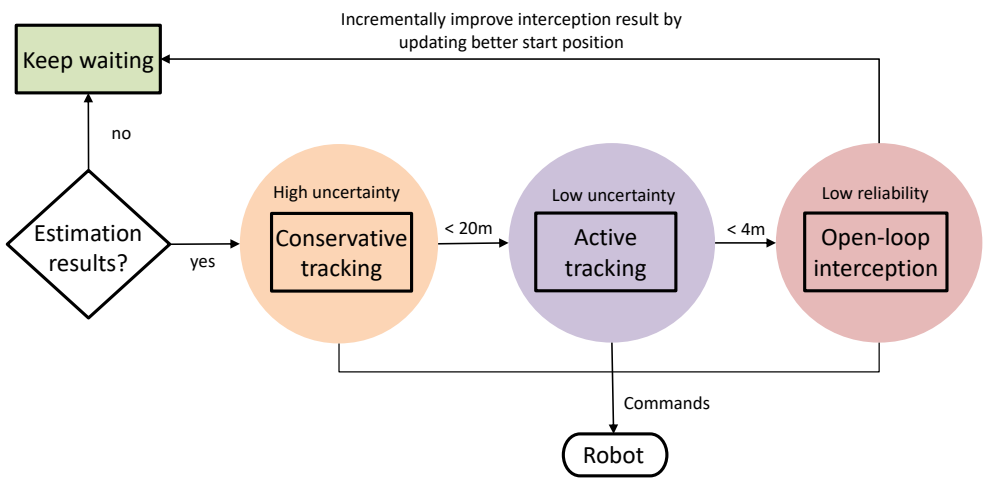

Figure 16. Pipeline of the tracking and interception, where the loop stops until the task is succeeded or time out. The input (i.e., "estimation results") is the estimated trajectory obtained from Figure 14, Equation 11, and Equation 12. The robot joints close after entering the "open-loop interception" phase, and re-open when back to the "keep waiting" phase.

need any estimation results. Note that, the open-loop interception phase is activated when the target distance is less than 4 m, which corresponds to the last 0.8 s (5 m/s) or 0.5 s (8 m/s) before the target is intercepted.

4.3. Incremental interception strategy

Because of the challenges in this task, including the short reaction time, fast target speed, and large uncertainties, it is common to fail in a single interception attempt, even for the team with the highest scores. In the 15 minutes challenge time, there are more than 20 full eight-shape routes and many interception attempt opportunities. Failure is not inevitable, but the global strategy could be developed to increase the success possibility based on the previous attempt. We develop an incremental interception strategy in the task level. In detail, the robot is not supposed to make the successful interception in only one shot; however, it is adjusted to a closer position to reduce the deviation and increase the success rate incrementally.

In the beginning, the robot moves to a pre-recorded GPS waypoint which is close to an end of a straight line where it is ensured that the target would fly towards the robot. Given that the orientation of the eight-shape trajectory varies in each trial, and normal GPS data has large errors, the distance to the real endpoint is more than 3 m in most cases. Once the target is detected, the desired interception position is computed. The interception position keeps updating until either the target is lost or the robot enters the open-loop interception state. Then the task-level planner would reset the state machine and make the robot stop at the last position to prepare for the next interception chance.

In the real challenge, the reactive time is less than 4 s for each interception attempt along with the large uncertainties in the detection and estimation results. Nevertheless, our incremental interception strategy is effective, and the interception could succeed within three iterations in most cases. More details would be analyzed in the following experiment chapter.

5. Evaluation and results

Our development began with a simulated system that allowed us to test different levels of abstraction. In particular, the planning and control method was evaluated using a life-size multilinked aerial robot platform in a physics-based simulation environment with Gazebo[14]. However, it is difficult to reproduce the real outdoor environment involving a complex background, motion blur, and sunlight conditions. Thus, the vision-based detection was trained and evaluated with various videos recorded from the on-board camera during system development. Furthermore, Challenge 1 required fast maneuvering for both our robot and target drone. Thus, it is important to anticipate all possibilities of interception motion in the real physical world and to conduct sufficient outdoor experiments in our development testbed, which had a configuration similar to that of the actual competition arena. The code of the entire aerial-interception system, including vision detection, aerial robot control, and interception planning can be found in our repository[15].

5.1. Offline training on target detector

For the SSD detector in the on-site experiment, we created an initial training dataset from onboard camera images under manual operation for both our robot and target drone. We also updated the dataset after each aerial interception experiment. The bounding box including both the drone and ball was manually annotated in each training image. Regarding data augmentation, we applied horizontal flip and image cropping randomly. We also applied transfer learning based on a pre-trained SSD model

[14] http://gazebosim.org/
[15] https://github.com/JSKAerialRobot/aerial_robot_demo/tree/master/mbzirc2020/mbzirc2020_task1

Table 1. mAP of dataset.

Location	On-site testbed	Real competition arena
Training mAP	0.666	0.405
Validation mAP	0.624	0.315

with CoCo dataset (ssd_mobilenet_v2_quantized_coco[16]). The weights of Mobilenet V2 backbone (i.e., CNN weights and batch normalization parameters) are frozen during training. Such a transfer learning allows us to save considerable time in training while maintaining feature-extraction accuracy. There are 128 images per minibatch and totally 5000 epochs performed with SGD optimization. We used a warmup learning rate which linearly increases from 0.09 to 0.2 for the first 100 epochs, and then use cosine decay to decrease learning rate from 0.2 to 0.000001 for the remainder. The total training time is approximately 6 hours with one NVIDIA GTX1080Ti. During the rehearsal and real competition, we collected sufficient image data from not only the onboard camera but also several handy cameras, and performed transfer learning based on the model trained with the on-site experiment dataset. The mean Average Precision (mAPs) of the training and validation datasets in both the on-site tested and real competition arena are summarized in Table 1. All datasets collected in both the on-site experiment and real challenge can be found in our repository[17].

5.2. Target tracking and localization

We first evaluated vision-tracking, the most critical component of the entire aerial interception system. We compared our tracking by the detection method presented in Section 3.1 with state-of-the-art visual trackers SiamRPN and DiMP-50. SiamRPN is a Siamese-network based tracker that uses AlexNet (Krizhevsky et al., 2012) as the backbone, and the structure is easy to quantize, indicating the potential to be deployed in the USB inference device (Google Coral TPU). DiMP-50 (Bhat et al., 2019) is a hybrid tracker that combines the Siamese-network and a correlation filter based on ResNet-50 (He et al., 2016) as the backbone. Thus, it has the best performance at the time of MBZIRC2020. However, both trackers require an initial template to start tracking, which is provided manually in this evaluation. Besides, to determine the best performance of these trackers, we ran them in a standard GPU (i.e., GTX1080Ti). However, our proposed method still ran in the USB inference device. To compare the three trackers, we chose a scenario from our testbed experiment. Figure 17 shows selections from a 5.5 s (166 frames) of that trail. We selected the initial bonding box from GUI to start two trackers at the first frame (our proposed method started tracking autonomously). We then counted the number of lost frames for each tracker, namely those where the estimated bounding box overlaps neither drone nor ball. As shown in Table 2, SiamRPN failed to track the target ball when the drone part was out of the image. Our proposed tracker (red) has almost the same performance as DiMP-50 (blue), which performed online updating for the target template. This result demonstrates that our SSD detector can extract the essential image features of the drone-ball model from the background properly even if there is only a partial component (e.g., only ball) inside the image.

We further evaluated the accuracy of the 3D position-estimation method presented in Section 3.3 and 3.4. As shown in Figure 18a, we manually operated the target drone with a constant distance from a static camera, and gradually increased the distance (i.e., 5 m, 10 m, 15 m, 20 m, 25 m and 30 m). We then calculated the error between the estimated ball position from the camera image and the ground truth obtained from RTK-GPS at separate axes as shown in Figure 18b,c,d, and also calculated the 3D localization RMSE as shown in Table 3. As shown in Figure 18d, the estimation

[16] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md
[17] https://github.com/JSKAerialRobot/aerial_robot_demo/blob/master/mbzirc2020/mbzirc2020_task1/mbzirc2020_task1_vision/script/install_train_data.py

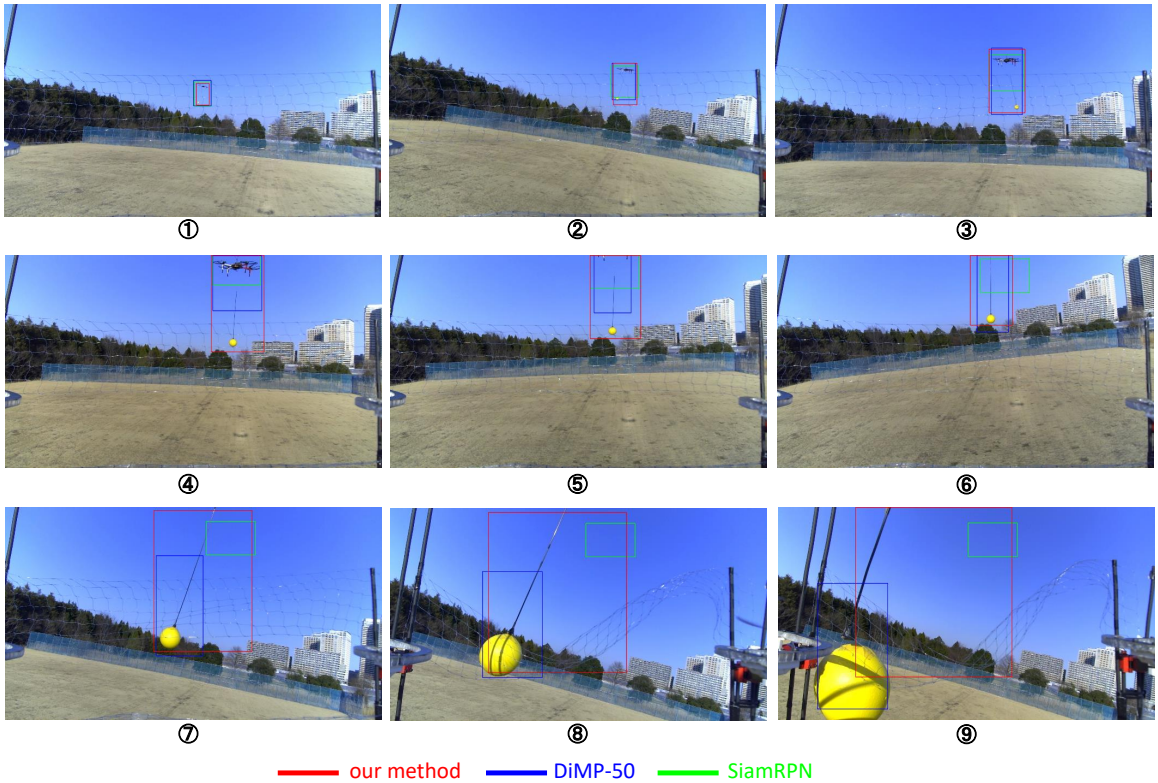

Figure 17. Comparison of performance of drone-ball tracking with different trackers (red: our proposed tracker presented in Section 3.1; blue: DiMP-50; green: SiamRPN). The bounding box in the first frame is given manually to start SiamRPN and DiMP-50, whereas our detector autonomously detects the drone-ball target. A related video can be found at https://youtu.be/s7TGY9uXLK8.

Table 2. Comparison of tracking performance between our proposed method and state-of-the-art trackers.

Trackers	**Our proposed SSD based method**	**DiMP-50**	**SiamRPN**
Backbone	MobileNet V2	ResNet 50	AlexNet
Inference machine	USB device (Google Coral TPU)	GTX 1080Ti	GTX 1080Ti
Inference time per frame [ms]	5	24	5
Lost frames (total frames: 166)	3	3	33

along the z axis demonstrated the largest error and deviation, and also increased significantly at long range ($> 15\,\mathrm{m}$). This is because the ball extraction at long range is difficult, thus, the z position (i.e., distance) as presented in Equation 8 and Equation 9 depends significantly on the width of the bounding box, which is not perfectly equal to the width of the drone. For such a problem, a ToF sensor or stereo camera, such as (Vrba et al., 2019), can directly provide the measured depth, achieving a better localization accuracy. However, our estimation result is still promising in the close and medium range ($< 15\,\mathrm{m}$), which is more crucial for an aerial interception. Furthermore, the presented trajectory estimation based on RANSAC can cover the low localization accuracy at long range when the target drone follows a known trajectory. It is also notable that our proposed method for target tracking and localization is inexpensive, requiring only a monocular camera, a standard CPU, and a USB inference device.

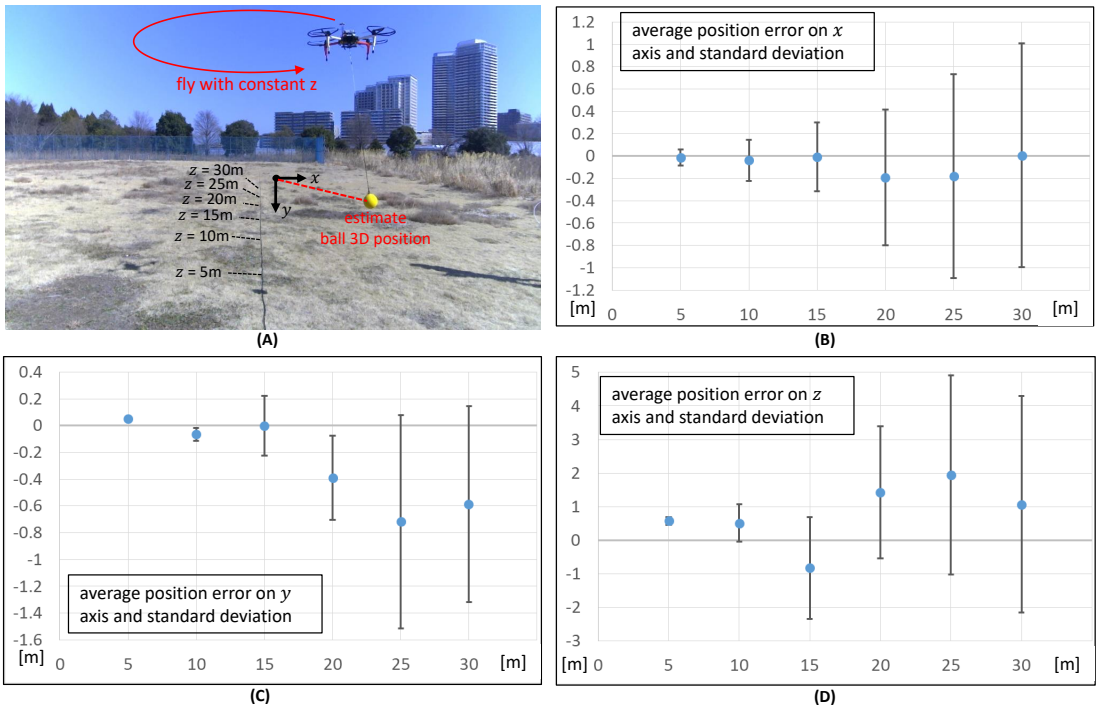

Figure 18. Evaluation of the proposed target ball position estimation. (A) Evaluation experiment where manually operated drone flew with a constant distance from the static camera. The coordinate origin is at the center of the camera image. (B)–(D) Average position errors on x, y, and z axes and their standard deviation under different distances (5 m, 10 m, 15 m, 20 m, 25 m and 30 m).

Table 3. 3D localization RMSE of evaluation experiment of Figure 18.

Distance [m]	5	10	15	20	25	30
RMSE [m]	0.59	0.78	1.72	2.56	3.81	3.98

5.3. On-site experiment

Our on-site testbed was a standard soccer field as shown in Figure 9. We developed the target drone based on DJI M100[18], suspending a yellow ball with a diameter of 0.15 m using a carbon pipe with a length of 1.5 m. Both the drone and ball were relatively larger than those in the actual competition. An eight-shape orbital trajectory (100 m × 40 m) with variant height (5 m to 15 m) was designed for this drone. The speed was set as 5 m/s, since a higher speed (i.e., 8 m/s) induced unstable trajectory-following performance in the DJI flight system.

Then, the proposed incremental interception motion was tested in such an environment, and Figures 19 to 21 demonstrate a typical result among various experiments. The interception with the yellow ball as shown in Figure 19⑧ and Figure 20⑧ was iteratively achieved by two cycles: ①–② and ⑤–⑧ correspond to the first interception (failure) and second interception (success), respectively, whereas ③–④ is a period in which the target drone flew away from our robot. As shown in the middle graph of Figure 19, the distance between our robot and the target ball was declined linearly when the target drone flew towards our robot and followed a straight segment (①–② and ⑤–⑧), indicating the correct detection of the target ball and position estimation. The bottom graph in Figure 19 demonstrates the position-tracking error relative to the desired encounter point, which

[18] https://www.dji.com/matrice100

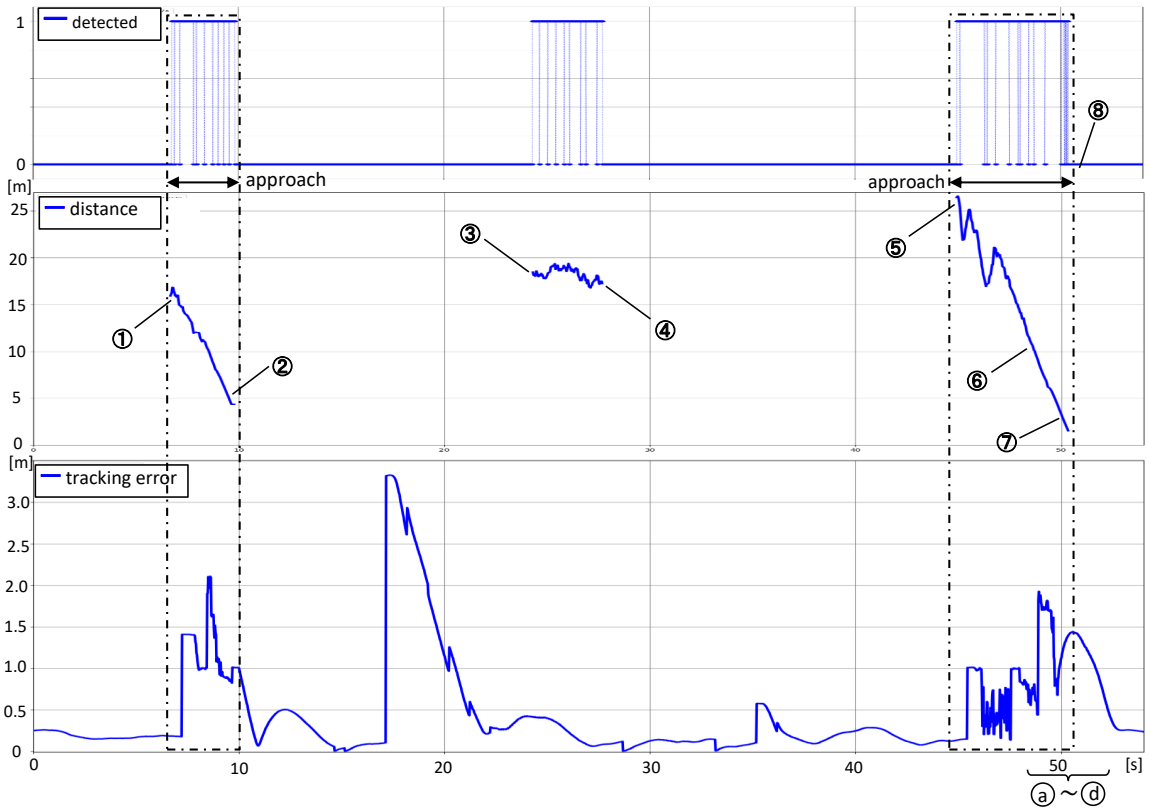

Figure 19. Onboard data collected during the on-site experiment corresponding to Figure 20. Top: ball detected flag (1: detected, 0: not detected); Middle: the relative distance between the target ball and our robot according to the estimated ball position as presented in Equation 10. ①–② and ⑤–⑧ imply the two approach periods, respectively; Bottom: the position tracking error between our robot and the desired encounter (interception) point.

later was incrementally updated according to our proposed interception planning. Although the first interception failed, the last encounter point updated at 10 s in Figure 19 provided a good waiting point for the next interception attempt, indicating the advantage of our incremental interception strategy. The desired altitude of our robot was updated to a neutral value of 7.5 m at 17 s as shown in Figure 19, since the robot height updated at 10 s had exceeded the upper limit of the orbital trajectory of the target drone. We infer that the drone's actual height might have exceeded the upper limit at this moment, since the target-drone's barometer-based height estimate was unreliable. The same phenomenon also occurred in the drone used in the real competition; however, the height recovery in our robot can obtain a relatively proper height for the next interception.

Figure 21 demonstrates the trajectories of our robot and the target ball during the second approach period. Although the real trajectory of the target ball should be a straight segment, the raw estimated trajectory (green path) fluctuated and was inaccurate at long-range (i.e., > 15 m) compared to the evaluation result in Figure 18. This was due to not only the inaccurate range-to-target estimation presented in Figure 12, but also the inaccurate time synchronization while transforming the 3D ball position from the camera frame to the world frame. Nevertheless, the proposed RANSAC filtering (yellow straight segment) increases robustness against these estimation errors. Furthermore, the straight segment estimated by RANSAC can also predict a good encounter point, which leads to successful interception, even if the vision-based detection on the target ball failed, particularly at the extremely close range as shown in Figure 20⑧. It is also notable that, position-tracking error at the interception moment (50.1 s in Figure 19) did not converge to zero, due to inaccurately

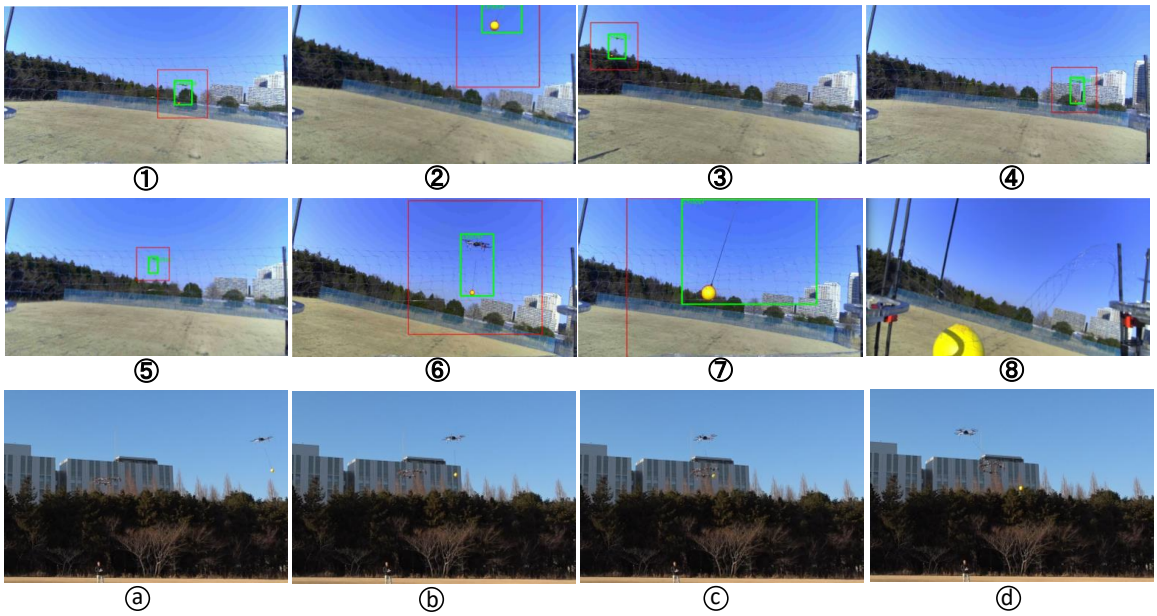

Figure 20. ①–⑧: representative detection results corresponding to Figure 19. ①–②: detection results during the first approach period; ③–④: detection results when the drone is flying away from our robot; ⑤–⑧: detection results during the second approach period (successful interception). ⓐ–ⓓ: successful interception moment corresponding to Figure 19, which dropped the ball from the drone. Related video can be found at https://youtu.be/s7TGY9uXLK8.

calculated intercept points near the end. However, the update of the desired encounter point during the entire approach period acted as a "low pass filter", enabling convergence to a point close to a true encounter point, but not to the point in the last RANSAC fitting segment (i.e., the yellow segment in Figure 21a).

To evaluate the integration performance of perception and planning, we analyzed the results of our final experiment day, in which the codes and parameters were the same as the settings during the MBZIRC competition. In the experiment day, there were 22 interception attempts with a total of 10 successful ones. The success rate in our local arena was 45.5%. Since we developed the incremental interception strategy, we also analyzed the required attempts for a successful interception, which involved an average of 1.8 attempts.

5.4. Results at MBZIRC2020

During the MBZIRC competition, we had two trials in Challenge 1 and one trial in the Grand Challenge to perform our interception system. We succeeded in dropping the suspended ball from the drone in the first trial of Challenge 1, and attained the third place in Challenge 1 out of the total of 22 competitors.

During the first trial of Challenge 1, we first tried to intercept the target at the left bottom endpoint of the straight segment as shown in Figure 2c. Our robot departed the start zone and moved to a GPS waiting-point established before the competition. However, the background captured from this waiting point was cluttered, and the target drone and ball almost blended into the surroundings, as shown in Figure 25a. Thus, we had few detection points to formulate the straight motion of the target ball, and our robot was unable to enter the interception phase during the first 9 min (8 min at 8 m/s and 1 min at 5 m/s). During that time, we also experienced a disconnection with the robot, since the arena was too large, and our operation room was far from the robot. After struggling with such a demanding condition, we decided to change the waiting point around the

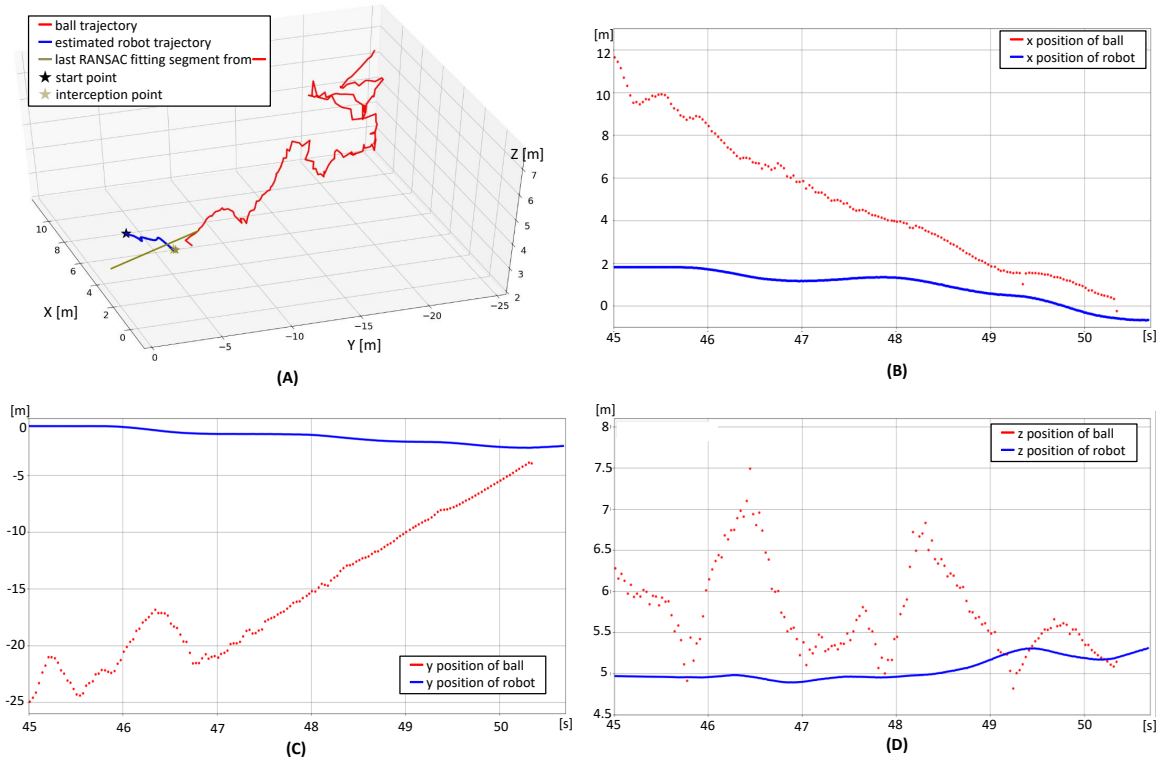

Figure 21. (A) 3D trajectories of our robot and the target ball during the second approach phase which ended with a successful interception (i.e., 45 s to 51 s in Figure 19). The red and blue trajectories correspond to the trajectories of the target ball and our robot respectively. The yellow segment denotes the RANSAC fitting resulted from the last 30 estimated points in the green trajectory, and the yellow star denotes the final interception point. (B)–(D) separate coordinates of trajectories of the target ball and our robot.

right bottom endpoint of another straight segment as shown in Figure 2c, which location was also closer to our operation room. We restarted our task for 10 min and succeeded in dropping the ball in 65 s as shown in Figure 22 and Figure 23. We had two approach opportunities (①–② and ⑥–⑧); however, our robot was in the waypoint phase from takeoff point to the waiting point (i.e, 26 s to 32 s), and missed the first approach chance. Before the second opportunity, the desired robot height was autonomously updated to 7.5 m at 51 s as shown in Figure 22 because the target ball was higher than our robot observed from ③–⑤. The height adjustment helped the robot to quickly converge to the desired encounter point for the second approach chance, which can also be confirmed from Figure 24 where the major change of the desired robot motion in the last two seconds (62 s to 64 s) was in the vertical direction rather than the horizontal direction. Finally, the edge of the robot net touched the suspending cable as shown in Figure 23ⓒ. It is notable that the target ball was not detected within the range of 5 m, indicating a more demanding condition for vision-based detection compared with our experimental environment.

During the second trial of Challenge 1 on the next day, our robot suddenly disabled its motors mid-competition (115 s) because of a problem with our original MCU board, resulting in a tragic crash from a height of 15 m. Druing the Grand Challenge on the last day, we were unable to takeoff because of machine trouble in the first 13 min. Finally, our robot succeeded in arming all motors at 13 min, and simultaneously fulfilled the interception challenge with other tasks. During the last 150 s for Challenge 1 in the Grand Challenge, we only had two chances to encounter the target drone, and the robot failed to intercept the ball at the end. However, according to the thorough analysis on the logged data, we determined that our robot could have intercepted the target at the next approach

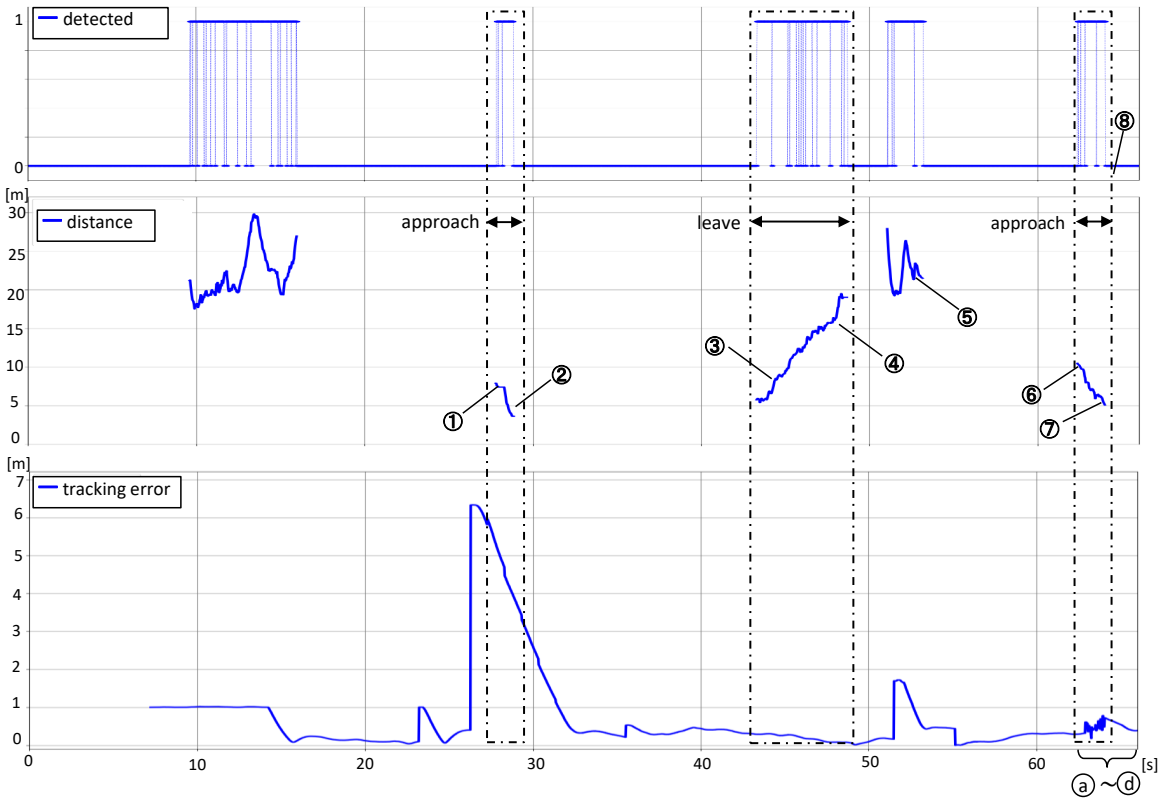

Figure 22. Onboard data collected during the first trial of Challenge 1 corresponding to Figure 23. Top: ball detected flag (1: detected, 0: not detected); Middle: the relative distance between the target ball and our robot according to the estimated ball position as presented in Equation 10. ①–② and ⑥–⑧ imply the two approach periods; Bottom: the position tracking error between our robot and the desired encounter (interception) point.

chance, since the robot position was getting iteratively closer to a proper interception point and was almost on the straight segment that the ball passed at the end. This post-mortem analysis supports the effectiveness of our proposed incremental interception planning method.

During the MBZIRC Challenge days, we attempted three intercepts in Challenge 1 and achieved one success, In the Grand Challenge, our single attempt failed to intercept. Although we only succeeded to drop the ball once in the real competition, the feasibility of our developed methods for target detection, trajectory estimation, and interception planning – along with the whole system – were effectively demonstrated, since Challenge 1 constituted a highly demanding task where only four teams (including Grand Challenge) out of all competitors were able to drop the ball.

5.5. Lessons learned

In terms of hardware, Challenge 1 is the most dangerous task, because aerial robots were required to perform fast maneuvering at a height of approximately 10 m for interception with another fast-moving object. Even if the robot can contact the ball, the propeller might become tangled in the detached ball or the suspending pole, inducing a crash from the 10 m height. Although many teams put a protecting net on the top of the propellers, such accidents can not be perfectly avoided. Thus, developing a platform that is easy to repair is one of the crucial keys to competing for multiple days. We also experienced a tragic crash during the competition; however, the highly modularized link structure of our robot platform allowed us to replace the minimum components which saved considerable time during the competition, and the unification of the robot platform in all challenges improved

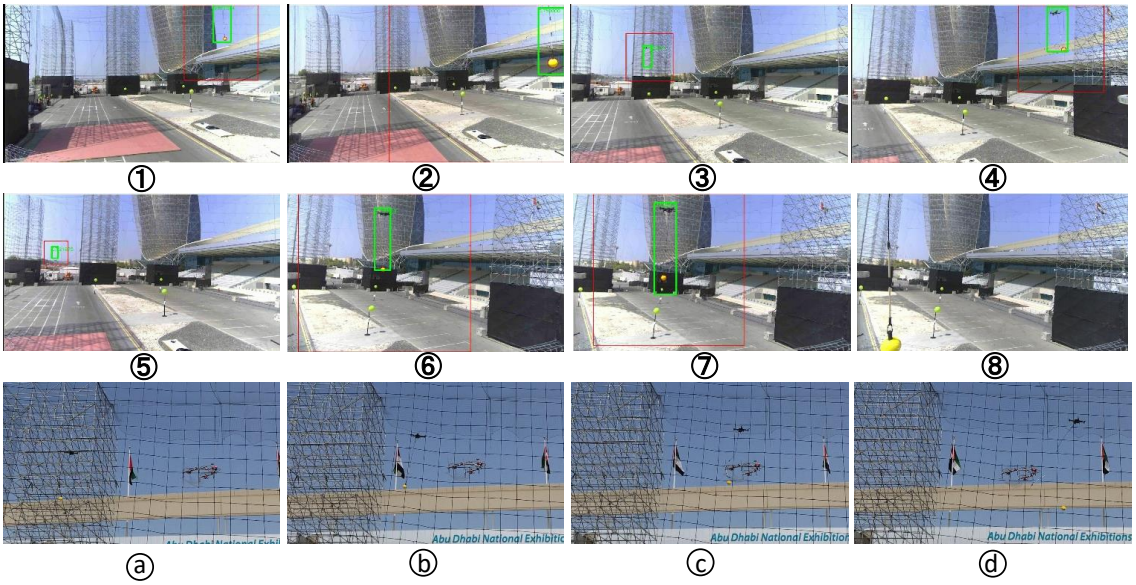

Figure 23. ①–⑧: representative detection results corresponding to Figure 22. ①–②: detection results during the first approach period; ③–⑤: detection results when the drone is flying away from our robot; ⑥–⑧: detection results during the second approach period (successful interception). ⓐ–ⓓ: successful interception moment corresponding to Figure 22, which dropped the ball from the drone. Related video can be found at https://youtu.be/s7TGY9uXLK8.

the robot management during the development and competition. Given that our robot is designed and built totally in-house and has a more complex structure than a common multirotor, we still have several bugs to resolve, indicating a trade-off between free customization and safety brought by off-the-shelf products. Nevertheless, many failures occurred during the on-site experiments and the real competition significantly facilitated the refinement of our original robot platform; we sincerely appreciate MBZIRC providing such an opportunity.

The main difficulty we experienced in the real competition is the demanding condition for the vision-based detection owing to the significantly more complex background, as well as the smaller drone and ball. Those unforeseen issues significantly increased task difficulty, and we were unable to detect the drone and ball during the rehearsals using our pre-trained SSD network based on our testbed experiments. Nevertheless, during rehearsal, we collected sufficient image data from not only the onboard camera but also several handy cameras; we first performed the transfer-learning with these image dataset based on the standard SSD network trained with the COCO dataset. However, the loss function during training could not converge. Then, we tried to perform the transfer-learning based on our task-tailored SSD network trained with the image data collected in our on-site experiments (mAP of on-site testbed dataset: 0.624), resulting in a relatively promising inference performance (mAP of real challenge dataset: 0.315). Such a transfer-learning allowed us to re-train a network with limited time and computation resources, leading to a relatively reliable detection result in the real competition. Besides, our combined drone-ball model improved robustness in the detection process improved the robustness. However, the detection performance was still worse than the performance in our testbed as shown in Table 4. In this table, we compare the detection performance in the last approach period. It is obvious that the number of detection (and tracking) points along with the detection range significantly reduces in the real competition compared with our on-site experiment. This is because the drone with a single black color in the real competition easily blended into the background, and the yellow ball and green balloons were very similar as shown in Figure 25. Fewer detection points indicating more iteration is required to converge to the proper encounter point in our interception planning, however, the competition time was limited. This reveals a weakness of

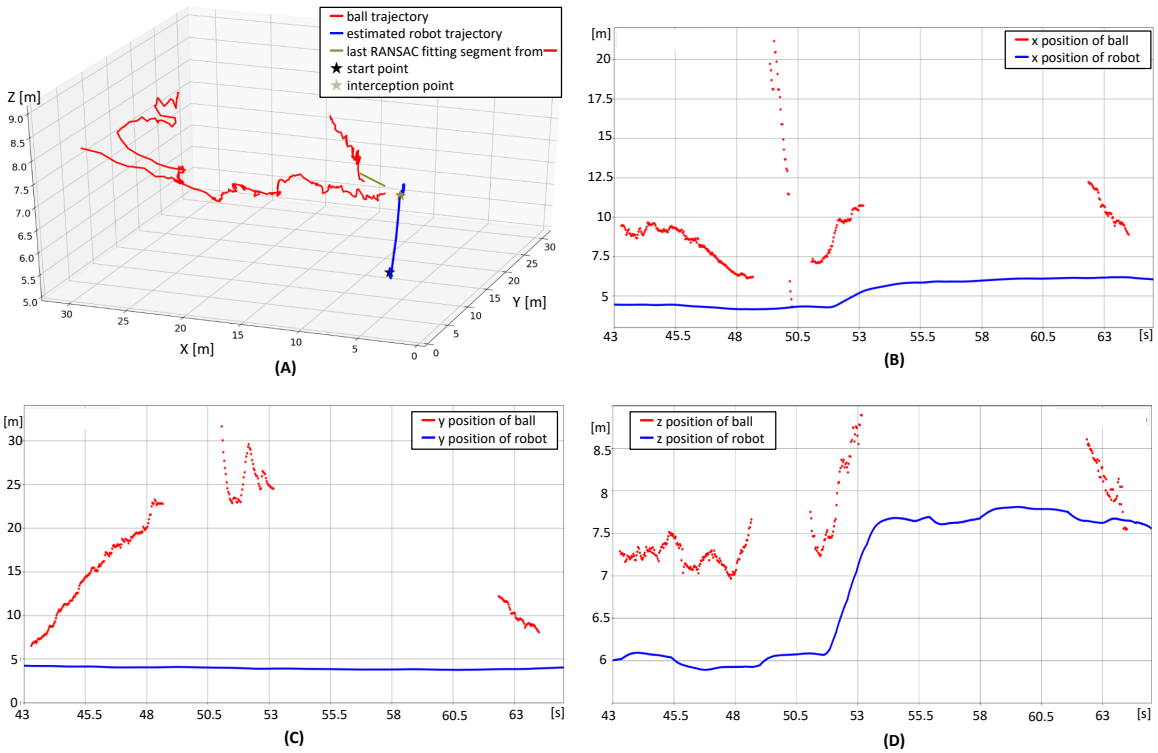

Figure 24. (A) 3D trajectories of our robot and the target ball during the leaving and second approach periods (i.e., 43 s to 64 s in Figure 22). The red and blue trajectories correspond to the trajectories of the target ball and our robot respectively. The yellow segment denotes the RANSAC fitting resulted from the last 30 estimated points in the green trajectory, and the yellow star denotes the final interception point. (B)–(D) separate coordinates of trajectories of the target ball and our robot.

Table 4. Comparison of detection performance in the on-site experiment and the real competition.

	On-site Experiment	**Challenge 1 Trial 1**	**Grand Challenge**
Interception	success	success	failure
Detection points	161	50	16
Detection duration [s]	5.332691	1.63	0.50
Average detection rate [Hz]	30.00	29.98	29.97
Estimated distance of first detection [m]	26.34	10.48 (△)	11.885377 (△)
Estimated distance of last detection [m]	1.53	5.09	5.21

△: unreliable value owing to inaccurate depth estimation

our "waiting" strategy compared to the active following strategy adopted by the winning team from BIT. To cover this shortage, a more robust visual tracking successively from the leaving period to the approach period (i.e., successive visual tracking from ③–⑦ in Figure 22) should be developed, facilitating a longer interception motion at each approach period.

Furthermore, the insufficient accuracy of the target ball position estimation is another factor that degrades interception performance. Although subsequent outlier filtering by RANSAC did improve the robustness, to some degree, we still need to improve the trajectory estimation from the perspectives of 1) temporal calibration among sensors and 2) the error model of depth estimation.

Although we came in third place for Challenge 1, our robot could not catch the ball. We found that the relative speed between the robot and target ball at the contact moment was too large ($> 5\,\mathrm{m/s}$), thus the ball always bounced off the net. In the competition, only the winning team

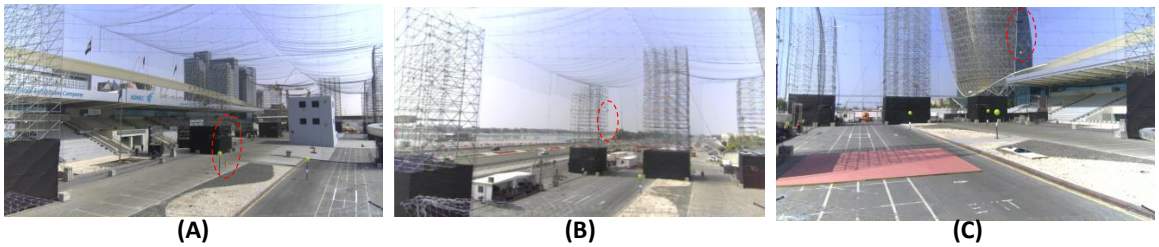

Figure 25. Several representative cases that our vision-based detection method cannot handle properly, since the target black drone and yellow ball entirely blended into the background of the real competition site. The target drone easily blended into the black structure (A), the scaffold (B), and the building (C).

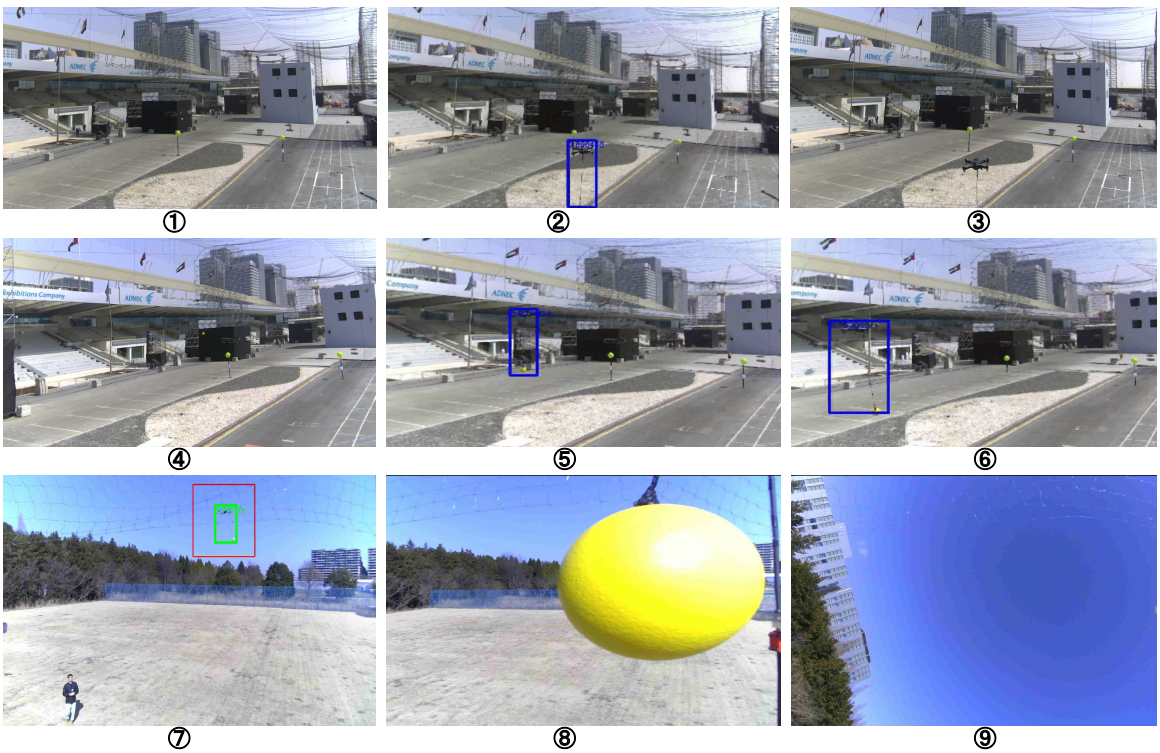

Figure 26. ①–③: Intercept-attempt failure in the first trail of Challenge 1, in which the target drone was moving at 8 m/s and the robot failed to track the height of the target well because of the limited reaction time. ④–⑥: the target drone moved 8 m/s as the previous failure, and the robot failed to move closer to the feasible horizontal interception position because of the short reaction time; ⑦–⑨: ball suspending stick sunk into the interception net and caused the crash of our robot during early development.

succeeded in capturing the ball. Given that our experiments before the competition mainly focused on the achievement of interception with a drone flying at the same speed as in the real competition, capturing motion was the second priority during our development. Therefore, advanced interception and capturing planning, which can suppress the relative speed by flying backwards against the coming target should be developed and testified by on-site experiments in the future. Nevertheless, we must highlight that only four teams out of 22 (including Grand Challenge) were able to intercept the ball autonomously, indicating the difficulties in both vision detection and motion planning to react quickly and avoid crashing during the challenge as shown in Figure 26. Simultaneously, the effectiveness of our simple but robust interception approach has been validated.

6. Conclusion

Our team demonstrated the autonomous capabilities of the prototype deformable articulated aerial robot platform in a task involving fast interception motion and won the bronze medal in Challenge 1 of MBZIRC 2020. This study presented the key considerations we addressed in developing the robot hardware platform, the vision-based detection, and the motion planning for intercepting a fast-flying target. The results from the on-site experiments and competition convey valuable insights toward the achievement of fast interception.

We addressed in advance possible issues regarding both the hardware and software; however, unforeseen problems occurred during the actual competitions, for example, the demanding condition to detect the target drone in a complex environment led to further development on the vision-based detection and tracking methods. Moreover, a successful capturing motion, which was not fully achieved by our deformable robot, is also an important future work.

Nevertheless, we believe that our contribution and experiences from Challenge 1 provide a good reference for operating aerial robots in the dynamic 3D real world for future participants. Team JSK plans to join the next MBZIRC in 2023 with our further evolved robot platform.

ORCID

Moju Zhao https://orcid.org/0000-0001-8361-5825
Fan Shi https://orcid.org/0000-0002-9202-1727
Tomoki Anzai https://orcid.org/0000-0002-4309-0750
Takuzumi Nishio https://orcid.org/0000-0002-0212-5507

References

Adcock, R. J. (1878). A problem in least squares. *The Analyst*, *5*(1), 53–54. https://doi.org/10.2307/2635758

Baca, T., Petrlik, M., Vrba, M., Spurny, V., Penicka, R., Hert, D., & Saska, M. (2021). The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, *102*(1), 1–28. https://doi.org/10.1007/s10846-021-01383-5

Bähnemann, R., Pantic, M., Popović, M., Schindler, D., Tranzatto, M., Kamel, M., Grimm, M., Widauer, J., Siegwart, R., & Nieto, J. (2019). The ETH-MAV team in the MBZ International Robotics Challenge. *Journal of Field Robotics*, *36*(1), 78–103. https://doi.org/10.1002/rob.21824

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). Fully-convolutional Siamese networks for object tracking. In G. Hua & H. Jégou (Eds.), *Lecture Notes in Computer Science: Vol. 9914. Computer Vision – ECCV 2016 Workshops* (pp. 850–865). Springer. https://doi.org/10.1007/978-3-319-48881-3_56

Beul, M., Nieuwenhuisen, M., Quenzel, J., Rosu, R. A., Horn, J., Pavlichenko, D., Houben, S., & Behnke, S. (2019). Team NimbRo at MBZIRC 2017: Fast landing on a moving target and treasure hunting with a team of micro aerial vehicles. *Journal of Field Robotics*, *36*(1), 204–229. https://doi.org/10.1002/rob.21817

Bhat, G., Danelljan, M., Van Gool, L., & Timofte, R. (2019). Learning discriminative model prediction for tracking. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 6181–6190. https://doi.org/10.1109/iccv.2019.00628

Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2544–2550. https://doi.org/10.1109/cvpr.2010.5539960

Chaudhary, K., Zhao, M., Shi, F., Chen, X., Okada, K., & Inaba, M. (2017). Robust real-time visual tracking using dual-frame deep comparison network integrated with correlation filters. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6837–6842. https://doi.org/10.1109/iros.2017.8206604

Dias, J., Lima, P. U., Seneviratne, L., Khatib, O., Tadokoro, S., & Dario, P. (2019). Journal of Field Robotics special issue on MBZIRC 2017 challenges in autonomous field robotics. *Journal of Field Robotics*, *36*(1), 3–5. https://doi.org/10.1002/rob.21851

Falanga, D., Kleber, K., Mintchev, S., Floreano, D., & Scaramuzza, D. (2019). The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, *4*(2), 209–216. https://doi.org/10.1109/lra.2018.2885575

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Gabrich, B., Saldaña, D., Kumar, V., & Yim, M. (2018). A flying gripper based on cuboid modular robots. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 7024–7030. https://doi.org/10.1109/icra.2018.8460682

García, M., Caballero, R., González, F., Viguria, A., & Ollero, A. (2020). Autonomous drone with ability to track and capture an aerial target. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 32–40. https://doi.org/10.1109/icuas48674.2020.9213883

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/cvpr.2016.90

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2015). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), 583–596. https://doi.org/10.1109/tpami.2014.2345390

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704–2713. https://doi.org/10.1109/cvpr.2018.00286

Jin, R., Owais, H. M., Lin, D., Song, T., & Yuan, Y. (2019). Ellipse proposal and convolutional neural network discriminant for autonomous landing marker detection. *Journal of Field Robotics*, *36*(1), 6–16. https://doi.org/10.1002/rob.21814

Kang, H., Joung, J., Kim, J., Kang, J., & Cho, Y. S. (2020). Protect your sky: A survey of counter unmanned aerial vehicle systems. *IEEE Access*, *8*, 168671–168710. https://doi.org/10.1109/access.2020.3023473

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 1097–1105. https://doi.org/10.1145/3065386

Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). SiamRPN++: Evolution of Siamese visual tracking with very deep networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4277–4286. https://doi.org/10.1109/cvpr.2019.00441

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with Siamese region proposal network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8971–8980. https://doi.org/10.1109/cvpr.2018.00935

Li, H. (2009). Consensus set maximization with guaranteed global optimality for robust geometry estimation. *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, 1074–1080. https://doi.org/10.1109/iccv.2009.5459398

Li, Z., Meng, C., Zhou, F., Ding, X., Wang, X., Zhang, H., Guo, P., & Meng, X. (2019). Fast vision-based autonomous detection of moving cooperative target for unmanned aerial vehicle landing. *Journal of Field Robotics*, *36*(1), 34–48. https://doi.org/10.1002/rob.21815

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2999–3007. https://doi.org/10.1109/iccv.2017.324

Lin, Y., Gao, F., Qin, T., Gao, W., Liu, T., Wu, W., Yang, Z., & Shen, S. (2018). Autonomous aerial navigation using monocular visual-inertial fusion. *Journal of Field Robotics*, *35*(1), 23–51. https://doi.org/10.1002/rob.21732

Liu, M., Zhu, M., White, M., Li, Y., & Kalenichenko, D. (2019, March). *Looking fast and slow: Memory-guided mobile video object detection*. arXiv: 1903.10172 [cs.CV].

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Lecture Notes in Computer Science: Vol. 9905. Computer Vision – ECCV 2016* (pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2

Nguyen, T., Shivakumar, S. S., Miller, I. D., Keller, J., Lee, E. S., Zhou, A., Özaslan, T., Loianno, G., Harwood, J. H., Wozencraft, J., Taylor, C. J., & Kumar, V. (2019). MAVNet: An effective semantic segmentation micro-network for MAV-based tasks. *IEEE Robotics and Automation Letters*, *4*(4), 3908–3915. https://doi.org/10.1109/lra.2019.2928734

Olsson, C., Enqvist, O., & Kahl, F. (2008). A polynomial-time bound for matching and registration with outliers. *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–8. https://doi.org/10.1109/cvpr.2008.4587757

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/cvpr.2016.91

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/cvpr.2017.690

Redmon, J., & Farhadi, A. (2018, April). *YOLOv3: An incremental improvement*. arXiv: 1804.02767 [cs.CV].

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/tpami.2016.2577031

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510–4520. https://doi.org/10.1109/cvpr.2018.00474

Štěpán, P., Krajník, T., Petrlík, M., & Saska, M. (2019). Vision techniques for on-board detection, following, and mapping of moving targets. *Journal of Field Robotics*, *36*(1), 252–269. https://doi.org/10.1002/rob.21850

Tzoumanikas, D., Li, W., Grimm, M., Zhang, K., Kovac, M., & Leutenegger, S. (2019). Fully autonomous micro air vehicle flight and landing on a moving target using visual–inertial estimation and model-predictive control. *Journal of Field Robotics*, *36*(1), 49–77. https://doi.org/10.1002/rob.21821

Vrba, M., Heřt, D., & Saska, M. (2019). Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system. *IEEE Robotics and Automation Letters*, *4*(4), 3402–3409. https://doi.org/10.1109/lra.2019.2927130

Young, P. C., & Willems, J. C. (1972). An approach to the linear multivariable servomechanism problem. *International Journal of Control*, *15*(5), 961–979. https://doi.org/10.1080/00207177208932211

Zhao, M., Anzai, T., Shi, F., Maki, T., Nishio, T., Ito, K., Kuromiya, N., Okada, K., & Inaba, M. (2021). Versatile multilinked aerial robot with tilted propellers: Design, modeling, control, and state estimation for autonomous flight and manipulation. *Journal of Field Robotics*, *38*(7), 933–966. https://doi.org/10.1002/rob.22019

Zhao, M., Kawasaki, K., Anzai, T., Chen, X., Noda, S., Shi, F., Okada, K., & Inaba, M. (2018). Transformable multirotor with two-dimensional multilinks: Modeling, control, and whole-body aerial manipulation. *The International Journal of Robotics Research*, *37*(9), 1085–1112. https://doi.org/10.1177/0278364918801639

Zhao, N., Luo, Y., Deng, H., Shen, Y., & Xu, H. (2018). The deformable quad-rotor enabled and wasp-pedal-carrying inspired aerial gripper. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–9. https://doi.org/10.1109/iros.2018.8594330

Zhou, X., Wang, D., & Krähenbühl, P. (2019, April). *Objects as points*. arXiv: 1904.07850 [cs.CV].

Zhu, M., & Liu, M. (2018). Mobile video object detection with temporally-aware feature maps. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5686–5695. https://doi.org/10.1109/cvpr.2018.00596

How to cite this article: Zhao, M., Shi, F., Anzai, T., Nishio, T., Maki, T., Ito, K., Kuromiya, N., Okada, K., & Inaba, M. (2021). Team JSK at MBZIRC 2020: Interception of fast flying target using multilinked aerial robot. *Field Robotics*, *1*, 70–101.