Received: 15 October 2017 | Revised: 9 March 2018 | Accepted: 12 July 2018

DOI: 10.1002/rob.21810

REGULAR ARTICLE

WILEY

Al-Robotics team: A cooperative multi-unmanned aerial vehicle approach for the Mohamed Bin Zayed International Robotic Challenge

Ángel R. Castaño | Fran Real | Pablo Ramón-Soria | Jesús Capitán | Víctor Vega | Begoña C. Arrue | Arturo Torres-González | Aníbal Ollero

Robotics, Vision and Control Group, Engineering School, University of Seville, Seville, Spain

Correspondence
Jesús Capitán, Robotics, Vision and Control Group, Engineering School, University of Seville, Av. Camino de los Descubrimientos, s/n, 41092, Seville, Spain.
Email: jcapitan@us.es

Funding information
Horizon 2020 Framework Programme, Grant/Award Number: 731667; Khalifa University

Abstract

The Al-Robotics team was selected as one of the 25 finalist teams out of 143 applications received to participate in the first edition of the Mohamed Bin Zayed International Robotic Challenge (MBZIRC), held in 2017. In particular, one of the competition Challenges offered us the opportunity to develop a cooperative approach with multiple unmanned aerial vehicles (UAVs) searching, picking up, and dropping static and moving objects. This paper presents the approach that our team Al-Robotics followed to address that Challenge 3 of the MBZIRC. First, we overview the overall architecture of the system, with the different modules involved. Second, we describe the procedure that we followed to design the aerial platforms, as well as all their onboard components. Then, we explain the techniques that we used to develop the software functionalities of the system. Finally, we discuss our experimental results and the lessons that we learned before and during the competition. The cooperative approach was validated with fully autonomous missions in experiments previous to the actual competition. We also analyze the results that we obtained during the competition trials.

KEYWORDS
aerial robotics, cooperative robots, robot competition

1 | INTRODUCTION

Multirobot teams are of interest for many applications where a single robot cannot perform all the tasks on its own or with the same efficiency. In aerial robotics, the same trend is arising, the use of teams of unmanned aerial vehicles (UAVs) to tackle autonomous missions is becoming commonplace. However, operating UAVs in outdoor and unstructured environments is still challenging, much more when they need to cooperate together. In those cases, the classic perception and control issues are complicated with additional communication constraints and the need for a more intelligent behavior.

Robot competitions are becoming popular, as they have proved to be helpful speeding up technological advances in certain robotics tasks. The idea is to replicate conditions from real life in simulated or testbed scenarios and push the community to propose efficient algorithms to solve specific challenges. Since all participants are forced to operate their robotic systems in the same controlled and standardized testbeds, competitions are also interesting in terms of robot benchmarking. They foster the replicability of results in robotics research and allow researchers to compare different approaches and methods under similar conditions.

Particularly, due to the recent advances in multi-UAV systems, there is an increasing need for testbed facilities and methodologies to compare existing methods in that field. Since competitions are a remarkable vehicle to develop specific technologies, aerial robot competitions are specially trending. The Mohamed Bin Zayed

International Robotic Challenge (MBZIRC)[1] is a new competition, which focuses on aerial robots operating outdoors, and cooperating between them and with ground robots.

In its first edition, which took place in March 2017 in Abu Dhabi, the MBZIRC gathered 143 applications from teams all around the world, out of which 25 top-teams were selected as finalists to compete in several outdoor challenges. In particular, the competition consisted of three challenges and a Grand Challenge integrating all together: Challenge 1 required a UAV to locate, track, and land on a moving vehicle; Challenge 2 required a ground autonomous robot to locate and navigate to a panel, and physically operate a valve stem on it with the appropriate tool; Challenge 3 required a team of UAVs to cooperate to search, track, pick up, and drop a set of static and moving objects; and the Grand Challenge required the team of UAVs and the ground robot to coordinate to solve Challenges 1, 2, and 3 simultaneously.

The Al-Robotics team was one of 25 finalists selected to participate in the first edition of the MBZIRC, and their members are researchers from the Robotics, Vision, and Control Group,[2] at the University of Seville. Even though the team participated in all the challenges, this paper focuses on the cooperative approach that was designed and implemented to address the Challenge 3. This task is particularly challenging for several reasons. From the point of view of the system architecture, a team of UAVs have to operate together in an outdoor scenario, showing cooperative behaviors. From the perception point of view, the UAVs must be able to locate and track different types of objects. Last but not least, they must interact physically with the environment by picking up, transporting and dropping static and moving objects.

In this paper, we describe in detail the approach and the systems used by the Al-Robotics team in the MBZIRC Challenge 3. First, a vision-based algorithm is proposed to detect objects based on color. The objects in the competition have known colors and sizes, so we developed color-based techniques assuming they were on the ground. Second, a data fusion approach is used to integrate observations from all the UAVs in the team and compute a probabilistic estimation of the objects' positions. Given the lack of communication issues (there are only few UAVs working in a short range), we opted for a centralized stochastic filter due to its robustness. Third, a mission planner is used for cooperative decision-making, sending the UAVs to search for objects, and later to perform pickup and dropping operations. After a detection phase, the UAVs are assigned objects heuristically so that hypothetical conflicts, that is, UAVs with crossing paths, are minimized. The pickup operations are carried out by means of a magnetic tool and a vision-based controller since the positioning systems of the UAVs are not accurate enough to pick up the objects.

In summary, the main contributions of this paper are: (a) to present our overall approach for the MBZIRC Challenge 3, combining multi-UAV data fusion and decision-making with vision-based algorithms; (b) to detail the design and implementation of our hardware and software architectures; (c) to describe our results and the lessons we learned before and during the competition, some of them even leading to system redesigns.

The remainder of the paper is structured as follows: Section 2 surveys the related work; Section 3 presents the overall approach followed by Al-Robotics to tackle the challenge; Section 4 provides details on the design of the aerial platforms; Section 5 describes the software architecture and functionalities; Section 6 discusses the evaluation of the system and the lessons learned; and Section 7 gives conclusions.

2 | RELATED WORK

This section presents related work relevant for the main components of our system. We also summarize the state of the art with respect to robot competitions.

2.1 | Robot competitions

Robot competitions are spreading fast due to the inherent difficulties associated with robotics benchmarking Stuckler, Holz, and Behnke (2012). Such competitions allow roboticists to test methods and compare them under the same conditions since they provide controlled testbeds where specific robotics challenges need to be solved. In this sense, there are recent initiatives like RoCKIn (Amigoni et al., 2015) to develop competitions where the focus is on coming closer to scientific experiments and enabling the replicability and repeatability of experimental results.

The Defense Advanced Research Projects Agency (DARPA) is one of the most active actors organizing robot competitions. They started with the DARPA Grand Challenge and Urban Challenge, which focused on autonomous ground vehicles; but they have organized recently their successor, the DARPA Robotics Challenge,[3] which fosters the development of humanoid robots solving complex tasks in disaster or emergency scenarios (e.g., driving a vehicle to the disaster site or manipulating valves). Another competition in the domain of rescue robotics is euRathlon,[4] which was inspired by the 2011 Fukushima accident and combines marine, aerial, and ground robots in an outdoor testbed.

The RoboCup[5] is a worldwide known competition involving different domains. It started as a league focused on cooperative teams of intelligent robots playing soccer, including humanoid leagues. However, they included later new leagues for rescue (RoboCup Rescue), industrial (RoboCup@Work), and service robots (RoboCup@Home). Indeed, this kind of competitions to develop home-assistant robots and to solve specific industrial challenges are

[1] http://www.mbzirc.com.

[2] https://grvc.us.es.

[3] http://archive.darpa.mil/roboticschallenge.

[4] http://www.eurathlon.eu.

[5] https://www.robocup2017.org.

becoming highly popular. Another example is the recent Amazon Robotics Challenge,[6] which proposes pick and stow tasks in unstructured industrial scenarios. In Europe, one of the challenges of the European Robotic Challenges (EuRoC)[7] is about plant servicing and inspection, and it targets the open problems in existing MAV (Micro Aerial Vehicle) solutions to enable their deployment in real, industrial applications.

2.2 | Vision-based object detection

Vision-based object detection is a complex task, which is not completely solved since most existing algorithms focus on detecting particular subclasses of objects to be more efficient. For instance, Viola and Jones (2001) use a boosted cascade classifier for detecting objects; Dalal and Triggs (2005) present another accurate classifier for general object detection, but it might be sensitive to appearance variations or changes in the object due to nonrigid deformations; and Felzenszwalb, Girshick, McAllester, and Ramanan (2010) propose an improved method for detection of deformable objects based on a multiscale model for deformable parts. Many approaches for object detection assume a model description using features selected by hand. However, recent works also use Artificial Neural Networks, as they have proved to be effective for learning a complex variety of objects (Goyal and Benjamin, 2014).

A common feature for object detection is their color. Algorithms for color segmentation have been widely studied for a long time, as many perception systems are based on RGB cameras, and the color is usually a quite distinguishable feature. The authors of Ilea and Whelan (2006) propose an adaptive technique for color segmentation based on the *K*-means algorithm. This algorithm presents the drawback that the parameter *K* for spatial color segmentation must be selected independently for each image, which results difficult because not all the scenes contain the same amount of objects. Thus, increasing the parameter might decrease the efficiency of the algorithm, whereas decreasing it might result in a mixture of colors within the same cluster. Tai, Jia, and Tang (2007) propose an automatic solution based on Gaussian Mixture Models. The previous methods focus on robust color segmentation but not on efficiency in terms of computation speed. Moreover, they usually perform color segmentation without keeping a track of the segmented zones, so another algorithm has to estimate later the position of the objects in the image.

In this study, we need to extract as much information as possible from the objects (e.g., size, shape, color, and position) and as fast as possible. Therefore, our color segmentation is an optimized version of the algorithm described in Bruce, Balch, and Veloso (2000). The results are then fused with information about the position and orientation of the camera to generate three-dimensional (3D) object positions by means of the corresponding homography.

[6]https://www.amazonrobotics.com//#/roboticschallenge.

[7]http://www.euroc-project.eu.

2.3 | Multirobot object tracking and decision-making

The use of multiple cooperative UAVs for missions where the positions of some objects or targets must be estimated and tracked is commonplace. These vehicles can provide enhanced sensing capabilities, faster dynamics, wider fields of view and they can access more hazardous areas; all of which are remarkable advantages for applications like surveillance and situational awareness in rescue robotics (Burdakov, Doherty, Holmberg, Kvarnstrom, & Olson, 2010; Hsieh et al., 2007; Beard, McLain, Nelson, Kingston, & Johanson, 2006).

From the point of view of perception, the problem of target tracking by means of a team of UAVs has been extensively studied. An estimation of the targets' positions and their associated uncertainties can be maintained by using different types of stochastic filters, which fuse observations coming from multiple sensors on board the team members. Depending on the models and sensors involved, some works assume Gaussian probability distributions and propose Kalman Filters (KFs; Morbidi & Mariottini, 2011) or Information Filters (Capitan, Merino, Caballero, & Ollero, 2011); whereas others deal with multimodal distributions through Bayes Filters (Cook et al., 2014), Particle Filters (Ong et al., 2006), or Gaussian Mixture Models (He, Bachrach, & Roy, 2010).

Besides the estimation problem, a decision-making problem needs to be solved, so that each UAV knows which are its best actions during the mission to locate the targets. One approach is to use stochastic optimal control to formulate the problem, trying to optimize some utility function based on the targets' uncertainties (Anderson & Milutinovic, 2013; Morbidi & Mariottini, 2011). These uncertainties can be quantified by means of different metrics, such as entropy or mutual information.

Another relevant approach for decision-making in this kind of missions is to split the scenario into survey areas or points to visit and assign them to the UAVs in an efficient manner. In this sense, coverage path planning algorithms can be useful, that is, algorithms to cover a certain area efficiently with a team of robots (Galceran & Carreras, 2013) present an extensive survey of those algorithms, providing a categorization for decomposition and coverage techniques in the literature. Task allocation techniques also play a key role in multi-UAV cooperation. The authors of Korsah, Stentz, and Dias (2013) provide an extensive literature review and propose a novel taxonomy. Traditionally, those algorithms allocate tasks to UAVs in an efficient manner, being typical tasks the points to visit for searching targets or the targets themselves, to be tracked. However, these tasks can vary depending on the techniques used. For instance, some people have proposed recently auctions to allocate behavior-based policies (Capitan, Merino, & Ollero 2016) or best-planned paths (Cook et al., 2014) among the UAVs. Moreover, the heuristics considered to solve the problem efficiently are important. Most works try to optimize the distance traveled or the energy consumed, but information-based heuristics can also be used.

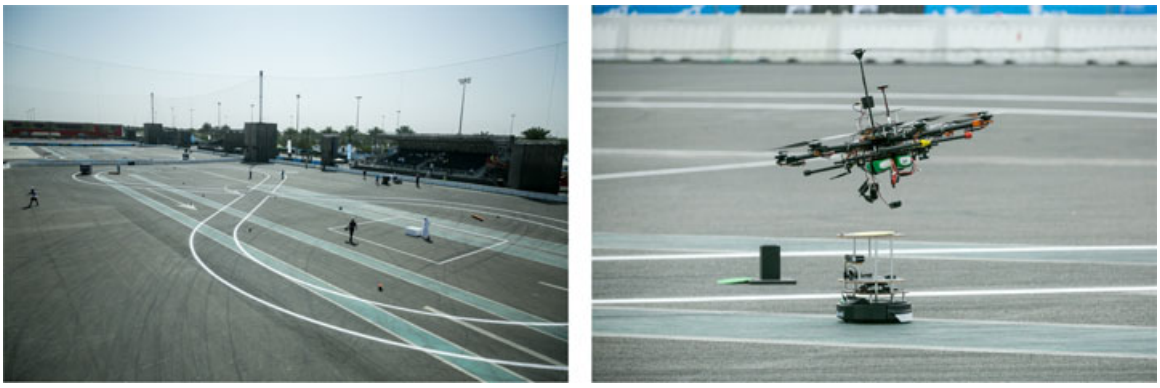

FIGURE 1 Competition site in Abu Dhabi. Left, the top view of one of the arenas with an eight-shaped track that was used to drive the vehicle of Challenge 1. The dropping box is white and lies in the middle of one of the track laces. Right, a UAV trying to pick up one of the moving objects. In the background, a green static object has fallen down from its pedestal. Images from the MBZIRC organization at http://www.mbzirc.com [Color figure can be viewed at wileyonlinelibrary.com]

3 | OVERALL APPROACH

In this section, we present an overview of our approach to address the MBZIRC Challenge 3. We summarize first the main features of the Challenge itself to ease the understanding of the required functionalities and constraints in the system. Then, we sketch our proposed architecture, with the different modules involved.

3.1 | Challenge description

Challenge 3 of the MBZIRC requires a team of UAVs (up to 3) to cooperate to search and find a set of static and moving objects. The UAVs will be equipped with magnetic, suction, or other types of effectors to pick up the found objects and drop them into a dropping box, whose position is known in the middle of a Dropping Zone (DZ). This challenge is expected to last for a maximum of 20 min and takes place in an outdoor arena with GPS signal accessible. The arena is approximately the size of a football pitch (around 100 m × 60 m).[8]

Communication between the UAVs and the ground station, and between the UAVs themselves, is allowed and based on an IEEE 802.11 network provided by the competition organizers. For safety reasons, the speed of the UAVs is limited to 30 km/hr. Their size is also restricted to a maximum volume of 1.2 m × 1.2 m × 0.5 m. All these technical constraints affect the platform design, as it will be explained in the next sections.

The objects randomly spread on the arena are of different types, all of them made of ferrous material (some pictures can be seen in Figure 1). There are 6 moving (with a speed lower than 5 km/hr) and 10 static small objects, as well as 3 static large objects. The small objects consist of circular disks on top of static pedestals that elevate them from the ground, or on top of small, moving platforms. There are three different colors and scores associated with the static objects: green, blue, and red. The moving objects are yellow, and the large ones orange. Moreover, the large objects are of rectangular shape (not exceeding the 2 kg) and may require of several UAVs to be picked up and transported. Thus, a higher score is associated with the large objects, and even higher when they are picked up by more than one UAV.

The score for each object is given only if the UAV drops it into the dropping box (1 m × 1 m). The large objects do not need to be placed into the box, but it suffices with the surrounding dropping zone. A lower score is obtained if the operation is not completed fully autonomously but with human intervention. The team collecting the maximum number of points is the winner. More details about the scoring scheme and the Challenge description can be seen in the official MBZIRC website.

MBZIRC Challenge 3 is an attempt to foster cooperative techniques due to the restrictions imposed. The main constraints are related to the level of autonomy of the aerial platforms, that need to fly for 20 min; to the onboard sensors needed to find objects; and to the design of the pickup device. Also, a high control precision is necessary to pick up and drop down objects in the right position; and cooperative approaches should be more beneficial when allocating the tasks among the team members.

3.2 | System architecture

A cooperative solution with several UAVs is proposed to address the MBZIRC. All the aerial platforms are homogeneous and equipped with the same hardware and functionalities, so they collaborate in the same manner to search and collect the objects in the arena. In particular, they all carry a camera for visual detection and a magnetic device for pickup operations. As it will be detailed in Section 4, the aerial platforms were designed to fulfill with requirements in terms of flight time and payload capacity. These requirements are given by the objects' sizes of the competition and the trials' length.

Figure 2 shows a diagram with the different blocks that compose our whole system. In the detailed view of the systems on board the

[8]During the competition in Abu Dhabi, two identical arenas were installed for the trials.

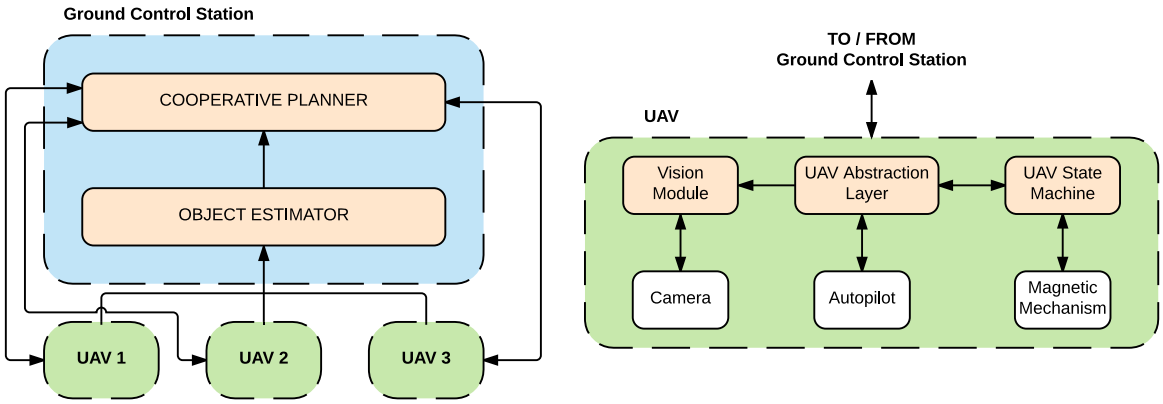

FIGURE 2 Block diagram of the proposed system architecture. Left, modules on the Ground Control Station and communication links with the UAVs. Right, detailed view of the blocks on board each UAV [Color figure can be viewed at wileyonlinelibrary.com]

UAVs, it can be seen that each UAV has a visual camera and runs a Vision Module to detect object positions and colors. Also, the UAVs carry an autopilot connected to the GPS and to the inertial sensors. This autopilot is in charge of providing localization in global coordinates, as well as navigation capabilities toward commanded waypoints. To abstract users from the low-level control and hardware, we implemented a UAV abstraction layer (UAL) through which higher-level commands can be issued (e.g., *take off*, *land*, or *go to waypoint*). Thus, all the architecture is independent of the particular autopilot and sensors used, which gives more flexibility to the system. Once a UAV is commanded to collect an object, it needs to navigate to the object's position, descend and activate a vision-based controller to pick up the object making contact with the magnetic device, navigate to the dropping box, and drop the object releasing the pickup device. All this level of autonomy is carried out by means of the UAV State Machine that runs on board and receives high-level tasks from a ground control station (GCS).

In addition to the processes that run onboard each UAV, there are also centralized modules that run on the GCS. In particular, all the vision-based observations from the UAVs are transmitted to this GCS and fused together into the object estimator, which keeps track of the estimated positions and other features of all the detected objects in the arena. With this information, the cooperative planner decides where each UAV should go and which object it should collect at each moment. This module is also in charge of resolving potential conflicts so that the vehicles do not collide.

In general, we apply in our architecture algorithms which are robust enough and heuristics that allow us to tailor the system to the competition objectives. Nonetheless, the methods used are widely used in the literature for different purposes, so our architecture could be seen as flexible and could be adapted for other domains without major modifications. All details about the algorithms developed will be given in Section 5. The vision module detects objects by means of their color. Given the fact that objects' colors and sizes are known and assuming that they will always be on the ground, we can apply color segmentation on the images and project detections on the ground plane with the 3D position of the cameras. To fuse all color-based detections from the UAVs, we use a centralized stochastic filter. We selected a centralized approach due to its simplicity and efficiency since there are only three UAVs operating in a relatively short range.

Regarding the cooperative planner, we also opted for a centralized scheme due to its robustness and to avoid inter-UAV conflicts as much as possible. The arena can be covered fairly fast with the three UAVs, so we divide the area to search for objects first. Then, we apply heuristics to assign objects to the UAVs, prioritizing assignments where UAVs crossing their paths are unlikely and collecting static objects first since they are simpler. Moreover, objects are picked up by means of a visual-based controller. This is needed because the localization system of the UAVs is not accurate enough to pick up objects, so we exploit the color-based detector already developed to approach the objects.

4 | AERIAL PLATFORM DESIGN

In this section, we describe the details of our aerial platforms for the MBZIRC Challenge 3. We overview the procedure that we followed to find the final design and to select and validate the components on board the UAVs. The weights of all the physical components on board the UAVs are also provided. The UAV design is based on three main restrictions imposed by the description of the Challenge:

1. The maximum duration of the Challenge is 20 min.
2. The maximum weight of the large objects is 2 kg.
3. The UAVs must fit within the volume 1.2 m × 1.2 m × 0.5 m.

We computed the minimum payload for each UAV considering the maximum weight of the competition objects and taking into account that the large objects can be transported by two UAVs. Thus, a payload of 2.5 kg was estimated: 1 kg corresponding to half of a large object; 1 kg for the electronics and sensors (i.e., onboard computer, camera, electronics battery, wireless link, etc.); and 0.5 kg for the pickup device. According to all these constraints, we needed

FIGURE 3 Aerial platform developed for the Challenge. Left, airframe without the mission electronics and the pickup mechanism. Right, fully equipped hexacopter [Color figure can be viewed at wileyonlinelibrary.com]

an aerial platform with a payload of at least 2.5 kg and able to fly for 20 min.

Well-known platforms from Ascending Technologies like the Asctec Firefly and the Pelican do not offer enough payload (600 and 650 g, respectively) nor flight time (14 and 16 min, respectively), and the promising Asctec Neo was not available for sale at the competition time. Then, we analyzed the two following platforms in detail: a Yuneec Tornado H920 and a DJI S900. On the one hand, the Yuneec Tornado H920 fits with the size requirements (1.06 m × 1.06 m × 0.5 m) but the payload and flight time are rather tight. It has a maximum payload of 2.3 kg and an estimated flight time for 1.6 kg of around 24 min. On the other hand, the DJI S900 is a little larger than the specifications (1.28 m × 1.28 m × 0.5 m), its maximum payload is 3.3 kg and its estimated flight time for 2.1 kg is 18 min.

We decided that the payload of the Tornado H920 was not enough. Regarding the DJI S900, it seemed plausible to make the frame arms a bit shorter to match the size specifications. However, the flight time was still quite tight. The estimated flight time given by the manufacturers is typically calculated with the vehicle hovering, whereas the Challenge implies maneuvers where the UAVs are mostly moving: searching for objects, descending and going up again to collect them, going to the dropping zone to drop them, and starting over. We also estimated a flight time reduction due to the expected high temperatures in Abu Dhabi in March. Therefore, we discarded the previous platforms and designed a custom hexacopter together with the Spanish company DroneTools[9] The final platform can be seen in Figure 3, and it is made of carbon fiber (CF) with a size of 1.18 m × 1.18 m × 0.5 m, including rotor blades.

We tested first our airframe with different configurations of motor, blades, and batteries. The platform only included a Pixhawk autopilot, a GPS receiver board (3DR uBlox LEA-6H High-Performance Receiver), an RC transmitter/receiver and a 433 MHz telemetry radiolink to communicate with the well-known QGroundControl software running on a laptop. It was also loaded with a dummy weight equal to the weight estimated for the electronics. To recreate the weather conditions in Abu Dhabi, these tests took place in summertime in Seville, which means an outdoor temperature above 30°C. The procedure for these tests was as follows:

1. Navigate autonomously a rectangular flight plan 200 m long.
2. Pick up and drop manually metallic objects continuously, until the batteries were discharged to 10%.
3. Check the flight time and motors temperature.

The first tests included AXI 2814/22 765 kV brushless motors with 14 × 4.8 and 13 × 6.5 CF propellers and 6S LiPo batteries. With this configuration, we achieved a flight time of 15 min without motor failures, but the motors reached really high temperatures. Therefore, after several tests, the following configuration was selected: T-Motor Antigravity MN4006 380 kV brushless motors, 15 × 5 CF propellers, JETI 40 A Opto ESC (Electronic Speed Controllers) and 2 Tattu batteries (7,000 mAh 22.2 V 25/50C 6S1P). This way, we reached a flight time of 23 min, while the motors temperature was normal.

Once the aerial platform was validated, it was equipped with all the sensors and devices required for the Challenge. Figure 4 depicts the spatial distribution of the onboard equipment, which is the following:

- Onboard computer: An Intel NUC5i7RYH computer with 16 GB RAM and a 256 GB Samsung 950 PRO M.2 SSD hard disk. This computer weights 1.1 kg mainly due to the metallic case, so we replaced it with a custom-made plastic case to reduce the weight to 460 g (see Figure 5). This computer is connected to the Pixhawk through a serial port and mounted on a quick-release system so that it can be easily replaced.
- Camera: A ZED stereo camera was selected after testing also a Basler daA1280-54 μm and an Intel R200. It is connected to the onboard computer through a USB 3.0 interface.
- Altimeter: A Lightware SF11C laser altimeter is integrated and directly connected to the Pixhawk autopilot. It gives a complementary measurement to the barometric pressure altimeter included in the Pixhawk.

[9]http://www.dronetools.es

FIGURE 4 Front view (left) and side view (right) of the aerial platform with all the sensors and electronic devices on board. The spatial distribution of the equipment is indicated [Color figure can be viewed at wileyonlinelibrary.com]

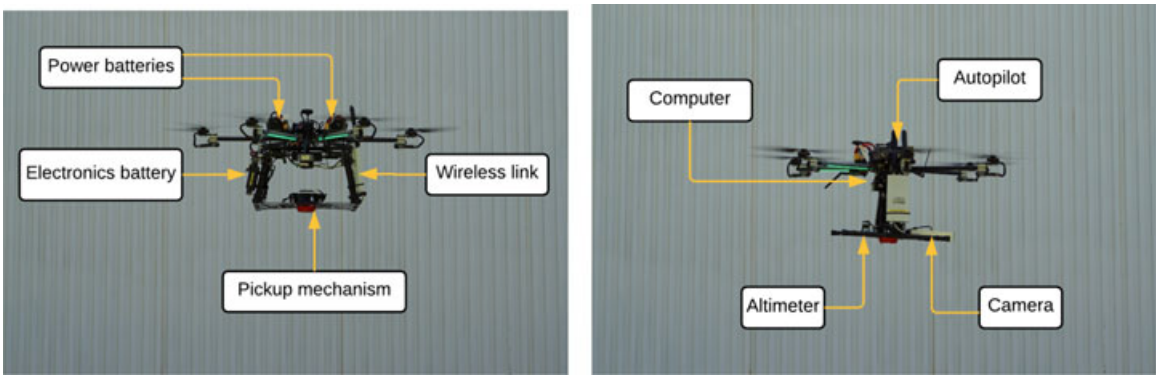

FIGURE 5 Detailed view of the core of the aerial platform. Left, Intel NUC computer with plastic case. Right, Pixhawk autopilot on top of the damped base [Color figure can be viewed at wileyonlinelibrary.com]

- Wireless link: An Ubiquiti Rocket M5 5.8 GHz radiolink is used for communications with the ground control computer and other UAVs. This device is connected through an Ethernet interface to the onboard computer.
- Electronics battery: A Hacker ECO-X Light 4S 3,500 mAh 10C independent battery for the electronics.
- Pickup mechanism: An object pickup device based on an OpenGrab EPM (Electro Permanent Magnet) by NicaDrone. This device is described in detail in Section 4.1.

Table 1 summarizes the weight distribution for the complete aerial platform. With this platform, we achieved a consistent flight time of 23 min, while having available enough payload to pick up the large objects of the Challenge (between 2 UAVs). Moreover, a voltage monitor with acoustic warning was attached to each battery to increase the safety of our operations. During our tests, we detected some vibrations in the internal IMU (Inertial Measurement Unit) of the Pixhawk, so a damped base was built to place the autopilot (see Figure 5). Our Pixhawk runs a modified version of the PX4 release v1.4.4 stack in order to integrate the laser altimeter and the control of the electromagnetic device.

4.1 | Pickup mechanism

The main objective of the Challenge is to pick up objects and drop them correctly, so the implementation of a mechanism for these operations is crucial. We describe in this section the design of the device that we developed for this purpose.

Given the metallic nature of the objects in the competition, an electromagnet seems to be the simplest option to grab them. Our pickup mechanism is based on an Electro Permanent Magnet (EPM), the

TABLE 1 Distribution of the weight for the aerial platform and the onboard equipment

Category	Component	Weight (g)
Airframe	Arms, motors, ESC, autopilot, RC receiver, and telemetry	2,800
	Power batteries	1,720
Mission electronics	Onboard computer	460
	Camera	170
	Wireless link	280
	Electronics battery	300
	Laser altimeter	40
Pickup mechanism	Carbon fiber lattice, plastic joints	210
	Electromagnetic device, plastic part, switch	160

Note. The total weight of 6.140 kg is distributed as 4.520 kg for the airframe, 1.250 kg for the mission electronics, and 370 kg for the pickup mechanism.

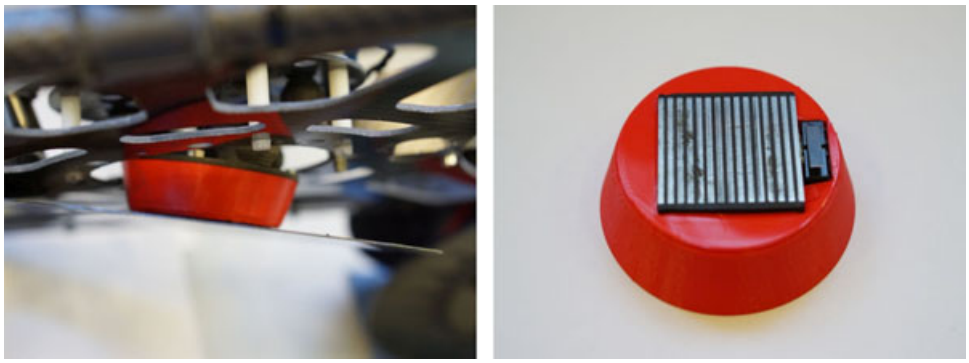

FIGURE 6 Prototype of the pickup mechanism. Left, detail of the damped platform. The EPM is grabbing a metallic disk and it has a certain degree of flexibility. Right, the holder with a single EPM and a contact sensor. EPM: electro permanent magnet [Color figure can be viewed at wileyonlinelibrary.com]

Opengrab EPM v3.[10] The EPM produces an external magnetic field that can be switched on or off by a pulse of electric current.[11] There are two sections of magnetic material and the electric current through a wire winding around one of them makes both sections be polarized in the same direction, creating the magnetic field. In particular, the device has a pulse width modulation (PWM) input, which lets us control the magnet status: an ON command results in a full magnetization; whereas an OFF command switches off the magnetic field by not magnetizing both sections in the same direction.

We tested different approaches to achieve a trustworthy device, all of them mounted on a CF lattice. This lattice is placed at the bottom of the hexacopter and it offers two different functionalities: first, it gives the aircraft a larger and solid base to land; and second, it centers the pickup device. In the first designs, the success rate picking up pieces was very low, and we found out that the problem was related to the contact surface. The rigid mounting for the EPMs made the aircraft require perfect flat contacts to pick up pieces. Any angle between the EPM surface and the contact surface of the pieces resulted in a failure. Therefore, we created a final prototype where a single EPM is mounted on a damped platform. Thus, the mechanism is not rigid but flexible, allowing the EPM to make a stronger contact with its whole surface and leading to a more stable grip.

Additionally, the lattice is made of CF to reduce the final weight of the mechanism. Attached to the lattice, there is a flexible platform with plastic dampers, where the magnet holder is mounted. Figure 6 shows the pickup mechanism with the damped platform and the EPM holder, which contains a contact sensor. That sensor provides readings to detect whether a piece is being transported and it includes a low-pass filter to avoid false positives.

The final design increased the success ratio to almost 100%, avoiding pieces from falling down during the flight. Although we obtained good results, a new issue appeared sending the ON/OFF commands with the Pixhawk. It seems that the PX4 firmware modifications (to control the auxiliary port) affect the way that Pixhawk manages the output mixer. Therefore, we experienced nonstable behaviors when issuing commands to the EPM, and we decided to design our own electronic interface.

This electronic interface receives commands from a serial port (e.g., from the Intel NUC). Then, it sends the ON/OFF commands to the EPM (using a PWM signal), controls activation/deactivation times and reads the contact sensor. The electronics are based on a dsPIC33FJ32GP302 microcontroller and its firmware is written in MPLAB C30.

5 | SOFTWARE FUNCTIONALITIES

This section explains the software architecture of our system and gives details about the techniques that are used within each of the modules of our overall approach in Section 3. These techniques provide the different functionalities that are required so that the whole system can address the MBZIRC Challenge.

5.1 | Vision-based object detection

The Vision Module runs on board each UAV and it is in charge of processing the images taken by the camera to detect the Challenge objects. An approach based on color segmentation and clustering is used to estimate object positions and other features on the image plane. Then, those measurements are integrated with the camera pose to produce object positions in the 3D space. In the following sections, we describe the techniques and steps to obtain the final detected objects from each image.

5.1.1 | Color segmentation

To extract objects from the scene, we first divide the color space into several clusters and classify each pixel in the image frame within those clusters. Even though RGB (red, green, and blue) is the most common color space, dividing it into clusters representing colors is not straightforward. Instead of using RGB, we use the HSV (hue, saturation, and value) color space (see Figure 7). HSV gathers most color

[10]https://kb.zubax.com/display/MAINKB/OpenGrab+EPM+v3.

[11]The EPM consumes 50 mW in steady mode and a peak of several watts during microseconds when it commutes.

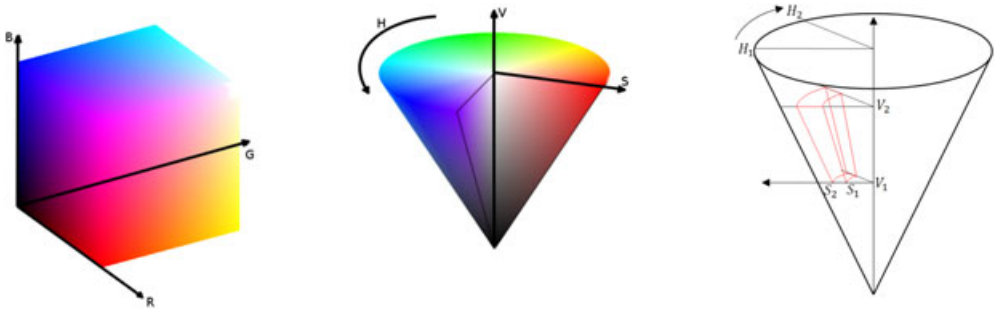

FIGURE 7 Left, color space in HSV and RGB channels. Right, division of the HSV color space into clusters. For instance, the cluster depicted in red can represent a blue color in a simple manner. However, in RGB, that color has to be defined by a region separated with a tilted plane [Color figure can be viewed at wileyonlinelibrary.com]

information in the Hue channel, and hence defining clusters for the colors is easier. In particular, the color space is divided into n_c clusters and each cluster is limited by six values as shown in Figure 7 (right).

When finding out the color cluster for a pixel, several conditional checks in cascade with the corresponding thresholds could be done. However, this procedure is not efficient because it may perform six conditional checks at runtime, which are not well vectorized by a CPU. Instead, we use an optimized implementation of the algorithm in Bruce et al. (2000), which consists of boolean thresholds defined at compile-time. We enhanced the algorithm by implementing a parallelized version, which reduces considerably the execution runtime.

First, each channel dimension is divided into n_h, n_s, and n_v discrete values, respectively. Hence, the values of a pixel (h_i, s_i, v_i) can be remapped into the discretized space as (h_i^d, s_i^d, v_i^d). Second, for each channel, the arrays A^h, A^s, and A^v, of sizes n_h, n_s, and n_v, respectively, are built. Each element of each array must store n_c bits, indicating each bit whether the corresponding discrete value of the channel belongs "1" or not "0" to that cluster. To check to which cluster a pixel belongs, its values (h_i^d, s_i^d, v_i^d) are used as indexes of the three arrays. Then, two bitwise comparisons are done to find out whether it belongs or not to each of the clusters: $A^h[h_i^d]$ & $A^s[s_i^d]$ & $A^v[v_i^d]$.

The significant advantage of this approach is that it can evaluate the belonging of a pixel to multiples color clusters simultaneously thanks to the parallelism of the bitwise operator. However, it is not adaptable at runtime. In the MBZIRC, we used $n_c = 5$, since the system only needs to distinguish between five colors. Figure 8 shows an example of a segmented image.

5.1.2 | Run-length encoding (RLE)

RLE is a simple form of data compression in which every group or run of data (i.e., a sequence of consecutive data with the same value) is compressed as a pair with the data value and the count. For instance, the data *WWWWWWWWBBBBBBBBBCCCCCCWWWWWWWWWWWW WWWW* would be compressed with RLE into 4 value/count pairs *W7B9C6W16*. We use RLE to reduce the image sizes, gathering groups of colors together. This kind of data compression is quite effective if the image color is homogeneous, which is the case in our images.

Once every row of the image is encoded with RLE, it is necessary to connect the runs that belong to the same object. This process is depicted in Figure 9. First, it goes from the top-down searching in consecutive lines for overlapping runs of the same color. If an overlap is detected, the upper run is assigned as parent of the lower run (see Figure 9a–c). Then, a second phase starts at the bottom row and goes up searching for disjointed objects. It checks whether adjacent runs of the same color have different parents (see Figure 9d).

5.1.3 | Parallel optimization

The previous color segmentation algorithm is computationally efficient as it optimizes the pixel-wise color classification. However, UAVs usually carry onboard computers, which have lower computational

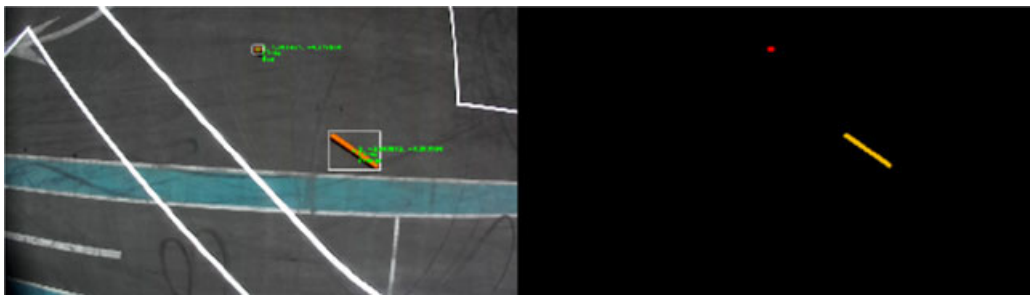

FIGURE 8 Output of the vision-based object detector. Left, original image from the arena with a red and an orange object. Bounding boxes and a text displaying extra information overlay the image. Right, processed image after the color segmentation. The two objects appear segmented and the background as black [Color figure can be viewed at wileyonlinelibrary.com]

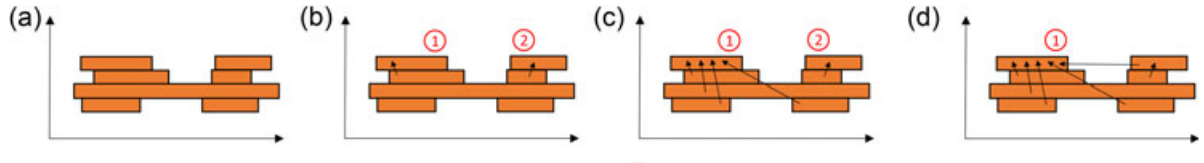

FIGURE 9 Steps of the algorithm to process an RLE image and connect runs that belong to the same object. Several runs of an orange object are depicted as example. All the runs are grouped together and they can be processed to extract the object information. (a) All runs start disconnected, (b) top-down phase connects adjacent runs, (c) Runs 1 and 2 become parents, (d) bottom-up phase makes 1 the single parent [Color figure can be viewed at wileyonlinelibrary.com]

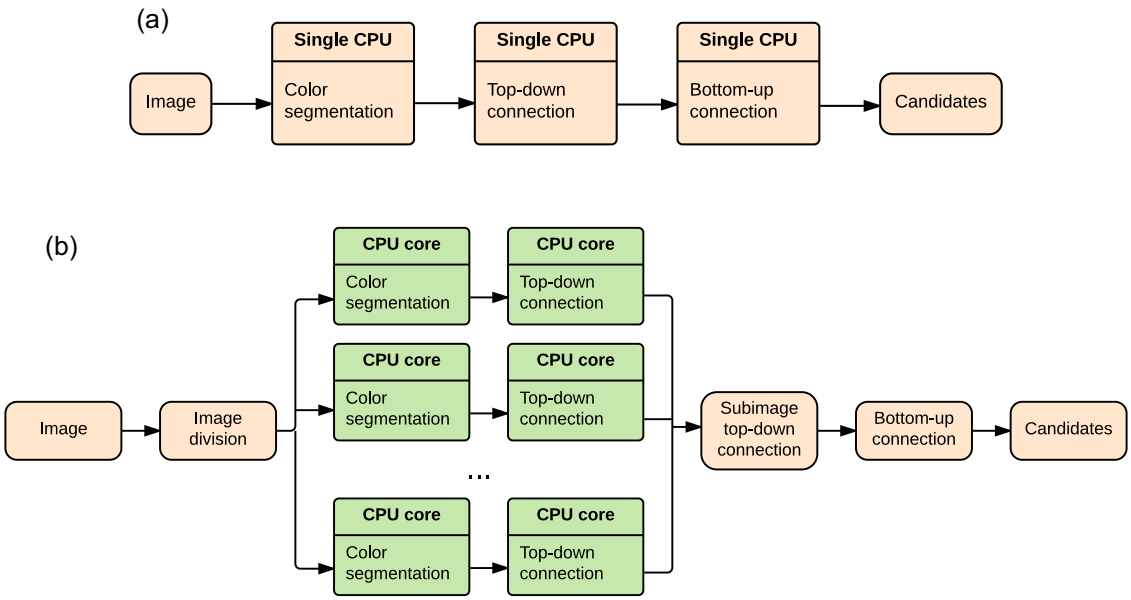

FIGURE 10 Block diagram for the complete vision-based object detector. An optimization of the algorithm is implemented by means of a parallelized version. (a) Nonparallel implementation and (b) parallel implementation [Color figure can be viewed at wileyonlinelibrary.com]

capabilities than common desktop computers. Thus, we contribute in this study with a new version of the algorithm for object detection, parallelizing its computation in the CPU. Figure 10 highlights the differences between the nonparallel and the parallel implementation of the algorithm. Basically, each image is split into fragments that are processed in parallel by a different thread each. Each thread performs pixel-wise color segmentation (as described in Section 5.1.1) in its fragment, and then, it compresses the fragment using RLE and carries out the top-down phase of Section 5.1.2. Finally, a single process synchronizes all the threads by fusing the results for each fragment. This is done by performing a new top-down phase that checks data from neighboring fragments, and the final bottom-up phase.

5.1.4 | Estimation of the 3D object positions

After all the candidates are obtained from the image, there is a final step to filter them out and compute their 3D positions in a global coordinate system. For this purpose, the pose of the camera is used together with the assumption that all objects lie on the ground. The Vision Module reads the UAV pose computed by the onboard localization sensors and applies a known transformation to obtain the camera pose. Then, a simple homography transformation is used to project the position of the candidates on the image plane onto the ground of a 3D coordinate system. Since we only need to estimate 2D positions on the ground (height is fixed for all objects), the Vision Module outputs for each candidate its observed color c_o, a vector $\mathbf{z}$ with its 2D position in global coordinates and an estimation of the error covariance matrix $\mathbf{R}$.[12] Moreover, as object sizes are known, the estimated size of each candidate can be compared with the actual ones, filtering out false positive detections, and improving the robustness of the results.

5.2 | Multi-UAV object estimation

The object estimator is in charge of implementing this functionality, which allows the system to estimate and track the positions of the objects detected in the arena. The objective is to keep a track for each new object detected, with all its information associated. These estimations are used by the planning module to assign objects to UAVs, which should navigate to the estimated positions and collect the objects.

We propose a centralized stochastic filter running on the GCS and receiving observations from all the Vision Modules in the team, and integrating them into a single data structure. The fact

[12]This parameter was fixed in our system and its adjustment will be discussed later.

that there are only three UAVs operating in a relatively short range, makes advisable to use this approach instead of a decentralized estimator. The communication bandwidth to send all the observations to the GCS is not critical, and at the same time, possible issues with inter-robot transmissions and delays are avoided. Therefore, we selected a centralized multitrack filter due to its simplicity and efficiency.

The object estimator maintains a belief over the pose and color of each object in the arena (i.e., a track). Anytime a new object is detected by any of the UAVs, this is communicated to the estimator, and a new track for that object is created. For each track, the state is composed of several factored variables associated with the object (x, y, v_x, v_y, c). The 2D position and velocity of the object are continuous variables, whereas the color is a discrete variable $c \in \{\text{red, blue, green, yellow, orange}\}$. The former is updated by means of a KF, and the latter with a discrete Bayes Filter.

We use a KF to estimate object positions because it is simpler for data fusion from different sources and less computationally costly than a Particle Filter. The main problem is that we cannot deal with multimodal distributions, but we alleviate that issue by reducing the integration of false positive observations into the filter. For that, we develop a technique to solve the data association problem between the observed objects and the current tracks, combining both color and distance information. The following sections give more details about the different probabilistic models used for the prediction step, the update step and data association.

5.2.1 | Initialization

As stated above, the initialization of a track occurs whenever a new object is detected by the team and must be incorporated into the filter. This happens when an observation received from some UAV is not associated with any previous track. Hence, a new track is created, initializing the position and velocity according to the observation received from the UAV and the color to a uniform probability distribution. Then, the color variable is updated with the information contained within the UAV observation, increasing the probability for the value of the observed color c_o.

5.2.2 | Prediction

A KF is used to maintain the belief over the position and velocity of the object. Therefore, if $\mathbf{x} = (x, y, v_x, v_y)^T$ is the state vector and $\mathbf{\Sigma}$ the covariance matrix, a linear kinematic model is used to predict this state from one time step to another separated a time interval Δt. The prediction and noise matrices are the following:

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_v^2 \Delta_t^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \Delta_t^2 \end{pmatrix}. \tag{1}$$

The parameter σ_v indicates the level of noise for the object velocity; the higher, the more the uncertainty grows after a prediction.[13] The belief color factor is never predicted, since the color is fixed for all the objects. Moreover, the KF is not predicted for the static obstacles, only for the moving ones. This is determined by means of the color: An object whose probability of being yellow is higher than its probability of not being yellow is labeled as a moving obstacle. Otherwise, the object is considered static. The same reasoning is applied to the probability of being orange to label each object as large or small.

5.2.3 | Update

If a UAV observes an object and this observation gets associated with a specific track, the belief of that track must be updated with this new information. The observation coming from the Vision Module consists of a 2D position in global coordinates of the object $\mathbf{z}$ and its corresponding covariance matrix $\mathbf{R}$; and an observed color c_o. To incorporate the position information, the KF is updated with a simple linear model:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{2}$$

The color belief is also updated with the observation c_o by means of the equations of a standard Bayes Filter:

$$p(c_t = i) = \eta \cdot p(c_o | c_t = i) \cdot p(c_t = i), \forall\, i, \tag{3}$$

where η is a normalizing constant and $p(c_o | c_t = i)$ is the probability of observing c_o given a color value $c_t = i$. We estimated empirically that the probability of detecting the actual color of an object with our vision algorithm was of 0.9, which is the value that we used to compute the previous probability. If the vision algorithm provides no information about the color because it could not be observed with enough certainty, the color belief is not updated.

Finally, the belief is only updated with recent observations. If an observation that is too old reaches the filter, it is discarded. In this way, we avoid spoiling the estimations with observations that were delayed too long due to network communication issues. This value was adjusted during the experimental trials after evaluating communication delays.

5.2.4 | Data association

A data association problem must be solved when new object observations arrive at the object estimator. Multiple tracks are maintained and it needs to be determined to which track the observations correspond, or whether new tracks should be created. We define a couple of heuristics based on probability to measure

[13] We set this value to $\sigma_v^2 = 0.2\,\text{m}^2/\text{s}^2$ and checked during the experimental trials that was reasonable for the moving objects.

how close an observation is to the current estimation of each track. Then, those heuristics are used to create associations. First, we define a probabilistic distance of the observed position:

$$d_p = \sqrt{(\mathbf{z} - \mathbf{Hx})^T \mathbf{S}^{-1} (\mathbf{z} - \mathbf{Hx})}. \quad (4)$$

Given a track with position belief ($\mathbf{x}$, $\mathbf{\Sigma}$) (mean and covariance matrix), the heuristic in Equation (4) measures the Mahalanobis distance between an observed object position $\mathbf{z}$ and the probability distribution of the predicted observation. With the current belief, the probability distribution of the predicted observations can be computed by projecting ($\mathbf{x}$, $\mathbf{\Sigma}$) into the observation space, that is, the probability distribution of observations would have mean and covariance matrix ($\mathbf{Hx}$, $\mathbf{S}$). The lower d_p, the higher the probability that the observation corresponds to that track.

To take into account the information about the color, we also compute the probability of the color observation c_o for each track:

$$p(c = c_o) = \sum_i p(c_o | c = i) \cdot p(c = i). \quad (5)$$

At each iteration of the estimator, the set of received observations is processed to associate them with the existing tracks. First, the heuristic d_p is computed for all possible pairs observation/track. Then, the best pair with minimum distance value is selected for association. If $d_p <= d_{th}$, the observation is likely enough[14] and the track is updated with that observation. Otherwise, the observation is not close enough to any of the existing tracks, so a new one is created and initialized with that observation. The same procedure is repeated until there are no more observations to associate. Note that more than one observation could be associated with the same track, since those may be observations of the same object coming from different UAVs. Finally, when a best pair is selected but the probability of its color observation is too low (Equation (5)), the association is discarded. We experimented with our color detection algorithm and estimated that this happened when $p(c = c_o) < 0.15$.

5.2.5 | Additional information

Besides the belief over the position and the color of each object, the estimator keeps additional information useful for other modules. First, for each object (track) a unique identifier is stored. This is useful for logging and visualization, and also to assign them to different UAVs. Second, each object has a status within the tuple {UNASSIGNED,ASSIGNED, CAUGHT, DEPLOYED, LOST,FAILED}. ASSIGNED and UNASSIGNED indicate whether the object has a UAV assigned to be picked up or not, respectively; CAUGHT means that the object has been picked up successfully; DEPLOYED that the object has been transported and dropped; an object is LOST when a UAV goes to pick it up and cannot find it; and an object is set to FAILED when a UAV goes to pick it up and the action is aborted after failing.

The Cooperative Planner and the UAV state machine use this status to keep a track of the objects' situation, and they are the ones in charge of modifying the values, as it will be explained in the next section. Furthermore, CAUGHT and DEPLOYED objects are not considered by the estimator for prediction nor update; whereas LOST objects are removed from the filter.

Finally, it is relevant to mention that the estimator also removes objects that are not observed for two long or were observed spuriously. After the experimental trials, we determined that an object that had been detected in less than five frames and had not been detected for 20 s, was a false positive and had to be removed. We adjusted those values during the trials not to have many spurious objects and to focus on those detections more likely to be real.

5.3 | Cooperative planning

The cooperative planner is in charge of planning paths and actions for the UAVs in a coordinated manner. It consists of a centralized module that runs on the GCS and that receives the current position from each UAV and the object estimations from the object estimator module. Then, it allocates different objects to the UAVs, that should go to their estimated positions, pick them up and drop them back into the dropping box.

Due to the fact that the arena can be covered fairly fast with the UAVs, and to avoid too many conflicts between the different vehicles collecting objects, we proposed a novel cooperative strategy with two phases. With this algorithm, objects are allocated to UAVs as tasks heuristically and the potential conflicts are minimized. First, the UAVs fly covering nonoverlapping zones of the whole arena and searching for the maximum number of objects. Second, once this search has ended, the planner starts to assign the UAVs objects that they must collect and drop.

During the search phase, the arena is divided into three longitudinal sectors, and each of them is covered by a different UAV with a straight-line path (return trip). Figure 11 depicts an example of the division and the paths followed, which can be computed geometrically so that all the segments are equally distributed. Also, in case that there were only two UAVs available (e.g., because one of them failed), the scenario would be split into two equal sectors to be covered in the same fashion as before. Note that the UAVs fly at the same height during this search phase, since their paths are nonoverlapping and no conflicts need to be solved.[15]

After the search phase, the UAVs should have a good estimation of most object locations. Then, a collecting phase starts; where the cooperative planner assigns them different objects to collect and drop. These assignments are asynchronous, that is, anytime a UAV is idle, it asks for a new task (object) and the planner decides the best

[14]The threshold d_{th} is a parameter to adjust how flexible the associations are. Its value will be discussed later.

[15]We flew with a height of 10 m during our trials, since we tested that that height was adequate for detecting most objects with our vision algorithm and covering a third of the arena.

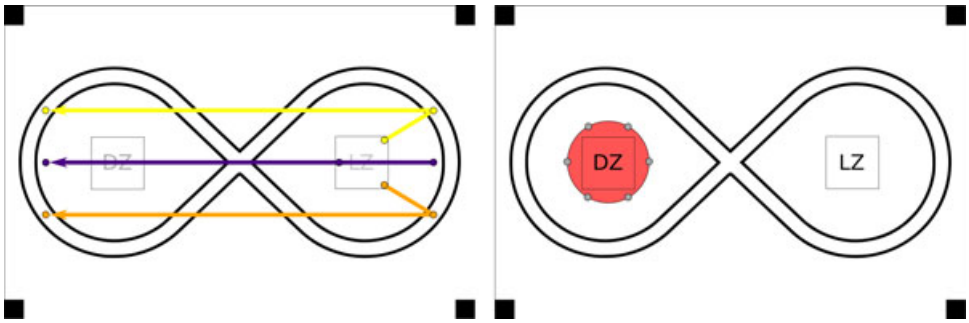

FIGURE 11 Scheme of the arena for the MBZIRC: DZ represents the dropping zone, where the box is placed; LZ represents the landing zone, where UAVs start the mission. Left, an example of the paths followed by three UAVs during the search phase to cover the whole arena (they go and return to the start position). Right, in red the roundabout around the DZ with the six waiting spots for the UAVs loaded with objects [Color figure can be viewed at wileyonlinelibrary.com]

one to be picked up, given the current situation. During this phase, a different height is assigned to each UAV for navigation, so that they can traverse the arena without colliding with each other.[16] Note that the vision modules and the object estimator are still running during the second phase, so new objects could be detected (or information from previous ones updated) as the UAVs navigate collecting and dropping objects.

When the planner needs to assign an object to a UAV, it follows several rules. First, it only considers objects that are not assigned to another UAV. Second, it prioritizes according to the color and assigns first those unassigned with the color of highest priority. In particular, we focused first on the static, small ones (the higher its score, the higher its priority); then on the moving ones (yellow); and finally on the large ones (orange). Given our aerial platforms, we estimated that scale of difficulty and decided to go from easier to harder. Last, to discriminate between obstacles with the same priority, the planner rates them with a heuristic based on distance, opting for the closest one. We also tested another heuristic weighting object scores and distances, but it turned out to be more effective the priority rule, that is, to pick up always the easiest ones first.

Although each UAV flies at a different horizontal plane while collecting objects, they may still collide when one of them is descending to pick up an object, since it could traverse others' planes. To minimize those situations, when assigning an object i to a UAV 1, the planner checks whether the straight line from that UAV 1 to the object i lies too close (distance measured on the horizontal plane) from any other object j assigned to some other UAV 2. In that case, this assignment is discarded because the UAV 1 could cause a potential conflict while the UAV 2 is descending to pick up its object j. Nonetheless, note that some conflictive situations may still arise, but their probability is significantly reduced. Indeed, being conservative and discarding assignments whose corresponding paths passed closer than 5 m to other assigned objects, we did not come across any conflict during all our experiments.

[16]We selected heights of 3, 7, and 11 m for the three UAVs during our trials, since we tested that those were still adequate to detect objects and keep a safe distance between the UAVs.

Finally, another source of conflict must be taken into account, the dropping zone. We solve this issue by treating that zone as a centralized shared resource where only one UAV can enter at a time. When a UAV enters that zone, it takes a token that needs to be freed before someone else uses it. Thus, as shown in Figure 11, an imaginary roundabout with six waiting positions is designed around the dropping zone. Any time a UAV has picked up an object and needs to drop it, it asks the cooperative planner for a waiting spot. The planner will assign to the UAV the closest spot not already assigned to another UAV also dropping. Then, the UAV will navigate there and wait until the token of the dropping zone is free. Note that UAVs navigating across the dropping zone toward their assigned objects (before picking them up) would still be conflictive, but this situation is highly unlikely since the dropping zone is placed in one of the extremes of the arena with no much space behind.

5.4 | UAV state machine

Each UAV runs a state machine on board that deals with all the tasks assigned by the cooperative planner. The UAV state machine is depicted in Figure 12 and it governs the UAV behavior by issuing commands through the UAL. State transitions may be triggered by the completion of UAL commands, by service calls from the planner, or by other external events (e.g., activation of the contact sensor in the pickup mechanism).

The state machine starts in REPOSE and it waits until the planner begins the mission by calling a service to take off the UAV. Then, it switches to TAKING_OFF and issues a TAKE_OFF command through the UAL with the corresponding height for the search phase (z_searching). When the take-off is completed, the UAV goes directly to the SEARCHING state, where it is issued a GOTO_WP command. This UAL command navigates the UAV to a single waypoint or through a list of waypoints. The cooperative planner indicates the specific search path for each UAV (search_path), as explained in Section 5.3. After finishing the path, the UAV goes to an idle state called HOVERING.

As explained in Section 5.3, the planner assigns asynchronously objects to the UAVs as they become idle and ask for new tasks. Hence,

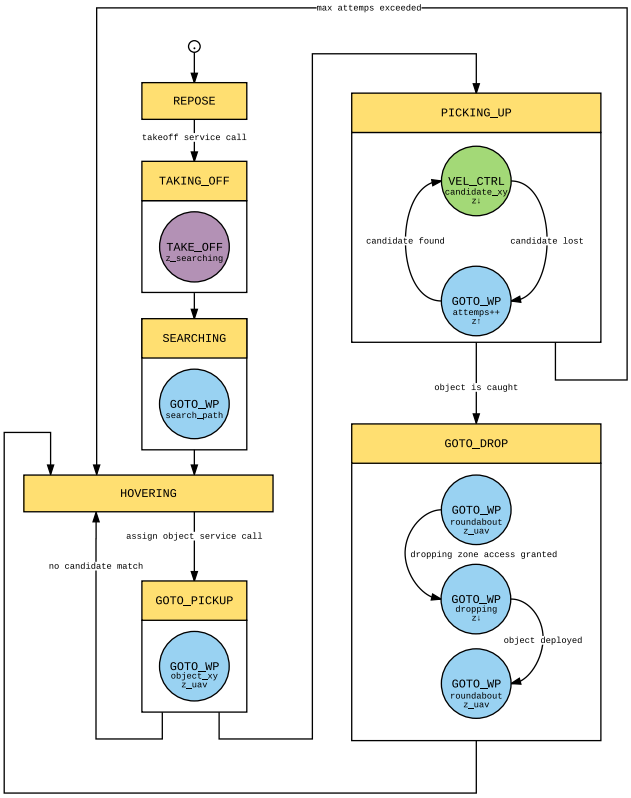

FIGURE 12 Diagram of the UAV state machine. States are represented by rectangles and transitions by arrows. The circles represent UAL commands to the UAV with specific parameters [Color figure can be viewed at wileyonlinelibrary.com]

any time a UAV is HOVERING, the planner selects the best object to collect (if there is any) and calls a service of the UAV state machine indicating information about that object. In the GOTO_PICKUP state, the state machine uses the object position (object_xy) to send the UAV there with a GOTO_WP command. Note that the navigation height z_uav during this phase varies from one UAV to another.

Once arrived at the object position, a pickup operation is attempted. The candidates generated by the vision module are explored to find one that matches the color of the assigned object. The best match is selected, that is, the closest one with the same color as the assigned object and inside the arena (see the discussion about geofencing in Section 6.4). If a match is found, the state machine transitions to PICKING_UP. Otherwise, it goes back to HOVERING and that assigned object is set to LOST.

In the PICKING_UP state, the UAL command VEL_CTRL is activated. This command controls the UAV in velocity (horizontally) to center the candidate position candidate_xy on the image, at the same time that the UAV descends to get closer. This visual servoing is based on a PID controller that works with local position errors, since global object positions are not accurate enough. Thus, the object position on the image (candidate_xy) with respect to the image center is used to center the UAV by means of horizontal movements. Different values for the controller gains are used to pick up static or moving objects. We tuned those values empirically to achieve a more aggressive control with the dynamic objects.

As the UAV descends, the corresponding candidate may be lost by the vision module. In that case, the UAV ascends back up to a maximum height or until the object is detected again (GOTO_WP command). The same procedure is repeated up to a maximum number of attempts, after which the object is set to FAILED and the UAV returns to HOVERING. Since FAILED objects are not considered again for assignment, in case there were not remaining objects, the object estimator would reset the ones FAILED to UNASSIGNED to attempt them over again. On the contrary, if the contact sensor of the pickup mechanism is activated, the object is set to CAUGHT and the state machine switches to GOTO_DROP.

In the GOTO_DROP state, the UAV goes back to its navigation height and asks for the closest free waiting spot in the roundabout (GOTO_WP command). Once arrived, it waits until the dropping zone is free, then it enters, descends to a dropping altitude, drops down the object and sets it to DEPLOYED. Afterward, the UAV returns to its original position at the roundabout at its navigation height and transitions back to HOVERING.

6 | EVALUATION AND LESSONS LEARNED

This section analyzes the performance of our system and the lessons that we learned during the MBZIRC competition. The system evaluation includes experimental results that we obtained during the development phase and results from the actual competition. We learned some lessons during the whole development process previous to the competition and during the actual competition, where we had to adjust parameters for our systems and even change the original design of some of them.

6.1 | Aerial platform design

During the competition in Abu Dhabi, we discovered that there were two kinds of approaches for the aerial platforms: some teams designed UAVs of similar size to ours; while others used smaller and lighter UAVs. These lighter platforms present advantages in terms of maneuverability and stability, but they were not able to pick up the large objects. However, there were two relevant facts during the competition that affected significantly the payload requirements. First, even though the weight of the small objects was originally specified as “less than 500 g,” the actual weight of those used in the competition was 350 g. Second, the organization allowed all teams to change batteries during the trials without penalty. Thus, it was feasible to fly with smaller batteries, having more payload available for the objects.

Given the above premises, we would have probably used a different platform. Actually, we also performed some experiments in Seville to test our software and pickup mechanism with the smaller and lighter DJI F550 airframe, as shown in Figure 13. Although this aircraft was more controllable and stable at low altitude, we originally considered that it would not offer enough flight time and payload. Nonetheless, the performance of our aerial platforms

FIGURE 13 Preliminary tests in Seville (Spain). A DJI F550 aerial platform with our pickup mechanism transporting a red object (top) and our final custom-made hexacopter transporting a blue object (bottom) [Color figure can be viewed at wileyonlinelibrary.com]

during the competition was excellent in terms of endurance. They behaved as expected according to the original design, being able to fly during 20 min and to perform complete missions. Hence, we never had to change the batteries during any of the trials.

Finally, we discarded more precise localization devices for the UAVs, such as an RTK GPS due to their price. Those devices provide more accuracy in the measurements and increase clearly the stability of the aircraft. However, we experienced that the level of precision provided by our autopilot (it achieved errors below 2 m by filtering GPS and IMU measurements) was enough to perform the missions, since we were using visual servoing with local coordinates to pick up the objects.

6.2 | Pickup mechanism

In our first rehearsal trials in Abu Dhabi, we experienced issues with our pickup mechanism described in Section 4.1, since the UAVs were not able to grab any of the competition objects. The issue was related to the layer of color paint that the official objects had. Our previous and successful tests in Seville were with similar mock-up metallic objects since the organization did not send instances of the competition objects. There were other teams experiencing the same critical issue, but instead of quitting the competition, we improvised a new design on site.

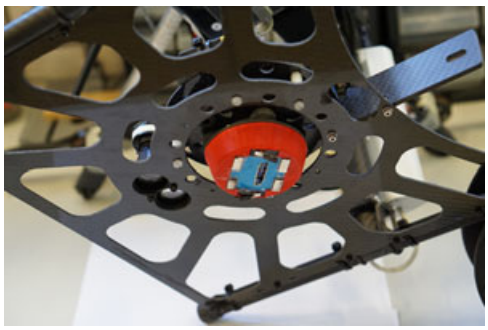

FIGURE 14 Final version of the pickup mechanism. In the middle of the carbon lattice, the red holder is mounted on the damped structure. The holder has four magnets on top, two contact sensors, and a lever in the middle actuated by a servomechanism [Color figure can be viewed at wileyonlinelibrary.com]

In particular, we used the same holder but replaced the EPM with an array of four permanent magnets, which had enough power to pick up the objects. We also included a radio-control servomechanism with a lever that was used to release the objects. Moreover, we mounted two contact sensors to detect the pieces. Figure 14 shows the new design of the whole mechanism. The contact sensors were in charge of confirming that an object had been caught, and the release action triggered the movement of the lever to push the object downwards. We tested our new device during the rehearsal trials successfully and were able to pick up eventually the official objects.

6.3 | System architecture and integration

We integrated all the modules of our architecture with the open-source Robotics Operating System middleware,[17] in particular, its version ROS Kinetic Kame. We also developed a simulated version of the MBZIRC arena and our aerial platforms based on the robotics simulator Gazebo.[18] A Software-In-The-Loop (SITL) scheme was used to integrate the actual software of our autopilot into the simulation, what allowed us to perform quite realistic simulations. Our platforms use PX4 as autopilot software, so we used an SITL module of the PX4 for Gazebo (Furrer, Burri, Achtelik, and Siegwart, 2016) integrated into our software architecture. This allowed us to implement a Gazebo model for our aerial platforms, which run the same software as the actual autopilot. The camera to feed the Vision Module and the pickup devices were also integrated into the simulation by means of Gazebo plug-ins. This simulation is not very reliable in terms of flight control, as we did not invest time identifying a dynamic model of the platform, but it is definitely accurate with respect to the autopilot behavior.

[17] http://www.ros.org.

[18] http://gazebosim.org.

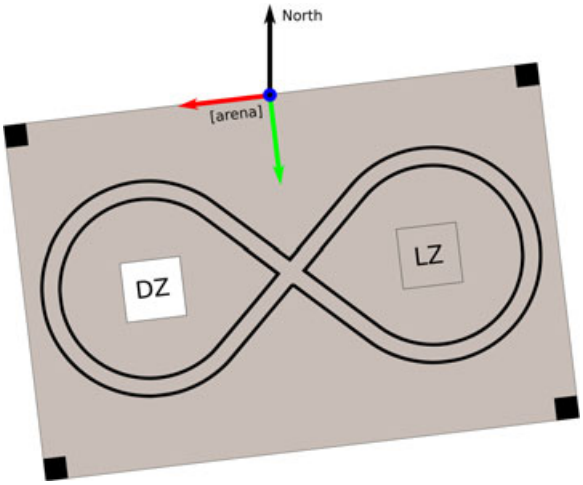

FIGURE 15 The MBZIRC arena with the [arena] coordinate system. It can be seen that the coordinates are aligned with the arena and can have a rotation with respect to the north. In gray, the valid area for the geofencing tool is also shown. Objects out of that area were not considered for estimation nor collection [Color figure can be viewed at wileyonlinelibrary.com]

Additionally, on top of the autopilot software, we developed our abstraction layer UAL based on ROS, what helped us to simplify and unify the commands to control the UAV. This UAL is publicly available at GitHub[19] and offers a simple interface with commands like *take-off*, *land*, and *go to waypoint*. On the other side, the abstraction layer has a back-end, which is in charge of the communication with the autopilot. For the MBZIRC, the communication with the PX4 was performed through MAVROS,[20] which is the ROS version of the *MAVlink* protocol.

In general, the UAL and the SITL simulator proved to be quite relevant for the integration of the whole system. The ability to simulate complete multi-UAV missions became a remarkable feature to test and debug the interfaces and functionalities of all the modules. Indeed, the whole system could not distinguish simulation from real flight behavior. Only the gains of the low-level controllers for the aerial platforms needed an additional adjustment to jump into experiments with the actual systems.

6.4 | System coordinates and geofencing

The autopilot provides the location of each UAV in global geodesic coordinates (latitude and longitude) and in local coordinates (in meters) with the origin in the place where the autopilot is booted and the axis aligned with the north. Since we have multiple UAVs which cooperate, we need a common coordinate system. Geodesic coordinates are global and seem to be an obvious option. However, our platforms have not the global precision of an RTK-GPS and the PX4 implements an enhancing filter (based on GPS readings) to estimate its pose on its own local coordinate system. Therefore, we preferred those local coordinates rather than the global ones, due to their accuracy and stability.

[19]https://github.com/grvcTeam/grvc-ual.

[20]http://wiki.ros.org/mavros.

We defined a global coordinate system called [arena] (see Figure 15) relative to the scenario map and we learned that specifying coordinates for the high-level modules in this common system avoided many issues. With this new system, we could also maintain the same configuration (in terms of UAV waypoints) for every arena (there were two). The only requirements to transform between the local geodesic coordinates of the UAVs and the [arena] system were to know the start UAV positions (we placed them in known points of the landing zone, e.g., the squares); and the arena rotation with respect to the north, because the local coordinates are defined in ENU (East-North-Up). Furthermore, we designed our UAL to deal with different coordinate systems, abstracting the end-user from that.

Besides defining different coordinate systems and managing them transparently, we implemented a geofencing tool. This tool is in charge of checking whether a hypothetical object is within the physical limits of the arena (see Figure 15). We discovered during the competition that this was quite relevant for safety reasons, to prevent the UAVs from attempting to pick up things out of the arena (i.e., false positives), or inside the dropping zone (i.e., dropped objects), where trying to catch an object could interfere with other UAV dropping. The geofencing tool solved the above issues in a simple fashion, double-checking object positions before creating them in the estimator. Also, objects not holding the geofencing constraints were not considered by the UAVs during pickup operations.

6.5 | Communication and network configuration

The network configuration and devices turned out to be critical for the competition. Although we had tested the wireless communication devices on board our aerial platforms extensively in the experiments previous to the competition, we experienced many communication issues during the trials in Abu Dhabi. We discovered eventually that it was a problem with the setup of our wireless links on board the UAVs and we solved our connectivity issues by updating the firmware of the Ubiquiti Rocket devices.

In addition, since the system is distributed and there are processes running on the UAVs and on the GCS, time synchronization is essential, especially for the algorithms of data fusion. Delays in the network communications led to situations where the object estimator discarded many observations for being too old or where those estimations were inconsistent. Therefore, we solved this issue by using the network time protocol (NTP) and a server configured on the GCS. NTP allows timing information to be distributed in local area networks with errors below one millisecond, which satisfies the time constraints of our distributed architecture.

6.6 | Vision module

The vision detector turned out to be a critical module for the execution of the mission. It feeds the object estimator to compute object positions and colors, but it is also used to control the UAV in velocity when it is picking up an object. An important parameter is

the resolution of the color space discretization, that is, n_h, n_s, and n_v. A coarse division reduces the sizes of the arrays in memory but may be insufficient for the correct segmentation of colors that occupy a small volume in HSV. For instance, yellow is quite thinner than blue (see Figure 7). We adjusted this resolution empirically and set all the array sizes to 36. That allowed us to segment images with a precision in each channel of 1/36 (i.e., 10° for the Hue channel). Theoretically, there is a limit for that resolution, which is determined by the color (from those that need to be detected) with smallest volume in the color space. In the case of the MBZIRC, the most critical colors with smallest volumes were yellow and orange. Increasing the resolution helps to divide the color space more accurately and enables the detection of more colors, but it increases slightly the computational cost of clustering.

Another relevant feature of the vision module is its frame-rate since we use it to feed the UAV controller while picking up objects. We evaluated the algorithm speed with two different image resolutions (available for our onboard cameras) and with/without the parallel optimization. Figure 16 shows the average processing time per frame. A significant difference can be observed varying image resolutions.

The selection of the image resolution for the cameras was done considering the frame-rate requirements and the accuracy to detect objects. During the search phase, each UAV should cover a third of the arena, which means that they should fly with an altitude of around 10 m (given the camera field of view). At this height, objects may appear too tiny on the images, so we chose a resolution of 1,280 × 720 to ensure that the objects were not of the size of noise. Furthermore, we achieved frame-rates faster than 20 FPS for that resolution, which was sufficient for the UAV controller.

In our trials in Abu Dhabi, we set the parameters as explained above and calibrated the system with the lighting conditions there. Then, we achieved positive results in a repetitive fashion in terms of object detections. Table 2 summarizes the results of the vision module over five different trials of the challenge. Most objects were detected correctly, true positives (TP). In these experiments, there were only a couple of false negatives (FN), caused by a yellow object that was missed during the searching phase due to the sunlight reflection. However, that object was later detected after the searching phase, with a UAV flying at a lower altitude. Regarding the false positives (FP), they were caused by participant T-shirts and a blue fence near the arena (geofencing was applied to discard most of them).

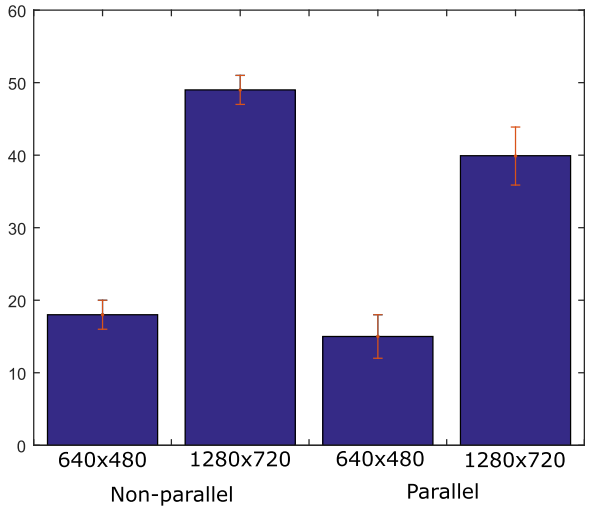

FIGURE 16 Average processing time per frame (milliseconds) of the vision module. Different image resolutions and parallelization options are compared [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Results of object detections over five different MBZIRC trials

	Set #1	Set #2	Set #3	Set #4	Set #5
Number of objects	6	10	7	9	8
TP	5	10	7	9	7
FP	0	1	1	1	2
FN	1	0	0	0	1

Note. The actual number of objects that appear throughout each trial is compared with the number of detections and misdetections.

6.7 | Multi-UAV object estimation and allocation

We integrated and tested successfully and repetitively our object estimator in the trials in Abu Dhabi, fusing information from the three UAVs. The navigation heights selected were appropriate to detect the objects in the arena and estimate their positions with enough accuracy to be found later by a UAV trying to collect them. The main source of error for the objects' positions came from the UAV positioning systems since those were used to project the image detections onto the 3D global coordinate system. Moreover, the time synchronization to match image detections and UAV telemetry was not perfect. Overall, with our GPS-based positioning system, we achieved an accuracy with errors below 2 m for the UAVs, and hence, for object estimations. We did not use RTK GPS and we had no ground truth either, so we could only get an empirical estimation of the UAV localization error. For that, we took large sets of measurements of a UAV at different static positions and compared them with the average value to extract a standard deviation. The UAV altitude was provided by a highly accurate laser altimeter, and hence it had a much lower vertical uncertainty. The cooperative planner worked also properly during the competition trials, performing the search phase and distributing later the objects between the three UAVs.

We were able to detect most of the objects in the arena, but we observed that two parameters were critical for the estimator performance, the error covariance for the observed positions **R** and the association threshold d_{th}. As expected, it is essential to adjust those parameters adequately so that the filter is not overconfident (what may make it diverge at some point) and the associations are reasonable. On the one hand, if the distance threshold for association is decreased, the filter considers many observations as new different objects instead of integrating them within previously existing estimations. On the other hand,

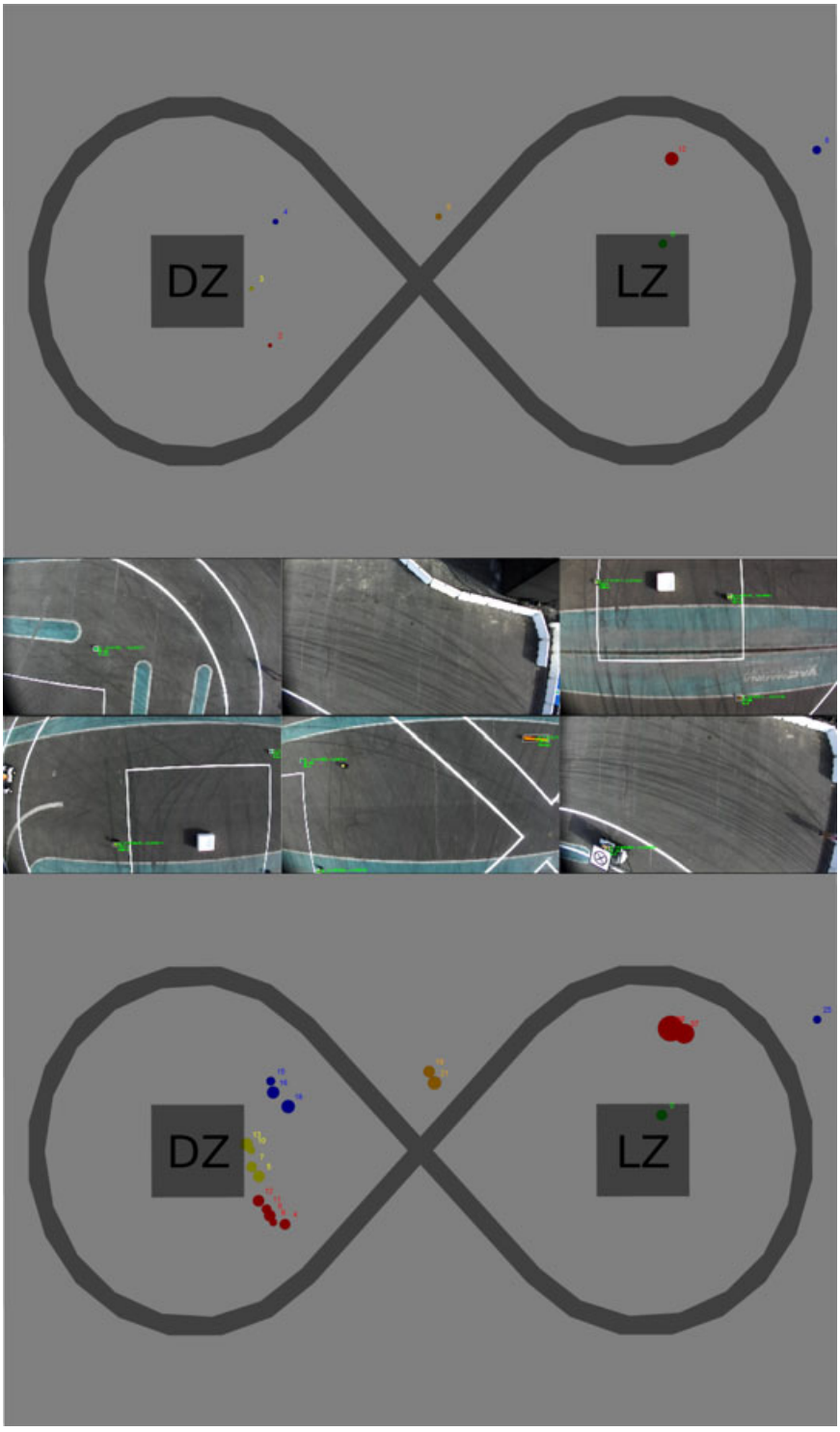

FIGURE 17 Results of the object estimator during a trial in Abu Dhabi with two UAVs. In the middle, some images taken from the UAVs during the experiment (each row comes from a UAV). Green marks indicate detections from the Vision Module. On top, the objects estimations after the search phase of both UAVs and with the Estimator parameters properly adjusted. At the bottom, the same without adjusting the parameters correctly. Each object has a number associated and a circle with the estimated position covariance. The color of the circle represents the most likely color according to the filter

increasing this threshold too much could cause that close objects are seen as the same.

Figure 17 shows some experimental results in Abu Dhabi[21] for the object estimator with and without adjusting the above parameters. When everything is tweaked correctly (top view), the system outputs estimations for the actual objects, whereas too many spurious objects appear without the correct parameters (bottom view). As explained above, this is due to the fact that many observations corresponding to the same objects are not associated well but seen as new objects. Moreover, the video of this experiment shows how a moving object (yellow) is detected within the dropping zone (second 50) but not included in the estimator due to the geofencing tool. The idea is to avoid the UAVs from going toward dropped objects again.

[21]A video of the experiment can be seen at https://youtu.be/38PnmsH4jOk.

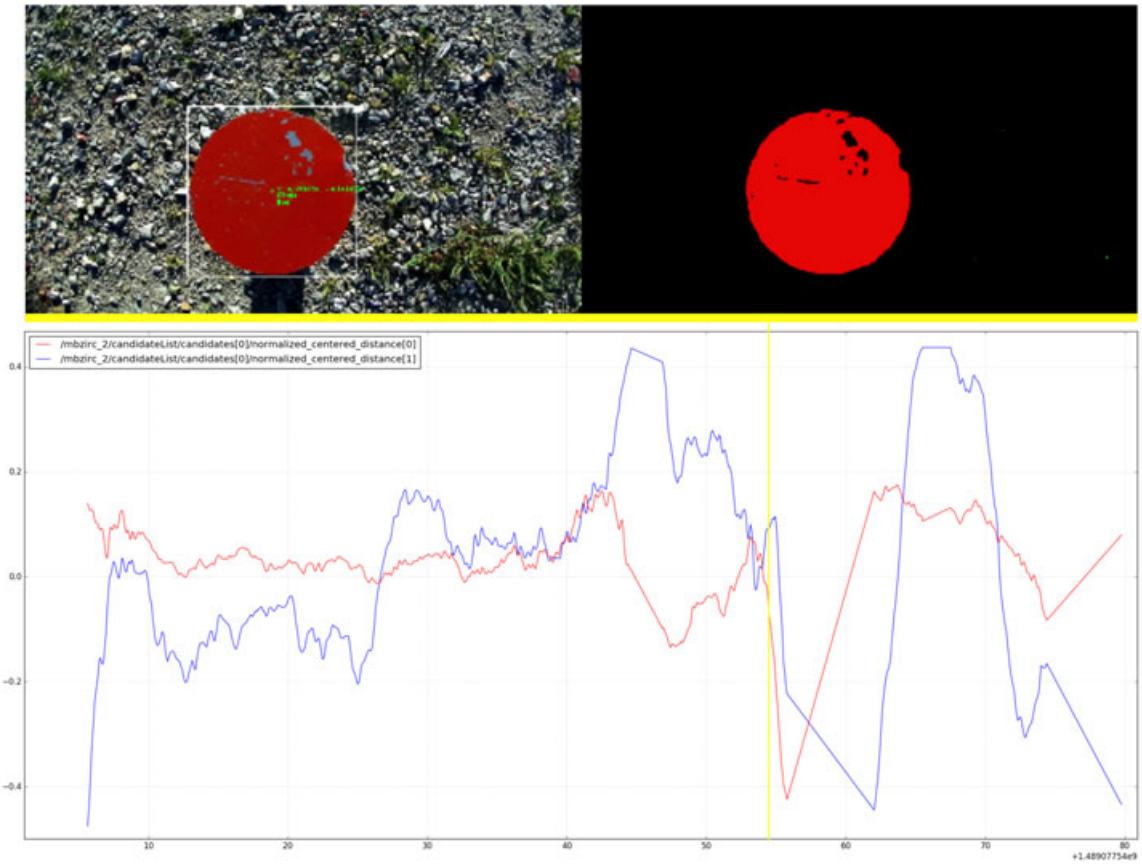

FIGURE 18 Results of a successful operation to pick up a red object autonomously. On the top left, a frame of the original image with the results of the vision detector. On the top right, the frame segmented. At the bottom, the evolution of the horizontal position errors. Those errors are measured with respect to the image center and normalized. The yellow line indicates the time instant corresponding to the example frames [Color figure can be viewed at wileyonlinelibrary.com]

Also, there are some false positive detections during the video that create new objects which are erased later (e.g., objects 1 and 6), as they are considered spurious after some time without detection. This is done for objects that are detected just in a couple of image frames.

Another interesting discovery during the competition trials was that the movement of the yellow objects was quite restrictive since they moved around the same zone where they started. In the beginning, our estimator was configured to remove moving objects that had not been detected for a while. That made sense because those predicted estimations were not reliable anymore after some time. However, we ended up treating them in the estimator as static ones, bounding their predictions and not removing them when not seen for a while. Regarding the cooperative behavior, it turned out to be wise the strategy of focusing first on the small, static objects, and then the moving ones, since those were harder to pick up and there was no team collecting all the static ones. Moreover, even though we managed to run missions with the three UAVs, many times we ended up with fewer due to hardware, software, or communication failures. This could lead to some issues due to synchronization constraints between the UAVs. For instance, originally the UAVs were waiting for each other after the search phase, to move together to the collecting phase. In the end, we removed those synchronizations to make the distributed system more robust.

6.8 | Picking up objects

In our previous experiments in Seville, we tested the software architecture to pick up our mock-up objects autonomously. Even though we did not have time to test the system extensively under a wide variety of conditions, we performed successful experiments, including autonomous complete missions picking up several pieces. For instance, Figure 18 shows the results of an experiment where one of our UAVs attempts to pick up a red object with the autonomous visual servoing.[22] In this experiment, the UAV centers the object on the image plane by means of its velocity control, at the same time that it descends gradually. The visual detector is stable enough and the UAV is able to recover when the object gets out of the field of view. This is done by ascending back slightly until the object is seen again. After the second attempt (second 70 of the video), the object is caught by the magnetic device successfully. This is noticed by the UAV, that starts going up again.

We also run some repeatability tests to assess the overall performance of the system.[23] In particular, we repeated multiple

[22]A video of the complete experiment is available at https://youtu.be/NQLvokGbVzM.

[23]A video showing an excerpt of these experiments is available at https://youtu.be/0n2B0wOoOZI.

autonomous pickups of different static objects an evaluated the success rate and the duration of each trial. On average, 20% of the trials failed, that is, the UAV was not able to pick up the object; a 30% of the trials were partially successful, that is, the UAV picked up the object but it fell down when returning to the dropping area; and a 50% of the trials were totally successful, with the UAV picking up and dropping the object correctly. Moreover, the average duration of each trial was 40 ± 4 s and the number of attempts 2.1 ± 0.34. In each trial, a single UAV started the pickup operation always at the same height, and performed several attempts (as explained in Figure 12) until either it picked up the object successfully or it made it fall down from its pedestal. We also picked up moving objects successfully, but we did not have enough time before the competition to run similar repeatability tests with moving objects.

During the first trials in Abu Dhabi, the performance of our controller was not satisfactory and we did not achieve the same successful results picking up objects autonomously. There were windy conditions and it turned out that our system was not robust enough to cope with that. We thought of tuning the controller to make it more aggressive, but it was too risky because we were not allowed to fly the UAVs for testing out of the trials. Instead, we decided to modify the final behavior, including a free fall of the aircraft (until the contact sensor was activated) when it managed to have the object centered and close enough.

After the free-fall implementation, we performed the last competition trials where our UAVs attempted to pick up several objects autonomously. However, they did not fall down with enough accuracy to contact the objects. Any subtle delay in the free-fall decision resulted in blindly trying to pick up a nonexistent object near the actual one. The problem may have been solved by tuning and testing better the controller, but we had no time available for that.

7 | CONCLUSIONS

In this paper, we presented a cooperative approach with multiple UAVs to address the MBZIRC Challenge 3. This Challenge takes place in an outdoor arena and it consists of searching, collecting, and transporting to a dropping box a set of static and moving colored objects. First, we presented the hardware and software architecture of our system. Then, we detailed the procedure to design our aerial platforms and all the onboard components. We also described the techniques used to develop all the software functionalities. Finally, we discussed our results before and during the first edition of the competition in Abu Dhabi (2017), as well as all the lessons learned during the process.

In terms of hardware, our aerial platforms performed well with all the devices correctly integrated. However, provided that battery replacement was not penalized eventually and that the payload requirements from the objects were not so high as expected, we conclude that we could have used UAVs with less payload. These would have been lighter and agiler platforms, and hence easier to control and stabilize.

Regarding the software modules, our participation in the competition entailed a tremendous and fruitful integration effort. As a result, our team managed to perform cooperative missions with the three UAVs and all the modules working together. In the competition trials, we always started in *Autonomous Mode* and flew simultaneously our three UAVs, except for one of the trials, where we lost communication with a UAV from the beginning. Our team always completed the search phase autonomously, finding most of the objects on the arena. Then, the team was also able to allocate objects to the UAVs autonomously, and the UAVs attempted to collect their assignments navigating without collisions in a coordinated manner.

In our experiments previous to the competition, we managed to pick up mock-up objects with an acceptable success rate, which we did not achieve with worse windy conditions in Abu Dhabi. We conclude that our system was more sensitive than others to the external conditions since it required to have the UAV stabilized to make contact with the pieces. We strongly believe that the system would have worked fully autonomous with some more time for a proper calibration and tunning process.

As a general conclusion, it seems that this first edition of the MBZIRC was more focused on hardware issues. Designing reliable aerial platforms and pickup mechanisms were the most crucial part. On the contrary, there was less focus on the implementation of cooperative and efficient strategies. After this first experience, we foresee that the next edition will push forward in that direction. Many participant teams will offer reliable hardware solutions and they will compete according to the efficiency of their strategies and methods.

ACKNOWLEDGMENTS

We would like to thank the Khalifa University and its sponsors for organizing the MBZIRC competition and providing fund for the participation of the Al-Robotics team. This study was also partially funded by the European Union's Horizon 2020 research and innovation program under Grant agreement No. 731667 (MULTI-DRONE). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains. The authors would also like to thank Luis Ramírez, Julián Delgado, and Carmelo Fernández-Agüera for their support developing and integrating our systems for the MBZIRC competition; and the people from DroneTools for their support during the integration of the aerial platforms.

ORCID

Ángel R. Castaño http://orcid.org/0000-0002-6235-9524
Pablo Ramón-Soria http://orcid.org/0000-0002-1411-0281
Jesús Capitán http://orcid.org/0000-0002-7534-0187
Begoña C. Arrue http://orcid.org/0000-0003-1777-2675
Aníbal Ollero https://orcid.org/0000-0003-2155-2472

REFERENCES

Amigoni, F., Bastianelli, E., Berghofer, J., Bonarini, A., Fontana, G., Hochgeschwender, N., & Schiaffonati, V. (2015). Competitions for benchmarking: Task and functionality scoring complete performance assessment. *IEEE Robotics Automation Magazine*, *22*(3), 53–61.

Anderson, R. and Milutinovic, D. (2013). Anticipating stochastic observation loss during optimal target tracking by a small aerial vehicle. International Conference on Unmanned Aircraft Systems (ICUAS), 278-287.

Beard, R., McLain, T., Nelson, D., Kingston, D., & Johanson, D. (2006). Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*, *94*(7), 1306–1324.

Bruce, J., Balch, T., and Veloso, M. (2000). On active target tracking and cooperative localization for multiple aerial vehicles.International Conference on Intelligent Robots and Systems (IROS), 2229-2234.

Burdakov, O., Doherty, P., Holmberg, K., Kvarnstrom, J., & Olson, P. M. (2010). Relay positioning for unmanned aerial vehicle surveillance. *The International Journal of Robotics Research*, *29*(8), 1069–1087.

Capitan, J., Merino, L., Caballero, F., & Ollero, A. (2011). Decentralized delayed-state information filter (DDSIF): A new approach for cooperative decentralized tracking. *Robotics and Autonomous Systems*, *59*(6), 376–388.

Capitan, J., Merino, L., & Ollero, A. (2016). Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs. *Journal of Intelligent & Robotic Systems*, *84*(1), 371–386.

Cook, K., Bryan, E., Yu, H., Bai, H., Seppi, K., & Beard, R. (2014). Intelligent cooperative control for urban tracking. *Journal of Intelligent & Robotic Systems*, *74*(1–2), 251–267.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection.IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), *1*, 886-893.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

Furrer, F., Burri, M., Achtelik, M., & Siegwart, R. (2016). RotorS–A modular Gazebo MAV simulator framework. In Koubaa, A. (Ed.), *Robot operating system (ROS): The complete reference* (1, *pp. 595–625). Springer International Publishing*

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258–1276.

Goyal, S., & Benjamin, P. (2014). Object recognition using deep neural networks: A survey. *CoRR*. abs/1412.3684

He, R., Bachrach, A., and Roy, N. (2010). Efficient planning under uncertainty for a target-tracking micro-aerial vehicle.IEEE International Conference on Robotics and Automation, 1-8.

Hsieh, M. A., Cowley, A., Keller, J. F., Chaimowicz, L., Grocholsky, B., Kumar, V., & MacKenzie, D. C. (2007). Adaptive teams of autonomous aerial and ground robots for situational awareness. *Journal of Field Robotics*, *24*, 991–1014.

Ilea, D. E. and Whelan, P. F. (2006). Color image segmentation using a spatial k-means clustering algorithm.International Machine Vision and Image Processing Conference (IMVIP).

Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, *32*(12), 1495–1512.

Morbidi, F. and Mariottini, G. L. (2011). Fast and inexpensive color image segmentation for interactive robots.International Conference on Intelligent Robots and Systems (IROS), 2229-2234.

Ong, L., Upcroft, B., Bailey, T., Ridley, M., Sukkarieh, S., and Durrant-Whyte, H. (2006). A decentralised particle filtering algorithm for multi-target tracking across multiple flight vehicles.International Conference on Intelligent Robots and Systems (IROS), 4539-4544.

Stuckler, J., Holz, D., & Behnke, S. (2012). RoboCup@Home: Demonstrating everyday manipulation skills in RoboCup@Home. *IEEE Robotics Automation Magazine*, *19*(2), 34–42.

Tai, Y. W., Jia, J., & Tang, C. K. (2007). Soft color segmentation and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(9), 1520–1537.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features.Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), *1*, 511-518.

How to cite this article: Castaño ÁR, Real F, Ramón-Soria P, et al. Al-Robotics team: A cooperative multi-unmanned aerial vehicle approach for the Mohamed Bin Zayed International Robotic Challenge. *J Field Robotics*. 2019;36:104–124. https://doi.org/10.1002/rob.21810