Distributed Mission Execution for Aerial Cinematography with Multiple Drones

Arturo Torres-González*, Alfonso Alcántara*, Vasco Sampaio†,
Jesús Capitán*, Bruno Guerreiro†‡, Rita Cunha† and Anibal Ollero*
*University of Seville, Seville, Spain
Email: [arturotorres,aamarin,jcapitan,aollero]@us.es
†Instituto Superior Técnico, Lisbon, Portugal
Email: [vsampaio,bguerreiro,rita]@isr.ist.utl.pt
‡ NOVA School of Science and Technology (FCT/UNL), Caparica, Portugal

***Abstract*—This paper presents a system for autonomous cinematography with multiple drones. Drones are becoming a trend for aerial cinematography. The price for buying a commercial platform is decreasing every year, while their quality increases. Drones allow for new shots and perspectives, and they can be automated. Despite their extended use, there are still challenges for teams of multiple drones which cooperate for autonomous cinematography. The proposed system tries to face these challenges, focusing on the actual execution of aerial shots. A set of canonical shots with specific parameters for autonomous implementation is compiled. This system includes a distributed scheduler to synchronize shots using an event-based mechanism, and an autonomous controller to provide smooth movements in both the drone and the camera, so the drones can take aesthetic shots. Moreover, the system considers safety in two levels: collision avoidance in the controller and emergency management to handle high level alarms (e.g. low battery).**

***Index Terms*—Multi-drone system, aerial cinematography, distributed mission execution.**

I. Introduction

Aerial cinematography with drones is awakening relevant attention lately, with many new commercial platforms for both amateurs and professionals. Drones are becoming so popular for filming due to their cost and maneuverability compared with static cameras or dollies. Moreover, drones allow us to take aesthetic shots from unique perspectives. The idea of using multiple drones to cover the same event cooperatively is novel and has special interest for outdoor, large-scale scenarios, where there may be several action points taking place at different locations simultaneously. Also, multi-camera shots with several drones open a wide spectrum of artistic possibilities for media production.

Autonomous drone cinematography entails problems such as target tracking or the generation of smooth trajectories for the cameras. However, multi-drone systems impose even additional challenges: the number of operators involved to control all cameras and drones increases; different shots have to be allocated to the drones efficiently; drones must avoid collisions with others when executing multi-camera shots; etc.

There are multiple works addressing camera motion planning for aerial filming [1]–[3]. The common idea is to formulate some kind of optimization problem to generate smooth camera trajectories that fulfill aesthetic and cinematographic constraints. There are also end-to-end solutions for aerial cinematographers [4], [5] where high-level commands can be specified. However, the focus of these previous works is on static scenes and single-drone settings. Moreover, there are works filming dynamic targets in outdoor scenarios and coping with obstacle avoidance [6]–[8]. There is no much work considering multi-drone shots for cinematography. Recently, some authors have proposed MPC-based optimization techniques to film with several drones in indoor settings [9], [10].

The EU-funded project MULTIDRONE [1] focuses on autonomous media production with multiple drones. The project objective is to develop a team of several drones that can film outdoor sport events in a coordinated manner. The project studies the whole process to define *shooting* missions for media production, translate them into feasible plans for a drone team and execute them. This implies assigning all requested shots to single drones or to subsets of drones, for the case of multi-view shots. This paper focuses on the autonomous execution of these shooting missions with the multi-drone team, once all shots have been allocated by a central planner, we propose a distributed system to execute aerial shots autonomously ensuring coordination and safety.

In particular, our main contributions are the following:

- We define a list of canonical shots for aerial cinematography. The type of shots come from previous works on drone cinematography, but we compile a set of parameters describing each shot to implement them autonomously.
- We propose a distributed scheduler for autonomous shot execution, and we devise an event-based mechanism for inter-drone shot synchronization. Safety is also considered by means of an emergency management component.
- An autonomous controller taking care of drone and gimbal motion to execute shots autonomously and safely.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

[1] https://multidrone.eu

II. System Overview

Figure 1 shows the architecture of our system. There is a central entity, so-called *Mission Controller*, which is in charge of interfacing the end-user to receive shot requests and then compute a feasible plan for the shooting mission. This module is out of the scope of this paper and produces a list of *actions* for each drone. These actions are to be executed by our distributed *Scheduler*, which runs on board each drone. Each drone Scheduler receives its lists of actions and needs to synchronize their start and end by calling another *Executer* module, which is the one actually controlling the drone and the camera. The Schedulers across multiple drones need to synchronize their actions for muli-drone shots. This is done by means of *Events* also sent by the Mission Controller. Thus, these Events are used to trigger actions.

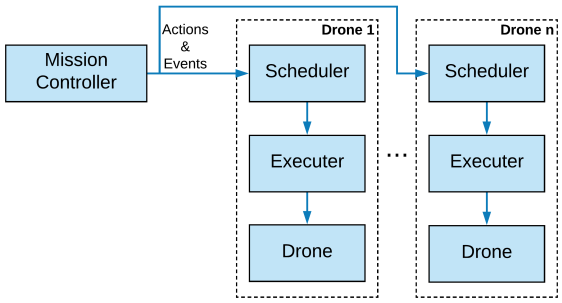

Fig. 1. System architecture with multiple drones.

III. Canonical Shots

In this section, we describe a set of canonical shots that have been implemented for our system. There is a lot of information about cinematographic rules and canonical types of shots in the literature [11]. In MULTIDRONE project, these canonical shots have been studied to come up with a taxonomy that include the most representative ones [12], which we implemented in our system.

A common relevant concept for all shots is the type of target. Each shot can have a *Reference Target* (RT) and a *Shooting Target* (ST). The former is used to drive drone movements, as they would go in formation following this RT. The latter is used to point the camera when filming. They may both coincide but not necessarily. The ST can be `virtual` if it is a predefined point or path or `real` if it is an actual physical target (e.g., a cyclist, a runner, etc.) whose position can be estimated, for instance through visual detection or with a mounted GPS. `None` ST can also be possible when the camera follows a pre-define motion. Additionally, we define three different motion modes for the drones during a shot:

- *VIRTUAL_TRAJ*: A predefined path and speed is specified for the RT, so-called RT trajectory. Drones will follow this RT trajectory at the specified speed.
- *VIRTUAL_PATH*: A predefined path is specified (RT trajectory) but no speed is provided. Drones will follow the rail specified by the RT trajectory at the speed of an actual ST.
- *ACTUAL_TARGET*: Drones follow an actual ST with no predefined trajectory. An RT trajectory could still be provided as an estimation of the target movement.

In the following we describe the parameters of the shot types that our system is able to execute. Table I summarizes them.

a) `STATIC`: The drone remains stationary above a fixed RT location (this height is indicated by parameter z_0), so an actual target as RT makes no sense. The ST could be `real` or `virtual` if the camera follows an actual or virtual target. With the ST as `none`, we can implement shots scene-centered moving the gimbal independently. In this case, parameters pan_s, pan_e, $tilt_s$ and $tilt_e$ indicate the pan and tilt initial and end angles.

b) `FLY-THROUGH`: The drone flies through the scene following a pre-defined path with no specific target to track. As in the previous shot, the RT is not an actual target, so a RT trajectory is required plus the flight altitude z_0. The ST is `none` and extra parameters are needed to indicate gimbal movement: pan/tilt initial and end angles (pan_s, pan_e, $tilt_s$ and $tilt_e$).

c) `ELEVATOR`: The drone moves vertically straight up or down tracking a target or a static position. The drone starts the shot above a given position (defined as the initial RT location) at altitude z_s, and it ends at z_e. Thus, a virtual RT with an RT trajectory is required. The ST could be `real` or `virtual`.

d) `CHASE/LEAD`: The drone chases a target from behind with constant or decreasing distance; or leads it in the front with decreasing or constant distance. The RT could be virtual or real, so all RT modes are possible. For the ST, only the `real` mode makes sense. Regarding parameters, z_0 determines the drone height over the RT and x_s and x_e, the initial and final distances in the X axis (pointing forwards) w.r.t. the RT.

e) `FLYBY`: The drone flies past a target normally overtaking the target as the camera tracks it. The RT could be virtual or real, so all RT modes are possible. For the ST, only the `real` mode makes sense. It needs as parameters distances w.r.t. the RT: z_0 for the altitude, x_s and x_e for the initial and final distances in the X axis and the constant lateral distance y_0.

f) `LATERAL`: The drone flies beside a target with constant distance as the camera tracks it. The RT could be virtual or real, so all RT modes are possible. For the ST, only the `real` mode makes sense. It needs as parameters the z_0 altitude w.r.t. the RT, and the constant lateral distance y_0.

g) `ESTABLISH`: The drone moves closer to a target from the front, typically with decreasing altitude. The RT could be virtual or real, so all RT modes are possible. For the ST, only the `real` mode makes sense. Both altitude and displacement in the X axis w.r.t. the RT change during this shot, so it needs as parameters z_s, z_e, x_s and x_e.

h) `ORBIT`: The drone moves around a target in a full or partial circle. The RT could be virtual or real, so all RT modes are possible. The ST could be `real` or `virtual`.

The parameters in this case include the altitude over the RT (z_0), the radius of the circle (r_0), the initial azimuth angle ($azimuth_s$) and the angular speed ($angular\_speed$).

TABLE I
PARAMETERS FOR EACH SHOT TYPE.

Shot name	Shooting parameters
STATIC	pan_s, $tilt_s$, pan_e, $tilt_e$, z_0
FLY_THROUGH	pan_s, $tilt_s$, pan_e, $tilt_e$, z_0
ELEVATOR	z_s, z_e
CHASE/LEAD	x_s, x_e, z_0
FLYBY	x_s, x_e, y_0, z_0
LATERAL	y_0, z_0
ESTABLISH	x_s, x_e, z_s, z_e
ORBIT	r_0, $azimuth_s$, $angular\_speed$, z_0

IV. DISTRIBUTED SCHEDULING

Multi-drone shot execution is carried out by means of a distributed Scheduler. Each drone runs on-board a Scheduler module that coordinates the execution of the shots assigned to it. Figure 2 shows a detailed scheme of this Scheduler, which consists of three components: the *Core*, a module for *Emergency Management* and a *Path Planner*.

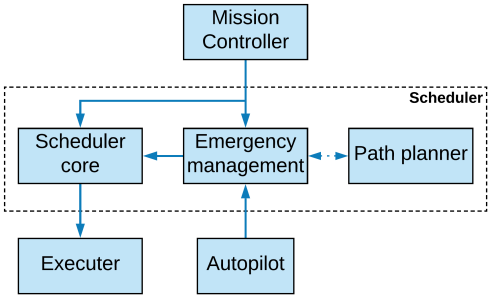

Fig. 2. Detailed scheme for the Scheduler on each drone.

Each Scheduler on board a drone receives plans to execute from the central Mission Controller. These plans consist of a list of actions for that particular drone that should be executed sequentially. The Scheduler, through its Core module, is in charge of synchronizing the start and end of each action and sending them to the drone's Executer. There are two types of actions that a drone can receive within a plan: navigation and shooting actions. Both types can have a start Event associated, which is the one triggering the action.

Navigation actions are those that do not include filming. These are basic commands such as landing, take-off or go-to-waypoint. In this case, a single waypoint or a list of waypoints to navigate the drone are provided.

Shooting actions are those that include filming. Therefore, they require a special controller to be executed, since drone and gimbal movement are needed. The data structure for these actions includes the shot type, its cinematographic (or shooting) parameters, its duration, associated targets, etc. The complete data structure is depicted in Table II. The fields related to the targets include some for the reference target (RT trajectory, RT speed, RT mode and RT ID) and others for the shooting target (ST type, ST ID). Section III described the shot types, how each type use the fields and which shooting parameters need.

TABLE II
STRUCTURE FOR THE DATA TYPE SHOOTING_ACTION.

SHOOTING_ACTION		
Field name	**Data type**	**Comment**
Start Event	String	Event that triggers this action
Shot type	Discrete value	Lateral, chase, orbit... (see section III)
Duration	Time	Duration of the shot
RT trajectory	List of global positions	Estimated path of the RT
RT speed	Float value	Speed of the RT if known
RT mode	Discrete value	*VIRTUAL_TRAJ*, *VIRTUAL_PATH*, *ACTUAL_TARGET*
RT ID	Natural number	Identifies the RT to follow if any
ST type	Discrete value	virtual, real, none
ST ID	Natural number	Identifies the ST to follow if any
Shooting parameters	Set of parameters	E.g., distance to the RT, angular velocity in an orbit... (see section III)

The Scheduler sends the actions to the Executer sequentially, but the start of some shooting actions or sequences of shooting actions are triggered by Events. These Events come from the Mission Controller, which sends them upon user request or when a specific condition is met. The Events are received by all drones, so the communication of Events will serve as a synchronization method between the individual shooting actions of several drones when needed. In order to have a proper synchronization between the ground station and all the drones, all the computers use NTP (Network Time Protocol) to keep their clocks synchronized.

In summary, the Scheduler ends up with a sequential list of navigation and shooting actions. Some of these are triggered by a specific Event, e.g., the start of a race or reaching a significant point of the race. The other actions are triggered as soon as their previous actions end. Section VI will depict an example shooting mission being executed by three drones.

A. Emergency management

Each Scheduler has integrated a sub-module for emergency management, which is crucial for safety. This module is in charge of monitoring the drone status, looking for any possible failure. If it detects failures such as low battery and lose of GPS, it activates an emergency status that is reported back to the Mission Controller on the ground. The Mission Controller may decide then to compute a new plan dismissing the affected drone and reassigning its tasks.

Simultaneously, the Scheduler will carry out a contingency plan. It will cancel the action being executed, it will command the drone to navigate to the closest base station and land. For that, the Path Planner component is used. This component is able to compute a safe paths (i.e., without collisions) to a given position. It has information about the positions of the base stations (KML-based) and a detailed map of the environment (point cloud). We use an off-the-shelf A* heuristic planner.

V. Autonomous Shot Execution

This section describes the controller for drone and gimbal motion. Figure 3 shows a detailed scheme of the Executer. It is divided in three modules, two for performing the autonomous shooting (drone and camera controls) and one for handling collision avoidance maneuvers.

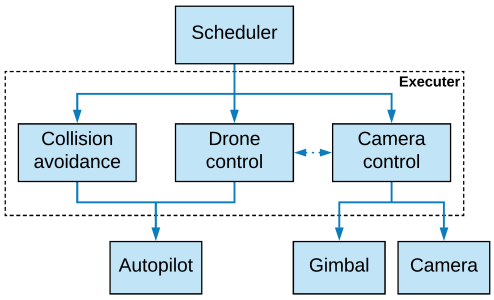

Fig. 3. Detailed scheme for the Executer on each drone.

All the modules inside the Executer need information about the drone and target positions. The target information could be provided, for instance, by a GPS mounted on the target or by visual tracking.

A. Drone control

Upon receiving a Shooting Action request from the Scheduler, the Drone Control module generates on-the-fly a desired drone trajectory in agreement with shot type and respective parameters. Based on this desired trajectory and the current state estimate, an error between current and desired position and yaw angle is then used to generate the velocity commands to the autopilot, using a simple saturated proportional controller together with a feedforward velocity term.

To generate the desired drone trajectory we consider the RT mode of the shooting action to determine whether to follow a virtual or real target and then impose the behavior of a trailer attached to target [13]. This approach effectively produces smooth reference trajectories for the drone to track. In addition, by generating a trailer trajectory, a reference frame tangent to path is also obtained, which can be directly used to define the relative displacements encoded in the parameters of each shot type. For example, given a FLYBY shot, where the drone is expected to overtake the target, the constant lateral and vertical displacements, y_0 and z_0, the initial and final displacements along the X-axis, x_s and x_e, as well as the speed to go from one to the other $v_r = (x_e - x_s)/\Delta t$, are all defined with respect to the trailer reference frame. The same applies to the CHASE/LEAD, LATERAL, ESTABLISH, and ORBIT shot types.

B. Camera control

The Camera Control module commands both gimbal and camera by means of dedicated interfaces. Depending on the requested shooting action, the gimbal may be controlled to describe predefined pan and tilt movements (STATIC and FLY_THROUGH shot types) or point at a virtual or real target. In all cases, the desired angular motion of the gimbal is computed independently from that of the drone and with respect to a world reference frame. This is made possible by the fact that the gimbal is equipped with an IMU and low-level controller that compensates for the motion of the platform and tracks angular rate commands. The option was made to match the heading of the vehicle with the direction of motion, allowing for the implementation of reactive collision avoidance based on the forward-looking LIDAR. Thus, conflicting objectives may arise leading to relative angles between drone and gimbal that exceed the mechanical limits. Such situations are avoided by setting to zero the angular rate commands whenever the relative angles approach their limits, using a bump function to smooth out the transition. Besides controlling the gimbal, this module is also responsible for sending commands to the camera that include start and stop recording, autofocus, and changing camera parameters, such as zoom, aperture, ISO, or white balance.

C. Collision avoidance

This sub-module performs reactive obstacle avoidance. These obstacles include static obstacles (e.g., trees, buildings, etc.) or moving obstacles (e.g., other drones in the team). The plan sent by the Mission Controller is supposed to be collision-free and, hence, drone trajectories should not cross. However, the actual execution may make drones to come closer than expected. Moreover, in case of following an actual target, its final trajectory will not be the same as estimated, so drones' trajectories may provoke collisions.

The Collision Avoidance component supervises drone trajectories and overwrites the Drone Control commands if collisions are detected. We use a reactive algorithm for collision avoidance [14]. The algorithm receives obstacles information from two sources: a LIDAR on board the drone pointing forwards; and a communication channel where neighboring drones share their positions in real time. Then, the algorithm defines safety cylinders around the drone to detect possible conflicts when they are invaded by obstacles. These conflicts are resolved by a set of maneuvers assuming that all drones follow the same predefined rules.

VI. Simulated Example

This section presents a use case example in simulation. We use the GAZEBO [15] simulator and the PX4 [16] SITL (Software In The Loop) functionality to simulate the autopilot. The simulator also uses UAL (UAV Abstraction Layer) [17], an open-source library to interact with autonomous drones, abstracting the user from the specifics of the platform used. With this simulator and the developed tools, it is easy to run and test different examples without flying with the real drones but using the very same software that would run in the real drones.

The use case example involves two drones performing shooting actions that are launched by Events. Figure 4 shows the temporal evolution of the actions carried out by the drones

during the mission of this example. First, the drones remain on the ground waiting for the `GET READY` Event. When this Event is triggered, the drones take off and go to the starting point of the first shooting action. The drones wait for the `START RACE` Event to execute the `CHASE` shot (Drone 1) and the `LATERAL` shot (Drone 2) simultaneously. This moment of the simulation is shown in Fig. 5. Drone 1 starts to perform the `FLYBY` shot when the duration of the previous shooting action is reached while Drone 2 keeps executing the `LATERAL` shot. When the target reaches the final line, the `FINISH LINE` Event is triggered and both drones carry out the `ORBITAL` shooting action. When the time is up, the drones return home and land at the base station.

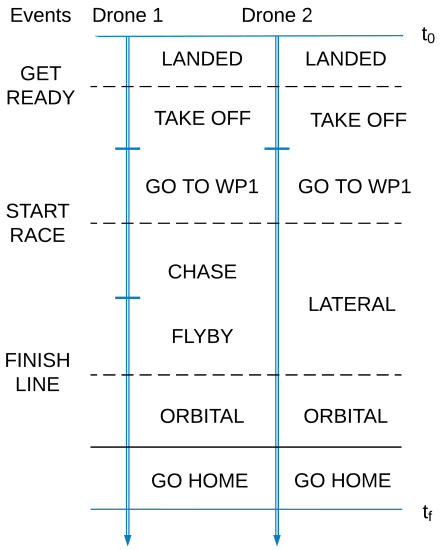

Fig. 4. Timeline of the simulated example mission.

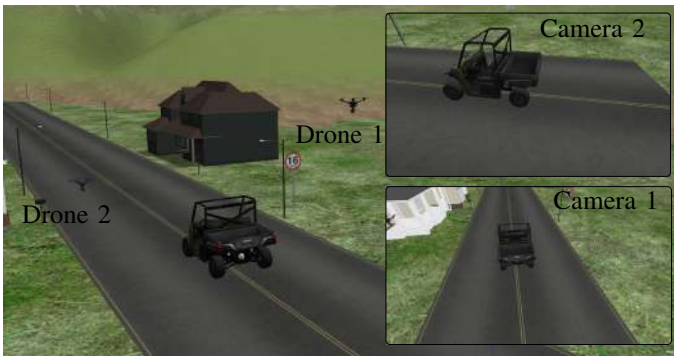

Fig. 5. Simulation with two drones. The target follows a straight line along the road. At that moment, Drone 1 and drone 2 are performing a `CHASE` and `LATERAL` shooting actions.

VII. Conclusions

This paper presented an architecture for autonomous shooting with multiple drones. The proposed system is composed of a distributed scheduler and an autonomous controller to execute the actions. Safety is considered in both parts. A list of canonical shots for aerial cinematography with specific parameters for autonomous implementation is also presented.

The system was tested in simulation with simple and more complex missions with up to three drones. Next steps include testing the execution of autonomous shooting missions with multiple drones in real experiments.

References

[1] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 1–11, oct 2015.

[2] C. Lino and M. Christie, "Intuitive and efficient camera control with the toric space," *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 82:1–82:12, jul 2015.

[3] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated Cinematography with Unmanned Aerial Vehicles," *Eurographics Workshop on Intelligent Cinematography and Editing*, 2016.

[4] N. Joubert, J. L. E, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, and P. Hanrahan, "Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles," *ArXiv e-prints*, 2016.

[5] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-Based Planning of Quadrotor Trajectories according to High-Level User Goals," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, ACM. New York, New York, USA: ACM Press, 2016, pp. 2508–2519.

[6] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Buelthoff, M. J. Black, and A. Ahmad, "Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3193–3200, Oct. 2018.

[7] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K. T. Cheng, "Act: An autonomous drone cinematography system for action scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7039–7046.

[8] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. Scherer, "Towards a Robust Aerial Cinematography Platform: Localizing and Tracking Moving Targets in Unstructured Environments," *arXiv e-prints*, p. arXiv:1904.02319, Apr 2019.

[9] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–10, jul 2017.

[10] M. Saska, V. Krátký, V. Spurný, and T. Báča, "Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control," in *22nd IEEE Intl. Conf. on Emerging Technologies and Factory Automation (ETFA)*, Sep 2017, pp. 1–8.

[11] C. Smith, *The Photographer's Guide to Drones*. Rocky Nook, Inc., 2016.

[12] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-uav cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, pp. 1–9, 2019.

[13] P. O. Pereira, R. Cunha, D. Cabecinhas, C. Silvestre, and P. Oliveira, "Leader following trajectory planning: A trailer-like approach," *Automatica*, vol. 75, pp. 77 – 87, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109816303363

[14] E. Ferrera, A. Alcántara, J. Capitán, A. R. Castaño, P. J. Marrón, and A. Ollero, "Decentralized 3d collision avoidance for multiple uavs in outdoor environments," *Sensors*, vol. 18, no. 12, 2018.

[15] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[16] L. Meier, T. Gubler, J. Oes, D. Sidrane, D. Agar, B. KÃijng, A. Babushkin, px4dev, M. Charlebois, R. Bapst, and et al., "Px4/firmware: v1.7.3 stable release," Jan 2018.

[17] F. Real, A. Torres-González, P. R. Soria, J. Capitán, and A. Ollero, "Ual: an abstraction layer for unmanned vehicles," in *2nd International Symposium on Aerial Robotics (ISAR)*, 2018.