

## Article

# Director Tools for Autonomous Media Production with a Team of Drones

Ángel Montes-Romero <sup>1</sup>, Arturo Torres-González <sup>1</sup>, Jesús Capitán <sup>1,\*</sup>,  
Maurizio Montagnuolo <sup>2</sup>, Sabino Metta <sup>2</sup>, Fulvio Negro <sup>2</sup>, Alberto Messina <sup>2</sup> and Aníbal Ollero <sup>1</sup>

<sup>1</sup> GRVC Robotics Lab, University of Seville, 41092 Seville, Spain; amromero@us.es (Á.M.-R.); arturotorres@us.es (A.T.-G.); aollero@us.es (A.O.)

<sup>2</sup> Centre for Research and Technological Innovation, Radiotelevisione Italiana (RAI), 10138 Turin, Italy; maurizio.montagnuolo@rai.it (M.M.); sabino.metta@rai.it (S.M.); fulvio.negro@rai.it (F.N.); alberto.messina@rai.it (A.M.)

\* Correspondence: jcapitan@us.es

Received: 6 February 2020; Accepted: 13 February 2020; Published: 21 February 2020



**Featured Application:** This work can be applied for media production with aerial cameras. The system supports media crew to film outdoor events with an autonomous fleet of drones.

**Abstract:** This paper proposes a set of director tools for autonomous media production with a team of drones. There is a clear trend toward using drones for media production, and the *director* is the person in charge of the whole system from a production perspective. Many applications, mainly outdoors, can benefit from the use of multiple drones to achieve multi-view or concurrent shots. However, there is a burden associated with managing all aspects in the system, such as ensuring safety, accounting for drone battery levels, navigating drones, etc. Even though there exist methods for autonomous mission planning with teams of drones, a media director is not necessarily familiar with them and their language. We contribute to close this gap between media crew and autonomous multi-drone systems, allowing the director to focus on the artistic part. In particular, we propose a novel language for cinematography mission description and a procedure to translate those missions into plans that can be executed by autonomous drones. We also present our director's *Dashboard*, a graphical tool allowing the director to describe missions for media production easily. Our tools have been integrated into a real team of drones for media production and we show results of example missions.

**Keywords:** autonomous media production; drone cinematography; multimedia tools

## 1. Introduction

There is a clear trend toward using drones for aerial cinematography and media production in general. The main reasons for the emergence of this technology are twofold: First, small drones can be equipped with high-quality cameras, but, at the same time, they are not too expensive, which makes them appealing for amateur and professional users. Second, due to their maneuverability, they broaden the aesthetic possibilities for media production, as they can create novel and unique shots. Besides, media production applications to cover outdoor events can benefit from the use of teams of drones, mainly to produce multi-view shots and film multiple action points concurrently. From a logistic perspective, a multi-drone system could be deployed easily (e.g., instead of camera cranes) to operate in such large-scale, outdoor scenarios without requiring the pre-existence of complex infrastructure.

The main issue with a multi-drone system for media production is its complexity of operation. Currently, two operators per drone are usually required: one to pilot the drone and another to handle

camera movement. The media *director* is the person in charge of the whole system from the production point of view. However, there are additional aspects that need to be accounted for: ensuring safety in operations avoiding collisions and no-fly zones, deciding which cameras allocate to each shot, considering battery levels of drones, etc. For that, enhancing the system with autonomous capabilities to plan and execute shots is rather helpful to alleviate the director's burden and allow her/him to focus on the artistic part.

There exist methods for autonomous mission planning with teams of multiple drones. Indeed, general purpose methods for multi-robot task allocation and scheduling could be adapted to tackle this problem. However, a media director is not necessarily familiar with these kinds of algorithms and their robotics language. Therefore, our objective is to close this gap between a media crew and these autonomous, intelligent systems, so that a director can use an autonomous fleet of drones for media production. In particular, we think that there is a need for a standard language and tools for cinematography mission description. With such tools, a director could talk to her/his autonomous cinematographers in a transparent manner, abstracting herself/himself from the planning and execution procedures behind.

In this paper, we propose a set of tools for mission description in media production. First, a novel language for mission description is presented. This language is used by the media director to specify the desired shots when filming an event. The output is an XML-based file that contains details for all shots specified by the director at each action point. Then, this *shooting mission* is interpreted by the system to be translated into a list of tasks that can be understood and executed by the drones. The proposed language acts as a scripting system for director story-telling, who can hence focus on the artistic part without specific knowledge in multi-robot autonomous planning. Last, we also propose a graphical tool, the so-called director's *Dashboard*, to create and manage XML-based shooting missions. This Dashboard allows the director to interact with the fleet of drones by sending/canceling missions and monitoring them during execution.

This paper continues our prior work [1], where we proposed a taxonomy of cinematography shots to implement and a preliminary version of our Dashboard. Here, we add the language to specify autonomous cinematography missions, and a complete description of the final version of the Dashboard, enhanced with new functionalities. Besides, we showcase the use of our tools in example scenarios to film sport and outdoor events, like a rowing or cycling race. We integrated our system with a real team of drones within the framework of the EU-funded project MULTIDRONE (<https://multidrone.eu/>), which aims at building a team of several drones for autonomous media production. We describe the whole process to write, translate, and execute example shooting missions, as well as details of the actual aerial platforms developed for media production.

The remainder of this paper is as follows. Section 2 reviews related work. Section 3 gives an overview of the system, Section 4 presents our language for shooting mission description. Section 5 describes our director's Dashboard. Section 6 depicts some experimental results showcasing the use of our tools, and Section 7 gives conclusions and future work.

## 2. Related Work

Media production is adopting drones as replacements for dollies (static cranes) and helicopters, as their deployment is easier and has less costs associated. Thus, the use of drones in cinematography has increased recently, in parallel with the improvement on their technology. For instance, the New York City Drone Film Festival (<https://www.nycdronefilmfestival.com>.) was the world's first film festival dedicated to drones in cinematography. Besides, drones are quite attractive for live coverage of outdoor events, as they can provide novel angles and visual effects. In October 2016, drones were relevant for news agencies in the coverage of the massive destruction caused by Hurricane Matthew on Haiti [2]. They have also been used in major international sport events, such as the Olympics [3,4].

In the current state-of-the-art solutions for media production with drones, the director usually specify targets subjects or points of interest to be filmed in pre-production, together with a temporarily

ordered script, the camera motion types, etc. Then, the drone pilot and the cameraman must execute the plan manually in a coordinated fashion. There are also commercial drones, such as DJI [5], AirDog [6], 3DR SOLO [7], or Yuneec Typhoon [8], that can implement certain cinematographic functionalities autonomously. However, they are typically prepared to track a target visually or with a GNSS and keep it on the image frame (*follow-me* mode), not focusing on high-level cinematography principles for shot performance.

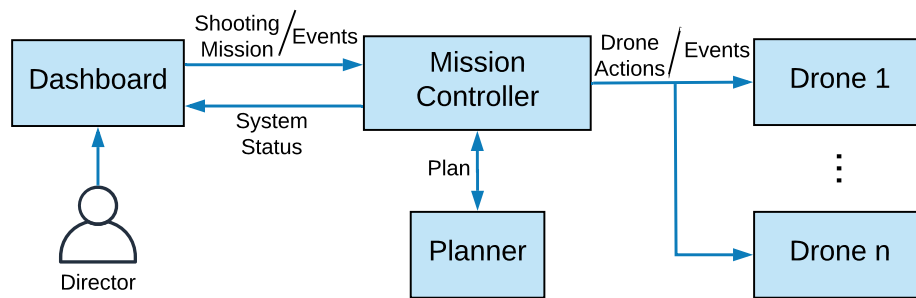
Besides, there are some end-to-end solutions for semi-autonomous aerial cinematographers [9,10]. In these works, a director specifies high-level commands such as shot types and positions for single-drone shooting missions. Then, the drone is able to compute autonomously navigation and camera movement commands to execute the desired shots, but the focus is on static scenes. In [9], an outdoor application to film people is proposed, and different types of shots from the cinematography literature are introduced (e.g., close-up, external, over-the-shoulder, etc.). Timing for the shots is considered by means of an easing curve that drives the drone along the planned trajectory (i.e., this curve can modify its velocity profile). In [10], an iterative quadratic optimization problem is formulated to obtain smooth trajectories for the camera and the look-at point (i.e., the place where the camera is pointing at). No time constraints or moving targets are included. In general, these approaches use algorithms to compute smooth camera trajectories fulfilling aesthetic and cinematographic constraints, which can be formulated as an optimization problem [11–13].

In a multi-drone context, there is little work on autonomous systems for media production. In [14], the optimal number of drones to cover all available targets without occlusion is computed. However, cameras must always be facing targets and smooth transitions are not considered. A more advanced method is presented in [15], where they propose an online planning algorithm to compute optimal trajectories for several drones filming dynamic, indoor scenes. User-defined aesthetic objectives and cinematographic rules are combined with collision and field-of-view avoidance constraints, but they do not address interfacing with the media director.

Regarding user interfaces, there exist several drone *apps* to support photographers, video makers, photogrammetrists, and other professional profiles during their activities. As relevant features, they usually include drone mapping, geo-fencing, and flight logging. In particular, based on regulatory information from each country, some apps can help users indicating no-flight zones such as airports, control zones (CTZ), etc. Local meteorological information (e.g., wind speed, direction, temperature, etc.) is also used in some apps to inform the pilot about flight safety. Moreover, these apps can support the creation of accurate, high-resolution maps, 3D models, and real-time 2D live maps. More specifically on media production, some apps offer autonomous tracking features for shooting video, see for example in [16]. This app supports the execution of several shooting modes, such as *Orbit me* and *Follow me*, as well as the planning in advance of specific flights and shots using a *Mission Hub* on a computer. Thus, the user can pre-program paths that will be later followed by the drone [17,18]. This is done specifying a set of temporally ordered, desired *key-frames* on a 3D reconstruction of the scene, as a rudimentary cinematography plan. Nevertheless, these apps are thought and implemented for single-drone flight and shooting. From a media perspective, the functionality of using more than one drone within the same shooting mission is not properly addressed.

### 3. System Overview

We present in this paper a set of tools so that a director can interface with an autonomous fleet of drone cinematographers and govern the system from an editorial point of view. The main contributions are a novel language for mission description in media production and a graphical tool to define those missions and interact with the autonomous system. The general architecture of the system is depicted in Figure 1.



**Figure 1.** General scheme of the system architecture. The director defines shooting missions with the Dashboard and sends them to the Mission Controller. This Mission Controller uses a Planner to compute plans that are then sent to the drones. During the execution of the plan, the Mission Controller sends events to the drones to trigger actions and reports periodically to the Dashboard about the execution/system status.

The director and the editorial team can specify a set of artistic shots and associate them with different events happening in time (e.g., a race start or racers reaching a relevant point). This is done through the *Dashboard*, which is a web-based graphical tool that allows her/him to interact with the rest of the system. The whole set of director shots, together with information of the events with which they are associated, constitutes the so-called *shooting mission*, which is saved in a database during the editing phase. After finishing this editorial phase, the shooting mission is encoded in an XML-based language and sent to the *Mission Controller*, which is the central module managing autonomous planning.

The Mission Controller can understand the director's dialect and is in charge of interpreting the shooting mission, as sequential or parallel *shooting actions* that will be assigned to the available drones in the team. A list of *tasks* to be executed by the drone fleet is compiled, where each task has a starting position and a duration, extracted from the shooting actions' descriptions. Then, the Mission Controller can use a *Planner* to obtain a feasible plan to accomplish that shooting mission.

The Planner should take into account spatial and temporal constraints of the tasks, as well as drone battery levels and starting positions, in order to assign tasks to drones efficiently. The plan consists of a list of *drone actions* for each drone, specifying where the drone should go at each instant and what to do. Basically, each drone gets assigned a sequence of shooting actions, so its list of actions is made up of single-drone shooting actions plus the required navigation actions in between. In general, this Planner module could be implemented by any standard planner for task allocation with temporal and resource constraints and it is not the focus of this paper. Some preliminary ideas can be seen in [19].

Once the plan is computed, it is sent to the drones, which are able to execute it autonomously. During execution, the Mission Controller provides some feedback to the director through the Dashboard, reporting on the status of each drone and the mission itself.

Section 4 describes the XML-based language to describe shooting missions and the process to translate them into plans made up of drone actions. Section 5 describes the design and functionalities of the Dashboard.

#### 4. Language for Cinematography Mission Description

There are professional drones for filming in the market, and many of them include autonomous capabilities to implement certain specific shots, for example, a panoramic view or tracking a moving target. However, there is no general framework to specify complete missions for autonomous cinematography. In this section, we present a novel language to describe aerial cinematography missions for multiple drones.

We use a vocabulary that the editorial team can understand and define an XML-based dialect to write shooting missions. In the following, we explain the structure of the XML files and how



they can be converted into lists of tasks for autonomous drones. The complete XML schema is also publicly available ([https://grvc.us.es/downloads/docs/multidrone\\_schema.xsd](https://grvc.us.es/downloads/docs/multidrone_schema.xsd)). Our XML schema to describe shooting missions has the following main information entities:

- **Event:** This is the central entity of information. An event represents a real-world occurrence with a spatial and temporal localization that is going to be filmed. For instance, a sport event like a rowing/cycling race. An event can have child events, i.e., events that occur relatively to the timeline of their parent. For example, different stages of a cycling race or different parts of a particular stage. Thus, there can be a tree-like structure of events where each of them can have a *planned start time* and *duration*.
- **Mission:** This describes the aerial shots designed by the editorial team for a given event. Missions can only be associated with leaf events, i.e., events with no children. A leaf event may contain an arbitrary number of missions, each of which has a specific *role*. This is because the director could design different missions for the occurrence of the same event, depending on contextual conditions. For example, the director might plan to have two distinct (and mutually exclusive) missions, in case of sunny or cloudy weather, respectively.

Each mission can have associated a specific *drone team*, made up of several drones. This choice has been made to foresee settings in which more than one multi-drone team are available, so the corresponding team for each mission needs to be specified. Moreover, a mission can have associated an *editorial team*, i.e., a group of Dashboard users, each with their own role, who will be granted permission to manage and modify the mission data.

- **Shooting Action Sequence:** A mission consists of one or more shooting action sequences (SASs), which can also have different *roles*. Thus, the director can plan different actions within the mission depending on contextual circumstances. For example, if the associated event is a race finish line, the director may decide to have two possible shooting action sequences: one in case of arrival in the sprint, another in case of a solo attack. All shooting action sequences with the same role within a mission will happen concurrently as the associated event occurs.
- **Shooting Action:** A shooting action sequence is made up of an ordered sequence of shooting actions (SAs). Therefore, shooting action sequences with the same role within a mission are aimed to run in parallel, whereas the shooting actions within each of them occur sequentially. A shooting action represents an aerial visual shot to be performed with one or several drones. A *duration* or *length* can be specified to indicate action endurance in terms of time or space (distance), respectively. A shooting action has a *reference target* (RT) that is tracked to move the drone formation alongside (it could be virtual just to define a desired formation movement). An *RT trajectory*, the *origin of the RT*, and *origin of the formation* need to be specified in global coordinates. Thus, it is indicated the origin of the center of the drone formation, which should then move along the RT trajectory as the shooting takes place. The *formation speed* is optional to indicate how fast the formation should move along the RT trajectory (mainly when RT is virtual).
- **Shooting Role:** In a shooting action, there can be one or several shooting roles, one per drone involved. This is helpful to indicate the role of each drone in shots with multiple drones. It has a *shooting type* defining the type of movement of the camera, e.g., static, lateral, orbit, etc. Each shooting role has some *shooting parameters* defining the geometry of the flying formation. These parameters vary depending on the specific type of shot, for instance, to indicate the lateral distance of the drone in a lateral shot or the radius and angle rate in an orbit. A shooting role has also a *framing type*: long shot, medium shot, close-up, etc. Even though all shooting roles in a shooting action have the same RT to move in a linked way, each shooting role could have a different *shooting target* to point the camera, which is specified as an identifier. This identifier may match a certain GNSS transmitter or visual tracker.

In summary, the relation of the main components of the XML schema is as follows.

```

<event> Main event: e.g., ~a race to film
  <event> Leaf event 1: e.g., ~start line
    <mission>
      <shootingActionSequence>
        <shootingAction>
          <shootingRole>
    <event> Leaf event 2: e.g., ~finish line
      <mission>
        <shootingActionSequence>
          <shootingAction>
            <shootingRole>

```

In a pre-production stage, the director and the editorial team will manage a database with all the above entities through the Dashboard. Then, an XML file with all events and missions associated is generated, i.e., a shooting mission. The Mission Controller receives that file and computes plans for all possible combinations of mission roles and shooting action sequence roles. Before starting the execution of the mission, the director will specify the final selected role for the missions and for the shooting action sequences, which will determine the actual plan to be executed. Note that multiple plans for the different roles are presented to the director, who must choose unique roles for the missions and the SASs. Once the plan is selected, the Mission Controller sends their corresponding actions to the drones and waits for events. Anytime a leaf event occurs, this should be notified to the drones, so that they can trigger the associated shooting actions. The occurrence of these leaf events is either indicated manually by the director (e.g., start of a race) or detected automatically by the Mission Controller (e.g., target reaching a relevant position).

To compute a plan, the Mission Controller extracts the relevant information from the XML file and creates a list of data objects of type SHOOTING ACTION, as indicated in Table 1. Most of the fields of the SHOOTING ACTION data structure come directly from the <shootingAction> XML element. However, some of them come from data in the <shootingActionSequence>, <mission>, and <event> elements, for instance, *Start event*, *Mission ID*, or *Action sequence ID*. Other fields are calculated from the received data, such as the *RT displacement*, which is the difference between the <originOfFormation and the <originOfRT>.

**Table 1.** Structure for the data type SHOOTING ACTION.

SHOOTING ACTION		
Field Name	Data Type	Comment
Start event	String	Identifies the leaf event that triggers this shooting action
Planned start time	Time in seconds	Leaf event time with respect to the parent event
Mission ID	String	Identifies the associated mission
Action sequence ID	String	Identifies the associated shooting action sequence
Action ID	String	Identifies this shooting action
Duration	Time in seconds	Duration of the shot
Length	Distance in meters	Expected length of the RT trajectory
RT trajectory	List of global positions	Estimated path of the RT
RT displacement	Vector of floats	Displacement between the origin of the drone formation and the RT trajectory.
Formation speed	Float	Speed for drone formation along RT trajectory
Shooting roles	List of SHOOTING ROLE	Look at Table 2

**Table 2.** Structure for the data type SHOOTING ROLE.

SHOOTING ROLE		
Field Name	Data Type	Comment
Shooting type	Discrete value	Lateral, chase, static, orbit, fly-through, etc.
Framing type	Discrete value	Long shot, medium shot, close up, etc.
Shooting parameters	Set of parameters	e.g., distance to the target in a lateral, angular velocity in an orbit, etc.
Shooting target	Natural number	Identifies the shooting target to film

The Planner would receive a list of these SHOOTING ACTION objects, each with one or multiple SHOOTING ROLE objects included. Then, each individual SHOOTING ROLE represents a task, and the Planner should solve the problem of allocating tasks to drones holding with constraints such as tasks start time and duration, drones' remaining battery, etc. For instance, if a shooting action implements a shot with several drones involved, this is translated into several tasks with the same starting time. Once that assignment is done, the plan for each drone is produced. This plan consists of a list of DRONE ACTION objects, some of them implementing navigation actions and other actual shots, which is specified by an *action type* field. If the action to perform is a shot, all the related information is included a SHOOTING ACTION object with a single SHOOTING ROLE for that drone. Table 3 shows the data structure for the DRONE ACTION objects. Last, the Mission Controller sends the corresponding list of DRONE ACTION objects to each drone and the mission execution can start.

**Table 3.** Structure for the data type DRONE ACTION.

DRONE ACTION		
Field Name	Data Type	Comment
Action type	Discrete value	Takeoff, land, go-to-waypoint or shooting
Action ID	String	Identifies the corresponding shooting action
Start event	String	Identifies the event that triggers this drone action
Shooting action	SHOOTING ACTION	The shooting action if type is "shooting"
Path	List of waypoints	List of waypoints to follow if type is "go to waypoint"

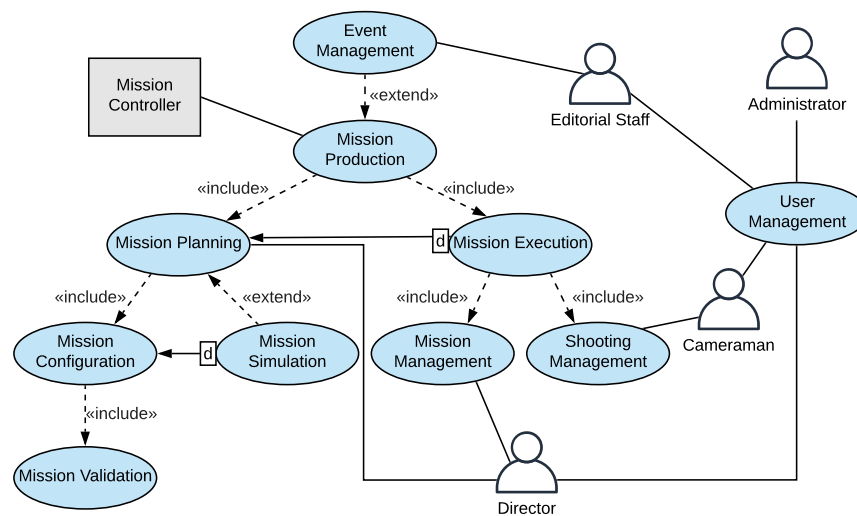
## 5. Director's Dashboard

The director's Dashboard is a Graphical User Interface (GUI) designed specifically to allow the editorial team to define shooting missions in an easy manner during a pre-production phase and interact with the autonomous system during mission execution. We designed this tool for media production in several steps. First, we defined the high-level interactions between the Dashboard and its users, to come up with an UML diagram depicting use cases. Second, we created a database to store all information entities considered in our XML-based cinematography language. Third, we used UML activity diagrams to define the workflows that shall be supported by the Dashboard software, and built upon them the GUI appearance and behavior.

### 5.1. Use Case Analysis

Figure 2 shows the use case breakdown of the Dashboard. The roles involved in the process include both human actors and machine system components (shown, respectively, as man icons and gray rectangles in Figure 2). The director is responsible for defining, leading, and coordinating the editorial shooting missions. The role of the cameramen is to operate the cameras on board the drones, thus replacing their autonomous operation when necessary due to production requirements. The editorial staff deals with the data entry of events, possibly organizing them hierarchically in

sub-events. The system administrator is responsible for effective provisioning, configuration, operation, and maintenance of the Dashboard.



**Figure 2.** High-level use case diagram of the Dashboard.

The Mission Controller is the autonomous module in charge of controlling the mission production workflow. It generates signals upon the occurrence of leaf events in to trigger associated shooting actions on board the drones. Some leaf events may be manually triggered by the director, e.g., to start a race. Others may be generated automatically, e.g., after detecting the appearance of a point of interest. The Mission Controller is also in charge of managing semantic maps, i.e., Keyhole Markup Language (KML) files [20] representing a set of geolocalized features such as landing/take-off spots, no-fly zones, or points of interest. These maps are first created offline during the pre-production phase and displayed on the Dashboard. Then, during mission execution, they might be automatically updated if new features were detected, e.g., obstacle or crowd regions detected by the drone cameras.

The human and machine actors defined above interact with each other in order to create and manage an editorial shooting mission (see Figure 2). There is a higher-level process preceding any shooting mission, which is that of event management, i.e., the process by which events are organized hierarchically, and by which specific events are associated with missions planned to be executed in reaction to them.

The process of mission production is split into mission planning and mission execution. Mission planning is the process in which all aspects related to each mission are defined. These include, e.g., expected starting time and duration, shot types, shooting targets like the leader of a race or a geographic points of interest within the race route, etc. The mission planning process can be further broken down into a mission configuration process, optionally supported by a mission simulation process, which depends on the former. A fundamental subprocess of mission planning is mission validation, i.e., the process by which the flight plan originated by a shooting mission is checked and validated in terms of safety and security.

Mission execution is the process during execution, and it always follows mission planning, i.e., a shooting mission cannot be started if it has not been previously planned and validated. Mission execution is broken down into two distinct subprocesses, namely, mission management and shooting management. Mission management is the process by which the director finally takes decisions about which among the several missions associated with an event will be actually executed, i.e., specific mission and shooting action sequence roles are selected. Shooting management is the process that allows the cameraman to watch the live streams coming from the available video sources (drone cameras) and to change some camera parameters such as zoom, translation and rotation.

## 5.2. Dashboard Implementation

The Dashboard is implemented as a 3-tier Java EE7 application, consisting of, from bottom to top, data access layer, business logic layer, and presentation layer.

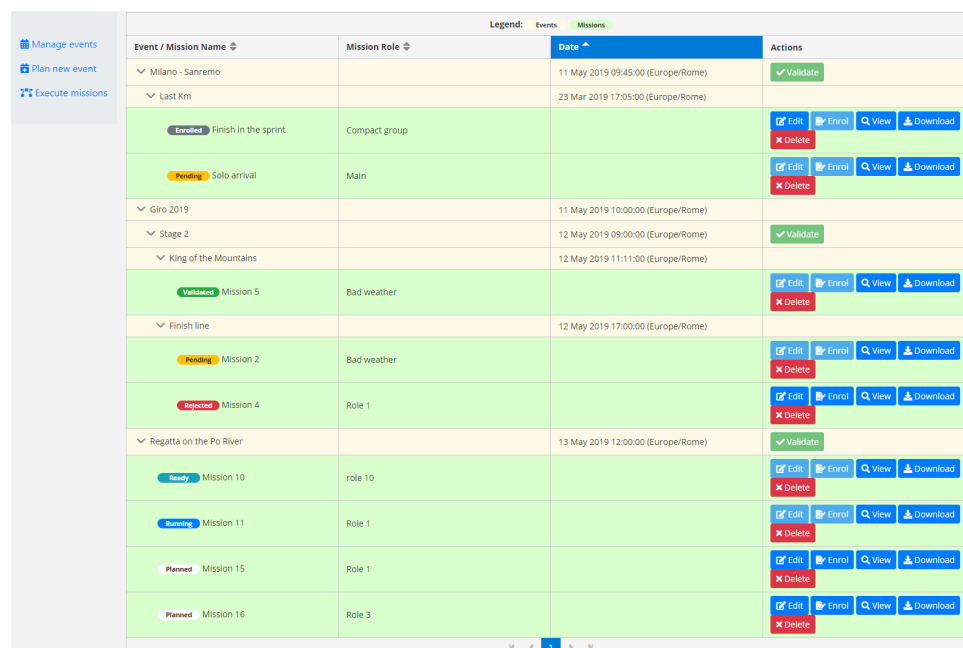
The data access layer communicates with a MySQL database, translating data from/to database format to/from domain format. The database was designed and built based on the information modeling described in Section 4. Java entity beans and data access objects are used to map database entries to Java objects and vice versa. The value of this approach is that if something changes in the database structure, changes are automatically reflected on the application data model.

The business logic layer manages user authentication/authorization, and it performs actions based on user inputs and expected outputs. It provides a set of RESTful HTTP endpoints used to drive most of the functionality of the Dashboard, including administration facilities, as well as CRUD (create, read, update, and delete) operations on the database data.

The presentation layer handles user requests and displays information to users. The GUI consists of a web application based on HTML5, Bootstrap [21], and Angular [22]. The GUI is designed to be responsive, meaning that one may access the platform through any kind of device from smartphones to desktop computers. The main functions of the GUI are described in the following.

### 5.2.1. Director Home Page

Once the user is logged into the system as a director, the homepage with the list of created missions is presented (see Figure 3), together with the related events, role of the mission, and date of the event. The status of each mission is also shown, so that the director may know the next steps of the planning or execution workflows that can be taken for every mission. For example, if a mission is in “Pending” status, it means that the director requested to validate the mission, and the Mission Controller is performing the required safety and security checks. Several actions can be performed from this page, as editing a mission or deleting it. The deletion of a mission implies the deletion of all the content of the mission itself (i.e., any related shooting action sequences and shooting actions). On the left side of the homepage, a menu allows the director to be redirected to the event management section, a specific page of the Dashboard used to manage already created events. Through this menu, the director has also access to the page for creating new events, and to the page for executing validated missions.



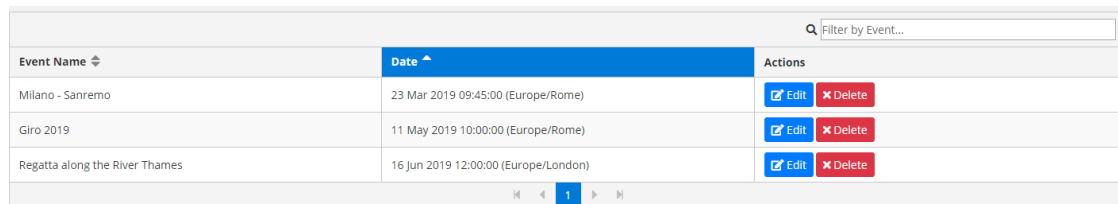
Event / Mission Name	Mission Role	Date	Actions
▼ Milano - Sanremo		11 May 2019 09:45:00 (Europe/Rome)	✓ Validate
▼ Last Km		23 Mar 2019 17:05:00 (Europe/Rome)	✓ Validate
Enrolled Finish in the sprint	Compact group		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
Pending Solo arrival	Main		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
▼ Giro 2019		11 May 2019 10:00:00 (Europe/Rome)	✓ Validate
▼ Stage 2		12 May 2019 09:00:00 (Europe/Rome)	✓ Validate
▼ King of the Mountains		12 May 2019 11:11:00 (Europe/Rome)	✓ Validate
Validated Mission 5	Bad weather		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
▼ Finish line		12 May 2019 17:00:00 (Europe/Rome)	✓ Validate
Pending Mission 2	Bad weather		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
Rejected Mission 4	Role 1		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
▼ Regatta on the Po River		13 May 2019 12:00:00 (Europe/Rome)	✓ Validate
Ready Mission 10	role 10		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
Running Mission 11	Role 1		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
Planned Mission 15	Role 1		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete
Planned Mission 16	Role 3		✗ Edit ✗ Enrol ✗ View ✗ Download ✗ Delete

Figure 3. Example screenshot of the director’s home page.



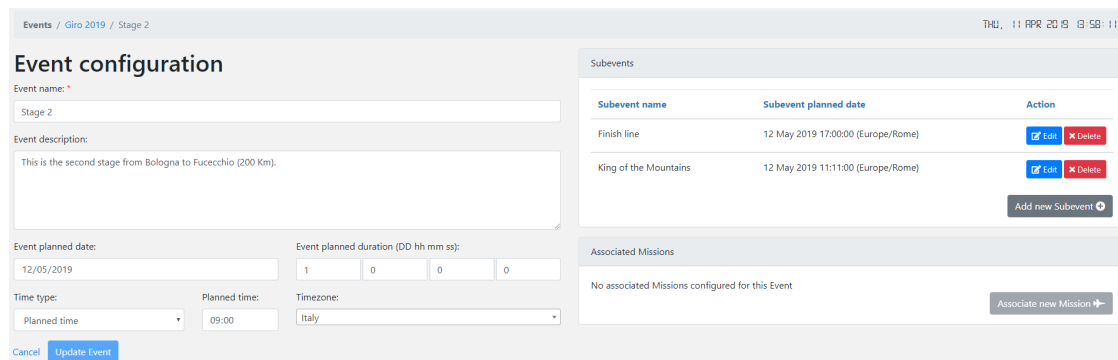
### 5.2.2. Event Management

The event management page shows both information on already created events, such as their planned date and time, and available actions to be performed on single events (i.e., edit and delete). An example is shown in Figure 4. If the director or the editorial staff need to create new events, they can select the “Plan new event” item on the left menu of the director home page. This action will redirect to the event configuration page, where a form containing information about the new event can be filled in and then saved on the database. For each event, users can create new child events or new missions. Figure 5 illustrates an example.



Event Name	Date	Actions
Milano - Sanremo	23 Mar 2019 09:45:00 (Europe/Rome)	<a href="#">Edit</a> <a href="#">Delete</a>
Giro 2019	11 May 2019 10:00:00 (Europe/Rome)	<a href="#">Edit</a> <a href="#">Delete</a>
Regatta along the River Thames	16 Jun 2019 12:00:00 (Europe/London)	<a href="#">Edit</a> <a href="#">Delete</a>

Figure 4. Example screenshot of the event management page.



**Event configuration**

Event name: \*  
Stage 2

Event description:  
This is the second stage from Bologna to Fucecchio (200 Km).

Event planned date: 12/05/2019  
Event planned duration (DD hh mm ss): 1 0 0 0

Time type: Planned time  
Planned time: 09:00  
Timezone: Italy

[Cancel](#) [Update Event](#)

**Subevents**

Subevent name	Subevent planned date	Action
Finish line	12 May 2019 17:00:00 (Europe/Rome)	<a href="#">Edit</a> <a href="#">Delete</a>
King of the Mountains	12 May 2019 11:11:00 (Europe/Rome)	<a href="#">Edit</a> <a href="#">Delete</a>

[Add new Subevent](#)

**Associated Missions**

No associated Missions configured for this Event

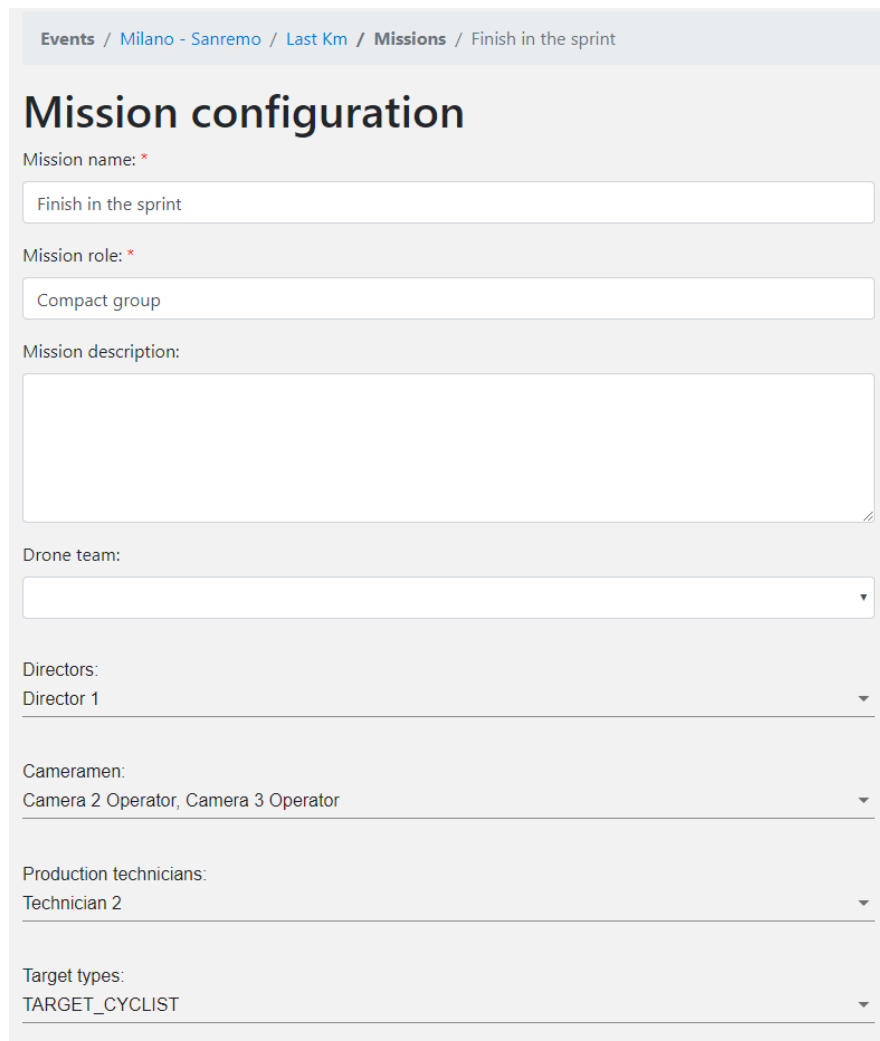
[Associate new Mission](#)

Figure 5. Example screenshot of the event configuration page, used to create a new event.

### 5.2.3. Mission Planning

Once the hierarchy of events has been created, missions for leaf events (i.e., events with no more children) can be configured. Figure 6 shows an example. The event is the shooting of the last kilometer of the “Milano-Sanremo” cycling race. The mission is called “Finish in the sprint” and it will be executed in case of a sprint finish of the race. The crew enrolled in the mission includes four people: one director, two cameramen, and one production technician.

The mission configuration page also shows, if present, a list of SASs related to the mission. The user can either add new SASs or update existing ones by simply filling in a short web-form. For each SAS, the user can add one or more shooting actions, detailing through appropriate web forms what the drone formation will do during the live event and what each single drone will do as well. Three tabs, as shown in Figure 7, compose the shooting action configuration page. The first tab describes the main metadata of the shooting action, like the duration. The second tab allows the user to fill in information about the drone formation. This includes, for example, the RT trajectory on a map and the speed of the formation. The third tab collects specific information about each shooting role in the drone formation, such as the shooting parameters (e.g., drones’ displacements), framing types (e.g., long shot, medium shot, close-up shot, etc.), and shooting target.



Events / Milano - Sanremo / Last Km / Missions / Finish in the sprint

## Mission configuration

Mission name: \*

Mission role: \*

Mission description:

Drone team:

Directors:

Director 1

Cameramen:

Camera 2 Operator, Camera 3 Operator

Production technicians:

Technician 2

Target types:

TARGET\_CYCLIST

**Figure 6.** Example screenshot of the mission configuration page. The people and drone team involved can be specified.

#### 5.2.4. Mission Execution

Once the plans for a shooting mission have been computed and validated, the director can select any valid combination of mission and SAS roles to execute. This is done by triggering a “run” command from the Dashboard, which will notify the Mission Controller to start the selected missions/SASs. The execution of a selected SAS can be monitored in real-time through a dedicated page of the Dashboard, as illustrated in Figure 8. The page shows information about the corresponding mission, including mission metadata (e.g., name, role, date, and a countdown timer), and associated SASs. Selecting a SAS will show the corresponding timelines with the shooting actions involved, as well as the video streams coming from drones cameras, when available. Furthermore, for each drone, a console to manually adjust camera parameters (i.e., gimbal orientation, zoom level, and focus) is shown.

Events / Milano - Sanremo / Last Km / Missions / Finish in the sprint / Shooting Action Sequences / SAS 1 / Shooting Actions / ELEVATOR THU, 11 APR 2019 5:26:42

Shooting Action config. Reference Target and Formation Drones roles config.

Shooting Action type: \*  
ELEVATOR

Shooting Action type description:  
This shot type is categorised by the drone moving straight up (ASCENT) or down (DESCENT). In this case, the camera gimbal will normally rotate, so as to always keep the target properly framed.

Duration (seconds):  
120

Cancel Update SA

---

Events / Milano - Sanremo / Last Km / Missions / Finish in the sprint / Shooting Action Sequences / SAS 1 / Shooting Actions / ELEVATOR THU, 11 APR 2019 5:33:52

Shooting Action config. Reference Target and Formation Drones roles config.

Absolute Azimuth (0° - 360°):  
Relative Azimuth (0° - 360°):  
Formation speed (km/h):  
Location/POI:  
sanremo

Longitude (-180° - 180°):  
7.7772  
Latitude (90° - 90°):  
43.81725

Download KML

Reference Trajectory  
Close Save changes

Cancel Update SA

---

Events / Milano - Sanremo / Last Km / Missions / Finish in the sprint / Shooting Action Sequences / SAS 1 / Shooting Actions / ELEVATOR THU, 11 APR 2019 5:35:07

Shooting Action config. Reference Target and Formation Drones roles config.

Drone role:  
ELEVATOR

Framing Type\*:  
FRAMING\_TYPE\_MCU

Shooting Type\*:  
RULE\_OF\_THIRDS

Target Type\*:  
VIRTUAL

Formation movement mode\*:  
☒ Virtual Trajectory ☐ Virtual Path ☐ Actual Target

The drone moves vertically straight up or down, normally tracking a target or with the camera fixed (e.g. facing vertically downwards).

Shooting parameters - all params express displacements relative to the (orIGIN) formation:  
Z-axis end (meters) - z,e:  
-4.5  
Z-axis start (meters) - z,s:  
1.55

Cancel Update SA

**Figure 7.** Example screenshots of the three tabs of the shooting action configuration page. From top to bottom: main data, drone formation data, and shooting role data.

Events / Giro 2019 / Stage 2 / King of the Mountains

Mission Information

Name:  
Mission 5

Role:  
Bad weather

Planned datetime:  
12 May 2019 11:11:00 (Europe/Rome)

Start in:  
0 0 0 0 0 0  
Years Months Days Hours Minutes Seconds

Shooting Action Sequences

Name	Role
SAS 1	Role 1
SAS 2	Role 2
SAS 3	Role 3

Shooting Action Sequence Timeline

Approach - Establish

Mixed Approach

Priority: 2  
Drones: 3  
Duration: 00:03:20  
Length: n/a  
Formation Speed: 20 km/h

Drone Camera Streams

Drone #1 - Target ("Role": "Race Leader")

Drone #2 - Target ("Role": "Race Leader")

Drone #3 - Target ("Role": "target\_1")

**Figure 8.** Example screenshot of the mission execution page.

## 6. Experiments on Shooting Mission Planning

In this section, we showcase the use of our tools in example scenarios to film outdoor events. Within the framework of MULTIDRONE project, we integrated our tools for autonomous cinematography into a real system with multiple drones. In the following, we first describe an example mission to illustrate the whole process to write, translate, and execute missions. Then, we show the execution of that example shooting mission in a simulator developed in MULTIDRONE. Last, we show some results of a real experimental media production performed in north Germany shooting a rowing race.

### 6.1. Example of a Shooting Mission

This section details an example of a shooting mission to cover an outdoor event with a single moving target to film. The objective is to illustrate the whole process to write the shooting mission through our language in Section 4, how to translate it into a list of tasks to feed a Planner, and a possible output plan computed for each drone.

#### 6.1.1. Shooting Mission XML

The example shooting mission consists of a parent event that is a race with a single target to be filmed by a drone team. There could be several leaf events to associate missions, as for example, the start of the race or the finish line. For simplicity, our parent event contains a single leaf event named “START\_RACE” with a duration of 20 s. Associated with the start of the race there is a mission with role “main” described by two shooting action sequences with the same role “main”, i.e., to be executed in parallel. One of the shooting action sequences has two consecutive shooting actions with a single shooting role each, i.e., single-drone shooting actions. The other shooting action sequence has a shooting action with two shooting roles, i.e., a shooting action to be performed by two drones simultaneously.

Figure 9 shows an example of the XML code. For the sake of simplicity, only the main elements of the XML are included. Tags relative to target information and positioning data (e.g., RT trajectory, origin of formation, shooting target, etc.) have been omitted. A complete version of the XML file of the example (also including alternative missions and shooting action sequences with other roles) can be seen online (<https://grvc.us.es/downloads/docs/example0.xml>).

#### 6.1.2. List of Shooting Actions

Once the Mission Controller receives the XML shown in the previous section, this is parsed to extract the relevant information and build a list of objects of type SHOOTING ACTION and SHOOTING ROLE. The next code block shows the resulting list with most relevant information, where SHOOTING ACTION data type is denoted as SA whereas SHOOTING ROLE data type as SR.

The idea in this example mission is to take a lateral shot with medium-shot framing for 10 s, followed by a static shot with close-up framing. In parallel, an orbital shot with two drones (one with medium-shot framing and another with long-shot framing) should be taken for 20 s. The shooting parameters depend on the type of shot; in this example, they indicate relative positioning of the drone formation with respect to the RT (Cartesian coordinates for the lateral and static shots and polar coordinates for the orbital shot).

In the XML, duration was not specified for shooting actions SA2 and SA3 (values set to 0). Therefore, they are estimated as the difference between the event duration (20 s in the XML) and the duration of previous shooting actions, if any (10 s for SA1). Thus, SA2 will last 10 s, whereas SA3 20 s. When the duration of shooting actions is specified in time, as in this example, the XML tag “length” is set to 0.

```

- <event>
  <description>Example 0, car race.</description>
  <name>Car race</name>
  <plannedStartTime>2019-08-01T09:25:00Z</plannedStartTime>
- <childEvents>
  - <event>
    <uuid>START_RACE</uuid>
    <plannedDuration>20</plannedDuration>
    - <missions>
      - <mission>
        <uuid>M1</uuid>
        <role>main</role>
        - <shootingActionSequences>
          - <shootingActionSequence>
            <role>main</role>
            <uuid>S1</uuid>
            - <shootingActions>
              - <shootingAction>
                <uuid>A1</uuid>
                <duration>10</duration>
                <length>0</length>
                <name>SA1</name>
                - <shootingRoles>
                  - <shootingRole>
                    <shootingTypeName>SHOOT_TYPE_LATERAL</shootingTypeName>
                    <framingTypeName>FRAMING_TYPE_MS</framingTypeName>
                    <params>{"y":-7,"z":8}</params>
                  </shootingRole>
                </shootingRoles>
              </shootingAction>
            - <shootingAction>
                <uuid>A2</uuid>
                <duration>0</duration>
                <length>0</length>
                <name>SA2</name>
                - <shootingRoles>
                  - <shootingRole>
                    <shootingTypeName>SHOOT_TYPE_STATIC</shootingTypeName>
                    <framingTypeName>FRAMING_TYPE_CU</framingTypeName>
                    <params>{"x":5,"y":-2,"z":10}</params>
                  </shootingRole>
                </shootingRoles>
              </shootingAction>
            </shootingActions>
          </shootingActionSequence>
        - <shootingActionSequence>
          <role>main</role>
          <uuid>S2</uuid>
          - <shootingActions>
            - <shootingAction>
              <uuid>A3</uuid>
              <duration>0</duration>
              <length>0</length>
              <name>SA3</name>
              - <shootingRoles>
                - <shootingRole>
                  <shootingTypeName>SHOOT_TYPE_ORBIT</shootingTypeName>
                  <framingTypeName>FRAMING_TYPE_LS</framingTypeName>
                  - <params>
                    { "radius":3, "initial azimuth":0, "z":3, "relative angular speed":0.52 }
                  </params>
                </shootingRole>
              - <shootingRole>
                  <shootingTypeName>SHOOT_TYPE_ORBIT</shootingTypeName>
                  <framingTypeName>FRAMING_TYPE_MS</framingTypeName>
                  - <params>
                    { "radius":3, "initial azimuth":180, "z":3, "relative angular speed":0.52 }
                  </params>
                </shootingRole>
              </shootingRoles>
            </shootingAction>
          </shootingActions>
        </shootingActionSequence>
      </mission>
    </missions>
  </event>
</childEvents>
</event>

```

Figure 9. Example XML describing our race shooting mission.



```

list<SA>:
  SA1:
    start_event: START_RACE
    planned_start_time: 0
    mission_ID: M1
    sequence_ID: S1
    action_ID: A1
    duration: 10
    shooting_roles:
      SR1:
        shooting_type: SHOOT_TYPE_LATERAL
        framing_type: FRAMING_TYPE_MS
        shooting_parameters:
          {'y':-7, 'z':8}
  SA2:
    start_event: -
    planned_start_time: 10
    mission_ID: M1
    sequence_ID: S1
    action_ID: A2
    duration: 10
    shooting_roles:
      SR1:
        shooting_type: SHOOT_TYPE_STATIC
        framing_type: FRAMING_TYPE_CU
        shooting_parameters:
          {'x':5, 'y':-2, 'z':10}
  SA3:
    start_event: START_RACE
    planned_start_time: 0
    mission_ID: M1
    sequence_ID: S2
    action_ID: A3
    duration: 20
    shooting_roles:
      SR1:
        shooting_type: SHOOT_TYPE_ORBIT
        framing_type: FRAMING_TYPE_MS
        shooting_parameters:
          {'initial_azimuth':0,
           'angular_speed':0.52,
           'radius':3, 'z':3}
      SR2:
        shooting_type: SHOOT_TYPE_ORBIT
        framing_type: FRAMING_TYPE_LS
        shooting_parameters:
          {'initial_azimuth':180,
           'angular_speed':0.52,
           'radius':3, 'z':3}

```

### 6.1.3. Generated Plan Per Drone

Once the list from previous section is sent to the Planner, this is in charge of solving the task assignment problem computing a plan fulfilling with time and battery constraints. Shooting actions with several shooting roles, i.e., multi-drone shots, are decomposed into several tasks with the same starting time, one per drone. With the location and timing information from all tasks, as well as the position and battery for the available drones, the Planner computes a schedule solving the task allocation problem. Different planners could be used and its implementation is not the focus of this paper (see [19] for more information). In our example, at least three drones are necessary to implement the shooting mission. One drone could perform two consecutive tasks (corresponding to SA1 and SA2), while the other two drones perform one single task each (corresponding to each of the shooting roles in SA3). The final plan could consist of the following list of DRONE ACTION objects per drone.

```
Drone 1:
  DA1-1:
    action_type: TAKE-OFF
    action_id: -
    start_event: GET_READY
    shooting_action: -
    path: -
  DA1-2:
    action_type: GOTOWAYPOINT
    action_id: A1
    start_event: -
    shooting_action: -
    path: (planned path to SA1
           start position)
  DA1-3:
    action_type: SHOOTING
    action_id: A1
    start_event: START_RACE
    shooting_action: SA1
    path: -
  DA1-4:
    action_type: GOTOWAYPOINT
    action_id: A2
    start_event: -
    shooting_action: -
    path: (planned path to SA2
           start position)
  DA1-5:
    action_type: SHOOTING
    action_id: A2
    start_event: -
    shooting_action: SA2
    path: -
  DA1-6:
    action_type: GOTOWAYPOINT
    action_id: -
    start_event: -
    shooting_action: -
```

```

        path: (planned path to
               landing station)
    DA1-7:
        action_type: LAND
        action_id: -
        start_event: -
        shooting_action: -
        path: -
Drone 2:
    DA2-1:
        action_type: TAKE-OFF
        action_id: -
        start_event: GET_READY
        shooting_action: -
        path: -
    DA2-2:
        action_type: GOTOWAYPOINT
        action_id: A3
        start_event: -
        shooting_action: -
        path: (planned path to SA3
              start position)
    DA2-3:
        action_type: SHOOTING
        action_id: A3
        start_event: START_RACE
        shooting_action: SA3' (as SA3 but
                           only with SR1)
        path: -
    DA2-4:
        action_type: GOTOWAYPOINT
        action_id: -
        start_event: -
        shooting_action: -
        path: (planned path to landing
              station)
    DA2-5:
        action_type: LAND
        action_id: -
        start_event: -
        shooting_action: -
        path: -
Drone 3:
    DA3-1:
        action_type: TAKE-OFF
        action_id: -
        start_event: GET_READY
        shooting_action: -
        path: -
    DA3-2:

```

```

    action_type: GOTOWAYPOINT
    action_id: A3
    start_event: -
    shooting_action: -
    path: (planned path to SA3
          start position)
DA3-3:
    action_type: SHOOTING
    action_id: A3
    start_event: START_RACE
    shooting_action: SA3'' (as SA3 but
                      only with SR2)
    path: -
DA3-4:
    action_type: GOTOWAYPOINT
    action_id: -
    start_event: -
    shooting_action: -
    path: (planned path to landing
          station)
DA3-5:
    action_type: LAND
    action_id: -
    start_event: -
    shooting_action: -
    path: -

```

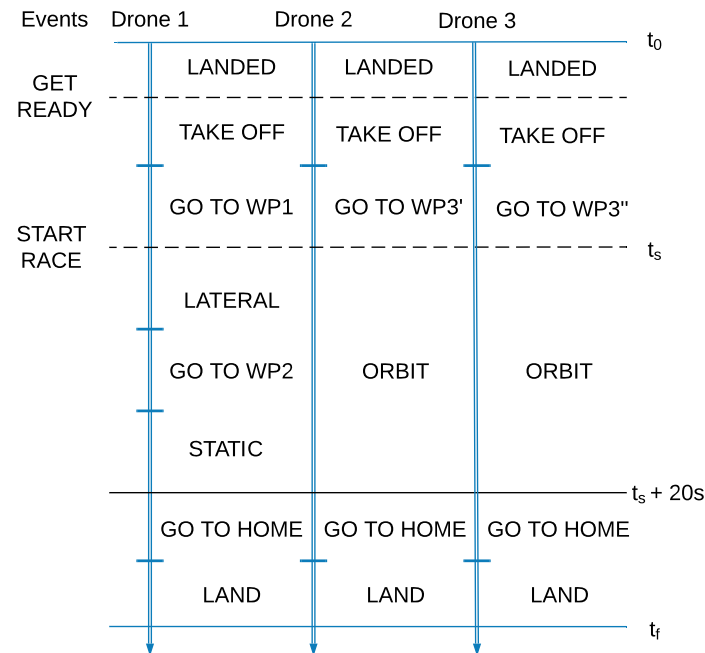
These lists of DRONE ACTION objects are sent to the respective drones. Shooting actions are interleaved with navigation actions. A GET\_READY event has been included. This is generated manually by the director at the beginning of the mission to get the system started and take off the drones. Thus, each drone takes off and navigates to its starting position in its associated shooting action. Then, it waits there for the start event of that shooting action, to trigger the shooting action itself. This process is repeated sequentially until the drone is commanded to land. In other missions, there may be landing operations in between shooting actions to recharge batteries, depending on the drones maximum flight time.

## 6.2. Simulation of Example Mission

This section shows a simulation of the example shooting mission described in the previous section. A simulator developed in MULTIDRONE project is used. It is based on GAZEBO [23] and the PX4 [24] SITL (Software In The Loop) functionality. GAZEBO is a well-known open source multi-robot simulator. It allows for fast robot prototyping and creation of 3D scenarios. PX4 is an open-source autopilot software, implementing a set of functionalities to fly a drone both manually and autonomously. The SITL is a functionality to use PX4 with GAZEBO, simulating drone's sensors such as Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU). Our simulator also uses UAV Abstraction Layer (UAL) [25], an open source library to interact with autonomous drones. With this simulator and our tools for media production, it is easy to run and test different examples without flying real drones but using the very same software that would run in the real system.

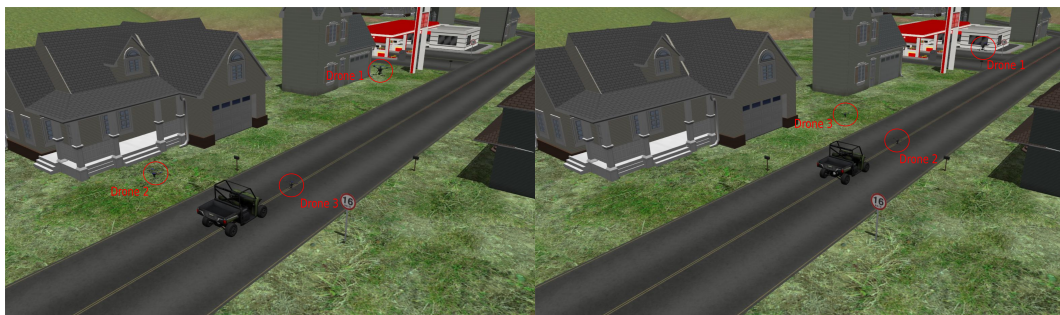
Recall that the mission consists of two shooting action sequences, one with a drone performing first a lateral shot (SA1) and then a static shot (SA2), and another with two drones performing an orbital shot together (SA3). In the simulation, the shooting target is simulated with a car following a straight road that the drones must track and film. Figure 10 depicts the timeline of the executed

mission. Everything starts with the GET\_READY event, which makes all drones take off and go to their corresponding waypoints. Then, the START\_RACE event triggers both sequences. Finally, after 20 s, the drones come back to the home position and land.



**Figure 10.** Execution of the example mission. WP1, WP2, WP3', and WP3'' are drone start positions for SA1, SA2, SA3', and SA3'', respectively.

Figure 11 shows *Drone 1* doing the lateral shot while *Drone 2* and *Drone 3* are orbiting around the target. Figure 12 shows *Drone 1* doing the static shot while *Drone 2* and *Drone 3* are still orbiting (The complete simulation can be seen in: <https://youtu.be/qRPXTid9dFI>).



**Figure 11.** Screenshots of drones executing SA1 and SA3 in our simulation.



**Figure 12.** Screenshots of drones executing SA2 and SA3 in our simulation.



### 6.3. Integration with Real Drones

After validating the system in simulation, we integrated our tools with custom-built drones to run media production on real outdoor events. Figure 13 shows one of our drones, based on an hexarotor platform (120 cm of diagonal distance) with a Pixhawk 2.1 autopilot running PX4 [24]. The payload consists of an Intel NUC computer, a Here + RTK GNSS, a communication board, and a gimbal with a Blackmagic Micro Cinema camera.



**Figure 13.** One of the drones used for autonomous media production.

The MULTIDRONE consortium ran some experimental media production in a countryside area near Bothkamp, in the north of Germany. We tested our system in mock-up, outdoor scenarios, implementing missions (e.g., shooting cyclists and rowers) with one to three drones and different shot types. The objective was to demonstrate the integration of the whole system and to explore the artistic possibilities of our tools for media production.

In particular, Figure 14 depicts some snapshots from an experiment where two drones have to shoot a rowing race in a lake. In that experiment, the original shooting mission consists of a lateral shot that runs concurrently with a fly-through followed by a static panoramic. The first shot is performed by one of the drones, which flies parallel to the boats over the yard. The other two shots are carried out by the other drone, which goes over the lake coming through the trees, and then keeps static following the rowers with the camera. In this case, one of the boats carries a GNSS receiver sending their position to the computers on board the drones, so that the gimbal cameras can track them all time as shooting targets. A video showing the whole shooting mission is available online (<https://www.youtube.com/watch?v=COay0hZsMzk&feature=youtu.be>).



**Figure 14.** Experimental media production of a rowing race with two drones. The images are taken from the cameras on board the drones: the drone taking the lateral shot from the yard (left), and the drone taking the static shot after the fly-through (right).

## 7. Conclusions and Future Work

This paper described a set of tools to support autonomous media production with multiple drones. We contributed with an XML-based language for shooting mission description. The language builds upon new concepts, such as events, shooting actions, or shooting roles, to create media production missions. High-level cinematography principles and different shot types can be implemented. Our language has proved to be flexible and adaptable enough to describe current cinematography operations, as new shots and parameters can easily be incorporated. Moreover, the system is able to bridge the gap between common vocabulary from editorial teams and robotics planning tools. We also contributed with our Dashboard, a graphical tool to support the editorial crew creating and executing shooting missions. The Dashboard has been designed to be intuitive and simple, and users can define complex shooting missions by navigating through a few configuration pages, describe shot geometry by clicking on aerial maps, and validate and execute plans for the missions. This tool generates XML files that can be interpreted by our Mission Controller, to compute mission plans and execute them.

Our system has been integrated and tested in a realistic simulator and during experimental media production with actual drones. As future work, we plan to explore the combination of more complex multi-camera shots to assess the capabilities of multi-drone teams for media production from an artistic and editorial point of view.

**Author Contributions:** Conceptualization, A.T.-G., J.C., M.M., F.N., and A.M.; software and validation, Á.M.-R., A.T.-G., M.M., and S.M.; writing and editing, A.T.-G., J.C., M.M., and S.M.; supervision, F.N., A.M., and A.O.; funding acquisition, A.M. and A.O. All authors have read and agreed to the published version of the manuscript.

**Funding:** 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.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

CRUD	Create, Read, Update and Delete
CTZ	Control Zone
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IMU	Inertial Measurement Unit
KML	Keyhole Markup Language
RT	Reference Target
RTK	Real Time Kinematic
SA	Shooting Action
SAS	Shooting Action Sequence
SITL	Software In The Loop
SR	Shooting Role
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
XML	Extensible Markup Language

## References

1. Mademlis, I.; Mygdalis, V.; Nikolaidis, N.; Montagnuolo, M.; Negro, F.; Messina, A.; Pitas, I. High-Level Multiple-UAV Cinematography Tools for Covering Outdoor Events. *IEEE Trans. Broadcast.* **2019**, *65*, 627–635. doi:10.1109/TBC.2019.2892585.
2. NBC News. New Drone Video Captures Scale of Haiti Hurricane Damage. 2016. Available online: <http://www.nbcnews.com/video/new-drone-video-captures-scale-of-haiti-hurricane-damage-784114243853> (accessed on 22 February 2020).
3. Charlton, A. Sochi Drone Shooting Olympic TV, Not Terrorists. 2014. Available online: <http://wintergames.ap.org/article/sochi-drone-shooting-olympic-tv-not-terrorists> (accessed on 22 February 2020).
4. Gallagher, K. How Drones Powered Rio's Olympic Coverage. 2016. Available online: <http://www.simulyze.com/blog/how-drones-powered-rios-olympic-coverage> (accessed on 22 February 2020).
5. DJI. 2019. Available online: <https://www.dji.com> (accessed on 22 February 2020).
6. Airdog. 2019. Available online: <https://www.airdog.com> (accessed on 22 February 2020).
7. 3DR. 3DR SOLO. 2019. Available online: <https://3dr.com/solo-drone/> (accessed on 22 February 2020).
8. Yuneec. Yuneec Typhoon 4K. 2019. Available online: <https://us.yuneec.com/typhoon-4k-overview> (accessed on 22 February 2020).
9. Joubert, N.; Goldman, D.B.; Berthouzoz, F.; Roberts, M.; Landay, J.A.; Hanrahan, P. Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *arXiv* **2016**, arXiv:cs.GR/1610.01691.
10. Gebhardt, C.; Hepp, B.; Nägeli, T.; Stevšić, S.; Hilliges, O. 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), San Jose, CA, USA, 7–12 May 2016; pp. 2508–2519. doi:10.1145/2858036.2858353.
11. Joubert, N.; Roberts, M.; Truong, A.; Berthouzoz, F.; Hanrahan, P. An interactive tool for designing quadrotor camera shots. *ACM Trans. Graph.* **2015**, *34*, 1–11. doi:10.1145/2816795.2818106.
12. Lino, C.; Christie, M. Intuitive and efficient camera control with the toric space. *ACM Trans. Graph.* **2015**, *34*, 82:1–82:12. doi:10.1145/2766965.
13. Galvane, Q.; Fleureau, J.; Tariolle, F.L.; Guillotel, P. Automated Cinematography with Unmanned Aerial Vehicles. In *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing*; Eurographics Association: Goslar, Germany, May 2016. doi:10.2312/wiced.20161097.
14. Saeed, A.; Abdelkader, A.; Khan, M.; Neishaboori, A.; Harras, K.A.; Mohamed, A. On Realistic Target Coverage by Autonomous Drones. *arXiv* **2017**, arXiv:1702.03456.
15. Nägeli, T.; Meier, L.; Domahidi, A.; Alonso-Mora, J.; Hilliges, O. Real-time planning for automated multi-view drone cinematography. *ACM Trans. Graph.* **2017**, *36*, 1–10. doi:10.1145/3072959.3073712.
16. VC Technology. Litchi. 2019. Available online: <https://flylitchi.com/> (accessed on 22 February 2020).
17. Garage, A. Skywand. 2016. Available online: <https://skywand.com/> (accessed on 22 February 2020).
18. FreeSkies. FreeSkies CoPilot. 2016. Available online: <http://freeskies.co/> (accessed on 22 February 2020).
19. Torres-González, A.; Capitán, J.; Cunha, R.; Ollero, A.; Mademlis, I. A multidrone approach for autonomous cinematography planning. In Proceedings of the Iberian Robotics Conference (ROBOT'), Seville, Spain, 22–24 November 2017.
20. Open Geospatial Consortium. OGC KML 2.3. 2015. Available online: <http://docs.opengeospatial.org/is/12-007r2/12-007r2.html> (accessed on 22 February 2020).
21. Otto, M.; Thornton, J. Bootstrap. 2018. Available online: <https://getbootstrap.com/> (accessed on 22 February 2020).
22. Google. Angular. 2018. Available online: <https://angular.io/> (accessed on 22 February 2020).
23. Koenig, N.; Howard, A. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 28 September–2 October 2004; pp. 2149–2154.

24. Meier, L.; Gubler, T.; Oes, J.; Sidrane, D.; Agar, D.; Küng, B.; Babushkin, A.; px4dev; Charlebois, M.; Bapst, R.; et al. PX4/Firmware: v1.7.3 Stable Release. 2018. Available online: <https://doi.org/10.5281/zenodo.1136171> (accessed on 22 February 2020).
25. Real, F.; Torres-González, A.; Ramón-Soria, P.; Capitán, J.; Ollero, A. UAL: An Abstraction Layer for Unmanned Aerial Vehicles. In Proceedings of the 2nd International Symposium on Aerial Robotics, Philadelphia, PA, USA, 11–12 June 2018.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).