Aerial Multi-Camera Robotic Jib Crane

Patricio Moreno[1], Juan F. Presenza[2], Ignacio Mas[3], and Juan I. Giribet[4]

***Abstract*—A formulation based on a team of unmanned aerial vehicles operating as a fully articulated multi-camera jib crane is proposed for the application of aerial cinematography. An optimization-based controller commands the formation to follow an artistic trajectory defined by the director of photography, while actively avoiding collisions and cameras' mutual visibility. The proposed scheme, based on the cluster-space formulation, presents an intuitive way of maneuvering the virtual camera fixture while automatically adjusting the motions by imposing artistic and safety constraints, facilitating the operator task.**

***Index Terms*—Aerial videography, autonomous cinematography, aerial robotics, multi-robot systems, cluster-space control**

I. Introduction

EVERY day there are more applications in which unmanned aerial vehicles (UAVs) are used for filming. Examples of these uses can be found in the film industry, news coverage, sporting events, etc., where UAVs are used to replace camera systems, such as helicopter-mounted cameras, cable-suspended cameras, dollies, and cranes. The use of UAVs reduces costs, allows a rapid relocation of the system to different types of shots, while it adds a new universe of shot types that are not suitable for the aforementioned camera systems. However, the use of UAVs in professional filming requires expert and qualified pilots for its use, and safety systems in the event of failures of a UAV. Progress is being made towards autonomous filming systems, such as the use of mobile robotic systems.

Multi-agent systems is one of the fields within robotics that autonomous cinematography can benefit from. Formation control strategies, where spatial constraints are defined among agents, are a powerful tool in cooperative aerial missions, as surveyed by [1]. Leader-follower configurations are one of the most common techniques found in multi-robot applications. This configuration defines one robot to be the leader, which follows a control objective, and one or more follower robots whose trajectory depends on that of the leader. An alternative approach, the cluster-space formulation [2] is an application oriented strategy where the formation represents a virtual entity described by state variables. Typically, this can be thought as an articulated mechanism that can be rotated, scaled, reshaped, and moved over time.

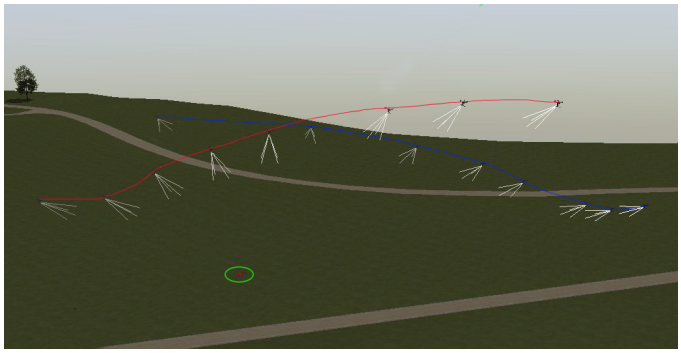


Fig. 1: Cameras mounted on two UAVs tracking a target (green circle) as they follow a trajectory that considers both artistic and collision constraints.

In the aerial cinematography field, UAVs are typically given the task of following a trajectory as close as possible, in order to produce a desired scene. Different trajectories may be defined for particular types of shots, and although it is not required, they might be specified in terms of a Point of Interest (POI), which can be a moving object or person to be filmed. Additionally, with the video feedback naturally available by the task, the director of photography (DOP) may want to adjust the trajectory in-flight, directly commanding the vehicles. We call this trajectory "artistic" in the sense that it is generated from artistic guidelines, as described in [3]. If multiple UAVs are being simultaneously used, the resulting trajectories must avoid collisions and mutual visibility, *i.e.*, UAVs appearing on another UAV camera's field-of-view (FOV). To accomplish this without any aid of the control system, the camera operators would have to coordinate their commands while taking into consideration the surroundings, which is a difficult task in dynamic environments. In order to make this possible, a control system must guarantee these requirements.

Several methods exist for single UAV trajectory generation [3]–[8]. A common approach is to generate position trajectories offline and control inputs in 3D euclidean space, which are then fed to an online controller. To accomplish this planning strategy, multiple inputs and constraints must be taken into account —typically as an optimization problem.

In [3], [7], the trajectory generation considers artistic principles (*e. g.* shot scale, screen position, and viewpoint angle), as well as obstacle and mutual visibility avoidance policies. In particular, [3] implements an unconstrained optimization prob-

Manuscript received: October, 15, 2020; Revised January, 14, 2021; Accepted February, 16, 2021.

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers' comments. This work was partially supported by Universidad de San Andres, Argentina, PDE-18-2019 and UBACyT 0421 projects from UBA, and CCUT-7731TC from UTN-FRSN, Argentina. Patricio Moreno and J. Presenza would like to thank the Peruilh Foundation and UBA for their Ph.D. scholarships.

[1]Patricio Moreno is with the Universidad de Buenos Aires (UBA), Facultad de Ingeniería, Departamento de Ingeniería Electrónica, Laboratorio de Automática y Robótica (LAR–FI–UBA), Argentina. pamoreno@fi.uba.ar

[2]Juan F. Presenza is with the LAR–FI–UBA, and Grupo de Redes Complejas y Comunicación de datos (CNet). jpresenza@fi.uba.ar

[3]Ignacio Mas is with CONICET and Instituto Tecnológico de Buenos Aires (ITBA), Argentina. imas@itba.edu.ar

[4]Juan I. Giribet is with LAR–FI–UBA and CONICET, Argentina. jgiribet@fi.uba.ar

Digital Object Identifier (DOI): see top of this page.

lem for online trajectory generation over long time horizons, so as to update the generated reference, which is then fed into a PID controller.

The work by Nägeli *et al.* [9] solves multiple optimization problems online for multiple UAVs following artistic principles, adding a cost to avoid mutual visibility of the cameras, and a constraint to avoid collisions. Marques *et al.* [10], consider a moving target tracking problem, with multiple UAVs, and solve offline for trajectory and command input generation, and online for command input only.

Some approaches to reduce the dimensionality of the optimization problem assume an independent controller for the camera [3], [10], which solves the aiming problem, or to consider less degrees of freedom (DOF) for the UAV or the camera [7], [9].

Unlike the aforementioned literature, in this work, a formation of UAVs is imagined as an articulated jib crane with multiple cameras, which can be remotely controlled by one or more operators, in a similar way as it is done nowadays with cranes of one camera (Fig. 1). Consequently, we propose a cluster-space formulation as the coordination technique for the camera-carrying UAVs. It has been shown that this strategy allows a single pilot to successfully command a UAV formation [11]. Moreover, the cluster-space formulation is versatile in the sense that the states definition can be changed depending on the task, for example, on the type of shot desired. Thus, a proper cluster state definition allows to work directly with application-oriented parameters for both specification and control. For example, typical formation trajectories can be specified with minimum amount of parameters. Following from the camera motion types defined in [12], it is easy to see that an orbital motion of the cameras—the UAVs circling around a point—can be specified solely with a rotation of the jib crane around its center. Two possible cluster state definitions for aerial filming are presented in section IV to illustrate this approach.

To generate compensation actions to guide the crane motion we propose a closed-loop control strategy based on an optimization problem. Due to the nature of the application, this scheme is appropriate since it can deal with compromises between trajectory tracking, multiple constraints, and different filming objectives. The proposed optimization problem, developed in section III, benefits from the cluster-space technique, as variables natural to the problem may be chosen for the articulated virtual crane, leading to a straightforward formulation. Moreover, the problem is solved simultaneously for the formation of UAVs equipped with actuated gimbals, giving a locally optimal solution for all the trajectory parameters, from a cinematographic perspective.

Our contributions rely on a technique that simplifies specification and control of multiple robots in aerial cinematography, giving the optimization problem for a multi-robot system as a jib crane. The proposed scheme presents an intuitive way, for crane operators, to maneuver multiple cameras using a virtual entity that is natural to the application, controlling the system states with variables defined for the cinematography task, rather than the vehicles positions. Moreover, both collision and artistic constraints are automatically contemplated, a functionality that provides safety and helps reducing scene re-shooting, further simplifying the operator's tasks.

The proposed method is validated on simulations. For this, the controller was implemented in ROS using the PX4 firmware and simulated in Gazebo. The results are discussed in Section V, and Section VI concludes this work with a discussion and future work.

II. Cluster-Space formulation

The cluster space approach [2] considers a group of robots as a single entity, a cluster, defined by state variables that capture relevant information for the application. Consider an n-robot cluster where, without loss of generality, each robot has p degrees of freedom, then the robot-space state vector is $\boldsymbol{r} \in \mathbb{R}^m$, where $m = np$. Let $\boldsymbol{c} \in \mathbb{R}^m$ be a state vector corresponding to the cluster variables. These states are related to the robot space states through m forward kinematic transformations $\text{fwd}_k(\boldsymbol{r})$, with $k = 1, \ldots, m$. The m inverse position kinematic transformations, denoted $\text{inv}_k(\boldsymbol{c})$, relate the k-th robot state parameter to the cluster parameters. These equations can be written as

$$\boldsymbol{c} = \text{FWD}(\boldsymbol{r}) = \begin{bmatrix} \text{fwd}_1(\boldsymbol{r}) & \cdots & \text{fwd}_m(\boldsymbol{r}) \end{bmatrix}, \tag{1}$$

$$\boldsymbol{r} = \text{INV}(\boldsymbol{c}) = \begin{bmatrix} \text{inv}_1(\boldsymbol{c}) & \cdots & \text{inv}_m(\boldsymbol{c}) \end{bmatrix}. \tag{2}$$

Now, let $\boldsymbol{J}(\boldsymbol{r})$ be the jacobian matrix obtained from (1), and $\boldsymbol{J}^{-1}(\boldsymbol{c})$, the jacobian matrix obtained from (2), the mappings between the velocities are, $\dot{\boldsymbol{c}} = \boldsymbol{J}(\boldsymbol{r})\dot{\boldsymbol{r}}$ and $\dot{\boldsymbol{r}} = \boldsymbol{J}^{-1}(\boldsymbol{c})\dot{\boldsymbol{c}}$, respectively.

III. Cluster-based Autonomous Filming Optimization

Consider an n-robot cluster $\mathcal{C} = \{0, \ldots, n-1\}$, and let $\boldsymbol{c}_k \in \mathbb{R}^m$ be the cluster state vector at time t_k. The state vector components will depend on the number of UAVs used and on the specific definition for the type of shot desired, however the formulation that follows is valid for any cluster state vector. Also, the artistic trajectory input, $\boldsymbol{c}_k^a \in \mathbb{R}^m$, indicates the desired cameras viewpoint as a result of combining the shot specifications, usually defined offline, and run-time maneuvers commanded by the operator to modify the shot. Additionally, the system considers the time-varying POI's position $\boldsymbol{s}_k \in \mathbb{R}^3$ and velocity $\dot{\boldsymbol{s}}_k$.

A control strategy is proposed to generate compensation actions $\boldsymbol{u}_k$ to guide the crane motion while autonomously contemplating the application's constraints. This scheme is based on the optimization problem stated in (3) where the function to be minimized is defined as the sum of three costs.

The cost $\mathcal{J}_a$ penalizes deviations from the artistic intention of the DOP, $\mathcal{J}_{\text{aim}}$ weights the error of all agents aiming at the POI, and $\mathcal{J}_u$ quantifies the control signal magnitude and smoothness. Also, the cluster state and control action are restricted to admissible sets, with functions to account for collision and mutual visibility avoidance as well as avoiding the saturation of the actuators. Cost functions and constraints are detailed in the following sections, where time dependence will be omitted unless necessary.

The following optimization problem is solved, at each t_k, for a receding horizon $\{t_{k,h}\}_{h=0}^{N-1}$, where $t_{k,h} = t_k + h\Delta t$, with time step Δt and length N.

$$\begin{aligned}\min_{\boldsymbol{u}_k} \sum_{h=0}^{N-1} & \mathcal{J}_a(\boldsymbol{c}_{k,h}, \dot{\boldsymbol{c}}_{k,h}, \boldsymbol{c}^a_{k,h}, \dot{\boldsymbol{c}}^a_{k,h}) \\ & + \mathcal{J}_{\text{aim}}(\boldsymbol{c}_{k,h}, \boldsymbol{s}_{k,h}) + \mathcal{J}_u(\boldsymbol{u}_k)\end{aligned} \tag{3}$$

$$\begin{aligned}
\text{s.t. } & \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max} && \text{(admissible effort)} \\
& \boldsymbol{c}_{k,h+1} = f(\boldsymbol{c}_{k,h}, \boldsymbol{u}_k) && \text{(cluster dynamics)} \\
& \boldsymbol{c}^a_{k,h+1} = f(\boldsymbol{c}^a_{k,h}, \dot{\boldsymbol{c}}^a_{k,h}) && \text{(artistic trajectory)} \\
& \boldsymbol{s}_{k,h+1} = g(\boldsymbol{s}_{k,h}, \dot{\boldsymbol{s}}_{k,h}) && \text{(POI expected path)} \\
& F_{\text{ca}}(\boldsymbol{c}_{k,h}, \boldsymbol{s}_{k,h}) > 0 && \text{(collision avoidance)} \\
& F_{\text{mva}}(\boldsymbol{c}_{k,h}, \gamma) > 0 && \text{(mutual visibility avoidance)}
\end{aligned}$$

The optimal solution, $\boldsymbol{u}^\star$, is a compensation signal that operates in the different degrees of freedom of the jib crane, and through the cluster space control formalism can be transformed into commands for the stabilized UAVs and their actuated gimbals. The loop is closed by feeding the controller at t_k with the cluster state $\boldsymbol{c}_{k,0} = \boldsymbol{c}_k$ and velocity $\dot{\boldsymbol{c}}_{k,0} = \dot{\boldsymbol{c}}_k$.

A. Artistic-based cost functions

The cost $\mathcal{J}_a$ defined in (4), used to penalize divergence from the desired artistic trajectory, contains two terms.

$$\mathcal{J}_a(\boldsymbol{c}, \dot{\boldsymbol{c}}, \boldsymbol{c}^a, \dot{\boldsymbol{c}}^a) = \|\boldsymbol{c} - \boldsymbol{c}^a\|^2_{\boldsymbol{Q}_c} + \|\dot{\boldsymbol{c}} - \dot{\boldsymbol{c}}^a\|^2_{\boldsymbol{Q}_v}. \tag{4}$$

The first term quantifies the divergence between cluster state $\boldsymbol{c}$ and that specified by the desired viewpoint, expressed by vector $\boldsymbol{c}^a$. The second one measures the difference between the input velocity $\dot{\boldsymbol{c}}^a$ and the cluster velocity $\dot{\boldsymbol{c}}$, by means of a quadratic function. These distances are induced by positive definite matrices $\boldsymbol{Q}_c, \boldsymbol{Q}_v \in \mathbb{R}^{m\times m}$, typically diagonal.

The cost $\mathcal{J}_{\text{aim}}(\boldsymbol{c}, \boldsymbol{s})$, is used to maintain a framing objective. Let $\hat{\boldsymbol{p}}_{is}$, be the unit vector in $\mathbb{R}^3$ that point from the i-th camera to the POI, we want to align $\hat{\boldsymbol{p}}_{is}$ with $\hat{\boldsymbol{n}}_i$, the unit vector of the image plane of camera i. The deviation from the desired aiming can be penalized simultaneously for all cameras, with the cost

$$\mathcal{J}_{\text{aim}}(\boldsymbol{c}, \boldsymbol{s}) = \sum_{i\in\mathcal{C}} q_i \left\|\hat{\boldsymbol{n}}_i - \hat{\boldsymbol{p}}_{is}\right\|^2, \tag{5}$$

which is greater than zero, except when every $\hat{\boldsymbol{n}}_i$ and $\hat{\boldsymbol{p}}_{is}$ are aligned ($\mathcal{J}_{\text{aim}} = 0$). The weight parameter $q_i > 0$ can be modified in order to adjust the relative importance of aiming for each vehicle.

B. Smoothness and control effort cost function

In order to generate smooth trajectories we penalize the norm of the control vector $\boldsymbol{u}_k$, and D derivatives of the predicted cluster trajectory, $\tilde{\boldsymbol{c}}(\boldsymbol{u}_k)$, for the horizon. Therefore, dropping the dependence of $\tilde{\boldsymbol{c}}$ on $\boldsymbol{u}_k$, we define the cost

$$\mathcal{J}_u\big(\boldsymbol{u}_k, \tilde{\boldsymbol{c}}\big) = \|\boldsymbol{u}_k\|_{\boldsymbol{Q}_u} + \sum_{d=1}^{D} \|D_d(\tilde{\boldsymbol{c}})\|_{\boldsymbol{Q}_d} \tag{6}$$

where $D_d(\cdot)$ is the d discrete difference operator, and $\boldsymbol{Q}_d$ is a positive definite matrix. This cost can also be written in alternative forms as in [13, Appendix A].

C. Model-based motion prediction

We propose evolution models for the dynamical systems involved in (3), *i.e.* those regarding the functions f and g.

These models are used, at each instant t_k, to update the expected formation and artistic trajectory and POI's expected path, which are used to predict corresponding costs and constraints. A cluster dynamic model, is developed in section IV-A due to its dependence on the cluster state choice. This model also is used to estimate future desired states of the artistic trajectory. Regarding the POI, our approach predicts its motion to instruct the cameras' aiming. Based on position and velocity measurements of a subject at time t_k, and assuming constant velocity, a function g can be derived as follows. If the POI position $\boldsymbol{s}$ is defined with its coordinates referred to inertial reference frame, then g it is an integrator. But if it's chosen w.r.t. a moving reference frame, or in spherical coordinates, it is necessary to perform the corresponding time derivatives in a similar way as it s done in section IV-A.

D. Collision avoidance

Let $f_{ij}(\boldsymbol{c})\colon \mathbb{R}^m \to \mathbb{R}$ and $f_{is}(\boldsymbol{c}, \boldsymbol{s})\colon \mathbb{R}^m \times \mathbb{R}^3 \to \mathbb{R}$ be inter-robot and robot-to-subject collision avoidance functions, respectively, where $i, j \in \mathcal{C}$ and $j > i$. These functions are defined as

$$f_{ij}(\boldsymbol{c}) = \|\boldsymbol{p}_{ij}\| - R_{\text{s}}, \qquad f_{is}(\boldsymbol{c}) = \|\boldsymbol{p}_{is}\| - R_{\text{s}}, \tag{7}$$

where $\boldsymbol{p}_{ij}$ is the vector pointing from i-th to the j-th robot. Then, the collision avoidance constraints for the cluster are written as $F_{\text{ca}}(\boldsymbol{c}, \boldsymbol{s}) > \boldsymbol{0}$, where this function comprises all inter-robot collisions and all robot-to-subject collisions. The cluster-space approach may simplify the expressions of these functions, however, it will depend on the cluster definition, as shown in section IV.

E. Mutual visibility avoidance

The controller must avoid situations where any vehicle enters into another vehicle's field of view (FOV). Let γ be an angle equal to half of the FOV (Fig. 2), then we define mutual visibility constraints $g_{ij}(\boldsymbol{c}, \gamma)$ representing the condition where vehicle j is in the FOV of vehicle i, for each vehicle. Fig. 2 shows a schematic top view of UAV$_i$ with a camera and UAV$_j$ near its field of view.

To ensure that mutual visibility is avoided, the angle between the unitary vectors $\hat{\boldsymbol{p}}_{ij}$ and $\hat{\boldsymbol{n}}_i$ must be greater than the angle $\tilde{\gamma} = \gamma + \epsilon$, for some small $\epsilon > 0$. This condition is equivalent to

$$g_{ij}(\boldsymbol{c}, \tilde{\gamma}) = \cos(\tilde{\gamma}) - \hat{\boldsymbol{n}}_i^T \hat{\boldsymbol{p}}_{ij} > 0. \tag{8}$$

The ϵ angle is included to take into account uncertainties, such as in the UAVs positions or in the cameras orientations.

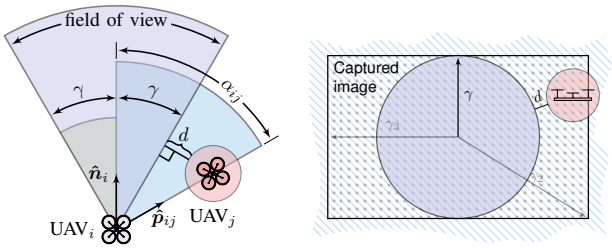
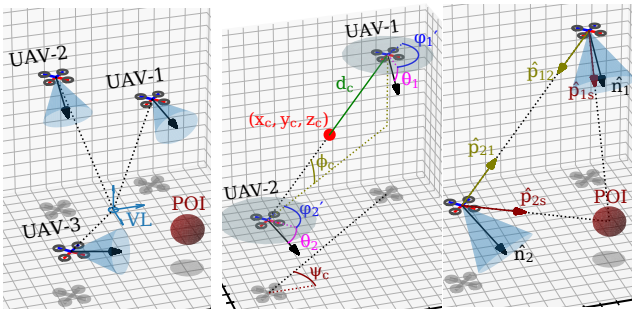


Fig. 2: The camera's FOV is represented as a right cone with apex at the camera's image plane center and its axis aligned with the normal of the image plane ($\hat{\boldsymbol{n}}_i$).

IV. Cluster Definition for Autonomous Filming

In this letter, it is assumed that the formation consists of stabilized UAVs with four degrees of freedom: three corresponding to position (x_i, y_i, z_i) and one to the yaw angle (ψ_i), which is assumed to remain constant, since we control the pan angle of the camera. Each vehicle is equipped with a camera mounted on an actuated 3-axis gimbal, bearing a stabilized pan-tilt (ρ_i, θ_i) camera, where the roll angle stays leveled at all times.

Depending on the task required and on the number of UAVs employed, distinct cluster state choices are possible, allowing different interpretations of the formation parameters. This abstraction provides the DOP and camera operators with a natural way of commanding the group of robots as an articulated virtual mechanism. For a small number of UAVs, it can be useful to choose geometric figures to represent the team, such as a segment ($n = 2$) or a triangle ($n = 3$), as described in section IV-B. For teams with a large number of UAVs, a cluster space definition based on leader-follower formation control might be more appropriate, as shown in the following section. Schematic views of the aerial jib crane for two cluster-space definitions are shown in Fig. 3a.



(a) Cluster Space state representations. (b) Aiming.

Fig. 3: Schematic views of the formation and the POI.

A. Formation definition for n agents

A cluster definition for autonomous filming using an arrangement of n vehicles can be specified as a multi-camera jib with its arms linked by a central pivot that we call virtual leader (VL) that can move and rotate. For many vehicles, it is natural to describe the position of the vehicles in spherical coordinates with respect to a reference frame fixed to the VL (VLRF). This allows to translate and rotate the formation as a virtual rigid mechanism with simple and intuitive commands (translations and rotations of the VL).

Let $\boldsymbol{p}_L, \boldsymbol{\xi}_L$ in $\mathbb{R}^3$ be the position and orientation of the VLRF, with respect to an earth-fixed, global reference frame. Each UAV state can be specified, in the robot space, with its cartesian position $\boldsymbol{p}_i$ along with camera's zoom, tilt and pan angles as in $\boldsymbol{\xi}_i = (\zeta_i, \theta_i, \rho_i)^T$. A convenient definition of the cluster space state consists of each UAV's position in spherical coordinates with respect to VLRF, *i.e.* distance to the VL, elevation and azimuth angles represented by $\boldsymbol{q}_i = (d_i, \epsilon_i, \alpha_i)^T$; and n vectors $\boldsymbol{\eta}_i \in \mathbb{R}^3$ where each represents the i-th camera's aiming expressed in the VLRF. The relationship between the robot and cluster space states is given by

$$\begin{aligned} \boldsymbol{c}_i &= \begin{bmatrix} \boldsymbol{q}_i \\ \boldsymbol{\eta}_i \end{bmatrix} = \mathrm{fwd}_i(\boldsymbol{r}) = \begin{bmatrix} \boldsymbol{\phi}\left(\boldsymbol{R}_L^T(\boldsymbol{p}_i - \boldsymbol{p}_L)\right) \\ \boldsymbol{R}_L^T \boldsymbol{\phi}^{-1}(\boldsymbol{\xi}_i) \end{bmatrix}, \\ \boldsymbol{r}_i &= \begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{\xi}_i \end{bmatrix} = \mathrm{inv}_i(\boldsymbol{c}) = \begin{bmatrix} \boldsymbol{R}_L \boldsymbol{\phi}^{-1}(\boldsymbol{q}_i) + \boldsymbol{p}_L \\ \boldsymbol{\phi}\left(\boldsymbol{R}_L \boldsymbol{\eta}_i\right) \end{bmatrix}, \end{aligned} \tag{9}$$

where $\boldsymbol{\phi}$ is the mapping from Cartesian to spherical coordinates, and $\boldsymbol{R}_L = \boldsymbol{R}(\boldsymbol{\xi}_L)$ is the $SO(3)$ matrix that represents the orientation of the VLRF. Individual vectors for each UAV (and for the leader as well) can be generated by defining $\boldsymbol{r}_i = (\boldsymbol{p}_i^T, \boldsymbol{\xi}_i^T)^T$ and $\boldsymbol{c}_i = (\boldsymbol{q}_i^T, \boldsymbol{\eta}_i^T)^T$, that can also be put together into the formation vectors

$$\begin{aligned} \boldsymbol{c} &= \mathrm{FWD}(\boldsymbol{r}) = \begin{bmatrix} \boldsymbol{r}_L^T, & \cdots & \boldsymbol{c}_i^T, & \cdots \end{bmatrix}^T \in \mathbb{R}^{6(n+1)}, \\ \boldsymbol{r} &= \mathrm{INV}(\boldsymbol{c}) = \begin{bmatrix} \boldsymbol{r}_L^T, & \cdots & \boldsymbol{r}_i^T, & \cdots \end{bmatrix}^T \in \mathbb{R}^{6(n+1)}. \end{aligned}$$

The dynamics of the cluster space state, can be obtained by taking the derivatives of (9). Denoting $\boldsymbol{J}_{\boldsymbol{\phi}}$ and $\boldsymbol{J}_{\boldsymbol{\phi}^{-1}}$ the 3×3 jacobian matrices of $\boldsymbol{\phi}$ and $\boldsymbol{\phi}^{-1}$, and let $\boldsymbol{S}(\cdot)$ be the cross-product matrix[1], then the relationship $\dot{\boldsymbol{c}}_i = \boldsymbol{J}(\boldsymbol{r}_i)\dot{\boldsymbol{r}}_i$ is in the form

$$\begin{aligned} \dot{\boldsymbol{q}}_i &= \boldsymbol{J}_{\boldsymbol{\phi}} \boldsymbol{R}_L^T \left[(\dot{\boldsymbol{p}}_i - \boldsymbol{v}_L) - \boldsymbol{S}(\boldsymbol{\omega}_L)(\boldsymbol{p}_i - \boldsymbol{p}_L)\right], \\ \dot{\boldsymbol{\eta}}_i &= \boldsymbol{R}_L^T \left[\boldsymbol{J}_{\boldsymbol{\phi}^{-1}} \dot{\boldsymbol{\xi}}_i - \boldsymbol{S}(\boldsymbol{\omega}_L)\boldsymbol{\phi}^{-1}(\boldsymbol{\xi}_i)\right]. \end{aligned} \tag{10}$$

Note that these expressions are in terms of the linear and angular velocity of the VL, namely $\boldsymbol{v}_L$ and $\boldsymbol{\omega}_L$, which are assumed to be known. To derive a dynamic model in the cluster space of the form $\dot{\boldsymbol{c}}_i = f_i(\boldsymbol{c}_i, \boldsymbol{u}_i)$ we first need to consider the dynamics of vector $\boldsymbol{r}_i$. For stabilized UAVs, they can be modeled as a simple integrator, so that $\dot{\boldsymbol{p}}_i$ and $\dot{\boldsymbol{\xi}}_i$ represent the control input in the robot space. The relationship between these variables and $\boldsymbol{u}_i$ is presented through the jacobian matrix

$$\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{\phi}} \boldsymbol{R}_L^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_L^T \boldsymbol{J}_{\boldsymbol{\phi}^{-1}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{p}}_i \\ \dot{\boldsymbol{\xi}}_i \end{bmatrix}.$$

Using (9), expressions in (10) can be restated as

$$\dot{\boldsymbol{c}}_i = \boldsymbol{u}_i - \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{\phi}} \boldsymbol{R}_L^T \boldsymbol{v}_L + \boldsymbol{J}_{\boldsymbol{\phi}} \boldsymbol{S}(\boldsymbol{R}_L^T \boldsymbol{\omega}_L)\boldsymbol{\phi}^{-1}(\boldsymbol{q}_i) \\ \boldsymbol{S}(\boldsymbol{R}_L^T \boldsymbol{\omega}_L)\boldsymbol{\eta}_i \end{bmatrix}.$$

[1] $\boldsymbol{S} : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ is a function that assigns to a vector x in $\mathbb{R}^3$ a unique skew-symmetric matrix such that $\boldsymbol{S}(x)y = x \times y$, for every $y \in \mathbb{R}^3$.

Then, compiling the control inputs in the vector $\boldsymbol{u} = (\cdots, \boldsymbol{u}_i^T, \cdots)^T \in \mathbb{R}^{6n}$, the cluster state dynamics can be expressed in the compact form

$$\dot{\boldsymbol{c}} = f(\boldsymbol{c}, \boldsymbol{u}, \boldsymbol{v}_L, \boldsymbol{\omega}_L).$$

B. Formation definition for two agents

Using the same definitions as before the formation state vector in the robot space $\boldsymbol{r} \in \mathbb{R}^{10}$ can be defined as

$$\boldsymbol{r} = (x_1, y_1, z_1, \rho_1, \theta_1, x_2, y_2, z_2, \rho_2, \theta_2)^T. \tag{11}$$

In this case, we define the cluster state variables to represent the cluster's centroid (x_c, y_c, z_c) and orientation (ψ_c, ϕ_c), the half distance between agents (d_c) and each camera orientation $(\varphi_1, \theta_1, \varphi_2, \theta_2)$. Now, the formation state vector in the cluster space $\boldsymbol{c} \in \mathbb{R}^{10}$ is

$$\boldsymbol{c} = (x_c, y_c, z_c, \psi_c, \phi_c, d_c, \varphi_1, \theta_1, \varphi_2, \theta_2)^T, \tag{12}$$

where

$$\begin{aligned}
x_c &= \frac{x_1 + x_2}{2},\ y_c = \frac{y_1 + y_2}{2},\ z_c = \frac{z_1 + z_2}{2}, && \text{(centroid)}\\
\psi_c &= \operatorname{atan}\left((x_2 - x_1)/(y_1 - y_2)\right), && \text{(yaw angle)}\\
\phi_c &= \operatorname{atan}\left(\frac{z_1 - z_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right), && \text{(roll angle)}\\
d_c &= \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2 + (z_1 - z_c)^2}, && \text{(half dist.)}\\
\varphi_1 &= \rho_1 - \psi_c, && \text{(heading)}\\
\varphi_2 &= \rho_2 - \psi_c. && \text{(heading)}
\end{aligned} \tag{13}$$

For this cluster, the unitary vectors $\hat{\boldsymbol{p}}_{12}, \hat{\boldsymbol{p}}_{21}$ that point between the agents (Fig. 3b) are given by

$$\hat{\boldsymbol{p}}_{12} = -R_z(\psi_c)R_x(\phi_c)\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}\cos(\phi_c)\sin(\psi_c)\\ -\cos(\phi_c)\cos(\psi_c)\\ -\sin(\phi_c)\end{bmatrix}, \tag{14}$$

and $\hat{\boldsymbol{p}}_{21} = -\hat{\boldsymbol{p}}_{12}$. With these definitions, we can obtain the vectors that point from each camera to the POI as follows

$$\boldsymbol{p}_{is} = \boldsymbol{s} - (\boldsymbol{p}_c + d_c\hat{\boldsymbol{p}}_{ji}), \tag{15}$$

where $i, j \in \{1, 2\}$ and $\boldsymbol{p}_c$ is the cluster centroid. Then, $\hat{\boldsymbol{n}}_i$, the normal vector of the image plane for camera i is written in terms of the cluster state parameters as

$$\hat{\boldsymbol{n}}_i = R_z(\psi_c + \varphi_i)R_y(\theta_i)\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}\cos(\psi_c + \varphi_i)\cos(\theta_i)\\ \sin(\psi_c + \varphi_i)\cos(\theta_i)\\ -\sin(\theta_i)\end{bmatrix}. \tag{16}$$

Next, we derive the formation model equation, the collision avoidance constraints, and mutual visibility avoidance constraint for this cluster definition.

1) Formation model: We model the UAV and gimbal as first order systems and, considering the equations from section II, it follows that $\dot{\boldsymbol{c}} = \boldsymbol{u}$.

2) Collision avoidance constraints: Here we follow the constraints definitions from section III-D. The inter-vehicle collision avoidance is straight forward, as $\|\boldsymbol{p}_{ij}\| = d_c$ and is part of the cluster definition. The robot-to-subject collision constraint uses $\boldsymbol{p}_{is}$, which was obtained in equation (15), for the aiming cost.

3) Mutual visibility avoidance: Although eq. (8) does not change it is worth noting that, using simple trigonometric identities, it can be seen that in cluster- space coordinates eq. (8) is independent of the cluster's center position (x_c, y_c, z_c) and the yaw angle (ψ_c). The remaining variables $(\phi_c, \varphi_1, \varphi_2, \theta_1, \theta_2)$ capture the cameras' relative aiming information and are sufficient to satisfy this cinematographic restriction.

V. Validation

To validate the proposed method, we present simulation results for the described cluster definitions, using an *Orbital shot*, which is a typical shot defined for one or more camera-equipped UAVs [14].

The simulation testbed consists of a small quadrotor that acts as a POI, and a cluster of hexacopters with pan-tilt cameras with a FOV of 120° but configured in the controller as of 60°. The reduced FOV allows us to see on video the behaviour of the UAVs as they approach other's FOV. For all these vehicles, the stabilizing controller is a POSIX version of PX4 stack v1.11.0, which runs within a ROS Kinetic environment and is simulated in Gazebo 7, using the `sitl_gazebo` package from PX4. The proposed controller, implemented using SciPy's *optimize* SLSQP method, runs on a ground station computer, which communicates with the vehicles using a Mavlink interface, provided by the `mavros` package. The system runs inside a container within the OpenUAV Simulation Testbed [15]. Example videos can be found in https://youtu.be/_OZ_zi9vtF8/.

A. Formation definition for two agents

During the orbital shot, the cluster rotates around a point in space —for this test in particular, $(0\,\mathrm{m}, 0\,\mathrm{m}, 10\,\mathrm{m}) \in \mathbb{R}^3$— while the moving target performs an ascending maneuver from $z = 0\,\mathrm{m}$ to $z = 20\,\mathrm{m}$ going through the center of the cluster. In this case, while tracking the target vehicle, the motions force the cameras to aim at each other when all the vehicles are at the same height, stressing the MVA constraint.

Fig. 4 shows the angles between $\hat{\boldsymbol{n}}_i$ and $\hat{\boldsymbol{p}}_{ij}$ or $\boldsymbol{p}_{is}$, that is $\alpha_{12}, \alpha_{1s}, \alpha_{21}, \alpha_{2s}$. For the mutual visibility constraint to hold, α_{12} and α_{21} must be greater than half the extended FOV $(\tilde{\gamma})$, while α_{1s} and α_{2s} move closer to zero the better aimed is the subject. Snapshots of the FPV from the UAVs between $t = 300\,\mathrm{s}$ and $t = 325\,\mathrm{s}$ are shown in Fig. 5.

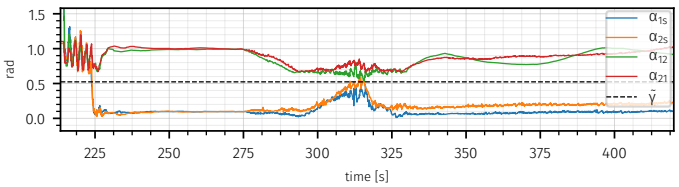


Fig. 4: Angles between the normal of each camera and the possible actors or objects of the scene (the other UAV and the POI) for the orbital shot.

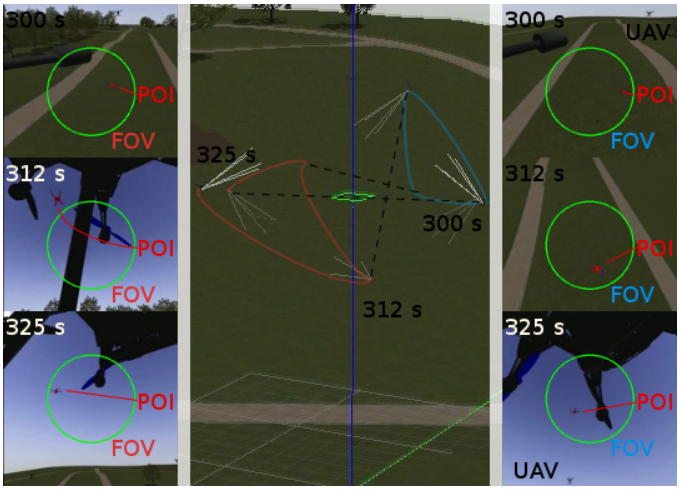


Fig. 5: Approximate trajectory and FPV for $t \in (300, 325)s$.

B. Formation definition for n agents

Similar to the above results, with this cluster state definition, 4 UAVs perform an orbiting motion describing a circle of radius $14.14\,\mathrm{m}$, while aiming a static object $30\,\mathrm{m}$ away from the VL. Despite this motion is easily commanded (only a rotation command in the z-axis of the VLRF), it is a compromising scenario in which every UAV has to repeatedly deviate from the artistic trajectory, in order to satisfy the MVA constraint. Two simulations where performed at different angular velocities, which translates into two tangential velocities of $v_1 = 2.2\,\mathrm{m\,s^{-1}}$ and $v_2 = 4.4\,\mathrm{m\,s^{-1}}$. The angles α_{ij} and α_{is} for $i, j = 1, 2, 3, 4$ are shown in Fig. 6.

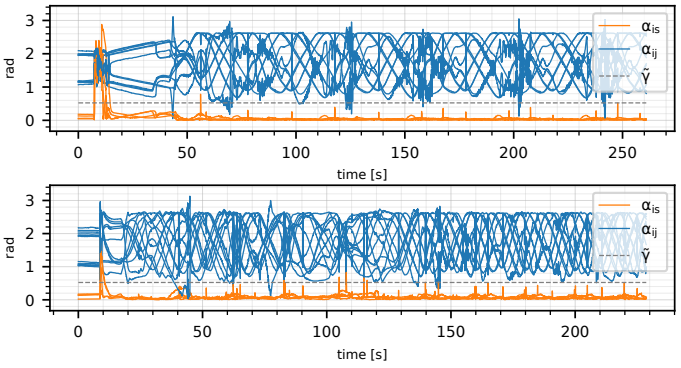


Fig. 6: Angles between the normal of each camera and the possible actors or objects of the scene (other UAVs and the POI). Above: $v_1 = 2.2\,\mathrm{m\,s^{-1}}$, Below: $v_2 = 4.4\,\mathrm{m\,s^{-1}}$.

VI. Conclusions

This work presented an autonomous filming setup based on the cluster-space control formulation. With this scheme, a multi UAV system with pan-tilt cameras is envisioned as an aerial multi-camera robotic jib crane, which can follow offline generated trajectories as well as run-time maneuvers commanded by the operator that alter the specified offline trajectory, while keeping a POI framed. As the cluster state variables are defined based on the application, the controlled parameters —which are the same as those that a ground operator would change for run-time trajectory modifications— are intuitive for the application, this can be seen for the chase or fly-by shots, where the operator moves the cluster's centroid or virtual leader, instead of commanding n vehicles. Moreover, the system guarantees that mutual visibility and collisions are avoided for each UAV, at the cost of deviating from the predefined trajectory, the operator's commands, or POI aiming.

The system was tested first using a pure python approach, then extended to ROS, Gazebo and the PX4 tools. This validation through results using common simulation tools which are known to work for a model continuity development approach, in our experience, has proven in the past to closely relate to the experimental validation of similar setups.

Acknowledgment

The authors would like to thank Harish Anand, from the OpenUAV group for his support using the OpenUAV testbed.

References

[1] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.
[2] C. A. Kitts and I. Mas, "Cluster space specification and control of mobile multirobot systems," *IEEE/ASME Trans. Mechatronics*, vol. 14, no. 2, pp. 207–218, 2009.
[3] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," *arXiv:1808.09563 [cs]*, 8 2018, arXiv: 1808.09563.
[4] C. Lino and M. Christie, "Intuitive and efficient camera control with the toric space," *ACM Trans. Graphics*, vol. 34, no. 4, pp. 1–12, 2015.
[5] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated Cinematography with Unmanned Aerial Vehicles," in *Workshop on Intell. Cinematography and Editing*. Eurographics Association, 2016.
[6] 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:1610.01691 [cs]*, 10 2016, arXiv:1610.01691.
[7] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-Time Motion Planning for Aerial Videography With Dynamic Obstacle Avoidance and Viewpoint Optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696 – 1703, 07 2017.
[8] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K.-T. T. Cheng, "ACT: An Autonomous Drone Cinematography System for Action Scenes," in *IEEE Int. Conf. Robotics and Automation*, 05 2018, pp. 7039–7046.
[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. 132:1–132:10, 2017.
[10] M. Marques, B. J. Guerreiro, R. Cunha, and C. Silvestre, "Trajectory planning and control for drone replacement for multidrone cinematography," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 334 – 339, 2019, 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019.
[11] P. Moreno, S. Esteva, I. Mas, and J. I. Giribet, "Multi-UAV specification and control with a single pilot-in-the-loop," *Unmanned Systems*, vol. 8, no. 3, 2020.
[12] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV Cinematography: A Tutorial and a Formalized Shot-Type Taxonomy," *ACM Comput. Surveys*, vol. 52, no. 5, pp. 1–33, 9 2019.
[13] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, 2020.
[14] A. Torres-González, J. Capitán, R. Cunha, A. Ollero, and I. Mademlis, "A multidrone approach for autonomous cinematography planning," in *Advances in Intelligent Systems and Computing*, 1 2018, pp. 337–349.
[15] M. Schmittle, A. Lukina, L. Vacek, J. Das, C. P. Buskirk, S. Rees, J. Sztipanovits, R. Grosu, and V. Kumar, "OpenUAV: a UAV testbed for the CPS and robotics community," in *ACM/IEEE Int. Conf. Cyber-Physical Systems*, 2018, pp. 130–139.