CineMPC: A Fully Autonomous Drone Cinematography System Incorporating Zoom, Focus, Pose, and Scene Composition

Pablo Pueyo, *Student Member, IEEE*, Juan Dendarieta, Eduardo Montijano, *Member, IEEE*, Ana Cristina Murillo, *Member, IEEE*, and Mac Schwager, *Member, IEEE*

***Abstract*—We present CineMPC, a complete cinematographic system that autonomously controls a drone to film multiple targets recording user-specified aesthetic objectives. Existing solutions in autonomous cinematography control only the camera extrinsics, namely, its position and orientation. In contrast, CineMPC is the first solution that includes the camera intrinsic parameters in the control loop, which are essential tools for controlling cinematographic effects such as focus, depth of field, and zoom. The system estimates the relative poses between the targets and the camera from an RGB-D image and optimizes a trajectory for the extrinsic and intrinsic camera parameters to film the artistic and technical requirements specified by the user. The drone and the camera are controlled in a nonlinear model predicted control (MPC) loop by reoptimizing the trajectory at each time step in response to current conditions in the scene. The perception system of CineMPC can track the targets' position and orientation despite the camera effects. Experiments in a photo-realistic simulation and with a real platform demonstrate the capabilities of the system to achieve a full array of cinematographic effects that are not possible without the control of the intrinsics of the camera. Code for CineMPC is implemented following a modular architecture in ROS and released to the community.**

***Index Terms*—Aerial robotics applications, autonomous drone cinematography, camera intrinsics, model predicted control (MPC).**

I. Introduction

REMOTELY piloted multirotor aircraft have already been widely adopted as mobile camera platforms for videography in television, film, commercial, and hobby applications. These platforms have also expanded the locations from which aerial footage can be obtained to include geographical regions previously inaccessible. However, a skilled human pilot and a human camera operator are still essential to operate these systems.

In this article, we propose CineMPC as a step toward making drone aerial videography truly autonomous. In contrast with other research in autonomous videography [1], [2], CineMPC controls both the camera pose as well as the focus, depth of field (DoF), and zoom—the so-called camera intrinsics—thereby doing the job of both the pilot and the camera operator. We accomplish this through a thin-lens model [3] of the camera optics, which exposes these camera intrinsic properties as control inputs. We, then, optimize a sequence of control inputs for both the camera pose and camera intrinsics while constraining the trajectory to be dynamically feasible for the drone. We close the loop by detecting the poses of the multiple targets in the scene in real-time, and reoptimizing the trajectory in the model predicted control (MPC) loop as new images are acquired. CineMPC

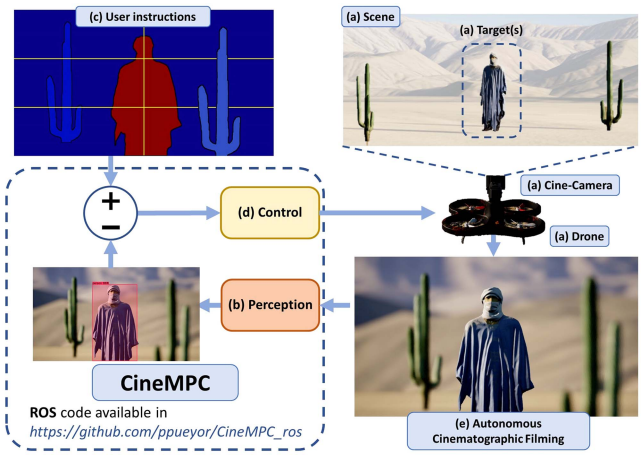

Fig. 1. **CineMPC pipeline.** (a) Drone holds a cinematographic camera, capturing footage of targets in a scene. (b) Perception module processes the recorded images to extract the targets' pose and calculates the error in comparison to user instructions. (c) Visual representation of user instructions, with the focused area highlighted in red, the blurry area in blue, and yellow lines depict the desired image position for the top and lower parts of the target. (d) Calculated error is the input to the control module, which determines the next N step for both the drone and the camera to minimize the error. (e) This process results in a new image acquired by the drone, restarting the loop for continuous refinement, producing autonomous cinematographic filming.

Manuscript received 3 November 2023; accepted 18 December 2023. Date of publication 12 January 2024; date of current version 16 February 2024. This paper was recommended for publication by Associate Editor F. Morbidi and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported in part by the DGA scholarship and project under Grant T45_23R, in part by the Spanish projects under Grant PID2019-105390RB-I00 and Grant PID2021-125514NB-I00, funded by MCIN/AEI/10.13039/501100011033, in part by the ERDF way of making Europe, in part by the European Union NextGenerationEU/PRTR, and in part by ONR under Grant N00014-18-1-2830. *(Corresponding author: Pablo Pueyo.)*

Pablo Pueyo, Juan Dendarieta, Eduardo Montijano, and Ana Cristina Murillo are with the Instituto de Investigación en Ingeniería de Aragón and DIIS, Universidad de Zaragoza, 50009 Zaragoza, Spain (e-mail: ppueyor@unizar.es; 538123@unizar.es; emonti@unizar.es; acm@unizar.es).

Mac Schwager is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: schwager@stanford.edu).

https://github.com/ppueyor/CineMPC_ros.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3353550, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3353550

TABLE I
RELATED WORK: COMPARISON OF EXISTING CINEMATOGRAPHIC PLATFORMS' MAIN PROPERTIES

Existing platforms	Control Extrinsics	Real Perception	Dynamic targets	Multitarget	Image comp.	Obstacle avoidance	Occlusion avoidance	Public ROS Code	Control DoF	Control Intrinsics
CineMPC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Alcantara et al.(2021) [2]	✓	×	✓	×	×	✓	✓	✓	×	×
Huang et al.(2018) [33]	✓	✓	✓	×	···	···	×	×	×	×
Bonatti et al.(2020) [1]	✓	✓	✓	×	···	✓	✓	×	×	×
Bucket et al.(2021) [34]	✓	✓	✓	×	×	✓	✓	×	×	×
Joubert et al.(2016) [29]	✓	×	×	✓	✓	×	×	×	×	×
Nagëli et al.(2017) [31]	✓	×	×	✓	✓	✓	✓	×	×	×

is able to track and record multiple dynamic targets (such as humans, animals, cars, or other aircraft) while taking footage to optimize artistic and technical objectives specified by the user. This framework is depicted in Fig. 1.

The core idea of the control module is to adapt the classic cinematographic concepts [4] to mathematical expressions that can be optimized using control techniques. The specifications are optimized, thanks to a nonlinear MPC formulation that transforms them into instructions to autonomously control the drone and the camera while recording footage. The drone position and orientation are controlled together with the intrinsic parameters of the camera lens in one unified control problem.

CineMPC's perception module identifies and estimates the pose of targets in images captured by a thin-lens cinematographic monocular camera. In images from this camera model, targets may appear blurry or distorted, posing a challenge to pose estimation. To address this issue, the module employs a neural network to extract target positions from RGB-D images and uses a Kalman filter (KF) and vector algebra to determine target orientation based on movement direction. Existing solutions [5], [6], [7] for 3-D orientation estimation with monocular cameras often require invasive wearables or involve heavy deep learning, making them impractical for real-time aerial cinematography of targets in the wild.

We release a modular implementation of the whole solution in robotic operating system (ROS) that also incorporates the tools and instructions to test it using CinemAirSim [8], an extension for cinematographic purposes of the robotics simulator AirSim [9]. In this environment, we conduct a battery of photorealistic experiments that, along with real-world experiments, demonstrate the potential of our approach.

The main contributions of this work are as follows.

1) *Optimal Control Problem:* We propose a novel optimal control problem within an MPC framework. This enables autonomous control not only over the extrinsic but also the intrinsic parameters of a drone and cinematographic camera. This facilitates capturing previously unattained cinematographic effects while handling different constraints.
2) *Integration with Perception Module:* The control solution is integrated with a perception module capable of tracking 3-D poses for multiple moving targets from RGB-D images. This capability remains unaffected by image distortions resulting from the modification of intrinsic camera parameters.
3) *ROS Implementation:* A mature ROS implementation is released with a modular software architecture, facilitating the adaptation of CineMPC to new drones and diverse aerial videography applications.
4) *Practical Implementation Aspects:* We delve into practical considerations associated with implementing a fully autonomous cinematographic platform in a real setup. We present an extensive array of experiments, in scenarios with both a real drone and camera, along with challenging filming sequences conducted in simulation.

This work is *an evolved version of [10]*, extending the core optimal control problem addressed in the conference paper. The control module now handles significant cinematography and robotics constraints, including collisions and occlusions. In addition, we introduce an extra cost term (J_f) to achieve a broader range of cinematographic effects and a low-level controller to ensure smooth trajectory execution. The remaining contributions are primarily introduced in this extended work.

II. RELATED WORK

The design of new user-friendly interfaces to direct drones with cinematographic purposes is essential for their use in cinematography. There are several efforts on this topic, for example, Gebhardt et al. [11] showed a new way to introduce a simplified trajectory, which is extended in [12] allowing the introduction of aesthetic requirements. In [13], a complete tool helps expert and novice cinematographers to achieve a visually pleasant drone trajectory, based on key-frames and some aesthetic user inputs. Other works, such as [14] and [15], develop touch interfaces to specify how to record a target. Although it is not the focus of this work, our implementation includes a user interface to ease the introduction of the control objectives.

In order to autonomously record aesthetically attractive footage while satisfying some cinematographic constraints some works present mathematical expressions to measure how good or bad the aesthetics of an image are [16], [17], [18]. These formulas are used to move a regular camera to a position that satisfies instructions from cinematographers in an autonomous way [19], [20], [21] or to find optimal views considering a static scene, enabling canonical static shots, such as the rule of thirds [22]. In contrast to CineMPC, these solutions only control the extrinsics of the camera, limiting the number of cinematographic options.

Other approaches focus on making the trajectory of the drone smoother while recording. Given a set of waypoints, different MPC formulations are used to control the drone to avoid unstable trajectories while passing through the established points [23], [24]. In [25], the director introduces the desired shot composition or a set of viewpoints, and a team of multiple drones records targets following a smooth trajectory, ensuring collisions and occlusion avoidance. Recent works attempt to imitate the trajectories run by a professional cinematographer.

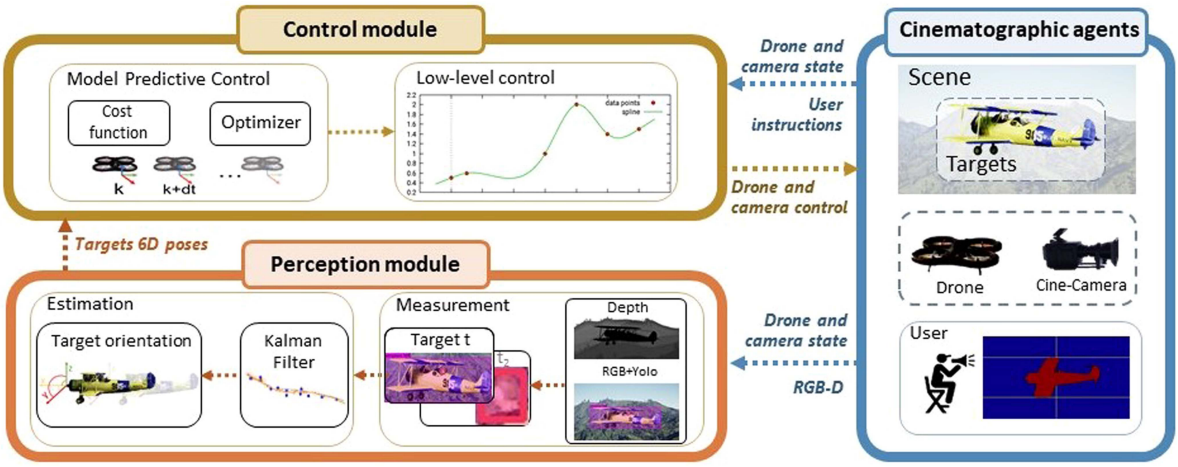

Fig. 2. **CineMPC System Overview.** Schematic summary of the platform, its modules, and their interactions. The cinematographic agents comprise the scene [containing the target(s)], the drone, the cine-camera, and the user providing instructions. The perception module utilizes camera images to extract the pose of targets, which are then fed to the control module. This module calculates the trajectory for the next N steps for both the camera and drone, optimizing the cost function within an MPC framework. This trajectory is transmitted through a low-level controller, ensuring smooth recording.

In [26], the drone imitates a walking camera operator that moves following one of the predefined patterns of movement to record a first-person view of a person, and the authors in [27] and [28] used imitation learning to reproduce the cinematographers' operations. These drone platforms are intended to take a single photo of a static scene or record footage either reproducing a determined kind of shot or following a predetermined trajectory of viewpoints, for a delimited amount of time. In opposition, CineMPC can track static and moving targets during an indeterminate time. Moreover, the control of the drone is not limited to the trajectory of the drone itself, but also the trajectory of the intrinsic camera parameters.

Other works can deal with multiple targets. For instance, Joubert et al. [29] guided a drone to record multiple targets following the rules of a predefined set of shots. In [2], a multidrone platform lets the user choose among a list of canonical drone shots according to [30]. MPC is used in [31] to film scenes while tracking and recording multiple targets, according to some cinematographic standards, i.e., the position of targets on the image. The multidrone platform presented in [32] uses MPC to record different targets, optimizing a trajectory of predefined viewpoints, while avoiding occlusions and collisions. Although these approaches represent substantial advances in cinematographic platforms, they do not use real perception to extract the pose of the targets.

The solutions presented in [33] and [1] use real perception to track and record people doing different kinds of activities while following a visually pleasant trajectory. The authors of [34] used some of the principles in [1] to present a multidrone approach, enabling multiview of the target and avoiding collisions, occlusions, and view-point similarities between drones. All these solutions focus on getting the best shots of human targets by only optimizing the extrinsic parameters, e.g., the position and orientation of the drones. Compared with them, our approach is the first that introduces an essential factor for high-quality photography into the control problem: The intrinsic parameters of the camera lens. Besides, we also use real perception to track different types of targets.

Table I shows the contributions of the most relevant existing works compared with CineMPC. The titles of the headers of the columns are reduced for the sake of space. The complete titles are, respectively: Control of Extrinsics, Real Perception, Dynamic Targets, Multitarget, Control Image Composition (position of elements in the image), Obstacle Avoidance, Occlusion Avoidance, Public ROS code, Control of the DoF, and Control of Intrinsic Parameters.

III. System Overview

The complete CineMPC system is represented in Fig. 2. This figure shows a high-level schematic of the modules involved in the system and the communication between them.

a) Cinematographic agents: These components are found in any real cinematographic setup involving drones; the *user*, that gives instructions, (e.g., a movie director, director of photography), the *drone*, responsible for holding and maneuvering the cinematographic *camera* through the *scene* or environment, where the *targets* are found. Section IV provides detailed explanations of these agents.

b) Control module: This module is in charge of performing the computations that give autonomy to the drone. It transforms the users' requirements in actions to apply to the drone and the camera to achieve the cinematographic objectives. The *MPC solver* solves the control problem, using the *optimizer* to resolve a *cost function*, and gives actions that the *low-level controller* transforms into commands sent to the drone and the camera to take actions accordingly. Section V describes each component in detail.

c) Perception module: This module processes *RGB* and *depth* images from the RGB-D camera, estimating the *6-D pose of the targets* and providing the poses to the control module. Target position is determined through *YOLO* detector [35] detection in the image and the depth map. A *KF* predicts the next N position and velocity values, used for calculating the target's orientation. Section VI details the steps for this module. Section VII outlines implementation. Section VIII presents experiments. Section IX describes future work and limitations. Finally, Section X concludes this article.

IV. Cinematographic Agents

This section describes the cinematographic agents present in a real-world setup with drones, namely, the user, the drone, the camera, and the scene.

A. Drone and Gimbal - Extrinsic Parameters

The drone is the flying vehicle that holds and moves the recording camera around the 3-D space. In this article, we consider a simplified model of this vehicle and leave the accurate control of its high-order complex dynamics to the low-level control module (Section V-B). This way, CineMPC remains flexible to be used with different platforms, as long as the manufacturers provide suitable low-level controllers.

We define the position and velocity of the drone at discrete time instant k by $\mathbf{p}_{d,k}$ and $\mathbf{v}_{d,k} \in \mathbb{R}^3$, respectively. We decouple the orientation of the camera from that of the drone, assuming the presence of a gimbal. While a gimballed camera is somewhat unusual in hobby drones, it is standard in high-quality cinematography drones [15]. Most gimbals also implement image stabilization strategies, as high velocity on the movement of the drone produces aggressive motions that can lead to shaky recordings. Nevertheless, for the sake of simplicity, we denote this orientation by $\mathbf{R}_{d,k} \in SO(3)$. Therefore, the extrinsic parameters of the system are

$$\mathbf{x}_{d,k} = (\mathbf{p}_{d,k}, \mathbf{v}_{d,k}, \mathbf{R}_{d,k}). \tag{1}$$

The actuators in the simplified model are the drone acceleration, and the angular velocity of the gimbal, and are represented by $\mathbf{a}_{d,k} \in \mathbb{R}^3$ and $\mathbf{\Omega}_{d,k} \in \mathbb{R}^3$ and are grouped into the drone actuators vector $\mathbf{u}_{d,k}$

$$\mathbf{u}_{d,k} = (\mathbf{a}_{d,k}, \mathbf{\Omega}_{d,k}). \tag{2}$$

According to this, for the optimal control problem, we consider double integrator dynamics for the position-velocity pair

$$\mathbf{p}_{d,k+1} = \mathbf{p}_{d,k} + \Delta_T \mathbf{v}_{d,k}, \quad \mathbf{v}_{d,k+1} = \mathbf{v}_{d,k} + \Delta_T \mathbf{a}_{d,k} \tag{3}$$

where Δ_T is the sampling time of the discrete model, and the rotation evolves according to

$$\mathbf{R}_{d,k+1} = \mathbf{R}_{d,k} \exp\left(\Delta_T \mathbf{\Omega}^{\wedge}_{d,k}\right) \tag{4}$$

where $\exp(\cdot)$ is the exponential map, used to compute the rotation matrix obtained by rotation at constant angular speed $\mathbf{\Omega}_{d,k}$ for Δ_T s.

B. Cinematographic Camera—Intrinsic Parameters

The cinematographic camera, a key component of CineMPC, captures the scene. Traditional cameras use the pin-hole camera model, considering only projection and geometric parameters. In this model, all the image rays pass through an aperture (hole) at the center of the sensor, showing the whole image in focus. However, the aperture of any real camera has a finite diameter, it is not a pinhole. A higher fidelity model of a camera is given by the thin-lens camera model, where a lens replaces the sensor's hole. This substitution allows the control of the image focus, the DoF, and other artistic features by adjusting the camera's intrinsics [36].

The camera's intrinsics that we model in CineMPC are the focus distance, the focal length, and the aperture.

The *focus distance*, F_k, represents the distance from the camera where the elements appear in perfect focus. The definition of the *focal length*, f_k is different in the pin-hole model and the thin-lens model. In pin-hole, it represents the distance in millimeters between the aperture and the sensor. In the thin-lens camera model, it is the distance from the optical center of the lens and the point of focus, where the parallel rays from the image intersect. The focal length affects different artistic effects, such as the field of view and the DoF (part of the scene that is in focus). The *lens aperture*, A_k, controls light intake by adjusting the size of the opening through which image light passes to the camera sensor. Expressed through the f-number (or f-stop), it influences image brightness, exposure, Bokeh effect, and DoF. Fig. 3 shows a graphical explanation of the effect of the intrinsics in the final image.

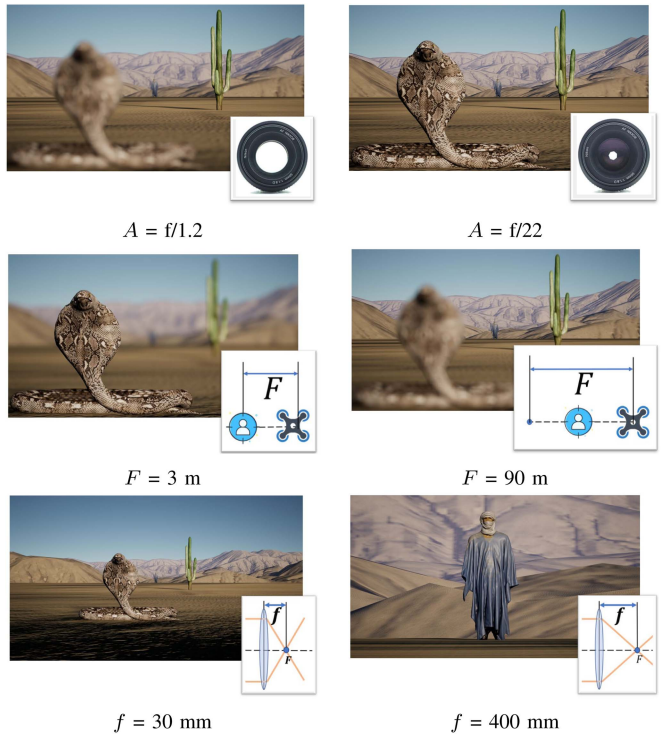

Fig. 3. **Effect of intrinsics in the final image.** The first row compares two aperture (A) values, affecting the portion of the scene shown in focus (DoF). The left side, with a low f-stop, has a wider aperture and shallow DoF. The right side, with a high f-stop, has a narrow aperture and a larger DoF. The second row contrasts two focus distance (F—distance from the camera to the center of the DoF) values, focusing on a closer distance (left) and a further distance (right). The third row compares different focal length (f) values, affecting the zoom, field of view, and DoF. The left side has a small focal length, providing a wide-angle view. The right side has a large focal length, resulting in a highly zoomed image. The camera maintains the same pose (same extrinsics) across all images.

The vector $\mathbf{x}_{c,k}$ represents the state of the intrinsics

$$\mathbf{x}_{c,k} = (f_k, F_k, A_k). \tag{5}$$

The relationship of these parameters with the extrinsics to determine the images acquired by the camera is detailed in Section V-A. In this section, we describe how we model their dynamic behavior. The intrinsic parameters can be set to any value within the physical camera range. We prevent large variations of the intrinsics in a short time, which can lead to aggressive image changes, not artistically pleasant in cinematography, by controlling their velocities instead of acting on their values directly

$$\mathbf{u}_{c,k} = (v_{f,k}, v_{F,k}, v_{A,k}) \tag{6}$$

where $v_{f,k} \in \mathbb{R}$ denotes the velocity of the focal length, expressed in mm/s; $v_{F,k} \in \mathbb{R}$ denotes the velocity of the focus distance, in m/s; $v_{A,k} \in \mathbb{R}$ is the velocity of the aperture, in f_stop/s all of them measured in the discrete-time step k.

This way, the intrinsic parameters evolve according to a single integrator model

$$\mathbf{x}_{c,k+1} = \mathbf{x}_{c,k} + \Delta_T \mathbf{u}_{c,k}. \tag{7}$$

C. Scene

The scene is the part of the environment captured by the camera, where many complex elements participate, e.g., foreground, background, people, and objects.

We model the scene as a set of n targets, represented by points of interest to be recorded. Similar to the drone, the state, $\mathbf{x}_{t,k}$, of each target is described by its position, $\mathbf{p}_{t,k} \in \mathbb{R}^3$, velocity $\mathbf{v}_{t,k} \in \mathbb{R}^3$, and rotation in the world, $\mathbf{R}_{t,k} \in \mathbb{R}^3$

$$\mathbf{x}_{t,k} = (\mathbf{p}_{t,k}, \mathbf{v}_{t,k}, \mathbf{R}_{t,k}). \tag{8}$$

Besides, we include additional information about the targets to describe their nature, t_{nature}, e.g., person, plane, etc., their estimated sizes in meters, i.e., width t_w and height t_h, and a preliminary orientation $t_R \in SO(3)$

$$\boldsymbol{\mu}_t = (t_{\text{nature}}, t_h, t_w, t_R). \tag{9}$$

This information is used in the control module to handle scene constraints and in the perception module to help in the estimation of the target state (8). We provide more details in the next sections.

D. User

The user gives the artistic and technical instructions to record the footage. Examples of individuals in this role include a movie director, a photographer, or an amateur user. In CineMPC, the user specifies the recording instructions and constraints, which are grouped into the sets $\boldsymbol{\mu}$ and $\boldsymbol{\mathcal{C}}$, respectively. Vector $\boldsymbol{\mu}$ contains the recording instructions, namely, the instructions on the nature of the targets $\boldsymbol{\mu}_t$, the composition $\boldsymbol{\mu}_{\text{im}}$, and DoF of the image $\boldsymbol{\mu}_{\text{DoF}}$, the desired values of the intrinsics $\boldsymbol{\mu}_f$, and the relative pose where the camera should be placed to record the targets $\boldsymbol{\mu}_p$

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_t, \boldsymbol{\mu}_{\text{DoF}}, \boldsymbol{\mu}_{\text{im}}, \boldsymbol{\mu}_f, \boldsymbol{\mu}_p). \tag{10}$$

All these parameters are described in the next sections of the article. The content and specification of the set of constraints $\boldsymbol{\mathcal{C}}$ are detailed in Section V-A3 of the article.

V. Control Module

CineMPC solves a nonlinear optimization problem inside an MPC framework [37]. Then, a low-level controller transforms the output of the MPC framework into commands to be sent to the drone and the camera. Fig. 4 shows a graphical explanation of the control module.

A. Model Predictive Control

At a given time k_0, CineMPC solves the following problem over a time horizon N:

$$\min_{\substack{\mathbf{u}_{d,k_0}..\mathbf{u}_{d,k_0+N} \\ \mathbf{u}_{c,k_0}..\mathbf{u}_{c,k_0+N}}} \sum_{k=k_0}^{k_0+N} J(\boldsymbol{\mu}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k})$$

$$\text{s.t.} \quad (3), (4) \text{ and } (7)$$

$$g(\boldsymbol{\mathcal{C}}, \mathbf{u}_{d,k}, \mathbf{u}_{c,k}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k}) \geq 0, \tag{11}$$

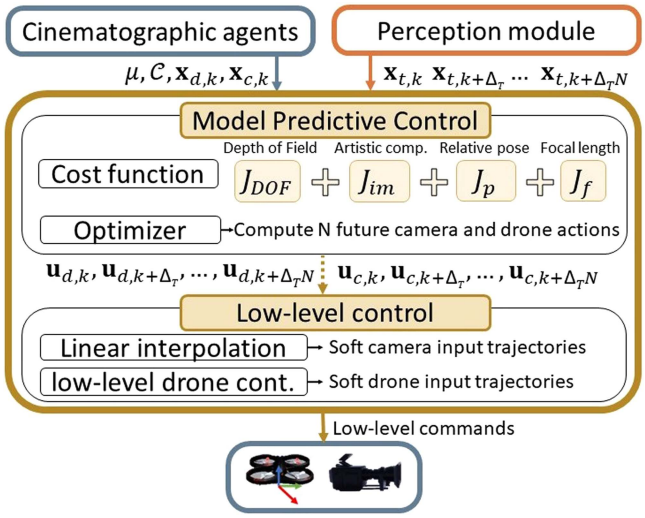

Fig. 4. **Control module diagram.** Components of the control module and their interaction. The module consists of the MPC framework, comprising the cost function, optimizer, and the low-level control submodule. The diagram incorporates inputs from other modules and the module's outputs.

where $J(\boldsymbol{\mu}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k})$ is the cost function, which considers the user instructions and $g(\boldsymbol{\mathcal{C}}, \mathbf{u}_{d,k}, \mathbf{u}_{c,k}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k})$ encodes the set of constraints. Note how the optimization problem computes all intrinsic parameters alongside extrinsic factors while considering scene and recording constraints. In the following, we provide more technical details on these two functions:

1) Cost Function: The cost function is composed of four terms

$$J(\boldsymbol{\mu}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k}) \equiv J_k = J_{\text{DoF},k} + J_{\text{im},k} + J_{p,k} + J_{f,k} \tag{12}$$

associated with the DoF, the artistic composition, the relative position between the camera and the targets, and the desired values of the intrinsics, respectively.

a) Focus of the image - DoF: Our solution autonomously controls the camera DoF, which represents the space of the scene that appears acceptably in focus in the image. According to specialized literature in cinematography and optics [38], the image DoF is delimited by two points, the near, D_n and the far, D_f distances. The region in the scene between D_n and D_f is in focus, whereas the rest appears blurry in the image.

To relate these distances to the camera intrinsics it is convenient to describe first the hyperfocal distance, H_k calculated as follows:

$$H_k = \frac{f_k^2}{A_k c} + f_k \tag{13}$$

where c is the circle of confusion, a constant parameter that depends on the model of the camera and expresses the limit of acceptable sharpness. The near distance $D_{n,k}$, represents the closest distance to the camera where the focus of the projected points is acceptable

$$D_{n,k} = \frac{F_k(H_k - f_k)}{H_k + F_k - 2f_k}. \tag{14}$$

Analogously, the far distance $D_{f,k}$, is the farthest distance to the camera where projected points are acceptably in focus

$$D_{f,k} = \frac{F_k(H_k - f_k)}{H_k - F_k}. \tag{15}$$

To determine the desired part of the scene to be in focus, the set of instructions $\boldsymbol{\mu}_{\text{DoF}}$ includes the desired near $D^*_{n,k}$, and far distances $D^*_{f,k}$, expressed in meters from the camera. In addition, w_{D_n} and w_{D_f} represent the weights associated with the cost terms of the near and far distances. The cost term of the DoF in the time step k penalizes intrinsic values that make actual distances depart from the desired values

$$J_{\text{DoF},k} = w_{D_n}\left(D_{n,k} - D^*_{n,k}\right)^2 + w_{D_f}\left(D_{f,k} - D^*_{f,k}\right)^2. \tag{16}$$

It is important to note that as f_k approaches low values, D_f tends toward infinity, implying that the image background is in focus. In contrast, D_n is always controllable. Therefore, it is common for w_{D_n} to be higher than w_{D_f}, or alternatively, setting $w_{D_f} = 0$ when f_k has a reasonably low value.

b) Artistic composition—Position of elements in image: The objective of this term is to show the targets placed in particular regions of the image. This term makes the elements appear in the final image so that they satisfy some cinematographic composition rules, e.g., the rule of thirds. Using the camera projection model, we define a cost term that penalizes deviations from the desired image composition. Let $\mathbf{K}$ be the calibration matrix of the camera [39]

$$\mathbf{K} = \begin{bmatrix} \beta_x f_k & s & c_u \\ 0 & \beta_y f_k & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

with c_u and c_v the image optical center coordinates and s the skew. The focal length affects the projection, thus, coupling the DoF and artistic composition objectives. The parameters β_x and β_y are other constants necessary to transform the units of the focal length, given in millimeters to pixels. Specifically, the ratios, $\beta_x = W_{px}/W_{\text{mm}}$ and $\beta_y = H_{px}/H_{\text{mm}}$, relates the width W_{mm}, and the height H_{mm}, of the camera sensor in millimeters with the width W_{px}, and the height H_{px}, of the image in pixels. The projection also requires the relative position between the camera and the target t, denoted by $\mathbf{p}_{dt,k} = \mathbf{R}^T_{d,k}(\mathbf{p}_{t,k} - \mathbf{p}_{d,k})$.

The target position in the image, $\mathbf{im}_{t,k} \in \mathbb{R}^2$, is

$$\mathbf{im}_{t,k} = \lambda \mathbf{K}\, \mathbf{p}_{dt,k} \tag{17}$$

where λ is the normalization factor to remove the scale component in the projection.

The subset $\boldsymbol{\mu}_{\text{im}}$ stores the desired image composition for the target t, denoted as $\mathbf{im}^*_{t,k}$, along with the associated weight, represented by $w_{\text{im},t}$, for all scene targets. The cost term penalizes deviations from this composition

$$J_{\text{im},k} = \sum_{t=1}^{n} w_{\text{im},t} \|\mathbf{im}_{t,k} - \mathbf{im}^*_{t,k}\|^2. \tag{18}$$

A target can be defined by multiple image coordinates, such as a person's face and body. Our solution also considers the option to control the position of various parts of a target within the image. For instance, a person's face could be positioned in the upper right third, while the knees are aligned with the bottom right third.

c) Relative position camera-target - Canonical shots: The target's depth $d_{dt,k}$, is the distance between the drone and the target, usually calculated using the Euclidean distance, $d_{dt,k} = \|\mathbf{p}_{dt,k}\|$. It is the only position-related value that cannot be controlled through J_{im}. When combined with a certain value of the focal length, $d_{dt,k}$ affects the amount of effective background visible and the image focus level.

The relative rotation between the camera and target, $\mathbf{R}_{dt,k} = \mathbf{R}^T_{d,k}\mathbf{R}_{t,k}$, determines the filming perspective. In the control problem, this is required to enable wide-angle shots and other types of aerial shots. This cost term is defined in terms of the subset $\boldsymbol{\mu}_p$, that contains the desired values of these two parameters $d^*_{dt,k}$ and $\mathbf{R}^*_{dt,k}$ for each target t, and their corresponding weights, w_R and w_d

$$J_{p,k} = \sum_{t=1}^{n} w_R \left\|\mathbf{R}^T_{dt,k} - \mathbf{R}^*_{dt,k}\right\|_F + w_d\left(d_{dt,k} - d^*_{dt,k}\right)^2 \tag{19}$$

and $\|x\|_F$ calculates the Frobenius norm of x.

d) Control of the focal length: To achieve some cinematographic effects, we need to adjust the focal length (f_k) of the camera, e.g., zooming in or out, *Dolly Zoom* [40], perspective distortion. The control of this term drives the focal length to the desired value, f^*_k, which is stored in the vector $\boldsymbol{\mu}_f \in \boldsymbol{\mu}$ along with its weight w_f

$$J_{f,k} = w_f\left(f_k - f^*_k\right)^2. \tag{20}$$

2) Optimizer: The MPC problem needs an optimizer that iterates the cost function and calculates the next control actuators that minimize that function in the future. The optimizer is a decision of implementation. In CineMPC, as the proposed optimization problem is nonlinear, we use interior point optimizer (Ipopt) [41], but this does not preclude the use of any other existing optimization libraries.

3) Constraints: The constraints are defined as a set of inequalities $g(\boldsymbol{\mathcal{C}}, \mathbf{u}_{d,k}, \mathbf{u}_{c,k}, \mathbf{x}_{d,k}, \mathbf{x}_{c,k}) \geq 0$ that depend on the states, the inputs, and the set $\boldsymbol{\mathcal{C}}$, specified by the user.

First, we consider upper and lower bounds on the control inputs, $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ in $\boldsymbol{\mathcal{C}}$, that depend on the cinematographic platform in which CineMPC is used, and are used to guarantee that the commands are physically feasible

$$\begin{aligned} &\mathbf{u}_{d,k} - \mathbf{u}_{d,\min} \geq 0, \quad \mathbf{u}_{d,\max} - \mathbf{u}_{d,k} \geq 0 \\ &\mathbf{u}_{c,k} - \mathbf{u}_{c,\min} \geq 0, \quad \mathbf{u}_{c,\max} - \mathbf{u}_{c,k} \geq 0. \end{aligned} \tag{21}$$

Similarly, we consider the possibility of adding upper and lower bounds and state constraints, e.g., to maintain the gimbal rotation in the allowed range

$$\begin{aligned} &\mathbf{x}_{d,k} - \mathbf{x}_{d,\min} \geq 0, \quad \mathbf{x}_{d,\max} - \mathbf{x}_{d,k} \geq 0 \\ &\mathbf{x}_{c,k} - \mathbf{x}_{c,\min} \geq 0, \quad \mathbf{x}_{c,\max} - \mathbf{x}_{c,k} \geq 0. \end{aligned} \tag{22}$$

The next set of constraints is used to prevent collisions of the drone with the targets

$$d_{dt,k} - d_{\min} \geq 0 \tag{23}$$

where $d_{\min}$ is the desired safety distance, introduced in $\boldsymbol{\mathcal{C}}$.

Finally, we consider a last set of constraints to handle potential occlusions of the targets. Using $\mathbf{x}_{t,k}$, the target height and width, t_h and t_w of (9), we can predict the bounding box of each target in the image using (17), which is represented by the left-top and right-bottom pixels in the image, $\mathbf{im}^{\text{lt}}_{t,k}$ and $\mathbf{im}^{\text{rb}}_{t,k}$.

Ideally, to guarantee occlusion-free trajectories, the bounding boxes of two targets should not intersect at any time. This can be formally introduced in the problem with a strong increase of the computational load, transforming it into a mixed-integer linear program. We consider instead a simplification that seems to work well in our experiments without increasing the complexity.

Before solving an instance of the optimization problem, we check the relative location of each pair of bounding boxes and analyze the potential risk of occlusion. We describe the process for the left-top horizontal coordinate of the bounding box, noting that the process can be done analogously for the other three coordinates of interest. Let $x^{\text{lt}}_{ti,k}$ represent the horizontal coordinate of $\mathbf{im}^{\text{lt}}_{ti,k}$ —the left-top of the target i. Then, we include the following constraint:

$$x^{\text{lt}}_{t1,k} - x^{\text{br}}_{t2,k} \geq 0 \tag{24}$$

if and only if the next condition holds for the initial configuration

$$(y^{\text{lt}}_{t1,k_0} > y^{\text{br}}_{t2,k_0}) \wedge (y^{\text{lt}}_{t2,k_0} > y^{\text{br}}_{t1,k_0}) \wedge (x^{\text{lt}}_{t2,k_0} > x^{\text{br}}_{t1,k_0}) \tag{25}$$

using the same notation as (24). Otherwise, we neglect the chance of occlusion and do not include this constraint.

B. Low-Level Control

The MPC calculates N high-level control actions that the drone should execute every Δ_T s. Since these actions are computed considering a simplified motion model, we include a low-level controller that transforms the MPC commands into actuator commands to achieve smooth trajectories that ensure suitable footage.

For the rotation and the intrinsics of the camera, we use linear interpolation to split each command into m smaller portions that are sent to the drone with a higher frequency, which corresponds to Δ_T/m. The choice of linear interpolation is made to prevent the overstepping of the commanded values, which would imply shaky images. Similarly, to smooth the position of the drone, we use a standard low-level drone controller that receives position key-points or velocity commands and transforms them into commands that the drone executes following smoother high-level trajectories.

VI. Perception Module

The perception module estimates the poses of all targets from step k until $k + \Delta_T N$ from the RGB images and depth data recorded by the drone. Fig. 5 summarizes the process. As detailed next, the perception process is done in two steps, measurement and estimation.

A. Measurement

The measurement process receives RGB-D images from the camera, extracting the relative position of the present targets.

1) Detection of Targets Position in the Thin-Lens Image: In this step, the system extracts the position in pixels of the targets present in the image. Our implementation uses YOLO [35], an off-the-shelf deep-learning approach for image segmentation. This choice is motivated by its low computational demand and the capacity to detect the targets even if they appear out of focus, as required by filming instructions [42].

We only consider the detections that belong to t_{nature}. For the remaining steps in the perception module methodology, we use the bounding box provided by the detector to select a pixel that identifies the target, $\mathbf{im}_{t,k}$. The decision on this pixel varies depending on the target nature, and t_w and t_h.

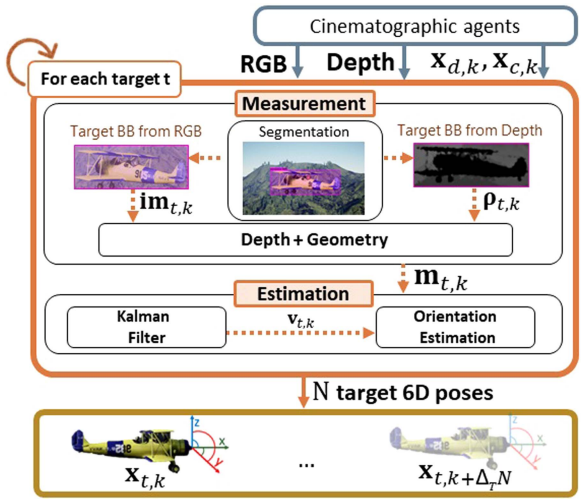

Fig. 5. **Perception module diagram.** Components of the perception module and their interaction. The module consists of depth and position measurements, along with the estimation of the targets' next poses. The diagram displays inputs from other modules and the outputs of this module.

2) Detection of Targets Depth: To obtain the 3-D position, we extract the relative distance between the drone and the target from the depth image. To make the measurement robust to noise, the depth value, $\rho_{t,k}$, is defined by the median of the minimum depth of each row of the bounding box

$$\rho_{t,k} = \underset{i}{\text{median}} \left(\min_{j} \left([\mathbf{D}_{t,k}]_{i,j} \right) \right) \tag{26}$$

where $[\mathbf{D}_{t,k}]_{i,j}$ represents the pixel in row i and column j of the bounding box in the depth image. This method helps to filter depth values from the background or foreground, as well as noisy readings. Alternatively, for distant targets where an RGB-D camera may lack sufficient resolution, the issue could be mitigated by employing a LiDAR or a laser sensor.

3) Relative Position of the Target: We use the image coordinates of each target, $\mathbf{im}_{t,k}$, its depth $\rho_{t,k}$, and the camera calibration matrix $\mathbf{K}$—which is dependent on the camera intrinsics, i.e., the focal length—to compute the relative position of the target with respect to the drone at time step k

$$\mathbf{p}_{dt,k} = \rho_{t,k} \mathbf{K}^{-1} \, \mathbf{im}_{t,k}. \tag{27}$$

Since the estimation step uses absolute positions, we convert the relative position to world coordinates using geometry

$$\mathbf{m}_{t,k} = \mathbf{p}_{d,k} + \mathbf{R}_{d,k} \mathbf{p}_{dt,k} \tag{28}$$

and the drone pose in the world, which is assumed to be available. The reason why we use absolute instead of relative positions is detailed in the next section.

B. Estimation

The estimation process receives 3-D absolute position measurements of the targets. With this information, it estimates the position, velocity, and orientation of the targets for the next N time steps, which are incorporated into the control module.

1) Kalman Filter: The central element of this process is a KF. The filter's state is defined by the position of the target in the world, $\mathbf{p}_{t,k}$, and its velocity, $\mathbf{v}_{t,k}$. The motion model for the prediction stage considers a double integrator with noise defined as small accelerations. The measurement used in the correction is the absolute targets' position in the world, $\mathbf{m}_{t,k}$ provided by the measurement module.

2) Estimation of Targets' Orientation: Extracting the targets' orientation from an RGB-D is not trivial. In cinematographic applications, the view directions are typically aligned with the movement direction of a target, i.e., recording a car from its front. Thus, it is reasonable to associate the rotation of a target in terms of its frontal plane, which matches the movement plane of the target. We use the targets' velocity from the KF and vector algebra to construct the rotation matrix associated with each target.

The three velocity vectors that form the targets' rotation matrix in the world, $\mathbf{R}_{t,k}$, are normalized and orthogonal to each other. The first vector, $\mathbf{r}_{1,k}$, represents the estimated velocity of the target in the world and is a component of the KF state: $\mathbf{r}_{1,k} = \mathbf{v}_{t,k} \in \mathbb{R}^3$. The remaining two vectors are calculated using vector algebra as follows. First, we associate the gravity vector, which is always pointing to the ground, to the target, $\mathbf{g}_k = [0, 0, -1]$. We obtain the second vector by taking the cross product of $\mathbf{g}_k$ and $\mathbf{r}_{1,k}$, $\mathbf{r}_{2,k} = \mathbf{v}_{t,k} \times \mathbf{g}_k$, which is orthogonal to $\mathbf{r}_{1,k}$. Finally, the cross product of the previous vector $\mathbf{r}_{2,k}$ and the velocity vector $\mathbf{r}_{1,k}$, returns the third vector, $\mathbf{r}_{3,k} = \mathbf{r}_{2,k} \times \mathbf{r}_{1,k}$, which is also orthogonal to $\mathbf{r}_{1,k}$ and $\mathbf{r}_{2,k}$. The rotation matrix of the target is composed of the three orthogonal vectors

$$\mathbf{R}_{t,k} = [\mathbf{r}_{1,k}, \mathbf{r}_{2,k}, \mathbf{r}_{3,k}]. \tag{29}$$

VII. Implementation

To promote the widespread use of CineMPC, we release a publicly available implementation that is integrated with the ROS [43] environment. Following standard ROS design principles, a simplified version of the system architecture is depicted in Fig. 6, showcasing key ROS nodes, topics, and services. Communication arrows represent topics, which facilitate information sharing between nodes, while services, enabling information exchange in a server–client system, are denoted with a squared form.

The code also contains all the necessary elements to be run inside the photo-realistic and popular AirSim simulator [9].

The next sections include a description of the main nodes together with their related topics and services.

A. CineMPC

CineMPC receives the RGB-D images from the camera, the drone and camera state, and the user instructions. It returns the next inputs for the drone and the camera to record the scene and targets according to the instructions and constraints. Using this input/output structure, CineMPC is transparent to the platform where it is used, i.e., simulation or real drones.

1) Control Module: These two nodes, shown in yellow in Fig. 6, implement the control module described in Section V. The `MPC` node implements the MPC solver, and the `low_level` node generates low-level commands to ensure a smooth trajectory and recording. The first node reads the estimation of the N future target states from the topic `current_state` and calculates the N high-level trajectory commands. These are sent to the drone and camera through the `next_n_states` topic every Δ_T s. This is then used in the second node in a linear or spline interpolation to split each command into several, sent to the drone and the camera at a higher frequency. The specific topics to which these commands are sent will depend on the platform used.

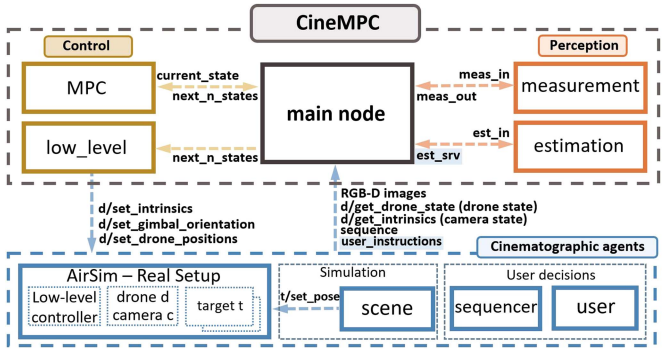

Fig. 6. **ROS software architecture of CineMPC.** Implementation components of CineMPC. The square solid lines represent the nodes of each module. The nodes communicate through ROS topics and services, depicted using lines. Similarly to Fig. 2, colors denote the system module of each ROS component: control module (yellow), perception module (orange), and cinematographic agents (blue). The main node has parts of every module.

2) Perception Module: This module implements the extraction of the pose of the targets from camera images, as detailed in Section VI. The two nodes are highlighted in orange in Fig. 6. The `measurement` node implements the measurement process of the perception module, whereas the `estimation` node is responsible for filtering noise and predicting the future steps of the targets.

The first node receives input from a message containing RGB and depth images and the drone state through the topic `meas_in`. To detect the bounding boxes of the targets, the node uses Darknet [44], a C++ implementation of YOLO [35]. The node outputs a message containing the list of absolute positions for each target, published in the topic `meas_out`.

The second node runs a KF for each target, estimating their next N poses, as detailed in Section VI-B2. The input of this node is the target's absolute position in the message `est_in`. This node outputs the next N absolute poses of the target, through service `est_srv`.

3) System Synchronization and Frequency - Main Node: Each topic is published at a different frequency. Thus, ROS is in charge of the synchronization. The `main node` acts as a coordinator between the nodes, dispatching the required information to each one at appropriate frequencies. Users should make sure to set Δ_T to a value higher than the solver's processing time, dependent on the computer's capabilities.

B. Cinematographic Agents

The source includes a `user` node to provide the instructions to the rest of the system through the service `user_instructions`, facilitating the introduction of different control objectives. The implementation features a user interface to simplify the definition of $\boldsymbol{\mu}$ (Section IV-D). This program provides a JSON file to the `user` node containing the

TABLE II
EXPERIMENT INDEX

Experiment (Scenario)	Real / Sim	No. Sequences	Control Extrinsics	Control Intrinsics	Control J_p	Control J_{DoF}	Control J_{im}	Control J_f	Dynamic targets	Multitarget	Blurred effects	Obs/Occ avoidance
Experiment 1 (A)	S	4	☑	☑	☑		☑		☑			
Experiment 2 (A)	S	4	☑	☑	☑		☑		☑		☑	
Experiment 3 (B)	S	2	☑	☑	☑	☑	☑	☑			☑	
Experiment 4 (B)	S	2	☑	☑	☑	☑	☑	☑			☑	☑
Experiment 5 (C)	R	4		☑			☑	☑		☑	☑	
Experiment 6 (D)	R	2	☑	☑	☑	☑	☑	☑				
Experiment 7 (D)	R	2	☑	☑	☑	☑	☑	☑				

TABLE III
SIMULATION EXPERIMENTS: CAMERA PARAMETERS

px		mm		β_x, β_y	c_u	c_v	s	c
W	H	W	H	40.40	480	270	0	0.03
960	540	23.76	13.365					

TABLE IV
SIMULATION EXPERIMENTS: SYSTEM CONSTRAINTS

	$\mathbf{p}_d, \mathbf{v}_d, \mathbf{R}_d$	$\boldsymbol{\Omega}_d, \mathbf{a}_d$	f, F, A	v_f, v_F, v_A
min	$-30, -40, -0.25$	$-0.25, -1$	$15, 4, 1.2$	$-7, -15, -3$
max	$30, 40, 0.25$	$0.25, 1$	$500, 2000, 22$	$7, 15, 3$

TABLE V
SIMULATION EXPERIMENTS: WEIGHTS OF COST FUNCTION TERMS

	w_{D_n}	w_{D_f}	$w_{\text{im}.x}$	$w_{\text{im}.y}$	w_d	w_R	w_f
units	10	10	1	1	10	100	1
E1-seq.1	0	0	1.25	0.5	20	100	0
E1-seq.2	0	0	2	0.5	20	2000	0
E1-seq.3	0	0	1.5	1	20	200	0
E1-seq.4	0	0	1.5	0.5	20	350	0
E3-seq.1	10	0	0.5	1	0	500	10
E3-seq.2	10	10	0.5	1.5	0	500	0.75

instructions. The `sequencer` node counts the delayed time since the beginning of the execution and splits the recording into `sequences`. Finally, in the case of the simulated experiments, the `scene` node automatically controls the AirSim scene to enable the recording of dynamic elements.

VIII. EXPERIMENTS

This section validates and demonstrates our system's capabilities with several experiments run in simulated (Section VIII-A) and real (Section VIII-B) setups. Table II shows an index of the conducted experiments, including the scenarios in which they were performed (in parentheses), along with the corresponding requisites they cover for clarification. We refer the reader to the video of the Supplementary Material for further visualization of the results of the experiments in simulation and the real world.

A. Simulation

Experiments in simulation enable more aggressive trajectories for both the cinematographic platform and targets compared with real setups. This allows for a comprehensive analysis of the platform under various situations and constraints. The experiments are conducted in two distinct scenarios. Scenario A focuses on evaluating the performance of the perception module, whereas scenario B involves testing the control module and conducting a simple user study to validate the system.

1) Experimental Setup: The experiments in simulation are run in Ubuntu 20 in an IntelCore i7-9700 8-Core CPU equipped with 64 GB of RAM and an NVidia GeForce GTX 1070. The experiments of Scenario A are simulated at $0.5\times$ speed, with a sample period of $\Delta_T = 0.2$ s and time-horizon of $N = 5$ time-steps. In Scenario B, the experiments are simulated at $1\times$ speed, with a sample period of $\Delta_T = 0.3$ s and time-horizon is $N = 5$ time-steps. Depth measurements are perturbed with Gaussian noise of zero mean and standard deviation of $\sigma = 0.04\,\text{m}^2$, and filtered with a KF. Table III contains the set of constants described in Section IV that describe the camera of the simulation environment. The constraints $\mathcal{C}$, (Section V-A3), vary for each platform. For instance, constraints on drone and camera control inputs maintain hardware realism or ensure nonaggressive trajectories, producing smooth footage. Table IV details the lower and upper bounds of the system constraints applied in the experiments for our platform. Table V depicts the cost function weights for each sequence of experiments E1 and E3. Each experiment is conducted multiple times, with the drone starting from random initial positions, capturing the actor within the field of view. The plots display the mean values across all runs with a solid line, whereas the standard deviation is represented by a lighter-shaded area.

2) Scenario A. Plane Flight Over a Forest: a) Goals on this scenario: We designed this scenario with two goals. The first goal is to test the control of the extrinsic parameters and the focal length by requesting wide variations in recording perspective and image composition. The second goal is to test the perception module and how it integrates with the control under different conditions and perspectives.

b) Experiment 1 (E1) - Filming a moving target from different perspectives: In this scenario, a plane moves with unknown and varying direction, orientation, and velocity over time, reaching a maximum speed of 10 m/s. The experiment consists of four sequences in which recording instructions are modified to capture the plane from different perspectives, e.g., *"High Angle Shot,"* and different image compositions e.g., *"Rule of thirds."* Consequently, we adjust the desired values associated with the cost terms J_p and J_{im} in each sequence. The weight of J_p, w_p, is higher than w_{im} to highlight its effect on the final recording, as shown in Table V. The cost term J_p controls the desired relative position between the camera and the plane. Different values of $\mathbf{R}^*_{p,k}$ and d^*_{dp} are considered for each sequence to force the drone to record the plane from different perspectives. The term J_{im} controls the position of the plane in the image. We define two targets inside the plane: The middle-top and middle-bottom

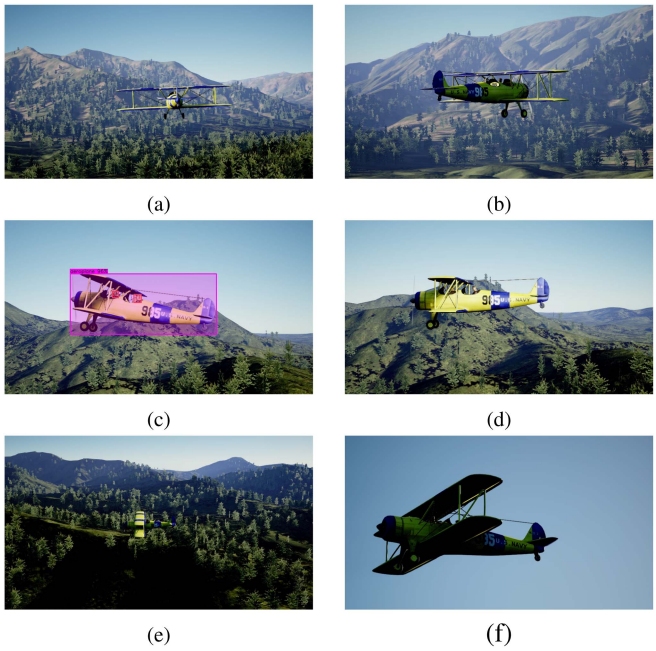

Fig. 7. **Experiment 1. Qualitative results:** (a) Initial frame. (b) End frame of Sequence 1. (c) Intermediate frame from Sequence 2 along with the bounding box detected by the perception module. (d) End frame of Sequence 2. (e) End frame of Sequence 3. (f) End frame of Sequence 4.

coordinates of its bounding box. The desired image positions for these targets change in each sequence, varying the vertical composition around the rule of thirds, depending on the portion of the frame that the plane should occupy.

Finally, concerning the intrinsics, this experiment just focuses on the control of the focal length for J_{im}. Therefore, the weights associated with J_{DoF} and J_f are set to zero.

Fig. 7 displays one frame for each sequence, providing a visual representation of all the recording perspectives. For quantitative results, we run the experiment 15 times and show the results in Fig. 8. Fig. 8(a) shows the position of the top of the plane. The ground-truth value and the estimation of the perception module are shown for each 3-D point $[x, y, z]$. Fig. 8(b) shows the estimated velocity by the perception module and the ground-truth values for comparison. Analogously, Fig. 8(c) depicts the estimated and ground-truth value of the rotation of the plane in the world, $\mathbf{R}_p$, converted to roll (not showing, always zero), pitch and yaw notation for simplicity.

Fig. 8(d) shows the position of the plane in the image. The upper line depicts the horizontal position in pixels, whereas the two lower lines represent the vertical position of the top and bottom parts of the plane, respectively. As mentioned, the higher weight of J_p may cause the other cost terms, such as J_{im} take longer to achieve their set-points.

Fig. 8(e) shows the drone-plane relative distance (d_{dp}) and the desired values. The focal length (f) is shown to demonstrate how the camera zoom helps to keep the plane shown in the image following the requested image composition, which changes over time Fig. 8(d). The relative distance is also controlled by J_p, so the controller automatically controls the focal length to adjust the composition of the image keeping the drone at the same relative distance to the target, satisfying both J_p and J_{im}.

Fig. 8(f) depicts the relative rotation, $\mathbf{R}_{\text{dp}}$, along with its varying desired values in each sequence, sometimes experiencing abrupt changes. After one of those, the drone's recording perspective must change significantly to keep the plane on screen and record it from the desired relative distance.

Finally, Fig. 8(h)–(j) represent each component of the drone's position ($\mathbf{p}_d$), illustrating the smoothness of the trajectories.

c) Experiment 2 (E2)- Analysis of the perception module: In this experiment, we manipulate the scene to show how the perception module is affected by changes in illumination and focus. First, we deliberately alter certain camera parameters to intentionally blur the scene throughout the entire experiment. Besides, we conducted illumination changes, running the experiment both in a dark scene and in an overilluminated scene (top row of Fig. 9). Fig. 9(b) and (e) and Fig. 9(c) and (f) compare the estimation of the position and orientation of the plane and their ground truth, respectively. In both cases, CineMPC was able to carry out the filming instructions without trouble.

3) Scenario B. Actor Standing in the Desert: a) Goals on this scenario: The *Dolly zoom effect* or *vertigo effect* [40], is a well-known kind of shot in cinematography. Important directors used it in awarded films.[1] In this kind of shot, the main target of the scene appears firstly centered vertically in the image. Then, the background suddenly appears to come closer to the viewer while keeping the target centered in the image with the same proportions, transmitting a feeling of vertigo and unreality.

The first goal of this scenario, addressed in Experiment 3, is to assess the control module. This entails showcasing the impact of each cost term in CineMPC on automatically reproducing the shot, with a focus on controlling both the extrinsic and, more importantly, the intrinsic camera parameters. Moreover, the DoF is varied over time and controlled automatically. In Experiment 4, we introduce changes to the scene to observe how the system can handle various environmental constraints.

b) Experiment 3 (E3) - Dolly zoom in the desert: Three targets are positioned in the middle of a desert. The primary target is a human actor standing. We intentionally place two cacti far from him—one behind and another in front—to show the effect of the variation of the DoF. The experiment is divided into two sequences. In the first sequence, the actor should adhere to the rule of thirds, centered horizontally in the image, while the camera maintains a stable focal length. The *Dolly zoom effect* and the changes in the DoF are conducted in the second sequence. The two sequences and their control requirements are described in detail next.

First sequence—Placing targets on the image*:* The goal of this experiment segment is to record a so-named *Cowboy Shot* [4], i.e., showing the upper part of the body on the image. The actor is recorded from the front (J_p), with the camera maintaining a stable focal length (J_f), only controlling the extrinsic parameters. To achieve the shot, the actor is treated as "two targets"—the center of his head (nose) and hips. The goal for the actor is to appear centered horizontally, with his nose and hips aligning with the top and bottom vertical thirds.

Second sequence—Dolly zoom effect*:* The costs associated with achieving this shot are set as follows.

1) J_{DoF}*:* This term controls the parts of the scene that appear in focus. We introduce significant variations in the desired

[1] See Vertigo (1958) by Alfred Hitchcock or Jaws (1975) by Steven Spielberg.

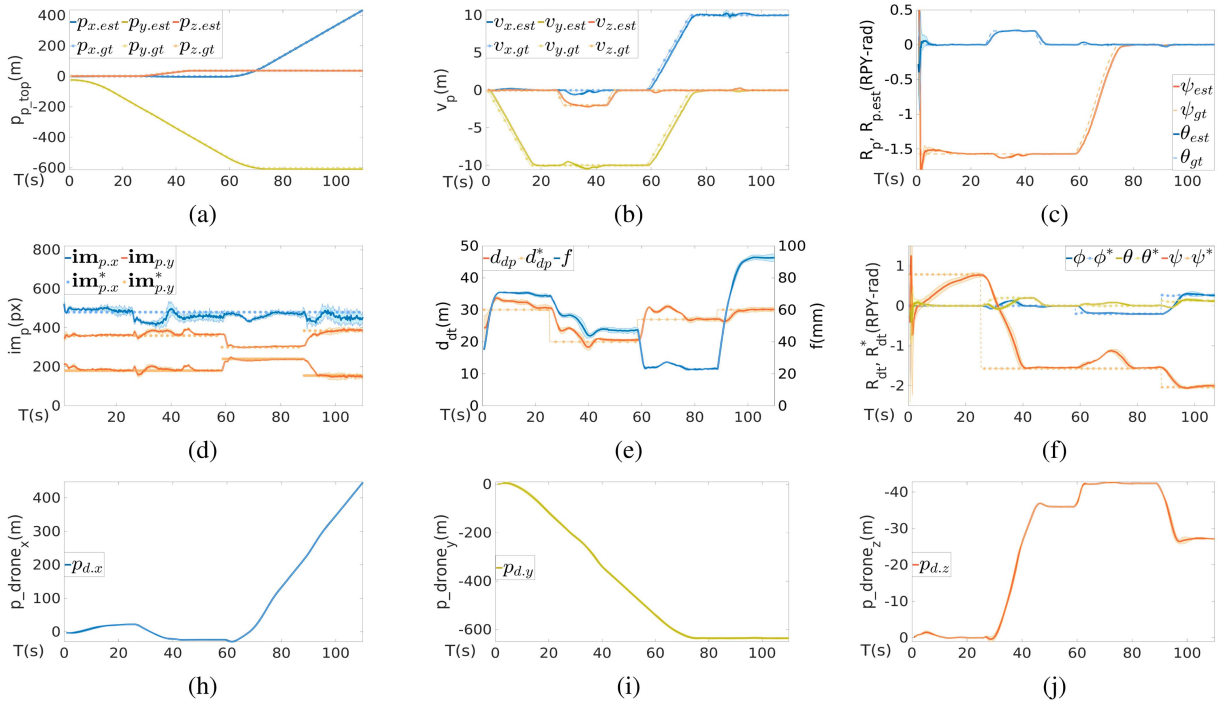

Fig. 8. **Experiment 1. Quantitative results:** Experiment was conducted 15 times from different starting points. The Z component is intentionally inverted for clarification. Solid lines represent the mean of the plotted value. Lighter areas depict the standard deviation. (a) Evolution of top position of the plane ($\mathbf{p}_{p\_\text{top}}$). Solid lines represent the estimated value from the perception module and dashed lines represent ground-truth values. (b) Evolution of the velocity of the plane ($\mathbf{v}_{\text{plane}}$). Solid lines depict estimated values extracted from the perception module. Dashed lines represent ground-truth values. (c) Evolution of the absolute rotation of the plane ($\mathbf{R}_p$) in roll, pitch, and yaw. Dashed lines are ground-truth and solid lines are the value from the perception module. (d) Evolution of position in the image of the plane ($\mathbf{im}_p$) (solid) and its desired values (dashed). (e) Evolution of relative distance drone-plane (d_{dp}) (solid), its desired values (dashed) and focal length (f). (f) Evolution of relative rotation of the plane ($\mathbf{R}_{\text{dp}}$) and desired value ($\mathbf{R}^*_{\text{dp}}$). Solid lines are actual values and starred lines are desired values. (h), (i), and (j) Evolution of the position of the drone ($\mathbf{p}_d$). (h) Component x. (i) Component y. (j) Component z.

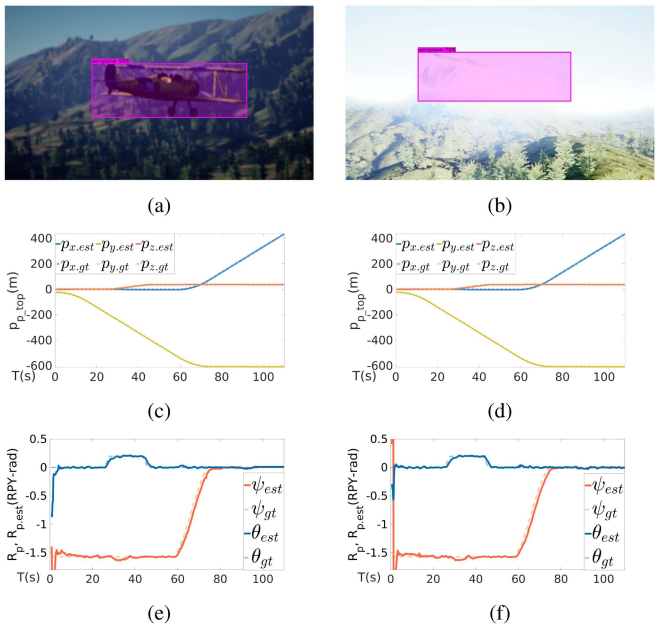

Fig. 9. **Experiment 2. Analysis of the perception module.** Recreation of E1 with changes in the scene focus and illumination. First column shows results when the scene is dark and blurry. Second column shows results when the scene is overilluminated and blurry. (a), (b) Frame of the experiment including the detected bounding box. (c) and (d) Position of the plane ($\mathbf{p}_{p\_\text{top}}$). (e) and (f) Absolute rotation of the plane ($\mathbf{R}_p$) in roll, pitch, and yaw. Solid lines are the estimated values by the perception module and dashed lines are ground-truth.

DoF to illustrate its effect. As the background comes closer, the DoF widens, showing more elements of the scene in focus. The near distance is requested close to the actor, $D^*_{n,k} = (d_{da,k} - 3)$m, and the far distance, $D^*_{f,k}$, ranges from $(d_{da,k} + 5)$m to $(d_{da,k} + 55)$m. When the two cacti appear in the image, the foreground and background of the scene get blurrier while keeping the actor sharp. This effect is performed by requesting a narrow DoF, setting the near and far distances close to the distance to the actor, $D^*_{n,k} = (d_{da,k} - 3)$m, $D^*_{f,k} = (d_{da,k} + 1)$m. Everything that is out of this range, e.g., background, foreground, and cacti, is shown out of focus.

2) J_{im}*:* This term keeps the actor's proportions in the image untouched from the first sequence.
3) J_p*:* This cost term adjusts $\mathbf{R}^*_{dt,k}$ to record the actor from the front. The weight of the relative distance term $d^*_{da,k}$ is set to zero so the solver decides about it freely.
4) J_f*:* The focal length plays a crucial role in executing the *Dolly zoom effect*. It is increased linearly at each iteration, starting from 35 mm and reaching 450 mm, to achieve the desired Dolly zoom effect.

Fig. 10 depicts some frames of the recording for qualitative results, whereas Figs. 11 and 12 show quantitative results of the experiment. Fig. 11 illustrates the controller's ability to track the desired values of the near (D_n) and far distances (D_f) of the DoF. The actor is positioned between those distances, appearing in focus. The other two lines represent the distance to the cacti, which may fall out of the focus range when requested.

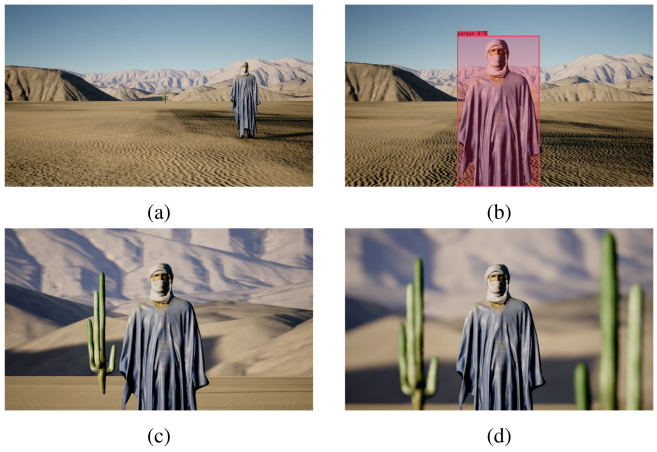

Fig. 10. **Experiment 3. Dolly zoom effect - Qualitative results:** (a) Initial frame of Sequence 1. (b) End frame of Sequence 1, including the bounding box detected by the perception module. (c) Intermediate frame of Sequence 2. (d) End frame of Sequence 2.

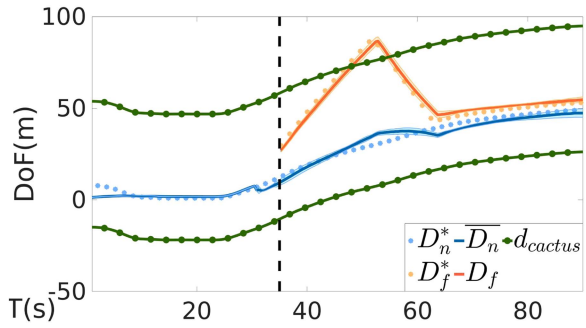

Fig. 11. **Experiment 3. Dolly zoom effect - Quantitative results (DoF):** Evolution of DoF—near and far distances—and desired values. The solid lines represent actual values and the starred lines depict desired values. The dashed line marks the initiation of the Dolly zoom effect in the video.

Fig. 12(a) displays the changes over time in the focal length and the relative distance between the drone and the actor, d_{da}. The zoom of the image increases with the focal length. To maintain the image composition, the actor should remain in the same place in the image. Therefore, the drone automatically flies farther away from the actor to compensate for the effect of the higher focal length. Fig. 12(b) shows the desired and real position of the actor in the image in pixels, $\mathbf{im}_{a,k}$. At the beginning of the sequence, the drone is positioned far from the actor, resulting in an initial transient period as the drone flies to a position to achieve the desired image composition. Subsequently, it maintains this position throughout the remainder of the experiment despite the zoom. CineMPC autonomously controls the camera's intrinsic parameters, with the aperture and focus distance depicted in Fig. 12(c). As explained in Section IV, the aperture plays a vital role in determining the DoF. In the initial phase of the second sequence, we request a wide DoF, causing the aperture to reach its maximum value. Consequently, as we request a narrow DoF toward the end of the sequence, the aperture decreases. Another parameter influencing the DoF is the focus distance, which the controller sets close to the distance of the actor.

c) Experiment 4 (E4)—Handling Environmental Constraints: The goal of this experiment is to demonstrate how CineMPC properly handles different constraints relevant to cinematography and robotics, such as collisions and occlusions. For this purpose, we add a cactus in a position that affects the recording. While obstacle detection is not the primary focus of the article, our modular architecture would facilitate the integration of advanced obstacle/occlusion avoidance techniques [1], [45], [46]. Therefore, we assume that the cactus's position is known in advance. However, the target's position is fully determined using the perception module.

We first test the *occlusion avoidance constraints*. We request CineMPC to record the actor from the front, following the rule of thirds, for 20 s. We conducted the experiment 60 times, randomly varying the initial positions of the drones and cacti (obstacles) within reasonable value intervals that ensure that the actor always remains in the field of view, although the cacti may initially occlude the actor. Fig. 13(a) and (b) depict two frames of this experiment. In Fig. 13(a) we do not request CineMPC to satisfy the occlusion avoidance constraints, causing an occlusion. In Fig. 13(b), CineMPC satisfies this hard constraint by redesigning the trajectory of intrinsic and extrinsic parameters to avoid occlusion. This is achieved, even if the primary recording objective (recording from the front) is not entirely met. To quantify the results of this experiment, we evaluate the average number of people detected by the perception module at each time. If the cactus is not occluding the actor, the module localizes one person. Fig. 13(c) and (d) show the number of detected people over time without and with occlusion constraints, respectively.

Second, we test the *collision avoidance constraints* using the same scenario. We conducted the experiment 10 times, with the recording instructions for each sequence remaining consistent. Without collision avoidance constraints, the drone impacts the cactus, stopping the recording. When collision constraints are in place (Section V-A3), the drone avoids approaching the cactus closer than d_{min} (2 m in this experiment) and continues recording. Fig. 14(a) displays the trajectory of the drone in the first sequence of the experiment without collision avoidance constraints, resulting in a collision with the cactus. Fig. 14(b) shows the trajectory of the drone for the same sequence if the collision constraints are included, altering the trajectory to avoid a collision. Fig. 14(c) shows the mean distance between the drone and the cactus over time without collision constraints. The distance goes sometimes below the security distance (dashed black line), causing a collision. Fig. 14(d) represents the drone–cactus distance over time when collision constraints are in place.

4) User Study: We conducted a small user study to gauge the potential interest in CineMPC among the general public and evaluate its user-friendliness. We engaged eight nonexpert users in a task involving manual control of a drone and camera within a simulation environment. The objective was to capture an image resembling a predefined frame while adhering to specific cinematographic guidelines. The study proceeded in two phases: First, users manually adjusted both the camera's extrinsic and intrinsic parameters using a simple keyboard interface. Then, they utilized the CineMPC user interface to input recording instructions, allowing our solution to position the drone and camera to fulfill these instructions while replicating the reference frame. To quantify the results, we compared the time required for manual adjustments with the time spent configuring all parameters within the CineMPC user interface. In addition, we assessed the final cost of CineMPC using (11) for each approach. Table VI presents the mean and standard deviation (*std*) of the times and costs for each case.

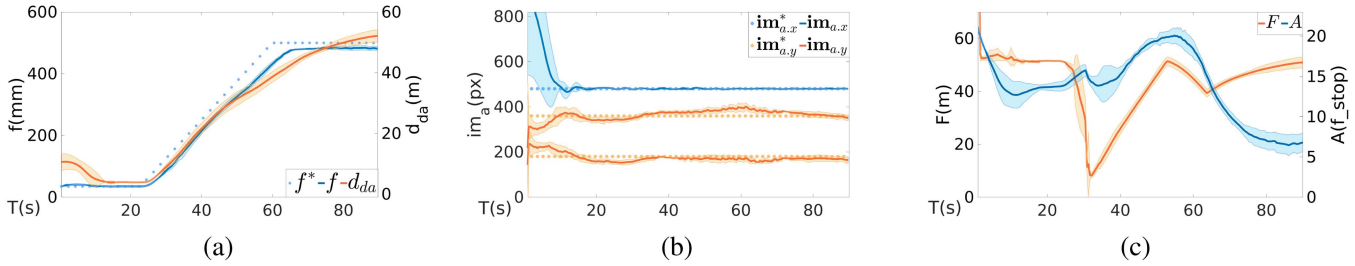

Fig. 12. **Experiment 3. Dolly zoom effect - Quantitative results:** Experiment was conducted 15 times from different starting points. Solid lines represent the mean of the plotted value. Lighter areas depict the standard deviation. (a) Evolution of focal length (f) and relative distance drone-target (d_{dt}). (b) Evolution of the image position of the actor ($\mathbf{im}_a$) and its desired value ($\mathbf{im}_a^*$). Solid lines represent actual values whereas starred lines depict desired values. The top line is the horizontal pixel and the bottom two lines are the vertical pixels. (c) Evolution of intrinsics [aperture (A) and focus distance (F)].

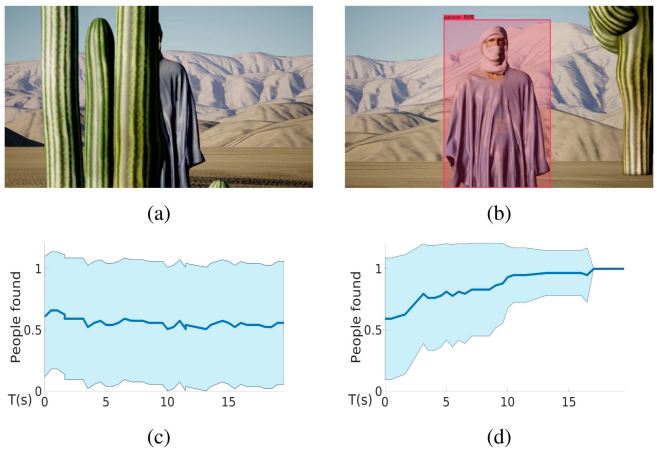

Fig. 13. **Experiment 4, Occlusion avoidance:** Left column depicts the experiment without occlusion avoidance constraints, leading to an occlusion of the main target caused by the cactus. This leads the perception module to lose track of the target. On the right, occlusion avoidance constraints prevent the cactus from occluding the target by readjusting the drone trajectory. (a) and (b) Final frames captured by the camera drone. (c) and (d) Plots with average (solid line) and standard deviation (light blue area) of the number of detected persons along the sequence. (c) Existing occlusions cause the perception module to lose track of the actor most of the time. (d) Occlusion avoidance constraints enable the perception module to localize the actor at the end of the sequence.

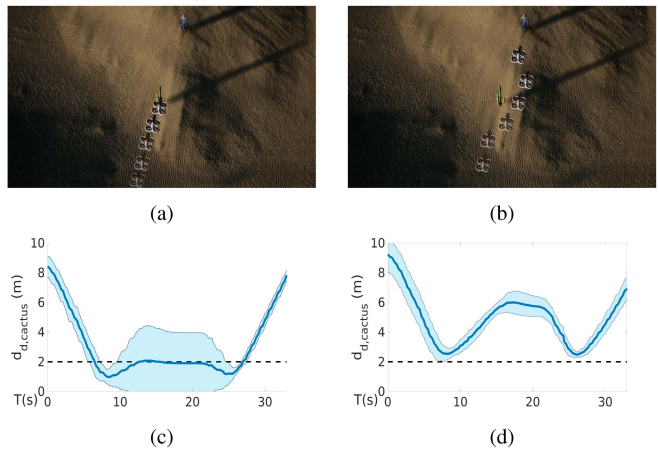

Fig. 14. **Experiment 4, Collision avoidance:** Left column illustrates the experiment conducted without collision avoidance constraints, leading to a collision with a cactus serving as an obstacle. In the right column, collision avoidance constraints prevent the drone from colliding with the cactus by readjusting its trajectory. (a) and (b) Third-person view of the drone trajectory. (c) and (d) Plots with average (solid line) and standard deviation (light blue area) of the distance drone-cactus. The dashed black line represents the security distance ($d_{\text{min}} = 2$ m). In the right column, this security distance is introduced as a constraint in the control module. This causes the drone never to approach the cactus closer than d_{min}, avoiding the collision and ensuring proper footage.

TABLE VI
USER STUDY EVALUATION: EXECUTION TIME AND CINEMPC COST

	mean, time (s)	*std*, time (s)	*mean*, cost	*std*, cost
CineMPC	81	15	1512	754
Manual	224	86	6850	1205

The results show that the users take much less time to understand and set the values in the user interface of CineMPC than controlling a camera manually, producing an image that is worse according to the CineMPC cost function in the last case. Supplementary Material includes a document with all the times and costs for every case as well as the final image of every user when they control the camera manually and the final image with CineMPC given as a reference.

B. Real Experiments

We conducted the real experiments in two different scenarios: Outdoors (C) and indoors (D). In the first experiment, we only controlled the camera's intrinsics, enabling us to conduct the experiment outdoors (Scenario C). We focus this experiment as an ablation study to show the qualitative differences in the image when controlling the intrinsics versus not controlling them, or the effect of incorporating only some cost terms of (11) in the control problem. Simulation experiments about the influence of the intrinsics are available in the conference paper [10]. In the second real experiment, we operated the drone indoors within a controlled area with safety nets (Scenario D). This setup allows us to fly the drone safely, and to present this experiment as a full test of the cinematographic platform, as we control all the cost terms of (11) and the intrinsics and extrinsics of the camera and drone.

1) Experimental Setup: The main hardware components of the drone used in the real setup are as follows:

- *Drone frame:* Quadrotor Holybro x500 v2;
- *Onboard Computer:* NVIDIA Jetson Nano;
- *Flight Controller:* Pixhawk 4;
- *Odometry-SLAM Camera:* Realsense T265; calculates the odometry and the pose of the drone in real-time;

(a) (b)

Fig. 15. **Experimental setup of hardware experiments.** (a) CineMPC drone platform. (b) Left—Close view of the cameras. Top to bottom: RGB-D camera, odometry camera, and cinematographic camera (inside the CineMPC box). Right—Close view of cinematographic camera (outside the CineMPC box). Note the inclusion of motors designed to control zoom and focus in real-time.

- *LiDAR:* Benewake TFMINI-S. Measures altitude;
- *RGB-D Camera:* Realsense D435i. Determines the depth of the targets of the scene; and
- *Cine-Camera:* Arducam PTZ 12 MP. This camera module incorporates a lens with two motors to the base camera chip Sony 12MP IMX477. The motors allow the modification of the focus and the zoom of the lens with different steps. We calibrated and transformed these steps into the actual focal length and focus distance. Aperture is not controlled in real experiments.

The real setup is depicted in Fig. 15 . Fig. 15(a) shows the CineMPC drone platform. The RGB-D camera, odometry camera, and Cine-camera are shown on the right side of Fig. 15(b), top to bottom. The Cine-Camera is embedded into a personalized box for safety and aesthetic reasons. This camera module is shown in detail on the left side of Fig. 15(b). The total weight of the setup is 2.25 kg.

The execution flow of the program is similar to the experiments in simulation. The onboard computer reads and sends the current state of the drone and the camera, along with the images recorded/taken by the cameras, to the desktop computer that is described in Section VIII-A1. Since the Jetson Nano lacks a GPU powerful enough to run the entire pipeline, this computer operates the CineMPC framework and computes the next values for the extrinsics and intrinsics to fulfill the experiment's instructions. These commands are then sent back to the onboard computer, which interprets and sends them to the drone and Cine-camera. The computers communicate using ROS. The Flight Controller implements the MAVLink communication protocol, allowing it to communicate with the onboard computer through MAVROS. The software controlling the Cine-camera is implemented in Python. In Experiment 7, we employed an Nvidia Jetson AGX Xavier to demonstrate the capability of running the entire pipeline onboard, eliminating the need for an external desktop computer. However, our current drone setup cannot accommodate this board due to the substantial weight of the Nvidia Xavier together with its development kit, which is 667 g, in stark contrast to the Nano's weight of 241 g. For the real experiments, we use a sample period of $\Delta_T = 0.5$ s and time-horizon of $N = 5$ time-steps.

Table VII contains the set of constants described in Section IV that describe the real cinematographic camera. Table VIII describes the lower and upper bounds of the system constraints that are determined by the hardware limits or stabilization requirements of the real experiments. Table IX shows the cost function weights for each sequence of experiments E5 and E6.

TABLE VII
REAL EXPERIMENTS. CAMERA PARAMETERS OF THE REAL CAMERA

px		mm		β_x	β_y	c_u	c_v	s	c
W	H	W	H						
675	380	6.29	4.71	107.3	80.6	337	190	0	0.001

TABLE VIII
REAL EXPERIMENTS: SYSTEM CONSTRAINTS

	$\mathbf{p}_d, \mathbf{v}_d$	f, F	v_f, v_F
E5-min	$0, 0$	$5, 0.4$	$-0.5, -0.5$
E5-max	$0, 0$	$10, 7$	$+0.5, +0.5$
E6-min	$-0.15, -0.15$	$5, 0.4$	$-0.1, -0.1$
E6-max	$+0.15, +0.15$	$10, 13$	$+0.1, +0.1$

TABLE IX
REAL EXPERIMENTS: WEIGHTS OF COST FUNCTION TERMS

	w_{D_n}	$w_{im.x}$	$w_{im.y}$	w_d	w_f
E5	300	1	0	0	1
E6	5	2	2	400	400

2) Experiment 5 (E5): Controlling the DoF and Image Position of Two Targets Using Only Intrinsics: This experiment is presented as an ablation study, where we exclusively control the cost terms of (11) where the intrinsics play a vital role. This approach showcases how controlling these parameters can positively impact the final recording. In this experiment, the position of the drone remains static, as shown in Fig. 16(h). Therefore, the position in the image and focus of the elements of the scene in real-time are uniquely determined using the focus distance and focal length of the camera. The aperture is not controllable in this camera. Consequently, we just control J_{im}, J_{DoF}, and J_f from (11) to meet the goals. CineMPC uses the RGB-D camera to calculate the position of the targets and adjust the desired DoF. First, the bounding box of the targets is detected from the RGB image [Fig. 16(e)]. Next, the distance to the object is calculated obtaining the depth of these bounding boxes from the depth image [Fig. 16(f)]. Finally, we adjust the requested near distance (D_n^*) to focus on the distance to the target. The focus distance of the camera is changed in real-time to satisfy this requirement. The experiment is divided into four sequences, each with two targets to position in the image and focus on: A bottle in the foreground and a car in the background. In the first sequence, the bottle should be placed in the left horizontal third, and the position of the car is not controlled. The bottle should be shown in focus, and the car out of focus. In the second sequence, the focus is the only parameter that changes, focusing on the car and showing the bottle out of focus. The third sequence maintains the focus but the position of the car should match the right horizontal third of the image. The last sequence maintains the positions of the elements but changes the focus, showing the bottle in focus and the car and background out of focus.

The results of this experiment are presented in Fig. 16. Fig. 16(g) illustrates the values of the intrinsics (focus distance and desired and actual focal length). Fig. 16(h) displays the evolution of the near distance along with the desired values. Fig. 16(i) depicts the desired and actual horizontal image positions of both targets. These quantitative experiments demonstrate the direct influence of the focus distance and the focal length on achieving the desired DoF and position of the targets in the image, respectively.

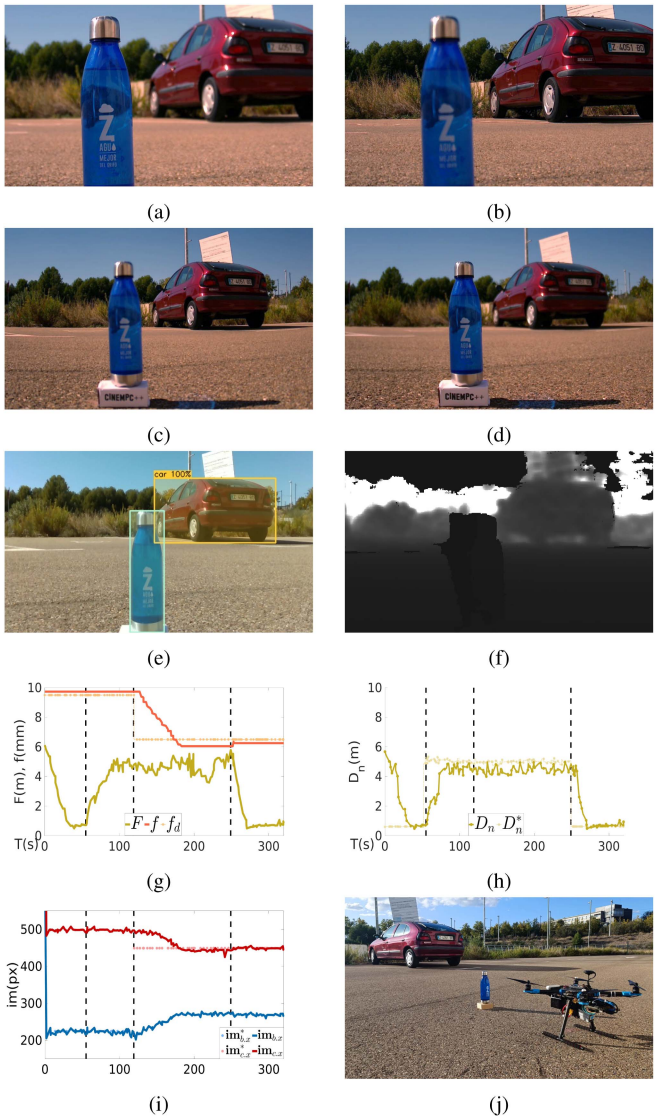

Fig. 16. **Experiment 5: Controlling the DoF and image position of two targets using only intrinsics.** Lighter lines represent desired values and darker lines represent actual values. Dashed black lines represent a change of sequence. (a), (b), (c), and (d) Frames captured by the cinematographic camera in each sequence (e) RGB output captured by the RGB-D camera, showcasing bounding boxes around objects detected by the perception module during the experiment. (f) Depth output captured by the RGB-D camera. (g) Intrinsics of the camera. Orange line is the focal length (f_d and f) and the yellow line focus distance (F). (h) Near distance (D_n) of the DoF. (i) Controlled vertical pixel for each target ($\mathbf{im}_{t.x}$). The blue line represents the image position of the bottle (controlled in Seqs. 1 and 2) and the red line represents the car (controlled in Seqs. 3 and 4). (j) Third-person view of the setup of this experiment, which remains static throughout the entire execution.

A representative frame of each sequence is shown in Fig. 16(a)–(d) as qualitative results. Notice how the target's sizes vary in the frames due to alterations in the focal length [e.g., Fig. 16(a) and (b) versus Fig. 16(c) and (d)], and how the focus of the scene changes due to the effect of the focus distance [e.g. car focused in Fig. 16(a) and (c) versus not focused in Fig. 16(b) and (d)]. All the cinematographic effects performed in this experiment, and represented in these frames, differ significantly from the output of the fixed camera shown in Fig. 16(e) where the image remains static during the whole experiment. This ablation study helps to demonstrate the importance of controlling the intrinsics of the camera in the final image result.

3) Experiment 6 (E6). Full Platform Test With a Flying Cinematographic Drone—Assessing Performance and Integration: The goal of this experiment is to demonstrate how CineMPC can control all cost terms by adjusting the extrinsics and intrinsics of a real drone and cinematographic camera to meet various cinematographic objectives. Each cost term in (11) plays a crucial role in filming this recording.

- J_{DoF}*:* Manages the focus distance (F) to ensure that the main target is always shown in focus, respecting the requested DoF (D_n^*).
- J_{im}*:* Controls the position of the drone ($\mathbf{p}_d$) and the focal length (f), placing the target in the requested image position ($\mathbf{im}_t^*$). In this experiment, the associated weight (w_{im}) is proportionally higher to highlight its effect.
- J_p*:* Places the drone at a desired recording distance (d_{dt}^*) by controlling its position $\mathbf{p}_d$.
- J_f*:* Controls the focal length (f_d) avoiding aggressive zoom variations.

To reduce complexity, and due to the hardware limitations, we do not control the orientation of the camera and the aperture of the camera in this experiment.

The experiment is divided into two sequences. In the first sequence, the chair in the scene should align with the left vertical third and the top of the chair should match the top horizontal third according to the rule of thirds. Moreover, the drone should be placed 4.5 m away from the target, keeping a stable focal length of 7 mm. In the second sequence, the chair should match the right vertical third, and the top of the chair the horizontal center of the image, while the drone flies at 3.5 m from the target, keeping a focal length of 5.4 mm. In both sequences, the control of the DoF ensures the chair appears in focus in the image.

Fig. 17 presents both qualitative and quantitative results of this experiment. Fig. 17(a) and (b) show a frame of the recording for each sequence. The bounding box of the target, as well as the horizontal and vertical guidelines denoting its desired position in the image, are depicted for clarification. Third-person views of the experiment with the drone in the air are illustrated in Fig. 17(f). For quantitative results, Fig. 17(c) depicts the actual and desired image position of the target. The plot of Fig. 17(d) shows the focal length and the actual and desired distance to the target. Finally, Fig. 17(e) depicts the evolution of the actual position of the drone. These plots demonstrate how CineMPC modifies the extrinsic parameters of the drone and the focal length of the camera to place the target in the desired image position while satisfying the other coupled requisites.

4) Experiment 7 (E7)—Computational Load: This experiment demonstrates the feasibility of using CineMPC on a real cinematographic platform. We installed and ran CineMPC on an Nvidia Jetson AGX Xavier. Utilizing a prerecorded ROS bag containing all camera images and drone state data (as described in Section VIII-B1), we executed the entire pipeline on the aforementioned board. To quantify the computational load, we measured the time required by the control and perception modules on this board, as these modules consume the most computational resources within the pipeline. The results (mean and standard deviation) are depicted in Table X. The sum of these two times is 631 ms. To ensure that the drone and camera execute the correct trajectory, the calculation time of each iteration should be shorter than the time horizon $\Delta_T N$ which is 2500 ms in the real experiments. The gap between

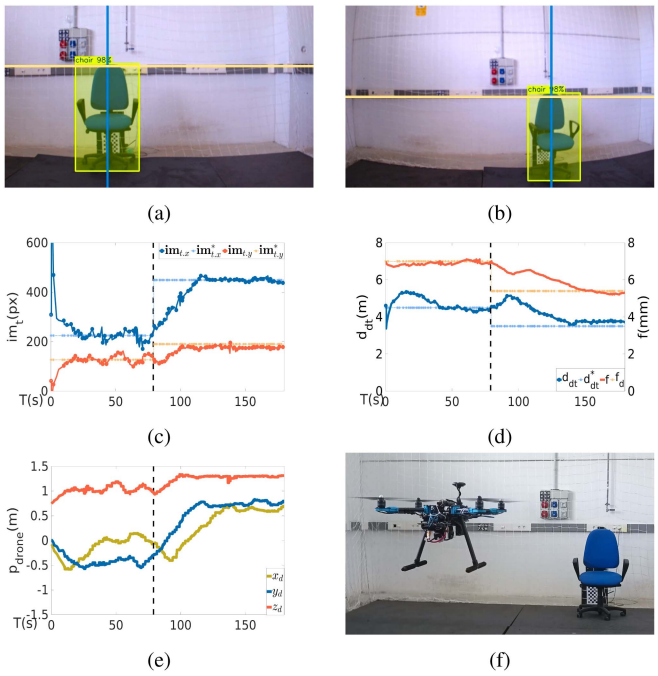

Fig. 17. **Experiment 6: Full platform test with a flying Cinematographic Drone—Assessing performance and integration.** Lighter lines represent desired values and darker lines represent actual values. (a) and (b) Frames captured by the cinematographic camera in each sequence. (c) Image position of the target ($\mathbf{im}_t$) and desired value ($\mathbf{im}_t^*$). The top blue line represents the horizontal pixel and the bottom orange line depicts the vertical pixel. (d) Focal length (f), in orange, and desired and actual distance drone-target (d_{dt}), in blue. (e) Position of the drone ($\mathbf{p}_d$), where x is yellow, y is blue, and z is orange. (f) Third-person view of platform and target while experimenting.

TABLE X
COMPUTATIONAL LOAD: MEAN AND STANDARD DEVIATION OF THE EXECUTION TIMES (IN MS) OF THE PERCEPTION AND CONTROL MODULES

	Mean	Std.
Perception	373	104
Control	258	84

the time horizon and computational time affirms the algorithm's feasibility for onboard execution.

IX. FUTURE WORK AND LIMITATIONS

CineMPC paves the way for new interesting and challenging problems in the context of autonomous cinematography. For example, on the control side, there is the issue of defining suitable values of $\boldsymbol{\mu}$ to achieve the different cinematographic effects. Currently, this requires human expertise, but it is something that could be automated as well. Some works such as [11], [14] approach this problem and present novel user interfaces that are more intuitive and visual, using selectors or images. Another interesting approach could be the use of data-driven techniques such as imitation or reinforcement learning, partly explored in [28], [47], [48], and [49] to determine all the intrinsic and extrinsic parameters from experts or existing footage. Consideration of the intrinsic parameters in a multidrone setup is also something worth analyzing when compared with existing multidrone approaches. CineMPC also raises new questions in the field of perception. The proposed perception module focuses on providing the necessary information for the controller to work, but it still relies on some human input. For instance, the current module is only able to estimate the orientation of moving targets and needs a preliminary orientation of static ones (t_R). The human also needs to specify the type of target to film and is limited to those available in YOLO. More advanced segmentation techniques [50] could be used to overcome these limitations, but the increase of computational load should also be considered.

X. CONCLUSION

We have presented CineMPC, a complete cinematographic platform that uses perception to track multiple types of targets from RGB-D thin-lens produced images and implements a MPC approach to control the intrinsic and extrinsic parameters of a camera for autonomous cinematography. This is the first approach to date to include the intrinsic information in this kind of control.

We have described the main modules of the system, namely, the perception and control modules and the cinematographic agents, and the role that they play in the cinematographic platform. The perception module is able to localize the targets that are present in the images taken by a thin-lens camera, that can be blurred or distorted, and extract their position and orientation that are filtered and processed with a KF. The control module is implemented inside an MPC framework whose cost function includes four different cost terms to achieve several artistic guidelines, such as the DoF and artistic composition of the image, the relative recording pose and canonical shots, and desired focal length. The optimization of these terms returns camera control values that generate semantically expressive images, closer to the ones seen in actual movies. We also release the system implementation of all these features to the community. A variety of experiments have been used to illustrate the potential of CineMPC in photorealistic simulation and in a real setup, successfully considering different guidelines and kinds of shots and a variety of targets in nature and dynamics.

REFERENCES

[1] R. Bonatti et al., "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *J. Field Robot.*, vol. 37, no. 4, pp. 606–641, 2020.
[2] A. Alcántara, J. Capitán, R. Cunha, and A. Ollero, "Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles," *Robot. Auton. Syst.*, vol. 140, 2021, Art. no. 103778.
[3] Z. Lu and L. Cai, "Camera calibration method with focus-related intrinsic parameters based on the thin-lens model," *Opt. Exp.*, vol. 28, no. 14, pp. 20858–20878, 2020.
[4] R. Thompson and C. Bowen, *Grammar of the Shot*. New York, NY, USA: Taylor & Francis, 2009.
[5] B. Lewandowski, D. Seichter, T. Wengefeld, L. Pfennig, H. Drumm, and H.-M. Gross, "Deep orientation: Fast and robust upper body orientation estimation for mobile robotic applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 441–448.
[6] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Comput. Vis. Image Understanding*, vol. 192, 2020, Art. no. 102897.
[7] T. Nägeli, S. Oberholzer, S. Plüss, J. A.-Mora, and O. Hilliges, "Flycon: Real-time environment-independent multi-view human pose estimation with aerial vehicles," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–14, 2018.
[8] P. Pueyo, E. Cristofalo, E. Montijano, and M. Schwager, "CinemAirSim: A camera-realistic robotics simulator for cinematographic purposes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1186–1191.

[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field Service Robotics*. Berlin, Germany: Springer 2018, pp. 621–635.
[10] P. Pueyo, E. Montijano, A. C. Murillo, and M. Schwager, "CineMPC: Controlling camera intrinsics and extrinsics for autonomous cinematography," in *Proc. Int. Conf. Robot. Autom.*, 2022, pp. 4058–4064.
[11] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," in *Proc. Conf. Hum. Factors Comput. Syst.*, 2016, pp. 2508–2519.
[12] C. Gebhardt and O. Hilliges, "WYFIWYG: Investigating effective user support in aerial videography," 2018, *arXiv:1801.05972*.
[13] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–11, 2015.
[14] Z. Lan, M. Shridhar, D. Hsu, and S. Zhao, "XPose: Reinventing user interaction with flying cameras," in *Proc. Robotics: Sci. Syst.*, 2017, pp. 1–9.
[15] H. Kang, H. Li, J. Zhang, X. Lu, and B. Benes, "FlyCam: Multitouch gesture controlled drone gimbal photography," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3717–3724, Oct. 2018.
[16] F. R. Perona, M. J. F. Gallego, and J. M. P. Callejón, "An application for aesthetic quality assessment in photography with interpretability features," *Entropy*, vol. 23, no. 11, 2021, Art. no. 1389.
[17] L. Mai, H. Le, Y. Niu, and F. Liu, "Rule of thirds detection from photograph," in *Proc. IEEE Int. Symp. Multimedia*, 2011, pp. 91–96.
[18] Y. Fang, H. Zhu, Y. Zeng, K. Ma, and Z. Wang, "Perceptual quality assessment of smartphone photography," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3677–3686.
[19] L.-W. He, M. F. Cohen, and D. H. Salesin, "The virtual cinematographer: A paradigm for automatic real-time camera control and directing," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 217–224.
[20] T.-Y. Li and X.-Y. Xiao, "An interactive camera planning system for automatic cinematographer," in *Proc. 11th Int. Multimedia Modelling Conf.*, 2005, pp. 310–315.
[21] C. Lino, M. Christie, R. Ranon, and W. Bares, "The Director's lens: An intelligent assistant for virtual cinematography," in *Proc. 19th ACM Int. Conf. Multimedia*, 2011, pp. 323–332.
[22] X. Xiong, J. Feng, and B. Zhou, "Automatic view finding for drone photography based on image aesthetic evaluation," in *Proc. 12th Int. Joint Conf. Comput. Vis. Imag. Comput. Graph. Theory Appl.*, 2017, pp. 282–289.
[23] G. Rousseau, C. S. Maniu, S. Tebbani, M. Babel, and N. Martin, "Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control," in *Proc. Eur. Control Conf.*, 2018, pp. 2897–2903.
[24] C. Gebhardt, S. Stevšić, and O. Hilliges, "Optimizing for aesthetically pleasing quadrotor camera motion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–11, 2018.
[25] Q. Galvane et al., "Directing cinematographic drones," *ACM Trans. Graph.*, vol. 37, no. 3, pp. 1–18, 2018.
[26] A. Ashtari, S. Stevšić, T. Nägeli, J.-C. Bazin, and O. Hilliges, "Capturing subjective first-person view shots with drones for automated cinematography," *ACM Trans. Graph.*, vol. 39, no. 5, pp. 1–14, 2020.
[27] C. Huang, Z. Yang, Y. Kong, P. Chen, X. Yang, and K.-T. T. Cheng, "Learning to capture a film-look video with a camera drone," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 1871–1877.
[28] C. Huang, Y. Dang, P. Chen, X. Yang, and K.-T. T. Cheng, "One-shot imitation drone filming of human motion videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5335–5348, Sep. 2021.
[29] N. Joubert et al., "Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles," 2016, *arXiv:1610.01691*.
[30] C. Smith, *The Photographer's Guide to Drones*. San Rafael, CA, USA: Rocky Nook, Inc., 2016.
[31] T. Nägeli, J. A.-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1696–1703, Jul. 2017.
[32] T. Nägeli, L. Meier, A. Domahidi, J. A.-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–10, 2017.
[33] C. Huang et al., "ACT: An autonomous drone cinematography system for action scenes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 7039–7046.
[34] A. Bucker, R. Bonatti, and S. Scherer, "Do you see what I see? Coordinating multiple aerial cameras for robot cinematography," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 7972–7979.
[35] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[36] M. Baba, N. Asada, A. Oda, and T. Migita, "A thin lens based camera model for depth estimation from defocus and translation by zooming," *Proc. ICVI*, pp. 274–281, 2002.
[37] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Berlin, Germany: Springer, 2013.
[38] M. Bass, *Handbook of Optics: Volume I-Geometrical and Physical Optics, Polarized Light, Components and Instruments*. New York, NY, USA: McGraw-Hill Education, 2010.
[39] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Germany: Springer, 2010.
[40] Y. Liang, R. Ranade, S. Wang, D. Bai, and J. Lee, "The "vertigo effect" on your smartphone: Dolly zoom via single shot view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 344–345.
[41] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[42] S. Zheng, Y. Wu, S. Jiang, C. Lu, and G. Gupta, "Deblur-YOLO: Real-time object detection with efficient blind motion deblurring," in *Proc. Int. Joint Conf. Neural Netw.*, 2021, pp. 1–8.
[43] A. Koubâa et al. *Robot Operating System (ROS)*. Berlin, Germany: Springer, 2017.
[44] J. Redmon, "Darknet: Open source neural networks in C," 2013–2016. Accessed: Feb. 2024. [Online]. Available: http://pjreddie.com/darknet/
[45] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5719–5726, Apr. 2022.
[46] X. Yang et al., "Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 156–167, Jan. 2021.
[47] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, "Can a robot become a movie director? Learning artistic principles for aerial cinematography," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1107–1114.
[48] H. Jiang, M. Christie, X. Wang, L. Liu, B. Wang, and B. Chen, "Camera keyframing with style and control," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–13, 2021.
[49] R. Bonatti, A. Bucker, S. Scherer, M. Mukadam, and J. Hodgins, "Batteries, camera, action! Learning a semantic control space for expressive robot cinematography," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 7302–7308.
[50] Z. Cao, G. HidalgoT. MartinezS. Simon Wei, and Y. A. Sheikh, "OpenPose: Realtime multi-person 2 D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021.

Pablo Pueyo (Student Member, IEEE) received the B.Eng. degree in computer science in 2015 from the Universidad de Zaragoza, Zaragoza, Spain, and the M.Eng. degree in mobile engineering in 2018 from the Universidad Pontificia de Salamanca, Salamanca, Spain. He is working toward the Ph.D. degree in robotics and systems engineering with the Departamento de InformÃ¡tica e Ingenieria de Sistemas, Universidad de Zaragoza, funded by a DGA grant.

His research interests include aerial cinematography, robotic simulators, MPC, perception, multirobot systems, and nonlinear control.

Juan Dendarieta received the B.Eng. degree in industrial engineering from the Universidad de Zaragoza, Spain, in 2011, and where he is currently working toward the master's degree in robotics, graphics, and computer vision.

He is actively engaged within the Robotics, Computer Vision, and Artificial Intelligence Group with the Universidad de Zaragoza, working on projects involving drones. His research interests include the progressive advancements in autonomous drone applications, particularly in the realms of navigation and cinematography.

Eduardo Montijano (Member, IEEE) received the M.Sc. degree in computer science and the Ph.D. degree in robotics and systems engineering from the Universidad de Zaragoza, Zaragoza, Spain, in 2008 and 2012, respectively.

From 2012 to 2016, he was a faculty member with Centro Universitario de la Defensa, Zaragoza. He is currently an Associate Professor with the Departamento de Informática e Ingenieria de Sistemas, Universidad de Zaragoza. His main research interests include distributed algorithms and automatic control in perception problems.

Ana Cristina Murillo (Member, IEEE) received the Ph.D. degree in computer science from the University of Zaragoza, Zaragoza, Spain, in 2008.

She is currently a Researcher and an Associate Professor with the Computer Science Department with the University of Zaragoza. Her current research interests include the area of computer vision and machine learning, with particular interest in scene-understanding problems for robotic and medical applications.

Mac Schwager (Member, IEEE) received the B.S. degree in mechanical engineering from Stanford University, Stanford, CA, USA, in 2000, and the M.S. and Ph.D. degrees in mechanical engineering from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2005 and 2009, respectively.

He is currently an Associate Professor with the Aeronautics and Astronautics Department, Stanford University. His research interests include distributed algorithms for control, perception, and learning in groups of robots.

Dr. Schwager was the recipient of the National Science Foundation CAREER Award in 2014.