Capturing Subjective First-Person View Shots with Drones for Automated Cinematography

AMIRSAMAN ASHTARI, Visual Media Lab, KAIST and AIT Lab, ETH Zürich
STEFAN STEVŠIĆ, AIT Lab, ETH Zürich
TOBIAS NÄGELI, AIT Lab, ETH Zürich
JEAN-CHARLES BAZIN, KAIST
OTMAR HILLIGES, AIT Lab, ETH Zürich

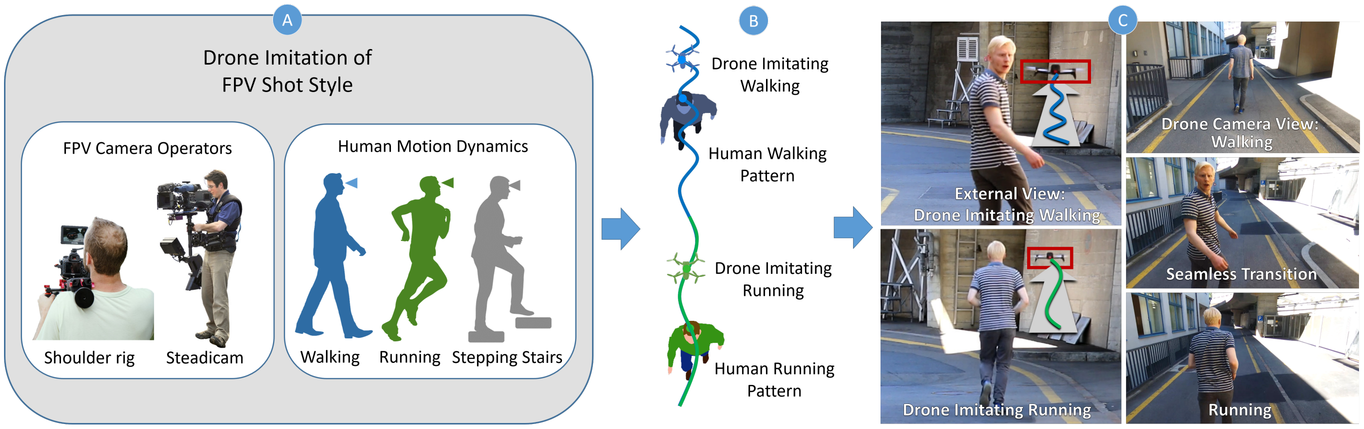

Fig. 1. We propose a computational method that leverages the motion capabilities of drones to imitate the visual look of first-person view (FPV) shots. These shots are usually obtained by a human camera operator that follows the action e.g., by walking or running (A). Such footage is intentionally shot to contain motion artifacts. Our method allows a drone to imitate such shots but offers more flexibility. For example, long shots that imitate a shoulder rig operator walking and then running (B). The result video is acquired in a single session, automatically, with a seamless transition between the operator's motion dynamics (C).

We propose an approach to capture subjective first-person view (FPV) videos by drones for automated cinematography. FPV shots are intentionally not smooth to increase the level of immersion for the audience, and are usually captured by a walking camera operator holding traditional camera equipment. Our goal is to automatically control a drone in such a way that it imitates the motion dynamics of a walking camera operator, and in turn capture FPV videos. For this, given a user-defined camera path, orientation and velocity, we first present a method to automatically generate the operator's motion pattern and the associated motion of the camera, considering the damping mechanism of the camera equipment. Second, we propose a general computational approach that generates the drone commands to imitate the desired motion pattern. We express this task as a constrained optimization problem, where we aim to fulfill high-level user-defined goals, while imitating the dynamics of the walking camera operator and taking the drone's physical constraints into account. Our approach is fully automatic, runs in real time, and is interactive, which provides artistic freedom in designing shots. It does not require a motion capture system, and works both indoors and outdoors. The validity of our approach has been confirmed via quantitative and qualitative evaluations.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Robotic planning**; **Motion path planning**.

Additional Key Words and Phrases: cinematography, aerial videography, filmmaking, quadrotor camera, human motion model

ACM Reference Format:
Amirsaman Ashtari, Stefan Stevšić, Tobias Nägeli, Jean-Charles Bazin, and Otmar Hilliges. 2020. Capturing Subjective First-Person View Shots with Drones for Automated Cinematography. *ACM Trans. Graph.* 39, 5, Article 159 (August 2020), 14 pages. https://doi.org/10.1145/3378673

Authors' addresses: Amirsaman Ashtari, Visual Media Lab, KAIST, AIT Lab, ETH Zürich, a.s.ashtari@kaist.ac.kr, aashtari@student.ethz.ch; Stefan Stevšić, AIT Lab, ETH Zürich, stefan.stevsic@inf.ethz.ch; Tobias Nägeli, AIT Lab, ETH Zürich, naegelit@inf.ethz.ch; Jean-Charles Bazin, KAIST, bazinjc@kaist.ac.kr; Otmar Hilliges, AIT Lab, ETH Zürich, otmar.hilliges@inf.ethz.ch.

1 INTRODUCTION

The cinematographic effects of different shot types can profoundly affect the way the audience interprets the scene and character [Katz 1991]. Common shot types can be separated into two main categories. The first category are third-person point of view or *objective shots*: this narrative style presents the action from the perspective of an ideal external observer. The second category are first-person point of view (FPV) or *subjective shots*: these let the audience experience the action through the eyes of a character and refer to the "I" of the story [Katz 1991].

Objective shots require smooth camera motion usually achieved by static cameras, dollies, rails or cranes. In contrast, *subjective shots* are intentionally not perfectly stabilized and contain camera motion that corresponds to walking patterns in order to increase the level of immersion of the audience into the scene. Typically such shots are filmed by a human camera operator using dedicated camera equipment, such as a shoulder rig or a Steadicam (see Figure 1) which dampen walking effects to different degrees.

Both camera types enable unique shots and are commonly used in Holywood productions. Directors value that their style of camera motion places the audience in the scene and gives the scene a frenetic feel, full of energy[1]. However, operating a Steadicam or a shoulder rig requires significant expertise and training[1] [Holway and Hayball 2013] and thus is only reserved for skilled camera operators. This leads to increasing operator- and movie production costs.

Recently, the flexibility, price and relative ease-of-use of drones have led to much attention in the computer graphics and robotics literature [Gebhardt et al. 2016; Gebhardt and Hilliges 2018; Gebhardt et al. 2018; Joubert et al. 2015; Roberts and Hanrahan 2016; Xie et al. 2018]. While paving the way for the use of drones in cinematography, existing approaches are focused on generating smooth camera motion, which is strongly correlated with aesthetically pleasing perception of the resulting footage when filming static scenes (cf. [Gebhardt et al. 2018]). Thus such approaches are only applicable to third-person objective shots, and hence cannot create the desired visual effect to capture *subjective* FPV shots.

In this paper, we propose the first computational approach to automatically capture *subjective* FPV shots with drones. Our method allows to evoke the same visual effect as created manually by a skilled camera operator but requires very little to no training. Therefore, it provides the video director the exact control over the type and amount of motion patterns and enables precisely controlled camera motion which can be replicated over multiple takes. Filming subjective shots with drones is a challenging task. Such a method must be precise to ensure repeatability of shots yet must run in real time in order to be able to react to the motion of the actor. Moreover, such a method must be capable of creating camera motion that evokes the immersive feeling associated to subjective shots while respecting the physical limitations of the drone (which are very different to those of a human).

Embracing these challenges, we propose a constrained optimization method to generate drone and gimbal control inputs in real time. At the core of the method lies the concept of imitating a physical system (the human) via a different dynamical system (the drone). To this end, we propose a method that leverages a parametric model of human walking to generate the desired velocities of the camera in the camera frame. A receding horizon closed-loop optimal control scheme then produces the drone and gimbal inputs to best match the desired camera motion and a user-specified trajectory along which the drone progresses.

[1]https://www.lightsfilmschool.com/blog/how-to-use-a-shoulder-rig-filmmaking-tutorial

This results in a close imitation of the visual effect achieved by experienced operators but provides the director more flexibility in terms of chaining shots and transitioning between shot styles. Furthermore, the method provides flexibility in terms of the environment in which filming is possible. For example, using a Steadicam while climbing stairs or filming on unsteady surfaces is normally difficult for human operators but becomes straightforward when using drones.

Our approach runs both in indoor and outdoor environments, and does not require a motion capture system. We demonstrate our approach in a number of scenarios, such as imitating different human dynamics (walking, running and stepping stairs), with different speeds, as well as different motion directions (e.g., forward, backward and sideways). Moreover, we demonstrate that our approach enables seamless transitions between different shot styles that were previously impossible, for example from walking FPV to smooth aerial dolly shots. We evaluate our method in a number of quantitative and qualitative experiments. Finally, a large perceptual study suggests that the resulting footage is virtually indistinguishable from footage captured by a professional camera operator.

2 RELATED WORK

Automated cinematography in virtual environments: Several methods have been proposed for automated camera placement [Lino et al. 2011], control [Drucker and Zeltzer 1994; Lino and Christie 2012, 2015] and motion planning [Li and Cheng 2008; Yeh et al. 2011] in the context of automated cinematography in *virtual* environments [Christie et al. 2008]. However, since they are designed for virtual environments, they do not consider the physical constraints of real systems, and thus might generate physically unfeasible drone trajectories.

Automated drone cinematography: Several tools have been proposed to plan aerial videography. For example, some existing apps and drones allow the users to place waypoints on a 2D map [APM 2016; DJI 2016; Technology 2016] or to interactively control the drone's camera as it follows a pre-determined path [3D Robotics 2015]. However, these tools generally 1) do not ensure the physical feasibility of the generated drone trajectories, and 2) are not designed to imitate the visual look of subjective FPV shots.

The (offline) planning of *physically* feasible drone camera trajectories for cinematography has recently received a lot of attention [Gebhardt et al. 2016; Gebhardt and Hilliges 2018; Gebhardt et al. 2018; Joubert et al. 2015; Roberts and Hanrahan 2016; Xie et al. 2018]. Such tools allow for planning of aerial shots and employ optimization that considers both aesthetic objectives and the drone's physical constraints. However, these methods are designed to replicate the visual look of *smooth* camera motions, usually acquired by dollies, rails and cranes for third-person objective shots. Moreover, they work *offline*, i.e., they cannot interactively react in real time to moving actors in dynamic scenes.

Online trajectory generation for drone cinematography: In the context of capturing dynamic scenes, several works have been proposed to generate real-time drone camera trajectories. For example, planning camera motion in a lower dimensional subspace, [Galvane et al. 2016; Joubert et al. 2016] achieved real-time performance. [Nägeli et al. 2017a] used a Model Predictive Controller (MPC) to locally optimize visual cinematographic constraints like the position and size of the captured targets on the screen. [Nägeli et al. 2017b] extended this work by using a Model Predictive Contour Control (MPCC) scheme for multiple drones and enabled actor-driven tracking on a geometric reference path i.e., their method does not require a predefined time-stamped reference camera path as MPC approaches do. It allows the user to design the reference camera path, referred to as virtual camera rails. [Galvane et al. 2018] proposed a solution for the computation of these virtual rails and provided a high-level coordination strategy for the placement of multiple drones. Similarly to these methods, our approach can fulfill various high-level user goals in real time for dynamic scenes, such as following a user-defined camera path, velocity and orientation. Our key novelty is that we optimize the drone commands to also imitate the dynamics of a walking camera operator in real time, and thus automatically capture subjective FPV shots. This means that, in contrast to previous methods, our approach considers two dynamical systems in its formulation (dynamics of a drone and a walking camera operator).

3 PRELIMINARIES

3.1 Notation

Here we introduce the most important notation used in the paper. For a full treatment we refer to Appendix A. Throughout this paper, we denote position, velocity and orientation vectors as $\mathbf{p}^{(\cdot)}$, $\mathbf{v}^{(\cdot)}$ and $\mathbf{o}^{(\cdot)}$, respectively. Superscripts q, h, m and s refer to the quadrotor, human walking model, imitation model and smooth reference path, respectively (e.g., $\mathbf{p}^q$ denotes the quadrotor's position vector). Subscripts x, y and z denote the directions in the corresponding world or body frame (e.g., v_x^h denotes the human walking model velocity in x direction). States, inputs and outputs of a dynamical system are denoted as $\mathbf{x}^{(\cdot)}$, $\mathbf{u}^{(\cdot)}$ and $\mathbf{y}^{(\cdot)}$, respectively (e.g., $\mathbf{y}^h$ refers to the output vector of the human walking model). For better understanding of the human walking model, we just denote its states as $\boldsymbol{\theta}^h$. The estimated value of any variable x is written $\hat{x}$, while its setpoint (i.e., its desired value) is written $\bar{x}$. All units are in SI system i.e., positions (m), velocities (m/s), orientations (rad) and angular velocities (rad/s).

3.2 Quadrotor Dynamical Model

Our method is agnostic to the quadrotor hardware. Our experimental setup is a Parrot Bebop 2, and we use a quadrotor dynamical model similar to [Nägeli et al. 2017b]. Let $\mathbf{p}^q \in \mathbb{R}^3$ denote the quadrotor's position, $\mathbf{v}^q = [v_x^q, v_y^q, v_z^q] \in \mathbb{R}^3$ its velocity and $\mathbf{o}^q = [\Phi^q, \Theta^q, \psi^q] \in \mathbb{R}^3$ its orientation (roll, pitch and yaw). The identified quadrotor model is of the form $\mathbf{x}_{k+1}^q = f^q(\mathbf{x}_k^q, \mathbf{u}_k^q)$ where

$$\begin{aligned}\mathbf{x}^q &= [\mathbf{p}^q, v_x^q, v_y^q, \Phi^q, \Theta^q, \psi^q, \theta_g, \psi_g]^T \in \mathbb{R}^{10}, \quad \mathbf{x}^q \in \boldsymbol{\chi} \\ \mathbf{u}^q &= [v_z^q, \phi^q, \theta^q, \omega_\psi^q, \omega_{\theta_g}, \omega_{\psi_g}]^T \in \mathbb{R}^6, \quad \mathbf{u}^q \in \boldsymbol{\zeta}\end{aligned} \tag{1}$$

and $f^q : \mathbb{R}^{10} \times \mathbb{R}^6 \rightarrow \mathbb{R}^{10}$ is an identified non-linear map which assigns to the current quadrotor state $\mathbf{x}^q$ and input $\mathbf{u}^q$, the successor state at each instant k. The state of the flying camera consists of its position $\mathbf{p}^q$, velocity (v_x^q, v_y^q) and orientation $(\Phi^q, \Theta^q, \psi^q)$, as well as the gimbal pitch and yaw angles (θ_g, ψ_g). The control inputs are the desired roll and pitch angles of the quadrotor (ϕ^q, θ^q), the translational and rotational velocities of its z-body axis (v_z^q, ω_ψ^q) as well as the pitch and yaw rates of the camera gimbal $(\omega_{\theta_g}, \omega_{\psi_g})$. $\boldsymbol{\chi}$ and $\boldsymbol{\zeta}$ are the set of admissible quadrotor states and inputs derived from its physical limitations.

3.3 FPV Camera Motion Pattern

The typical look of subjective FPV shots is due to the walking pattern of camera operators. Therefore, to imitate FPV shots with drones, we first need to model the human walking dynamics. In our context of cinematography, we need a model that fulfills the following requirements. First, the human walking pattern consists of several components (i.e., the vertical, lateral and rotational walking patterns). Therefore, we need a realistic walking model that simultaneously encompasses all these components. Second, video directors must be able to react to the actor's motion. Hence, the model must be adaptive, such that it can automatically adjust, for example, the operator's step frequency to match the desired walking velocity. This velocity can be defined interactively and in real time by the video director. Third, the model must be usable in real-time drone control schemes. In our drone control scheme, it is important to have a *segment-free* model instead of hybrid and multi-segment models commonly used to simulate human walking pattern [Gregg et al. 2012; Hasaneini et al. 2013; Manchester and Umenberger 2014; Sharbafi and Seyfarth 2015]. In other words, the model must have a single function in the time-domain to explain the different phases of the human walking instead of having multiple functions to explain each phase. The segment-free model simplifies our controller design because we do not have to consider multiple models, nor the switching effect between each of them.

Camera operator walking pattern: Several methods have been proposed in the fields of biomechanics and biped robotics to simulate realistic mechanical models of human walking, such as the inverted pendulum model, passive dynamics walking and the zero moment point(ZMP)-based method, see [Xiang et al. 2010] for a review of existing methods. However, these models cannot be directly used for our drone FPV shot imitation purpose because they are not segment-free. The model of [Carpentier et al. 2017] is the first segment-free time-domain model developed and demonstrated to explain human walking. However, it does not consider the lateral or rotational walking patterns, which are important to replicate subjective FPV shots. [Faraji and Ijspeert 2017] proposed a 3D human walking model appropriate for Model Predictive Control (MPC) schemes. However, the center of mass height is constant in their model (e.g., no vertical displacement). Zijlstra and Hof [1997] showed that lateral movement of the walking pattern is a simple sinusoidal signal based on a 3D inverted pendulum model. We will build upon the ideas of [Carpentier et al. 2017] and [Zijlstra and Hof 1997] to make a single adaptive walking camera model (see Section 4.2 for more details).

4 METHOD

4.1 Overview

We propose a computational method to imitate the visual characteristics of FPV shots with a drone. Our real-time algorithm allows drones to imitate the dynamics of a walking camera operator while following a user-defined trajectory and considering the physical limitations of the drone. Our method allows to switch between different shot styles with seamless transitions (e.g., switching from a smooth dolly shot to a FPV shoulder rig shot). Our method also enables a director to interactively adjust the parameters of the camera walking model, such as the walking speed of the operator or the amount of camera shake. Our algorithm is illustrated in Figure 2 and iteratively performs the following steps (letters below correspond to those in Figure 2).

Visual style: The video director can interactively define and adjust in real time the following (collectively called user preferences):

(A) the desired trajectory as a global guidance for the drone camera path and orientation, i.e., the user only needs to define the desired key-frames similar to [Gebhardt et al. 2018; Nägeli et al. 2017b] (desired position and orientation of the camera for sparse locations in 3D space).
(B) the desired shot style (camera equipment) for drone imitation, such as FPV shoulder rig or Steadicam shot styles.
(C) the camera velocity. For example, to smoothly accelerate from low to high walking speed, up to running.

FPV Shot Generation: At each iteration,

(D) corresponding to the user preferences, our method adaptively generates the walking camera model. The generated model also includes damping parameters based on the camera equipment selected by the user. The walking camera model then predicts positional and angular velocity set-points over its prediction horizon.
(E) the predicted set-points, the desired trajectory and the drone's on-board sensor data (e.g., IMU and optical flow sensor) are used as inputs to compute the drone control commands via a receding horizon closed-loop optimal controller.

Using our approach, the drone dynamics converge to the desired walking camera operator dynamics (see the blue curve illustrating the drone motion as a walking operator in the output block of Figure 2) while it follows the desired trajectory (the red dashed curve in the output block of Figure 2).

4.2 Walking Camera Model

Following the discussion of Section 3.3, we will build upon the ideas of [Carpentier et al. 2017] and [Zijlstra and Hof 1997] to model the vertical and lateral displacements of the human walking pattern, respectively. We will combine them into one single model, make it adaptive, include rotational displacements and further extend it to simulate the damping effects of cinematographic equipment.

Human walking model: We now present our adaptive and segment-free camera operator walking model. Let $\mathbf{p}^h = [p_x^h, p_y^h, p_z^h] \in \mathbb{R}^3$ denote the position of the human walking model in its body-frame and $\mathbf{v}^h = [v_x^h, v_y^h, v_z^h] \in \mathbb{R}^3$ its velocity while ψ^h and ω^h are the human walking rotation and angular velocity around its z-body axis, respectively (see Figure 3). We model the human lateral (left-right) walking motion p_y^h and its rotation around the z-body axis ψ^h as a sinusoidal signal while we formulate its vertical (up-down) displacement p_z^h as a parametric curtate cycloid curve[2] w.r.t. the human walking time[3] τ_z^h (green, black and blue curves in Figure 3, respectively). We use the simple constant velocity model at each sampling time to model the human walking motion in the x-axis $\dot{p}_x^h = v_x^h$ (orange curve in Figure 3). The only input to our model is the user-defined walking velocity v_x^h at each time instant. The outputs, states, and the initial conditions are

$$\begin{aligned} \mathbf{y}^h &= [p_y^h, v_y^h, p_z^h, v_z^h, \tau_z^h, \psi^h, \omega^h]^T, \\ \boldsymbol{\theta}^h &= [\theta_y^h, \theta_z^h, \theta_\psi^h]^T, \quad \boldsymbol{\theta}^h(0) = [0, \pi/2, 0]^T. \end{aligned} \tag{2}$$

The outputs $\mathbf{y}^h$ are the human lateral (p_y^h, v_y^h) and vertical (p_z^h, v_z^h, τ_z^h) walking pattern, as well as its rotation and angular velocity (ψ^h, ω^h) around the z-body axis. The states $\boldsymbol{\theta}^h$ of our model are the phase of the lateral, vertical and rotational walking pattern denoted as θ_y^h, θ_z^h and θ_ψ^h, respectively. Our human walking model is represented as a continuous non-linear state space model

$$\begin{aligned} \dot{\boldsymbol{\theta}}^h &= \begin{bmatrix} \omega_y^h & \omega_z^h & \omega_\psi^h \end{bmatrix}^T, \\ \mathbf{y}^h &= [a_y \sin(\theta_y^h) \quad a_y\omega_y^h \cos(\theta_y^h) \quad h^h - r\sin(\theta_z^h) \\ &\quad -r\omega_z^h \cos(\theta_z^h) \quad t + r\cos(\theta_z^h) \quad a_\psi \sin(\theta_\psi^h) \quad a_\psi \omega_\psi^h \cos(\theta_\psi^h)]^T, \end{aligned} \tag{3}$$

where a_y, a_ψ and r are the amplitudes of the human lateral, rotational and vertical walking pattern. h^h is the height of a human and t is the time. ω_y^h, ω_z^h and ω_ψ^h denote the lateral, vertical and rotational human walking pattern angular frequencies, respectively. We adaptively compute these walking frequencies $(\omega_y^h, \omega_z^h, \omega_\psi^h)$ by computing the corresponding step length l_s^h and step frequency f_s^h from the user-defined walking velocity v_x^h as

$$\begin{aligned} l_s^h &= \beta_0 + \beta_1 |v_x^h| + \beta_2 {v_x^h}^2 && \text{(step length)} \\ f_s^h &= \frac{|v_x^h|}{l_s^h} && \text{(step frequency)} \\ \omega_y^h &= \omega_\psi^h = \pi f_s^h && \text{(walking model frequencies)} \\ \omega_z^h &= 2\pi f_s^h \end{aligned} \tag{4}$$

where β_0, β_1 and β_2 are fixed known constants identified for a walking person by [Seitz and Köster 2012]. We discretize our human walking model and use it to predict the walking pattern over a finite prediction horizon N. Since τ_z^h is a non-linear function of time, we re-sample the vertical motion pattern over the horizon. Thus attaining samples $(\tau_{z_k}^h, v_{z_k}^h)$ where $k \in 1, \cdots, N$, at the sampling rate required for drone control.

[2]A parametric curtate cycloid is the curve described by a point rigidly attached to a wheel rolling on a flat surface (see blue curve and dotted circles in Figure 3).

[3]τ_z^h is the x-axis of the parametric curtate cycloid curve p_z^h and is a non-linear function of time t.

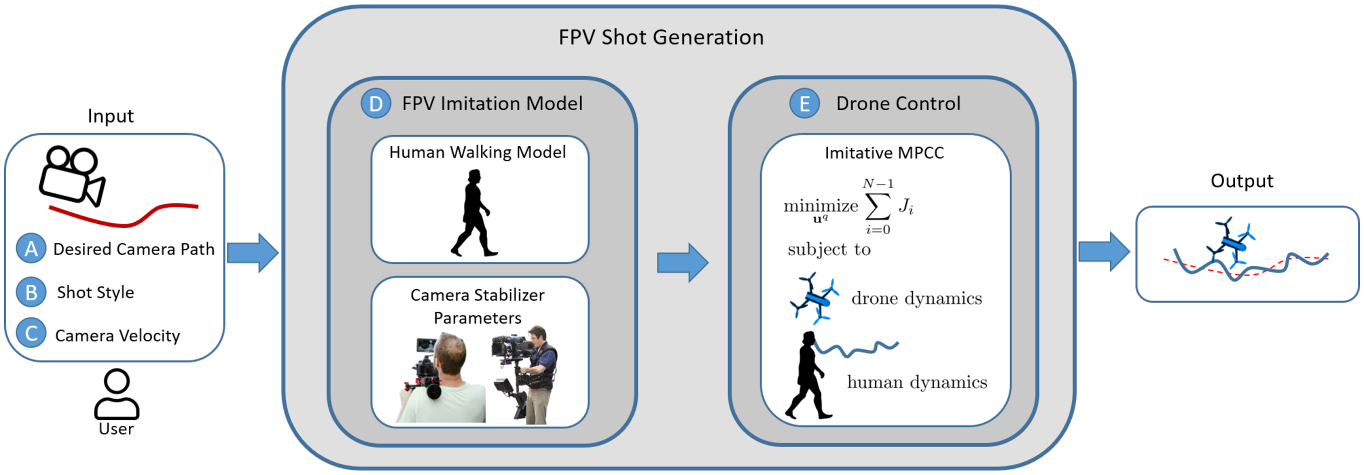

Fig. 2. Overview of our FPV shot generation method. Left to right: A user defines the reference camera path, shot style and forward velocity of the camera operator (A, B, C). Then we predict the velocity and orientation profile of the desired imitation model in a prediction horizon (D). Finally our MPCC formulation computes the control commands (E) such that the drone flies closely to the user-defined camera path (dotted red curve) while it replicates the desired imitation model dynamics (blue curve).

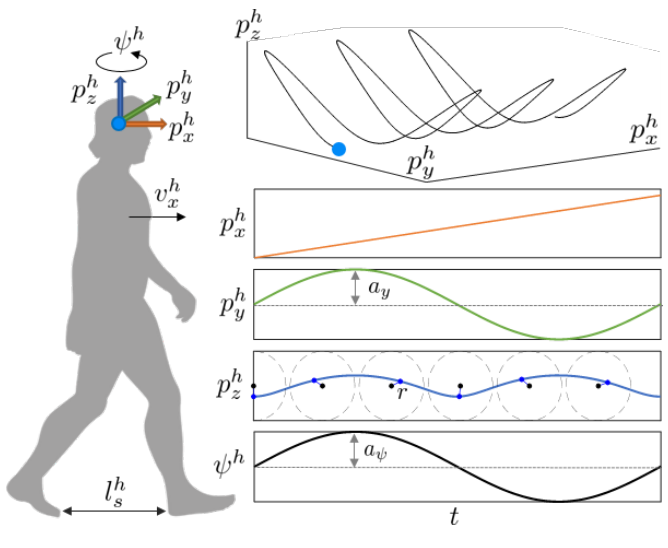

Fig. 3. Our human walking model in its body-frame. See text for details.

Camera stabilizers: Each camera stabilizer (e.g., shoulder rig, Steadicam or dolly) is designed to damp some components of the human walking motion. In our human walking model, the parameters r, a_y and a_ψ in Eq. (3) define the range of the vertical, lateral and rotational human walking pattern. These parameters can be adjusted to imitate each of these camera stabilizers. For example, to capture a smooth dolly shot, the video director sets them all to zero in our human walking model. To imitate shakier human shots, the director can simply set them to higher values in an interactive manner via online visual feedback (see Section 8.3). Hence, our method can both imitate and seamlessly switch between various shot styles (e.g., objective smooth dolly or subjective FPV shots) using the same algorithm.

4.3 Imitative MPCC Formulation

Given the dynamical models of drones (Section 3.2), walking camera operator and camera stabilizer (Section 4.2), we now present our approach to compute the drone commands to imitate the target camera's motion pattern. We express this task as a constrained optimization problem, where we aim to fulfill the high-level user-defined goals, while imitating the dynamics of the walking camera operator and taking the drone's physical constraints into account. In the following, we first define the different cost terms of our optimization and then present our general optimization formulation.

Following the human walking model dynamics: To enable a quadrotor to imitate the dynamics of a walking camera operator (defined in Eq. (3)), the quadrotor's dynamical model must follow the corresponding human walking model dynamics. This imitation constraint means that the quadrotor position $\mathbf{p}^q$ and orientation ψ^q states must follow the corresponding position $\mathbf{p}^h$ and orientation ψ^h of the human walking model at each time stage k. We use penalty functions to convert our constrained problem (e.g., imitating human dynamics) into an unconstrained problem by introducing an artificial penalty for violating the constraint. First, we use the human walking model dynamics to predict its velocity and angular velocity over a prediction horizon i.e., $\mathbf{v}_k^h = [v_{x_k}^h, v_{y_k}^h, v_{z_k}^h]$ and ω_k^h for all k from 1 to N. At each instant k, we set the initial conditions of the human walking model with its current state. Then, to ensure the convergence of the velocity and angular velocity of the quadrotor to the human walking model states in a prediction horizon, we define the following imitation cost term c^{im} as:

$$c^{im}(\mathbf{v}^q, \mathbf{v}^h, \omega_\psi^q, \omega^h) = ||\mathbf{v}^q - \mathbf{v}^h||^2 + ||\omega_\psi^q - \omega^h||^2. \quad (5)$$

When the velocities and angular velocities of the quadrotor converge to the corresponding human walking model velocities, the quadrotor position $\mathbf{p}^q = \int \mathbf{v}^q dt$ and its orientation $\psi^q = \int \omega_\psi^q dt$ will also follow the human walking model position and orientation states. Since our drone inputs are velocity and angular velocity in and around its z-body axis (see Eq. (1)), we define the imitation term based on the velocities and angular velocities, instead of positions and orientations, to directly compute the drone commands. In addition, it may be more intuitive for a user to interactively define the desired camera velocity instead of its position.

Following a desired trajectory: Our goal is to imitate the dynamics of a walking camera operator on a user-defined smooth path. Since the human walking model is formulated in the human body-frame, we need to reformulate our imitation cost (see Eq. (5)) based on the drone velocities in the drone's body-frame and consider the effects of the user-defined smooth path (see the dashed black line in Figure 4). We assume that a human walks on a desired path while its forward velocity is in the tangent direction of the path at each instant, and its vertical displacement (up-down) is along the z-direction in the world frame (see Figure 4).

Denoting with $\mathbf{a}_t$ the normalized tangent vector of the desired trajectory, let $\mathbf{a}_z$ define the unit vector along the z-direction in the world frame. Therefore, the normalized vector orthogonal to the desired trajectory is obtained by $\mathbf{a}_n = \mathbf{a}_z \times \mathbf{a}_t$ i.e., the vector in the lateral (left-right) direction of the human walking model. To imitate the motion of a camera carried by a walking operator, we project the quadrotor velocity $\mathbf{v}^q$ onto the $\mathbf{a}_t$, $\mathbf{a}_n$ and $\mathbf{a}_z$ directions and encourage it to be similar to the human walking velocities (v_x^h, v_y^h, v_z^h) in its body-frame as (see Figure 4)

$$\begin{aligned} c^{\mathbf{a}_t}(\mathbf{v}^q, v_x^h, \mathbf{a}_t) &= \|e_t\|^2 \quad \text{where} \quad e_t = \langle \mathbf{v}^q, \mathbf{a}_t \rangle - v_x^h, \\ c^{\mathbf{a}_n}(\mathbf{v}^q, v_y^h, \mathbf{a}_n) &= \|e_n\|^2 \quad \text{where} \quad e_n = \langle \mathbf{v}^q, \mathbf{a}_n \rangle - v_y^h, \\ c^{\mathbf{a}_z}(\mathbf{v}^q, v_z^h, \mathbf{a}_z) &= \|e_z\|^2 \quad \text{where} \quad e_z = \langle \mathbf{v}^q, \mathbf{a}_z \rangle - v_z^h. \end{aligned} \tag{6}$$

Furthermore, we encourage its angular velocity ω^h to be similar to the corresponding quadrotor state:

$$c^{\omega_\psi}(\omega_\psi^q, \omega^h) = \|\omega_\psi^q - \omega^h\|^2. \tag{7}$$

We also allow a user to interactively and in real time control the orientation of the drone camera by adjusting both the drone pan-tilt gimbal[4] and the drone rotation around its z-body axis. To this end, we define a cost term for the desired drone rotation around its z-axis:

$$c^{\psi}(\psi^q, \bar{\psi}^q) = \|\psi^q - \bar{\psi}^q\|^2, \tag{8}$$

where $\bar{\psi}^q$ denotes the desired reference yaw angle of the camera.

To follow the desired trajectory and avoid drift from the desired path, we add a path following cost composed of the lag and contour terms c^l and c^c based on the approach proposed in [Gebhardt et al. 2018] (see Appendix B, Eq. (12) for more details). In contrast to Model Predictive Control (MPC) approaches which require a time-stamped reference trajectory, our MPCC-based formulation enables the video director to interactively and in real time adjust the desired walking velocity for the drone to react to the motion of the actor. To avoid excessive use of the control inputs and limit progress acceleration rate on the path, we use the cost term c^{inp} (see Appendix B, Eq. (14) for more details).

[4]Parrot Bebop2 has a fast electrical gimbal, and its SDK allows a user to directly set the desired pan-tilt gimbal angles.

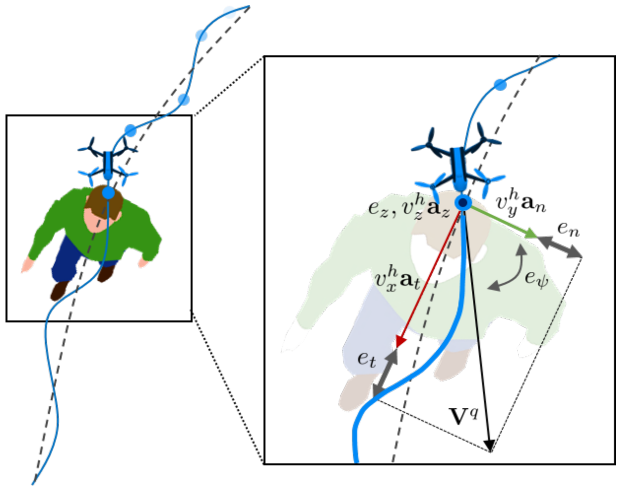

Fig. 4. To track the dynamics of the human walking model, our optimization method encourages the quadrotor velocity $\mathbf{v}^q$ to be similar to the human walking velocities at the directions $\mathbf{a}_t$, $\mathbf{a}_n$ and $\mathbf{a}_z$ on the smooth spline. We minimize the associated errors e_t, e_n and e_z in a prediction horizon, which enables a drone to follow the walking camera operator dynamics (blue curve) on a user-defined smooth path (dashed black curve).

Optimization formulation: Finally, we define our drone imitation objective function by linearly combining the cost terms described above, grouped into four categories: 1) Imitation of the walking camera model: $(c^{\mathbf{a}_t}, c^{\mathbf{a}_n}, c^{\mathbf{a}_z}, c^{\omega_\psi})$ in Eq. (6) and Eq. (7). 2) Camera orientation: c^ψ in Eq. (8). 3) Path following: (c^l, c^c) in Eq. (12). 4) Limiting control inputs: c^{inp} in Eq. (14) to avoid excessive use, leading to jerky camera motion. The final cost term is given by:

$$\begin{aligned} J_k = &\Big(w_{\mathbf{a}}\big(c^{\mathbf{a}_t}(\mathbf{v}_k^q, v_{x_k}^h, \mathbf{a}_{t_k}) + c^{\mathbf{a}_n}(\mathbf{v}_k^q, v_{y_k}^h, \mathbf{a}_{n_k}) + c^{\mathbf{a}_z}(\mathbf{v}_k^q, v_{z_k}^h, \mathbf{a}_{z_k})\big) \\ &+ w_{\omega_\psi} c^{\omega_\psi}(\omega_{\psi_k}^q, \omega_k^h)\Big) + w_\psi c^\psi(\psi_k^q, \bar{\psi}_k^q) \\ &+ \Big(w_l c^l(\theta_k^s, \mathbf{p}_k^q) + w_c c^c(\theta_k^s, \mathbf{p}_k^q)\Big) + c^{inp}(a_k^s, \mathbf{u}_k^q), \end{aligned} \tag{9}$$

where the weights $w_{\mathbf{a}}$, w_{ω_ψ}, w_ψ, w_l and w_c are adjusted for trade-off between imitation of the dynamical walking model and following the desired path. We used the same weights for all the results shown in this paper and their values are tabularized in Appendix D.

To compute the drone commands, the final optimization problem is then formulated as:

$$\begin{aligned} &\underset{\mathbf{x}^q, \mathbf{u}^q, \Theta^s, a^s}{\text{minimize}} \sum_{k=0}^{N-1} J_k + w_N J_N && (10) \\ &\text{subject to } \mathbf{x}_0^q = \hat{\mathbf{x}}^q(t) && \text{(initial state)} \\ &\Theta_0^s = \hat{\Theta}^s(t) && \text{(initial progress)} \\ &\mathbf{x}_{k+1}^q = f(\mathbf{x}_k^q, \mathbf{u}_k^q) && \text{(system model)} \\ &\Theta_{k+1}^s = \mathbf{A}^s \Theta_k^s + \mathbf{B}^s a_k^s && \text{(progress along path)} \\ &\mathbf{x}_k^q \in \chi && \text{(state constraints)} \\ &\mathbf{u}_k^q \in \zeta && \text{(input constraints)} \\ &\mathbf{0} \le \Theta_k^s \le \Theta_{max}^s && \text{(path progress bounds)} \\ &a_{min}^s \le a_k^s \le a_{max}^s && \text{(progress input limits)} \end{aligned}$$

where the vectors $\hat{\mathbf{x}}^q(t)$ and $\hat{\mathbf{\Theta}}^s(t)$ denote the estimated or measured values of the current quadrotor $\mathbf{x}^q$ and path progress $\mathbf{\Theta}^s$ states. For more information about the progress along the path, path progress bounds and progress input limits, see Eq. (13) and Eq. (14) in the Appendix B. The scalar $w_N > 0$ is a weight parameter used to weight a so-called terminal cost J_N (i.e., Eq. (9) where $k = N$ and N is the prediction horizon). The terminal cost is usually weighted more than the costs in previous stages (i.e., $\sum_{k=0}^{N-1} J_k$), which provides a solution that is closer to the infinite horizon (i.e., $\sum_{k=0}^{\infty} J_k$) solution [Nägeli et al. 2017b]. Solving this optimization problem at each step enables a drone to imitate the dynamics of a walking camera operator while following a desired reference path.

Our optimization formulation is general in the sense that it is not limited to human walking, which we focus on in our experiments, and can imitate other dynamical systems. Please refer to Appendix C for more information about drone imitation of a general dynamical system.

5 IMPLEMENTATION

Optimization: Our experiments are conducted on a standard laptop (Intel(R) Core(TM) i7-7700HQ CPU @2.8 GHz). We use the drone on-board sensors and its visual odometry data in our control algorithm. Our optimization system (see Eq. (10)) is implemented in MATLAB and solved by the FORCES Pro software [Domahidi and Jerez 2017] which generates fast solver code, exploiting the special structure in a non-linear program (NLP). Our solver generates feasible solutions in real-time at 20Hz. We initialize the solver of Eq.(10) with the solution vector computed at the previous time-step, perturbed by random noise. The method is robust to initialization as we did not observe significant changes in solve time even if the initialization is drastically perturbed.

Drone hardware: In all our experiments, we use an unmodified Parrot Bebop2 drone with an integrated electronic gimbal. We directly send the control commands at 20Hz to the drone, and read on-board sensor and visual odometry data via ROS [Quigley et al. 2009]. The sensory data is up-sampled from approximately 5Hz to 20Hz via a Kalman estimator. No motion capture system is used in any of the experiments.

User interaction: The user interactively controls the operator walking model parameters via a joystick in real time. We use separate keys on the joystick to interactively change the desired walking velocity of the camera operator, the user-defined shot type and the desired drone's yaw angle as well as the desired gimbal's pan and tilt. Since we conduct our experiments outdoors without a motion capture system, the actor and drone move on pre-defined paths. The actor moves at an arbitrary speed, and the user interactively adjusts the drone camera speed to correspond with the actor's motion. The user can also adjust the amplitudes of the lateral, vertical and rotational human walking pattern. We asked a professional camera operator to tune these parameters based on her preferences for different shot types.

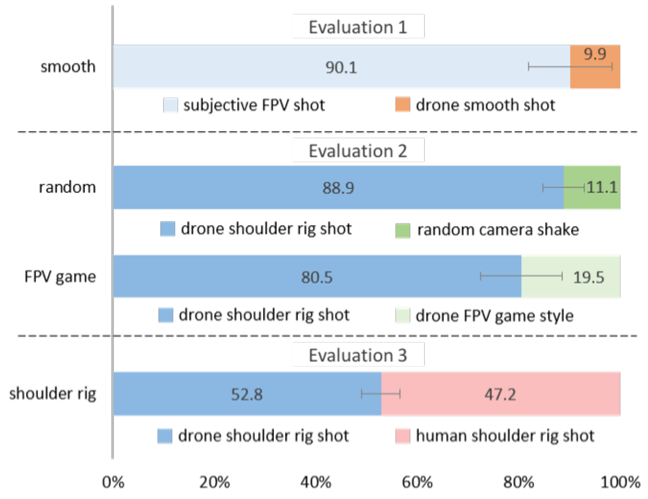

Fig. 5. User study results. Participants' preference in selecting between various shot types based on their quality in representing FPV shots. See text for details.

6 PERCEPTUAL STUDY

6.1 Experimental Setting

To qualitatively asses our drone imitation algorithm, we conducted three evaluations, as described in the following.

Evaluation 1 [smooth drone shots vs. FPV shots]: The goal is to check whether existing drone cinematography methods designed to replicate smooth drone shots can capture subjective FPV shots. In addition, we want to investigate whether people can distinguish between smooth and FPV shots. For this, we compared 1) smooth drone shots captured by a state-of-the-art drone cinematographic method [Gebhardt et al. 2018] with 2) subjective FPV shots captured by a Steadicam or a shoulder rig or our algorithm imitating a human shoulder rig operator.

Evaluation 2 [human motion model]: Furthermore, we compared shots obtained by 1) our drone imitation of a human shoulder rig operator vs. 2) random shaky motions of the drone and vs. 3) our drone algorithm imitating a simple walking style proposed by [Lécuyer et al. 2006] in the context of video games (that we refer to as "FPV game" style).

Evaluation 3 [our drone shots vs. human shoulder rig shots]: To verify whether our drone imitation method can capture FPV shots that look like a shot captured by a human operator, we compared 1) shots captured using a shoulder rig operated by a camera professional vs. 2) shots captured by our drone imitation algorithm set to the shoulder rig style.

For fair visual comparisons of the shots, we used the same camera on the drone and on the shoulder rig in all the experiments. The study was conducted online. For each comparison, we placed two videos side-by-side, randomly assigned to the left or right, and each video pair was placed in a random order. Each participant had to compare 17 videos (4, 6 and 7 video pairs for evaluation 1 to 3, respectively). The videos in each pair were captured at the

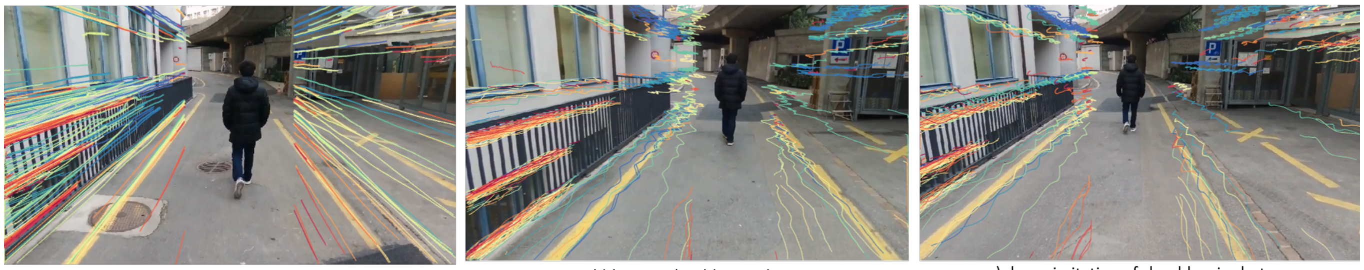
a) smooth drone shot
b) human shoulder rig shot
c) drone imitation of shoulder rig shot

Fig. 6. Representative results of feature trajectories tracked in the image space of a) a smooth drone shot [Gebhardt et al. 2018; Nägeli et al. 2017b], b) human shoulder rig shot and c) our drone imitation of a shoulder rig shot. Each feature point trajectory is shown with a different color. The feature trajectories by our drone imitating a shoulder rig operator (c) are similar to those of the human shoulder rig shot (b), while different to those of the smooth drone shot that resemble line directions (a).

same scene, following an actor from behind. The participants were asked to answer "which video represents the first-person's point of view, i.e., feeling more like the view of a person walking behind the actor, better". They had to answer a forced binary choice: "left video" or "right video".

6.2 Results of the Perceptual Study

In total, 106 participants answered the online survey and the results are shown in Figure 5. Based on the user study results, we draw the following conclusions.

Evaluation 1 in Figure 5 shows that 90.1% of participants preferred subjective FPV shots (captured by our algorithm or a Steadicam or a shoulder rig) over smooth shots. This suggests that people can easily distinguish the visual differences between objective smooth drone shots and subjective FPV shots. It also confirms that state-of-the-art drone methods cannot capture subjective shots, due to the fact that they aim to optimize the smoothness of the drone camera trajectory or follow a smooth path.

Moreover, 88.9% and 80.5% (see Evaluation 2 in Figure 5) of participants preferred our drone algorithm imitating a human shoulder rig operator over the random camera shakes and our drone algorithm imitating the FPV game style, respectively. This shows that our human walking model (Section 4.2) leads to a higher level of cinematographic FPV shot imitation than simply applying random perturbations or the simple FPV game walking model.

Finally, Evaluation 3 in Figure 5 shows that the preference of participants w.r.t. real human operator and our drone shots is similar to chance level (47.2% vs. 52.8%). This indicates that our automatic drone method can capture subjective FPV shots that are visually indistinguishable to those manually captured by a human camera operator.

7 EVALUATION IN IMAGE SPACE

The above user preferences provide evidence for the utility of our method. We note that effects in image space are the underlying factors that influence aesthetics of the shot. In this section, we qualitatively and quantitatively measure such visual features. Each shot style results in a different motion pattern of feature points that can be tracked in the image space. For example, feature trajectories of a smooth, linear shot are expected to be similar to lines, while they

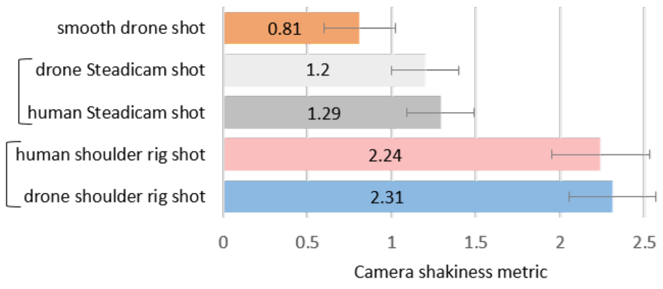

Fig. 7. Quantitative comparison of camera shakiness in image space.

should be shakier for a shoulder rig shot. In this section, we compare the trajectories in image space on videos obtained by different approaches both visually and quantitatively. To obtain the feature trajectories, we extract corner points [Shi and Tomasi 1993] and track them via the KLT algorithm [Tomasi and Kanade 1991].

7.1 Qualitative Comparison

Figure 6 provides representative examples of feature trajectories for a) a smooth drone shot captured by a state-of-the-art drone cinematographic method [Gebhardt et al. 2018; Nägeli et al. 2017b], b) a shot manually captured with a shoulder rig and c) a shot captured by our drone imitating a shoulder rig operator. It shows that the trajectories in the smooth drone shot resemble lines, while they display more variance in the human shoulder rig and our FPV drone shots. Moreover, the trajectories in our drone-based imitation shot resemble those in the human shoulder rig shot.

7.2 Quantitative Comparison

In addition to the visual comparison of the trajectories, we also conduct a quantitative analysis by comparing the amount of shakes of the trajectories. As a metric of shakiness, we compute how much a trajectory deviates from a straight line. For this, we compute the covariance matrix of each feature point trajectory, and then compute its eigenvalues and eigenvectors. The second largest eigenvector is orthogonal to the main direction of the feature trajectory, and its associated eigenvalue thus corresponds to the amount of deviation from this main direction. We compute this value for all the trajectories and compute the average, which provides a measure of shakiness for the video.

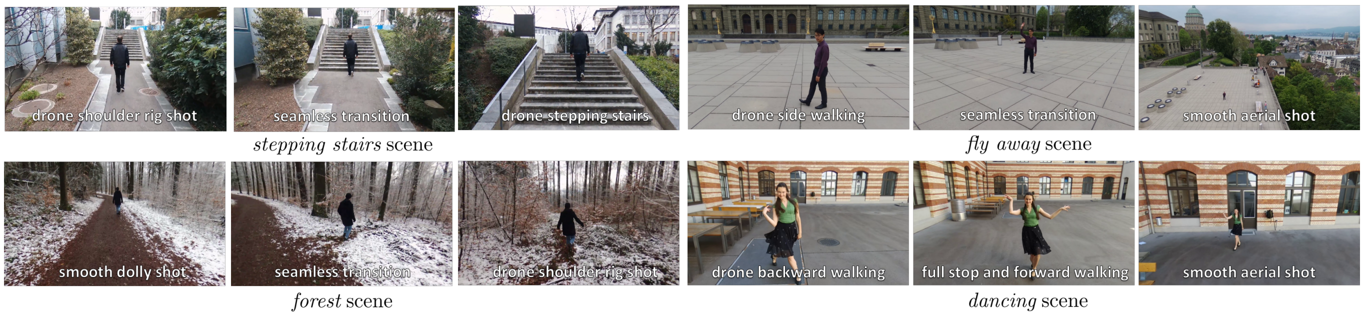

Fig. 8. Thumbnails from some of our representative drone result videos.

We compute the shakiness metric for each video of our user study, and show the average metric value per shot type in Figure 7. Several observations can be made. First of all, the smooth drone shots by [Gebhardt et al. 2018; Nägeli et al. 2017b] provide the lowest value (0.81) since these methods (and other methods of drone cinematography, see related work) aim to optimize for smooth motion. Second, our drone imitation of shoulder rig shots has a similar value to the shoulder rig shots captured by a human operator (2.31 and 2.24 respectively). This is an additional indication that our proposed approach can imitate shoulder rig shots. Third, (human) shoulder rig shots (2.24) are more shaky than (human) Steadicam shots (1.29). Fourth, our drone imitation of Steadicam shots has a similar value to the human captured Steadicam shots (1.2 and 1.29 respectively), which also indicates that our approach can imitate Steadicam shots.

8 QUALITATIVE EXPERIMENTS

8.1 Drone Imitation of Other Human Motions

The results shown so far in the paper are based on the human walking model described in Section 4.2. In this section, we show that our approach is generalizable to different human motions, including stepping stairs and running.

Stepping stairs: Operating a shoulder rig or a Steadicam on stairs is challenging for human operators due to the unwanted jerks transferred to the camera stabilizer while they step the stairs. In contrast, our drone approach can simply imitate human stepping stairs with slight modifications of our general formulation. To model human stepping on stairs, we adjust the step length in our human walking model formulation so that it corresponds to the approximate height and depth of the stairs. In order to consider small delays of each foot stepping on the stairs to reach the next step, we slightly reduce the user-defined walking velocity on each step with a sinusoidal pattern with the same frequency as stepping. Figure 9 and the supplementary video show a representative result (see Figure 8 for representative frames).

Walking speed and running: Our method can also be used to imitate a human accelerating from very low to high speed, up to running. We let a user interactively and in real time increase the desired camera velocity with a joystick. A representative result is provided in Figure 1 and the supplementary video. In the first part of the video sequence, we show how our algorithm adaptively and in real time tunes the step length and step frequency corresponding to the desired camera velocity.

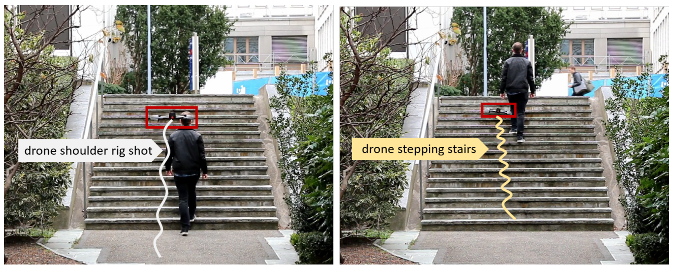

Fig. 9. Drone imitation of human stepping stairs. Left: drone imitation of a shoulder rig shot while the walking actor approaches the stairs. Right: drone imitation of a human stepping stairs (another dynamical model). Transition between the two imitation models is seamless. The drone is highlighted in red for better visibility.

In the second part, our approach is used to imitate human running which is given by another dynamical system. To model human running, we build upon our human walking model (see Eq. (3)) and modify two components. First, we change its initial condition in Eq. (2) to $\boldsymbol{\theta}^h = [0, 0, 0]^T$ because a running person reaches the peak height in the middle of the "flight" phase of running, in contrast to walking (see Figure 3). In addition, we adaptively adjust the human running step length and step frequency of Eq. (4) where we change the fixed known constants β_0, β_1 and β_2 to the corresponding values of a running human identified by [Bailey et al. 2017].

Overall, this experiment shows that our human walking model adaptively tunes the imitation walking step length and frequency corresponding to the actor's forward walking speed to convey the feeling of walking velocity increase in the shot up to running.

8.2 Seamless Transition

In the following, we show how our approach can be used to capture seamless transitions between different shot styles and different walking patterns.

From walking to stepping stairs: In this experiment (see Figure 8 and Figure 9), the drone first imitates a shoulder rig shot style while the walking actor approaches the stairs. Then, the drone imitation model is seamlessly switched to our human stepping stairs model and starts stepping up the stairs as a human. The full length shot is available in the supplementary video.

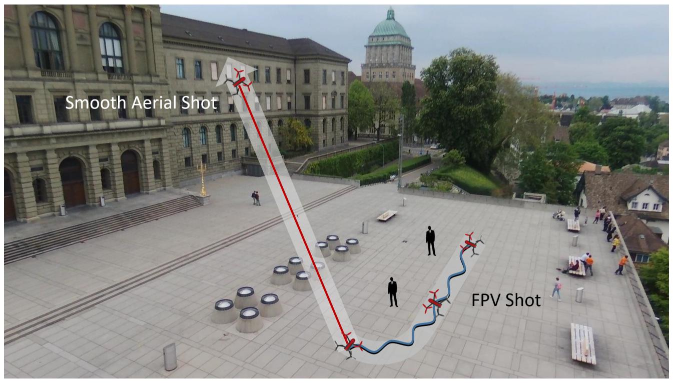

Fig. 10. Transition from a shoulder rig FPV shot following an actor (blue curve) to a smooth aerial dolly shot (red curve) in a seamless manner.

From shoulder rig to dolly shot: Switching between different shot types in one continuous video take (without cuts in between) is challenging, if not impossible, in traditional cinematography. For example, it is not possible in practice to have a seamless transition from a shoulder rig to a crane shot because the camera must be detached and re-attached to the other rig instantaneously. To show the capability of our algorithm to achieve seamless transitions between different shot types in a single session, we designed an experiment where a drone is following an actor in the shoulder rig shot mode, and then flies away in the smooth dolly shot mode (see Figure 10 and *fly away* scene in Figure 8). The resulting video is available in the supplementary material.

An additional representative result is the seamless transition from a smooth dolly shot to a FPV shoulder rig shot, see *forest* scene in Figure 8 for representative frames and its full length shot in the supplementary video.

8.3 Real-Time Interactions

Adjusting the amount of camera shakiness: Our method enables directors to interactively tune the amount of camera shakiness via a joystick and to see the video result in real time. They can increase and decrease the amount of vertical, lateral and rotational shakiness of the camera, see Figure 11. In this way, they can make the shot as smooth as dolly shots or as shaky as shoulder rig or Steadicam shots. We asked a camera operator to interactively design her own Steadicam shot style, and used the resulting drone shots in the quantitative comparison of Section 7.2. Our shakiness metric (see Figure 7) demonstrated that the drone (Steadicam) shots look similar to the human (Steadicam) shots.

We provide directors the artistic freedom to design their own camera stabilizer on top of the operator walking pattern. For example, a director might just be interested in vertical camera shakiness. It is very challenging for a human operator to capture the scene in a way that the camera just goes up and down, or have a specific amount of camera shakiness and precisely repeat this exact style over several video takes. In contrast, our automatic approach allows the directors to design their own style and the visual look can be consistently replicated in different takes and videos.

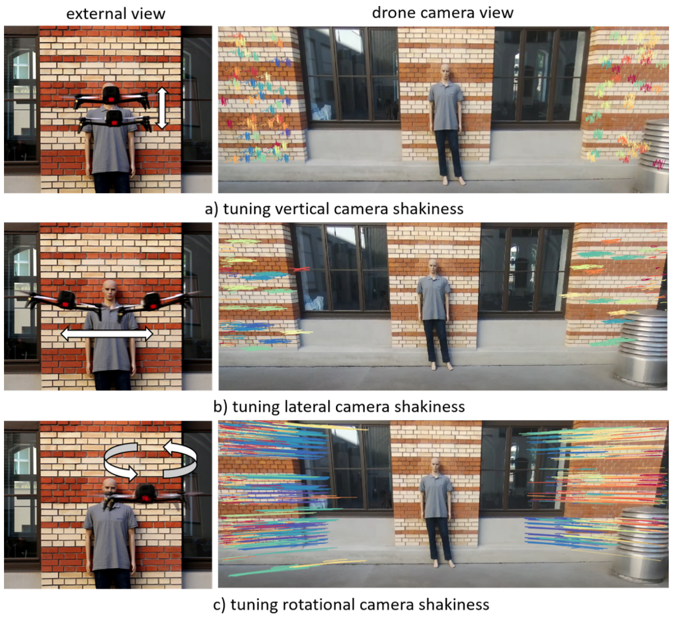
external view
drone camera view
a) tuning vertical camera shakiness
b) tuning lateral camera shakiness
c) tuning rotational camera shakiness

Fig. 11. Interactive tuning of the camera shakiness by adjusting a) vertical, b) lateral, and c) rotational camera shakiness. Left column: external camera view. Right column: drone camera view showing feature trajectories tracked in the image space. Each feature trajectory is shown with a different color.

Dancing: In this experiment, we show that our method can imitate backward walking, forward walking and stationary shots. Moreover, the directions and velocity of the drone are controlled interactively and in real time by the video director to follow the dancer's movements in the scene (see *dance* scene in Figure 8 for representative frames and in the supplementary video). We also switch from FPV shots to aerial shot style in a seamless manner.

9 EVALUATION OF OUR WALKING CAMERA MODEL

We use the Carnegie Mellon University motion capture database [Hodgins 2015] to analyze the accuracy of our human walking model. Since the only input to our model is the walking velocity v_x^h, we compare our model to the human walking data at different walking speeds. To this end, we extract the 3D motion trajectory of the 7^{th} cervical spine vertebrae (CV7) marker (see Figure 12) and use it as ground truth. CV7 is the largest vertebrae located at the most inferior region of the neck[5], and its function is to support the skull and enables head movements. Hence, CV7 represents the head's general motion pattern independent of its rotation.

We conducted both qualitative and quantitative comparisons. For qualitative comparison, we compare the output of our fitted model to the ground truth at different walking speeds. Our model automatically computes the step-length, step-frequency, and lateral and vertical motion patterns that correspond to the real data. Qualitatively, the results confirm that our model follows the general motion pattern of the ground truth (see the supplementary video and a representative result in Figure 12). For quantitative evaluation, we compute the Root Mean Squared Error (RMSE) between

[5]https://v20.wiki.optitrack.com/index.php?title=Biomech_(57)

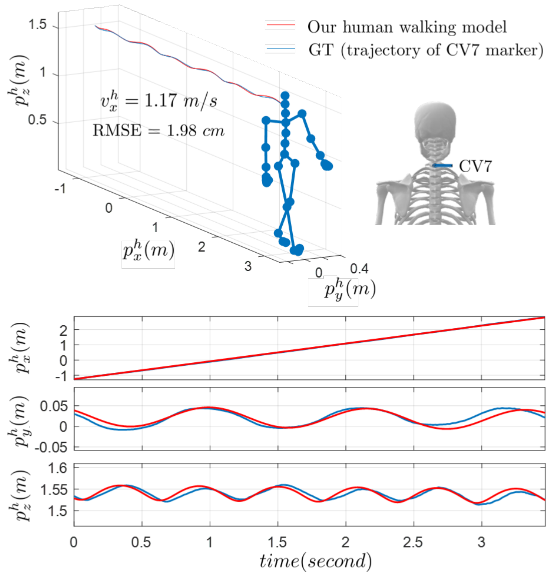

Fig. 12. Evaluation of walking camera model. A representative result of our fitted model (red curve) to a walking person motion (blue curve) at a speed of 1.17 m/s. GT is extracted from the CMU database (data id: 35-28). RMSE between the GT and our fitted model is 1.98 cm.

the ground truth and our fitted model trajectories (i.e., the distance between the corresponding 3D ground truth path and the fitted model trajectory). The RMSE is in the range of 1.98 to 2.60 cm with a mean value of 2.34 cm at different walking speeds.

10 LIMITATIONS AND FUTURE WORK

Our approach takes the drone's physical limitations into account to imitate various FPV shot styles, and we show it is applicable in different scenarios such as imitating walking, running and stepping stairs. However, the drone's physical limitations restrict the feasible space of the optimal solution. For example, drones have a maximum torque velocity, and therefore cannot be used to imitate exactly the same behavior as a system with much faster velocities.

Another example is imitating human jumping with drones. Drone imitation of human jumping requires to (1) completely turn off the drone propellers to imitate the free-fall phase of the jump and then, (2) immediately requires to send drone control commands to turn on the propellers for "landing". However, our drone SDK (Bebop 2) does not allow to completely turn off and immediately turn on the propellers in such a short duration.

Our work is dedicated to human walking imitation. An interesting direction for future work is to extend it to animal motions. For example, our general formulation could be used to imitate the specific dynamics of the head motion of a horse or a dog with drones and see the world from their perspective.

In this paper, we imitated FPV shots, for example acquired by a shoulder rig and a Steadicam. A future direction worthwhile to explore is to mimic other kinds of camera rig equipment, such as a car-mounted camera rigs, e.g., a Russian Arm[6], to capture car chasing scenes in action movies.

Our work imitates a single camera operator motion style. A direction worthwhile to explore is to study multi-person scenario, for example, when there are multiple people in the scene, how to design the drone motion to smoothly switch from FPV of one person to another one. In addition, it would also be interesting to conduct a systematic study with professional cinematographers.

11 CONCLUSION

We presented the first approach to automatically capture subjective FPV shots in the context of drone cinematography. Our key technical contribution is a computational method that enables a drone to imitate the motion pattern and dynamics of a walking camera operator. In addition, our method is interactive, runs in real time and also satisfies high-level user goals such as the user-defined reference camera path, camera velocity, and shot style (e.g., smooth dolly shot or FPV shot). The validity of our approach has been confirmed by both quantitative and qualitative evaluations.

Our method is interactive, which provides video directors the artistic freedom to design their own FPV shot style and tune the amount of camera shakiness based on the online visual feedback.

Finally, we have shown that our approach allows to capture seamless transition videos (such as from FPV to dolly shots), which is impossible in practice using traditional cinematographic equipment for a human camera operator. Overall, we believe that our work, and more generally automated drone cinematography, offer exciting opportunities to capture new shot styles, bring novel expression formats and new ways to design video storytelling.

ACKNOWLEDGMENTS

We are grateful to Seonwook Park for narrating the video. We thank David Lindlbauer and Anna Maria Feit for their valuable comments in designing our user-study. A special thanks goes to all wonderful AIT lab members, especially Christoph Gebhardt, Sammy Christen, Xu Chen and Lukas Vordemann as well as Cecile-Marie Lissens and Christina Welter for their participation in our experiments. We thank Prof. Junyong Noh for his valuable comments and insights. We are grateful to embotech for technical support as well as the BK21 Plus Program. This research was partially supported by the National Research Foundation of Korea (NRF-2017R1C1B5077030) funded by the Korean government (MSIT). This work was also partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-00450, A deep learning based immersive AR content creation platform for generating interactive, context and geometry aware movement from a single image).

[6]http://filmotechnicusa.com/russian-arm-6.html

REFERENCES

3D Robotics. 2015. 3DR Solo. Retrieved September 13, 2016 from http://3drobotics.com/solo

APM. 2016. APM Autopilot Suite. Retrieved September 13, 2019 from http://ardupilot.com

Joshua Bailey, Tiffany Mata, and John A. Mercer. 2017. Is the relationship between stride length, frequency, and velocity influenced by running on a treadmill or overground? *International Journal of Exercise Science* 10, 7 (2017), 1067.

Justin Carpentier, Mehdi Benallegue, and Jean-Paul Laumond. 2017. On the centre of mass motion in human walking. *International Journal of Automation and Computing* 14, 5 (2017), 542–551.

Marc Christie, Patrick Olivier, and Jean-Marie Normand. 2008. Camera Control in Computer Graphics. *Computer Graphics Forum* 27, 8 (2008), 2197–2218.

DJI. 2016. PC Ground Station. Retrieved September 13, 2019 from http://www.dji.com/pc-ground-station

Alexander Domahidi and Juan Jerez. 2017. FORCES Pro: code generation for embedded optimization. Retrieved September 4, 2019 from https://www.embotech.com/FORCES-Pro

Steven M. Drucker and David Zeltzer. 1994. Intelligent Camera Control in a Virtual Environment. In *Proceedings of Graphics Interface (GI)*. 190–199.

Salman Faraji and Auke J Ijspeert. 2017. 3LP: a linear 3D-walking model including torso and swing dynamics. *The International Journal of Robotics Research* 36, 4 (2017), 436–455.

Quentin Galvane, Julien Fleureau, François-Louis Tariolle, and Philippe Guillotel. 2016. Automated Cinematography with Unmanned Aerial Vehicles. In *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing (WICED)*. 23–30.

Quentin Galvane, Christophe Lino, Marc Christie, Julien Fleureau, Fabien Servant, Fran Tariolle, Philippe Guillotel, et al. 2018. Directing cinematographic drones. *ACM Transactions on Graphics (TOG)* 37, 3 (2018), 34.

Christoph Gebhardt, Benjamin Hepp, Tobias Nägeli, Stefan Stevšić, and Otmar Hilliges. 2016. Airways: Optimization-Based Planning of Quadrotor Trajectories According to High-Level User Goals. In *CHI*. 2508–2519.

Christoph Gebhardt and Otmar Hilliges. 2018. WYFIWYG: Investigating Effective User Support in Aerial Videography. *arXiv preprint arXiv:1801.05972* (2018).

Christoph Gebhardt, Stefan Stevšić, and Otmar Hilliges. 2018. Optimizing for aesthetically pleasing qadrotor camera motion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 90.

Robert D. Gregg, Adam K. Tilton, Salvatore Candido, Timothy Bretl, and Mark W. Spong. 2012. Control and planning of 3-D dynamic walking with asymptotically stable gait primitives. *IEEE Transactions on Robotics* 28, 6 (2012), 1415–1423.

S. Javad Hasaneini, C.J.B. Macnab, John E.A. Bertram, and Henry Leung. 2013. The dynamic optimization approach to locomotion dynamics: human-like gaits from a minimally-constrained biped model. *Advanced Robotics* 27, 11 (2013), 845–859.

Jessica Hodgins. 2015. CMU graphics lab motion capture database.

Jerry Holway and Laurie Hayball. 2013. *The Steadicam® Operator's Handbook*. CRC Press.

Niels Joubert, L. E. Jane, Dan B. Goldman, Floraine Berthouzoz, Mike Roberts, James A. Landay, and Pat Hanrahan. 2016. Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *arXiv preprint arXiv:1610.01691* (2016).

Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 238.

Steven Douglas Katz. 1991. *Film directing shot by shot: visualizing from concept to screen*. Gulf Professional Publishing.

Anatole Lécuyer, Jean-Marie Burkhardt, Jean-Marie Henaff, and Stéphane Donikian. 2006. Camera motions improve the sensation of walking in virtual environments. In *IEEE Virtual Reality Conference (VR)*. 11–18.

Tsai-Yen Li and Chung-Chiang Cheng. 2008. Real-Time Camera Planning for Navigation in Virtual Environments. In *International Symposium on Smart Graphics (SG)*. 118–129.

Christophe Lino and Marc Christie. 2012. Efficient Composition for Virtual Camera Control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*. 65–70.

Christophe Lino and Marc Christie. 2015. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 82.

Christophe Lino, Marc Christie, Roberto Ranon, and William Bares. 2011. The Director's Lens: An Intelligent Assistant for Virtual Cinematography. In *ACM International Conference on Multimedia*. 323–332.

Ian R. Manchester and Jack Umenberger. 2014. Real-time planning with primitives for dynamic walking over uneven terrain. In *ICRA*. 4639–4646.

Tobias Nägeli, Javier Alonso-Mora, Alexander Domahidi, Daniela Rus, and Otmar Hilliges. 2017a. Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters* 2, 3 (2017), 1696–1703.

Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017b. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 132.

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source Robot Operating System. In *IEEE ICRA Workshop on Open Source Software*.

Mike Roberts and Pat Hanrahan. 2016. Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 61.

Michael J. Seitz and Gerta Köster. 2012. Natural discretization of pedestrian movement in continuous space. *Physical Review E* 86, 4 (2012), 046108.

Maziar A. Sharbafi and Andre Seyfarth. 2015. FMCH: a new model for human-like postural control in walking. In *IROS*. 5742–5747.

Jianbo Shi and Carlo Tomasi. 1993. *Good features to track*. Technical Report. Cornell University.

VC Technology. 2016. Litchi Tool. Retrieved September 13, 2019 from https://flylitchi.com/

Carlo Tomasi and Takeo Kanade. 1991. *Tracking of point features*. Technical Report. Tech. Rep. CMU-CS-91-132, Carnegie Mellon University.

Yujiang Xiang, Jasbir S. Arora, and Karim Abdel-Malek. 2010. Physics-based modeling and simulation of human walking: a review of optimization-based and other approaches. *Structural and Multidisciplinary Optimization* 42, 1 (2010), 1–23.

Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. 2018. Creating and chaining camera moves for quadrotor videography. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 88.

I-Cheng Yeh, Chao-Hung Lin, Hung-Jen Chien, and Tong-Yee Lee. 2011. Efficient camera path planning algorithm for human motion overview. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 239–250.

Wiebren Zijlstra and A.L. Hof. 1997. Displacement of the pelvis during human walking: experimental data and model predictions. *Gait & posture* 6, 3 (1997), 249–262.

Symbol	Description
$\mathbf{x}^q$, $\mathbf{u}^q$	Quadrotor states and inputs
$\mathbf{p}^q$, $\mathbf{v}^q$, $\mathbf{o}^q$	Quadrotor position, velocity and orientation vector
v_x^q, v_y^q, v_z^q	Quadrotor velocity in x, y, z directions
Φ^q, Θ^q, ψ^q	Quadrotor roll, pitch and yaw
ϕ^q, θ^q	Quadrotor desired roll and pitch
ω_ψ^q	Quadrotor desired angular velocity around body-z
θ_g, ψ_g	Gimbal pitch and yaw
ω_{θ_g}, ω_{ψ_g}	Gimbal pitch and yaw rate
$\boldsymbol{\theta}^h$, $\mathbf{y}^h$	Human walking model states and outputs
$\mathbf{p}^h$, $\mathbf{v}^h$	Human walking position and velocity vector
p_x^h, p_y^h, p_z^h	Human walking position in x, y, z directions
v_x^h, v_y^h, v_z^h	Human walking velocity in x, y, z directions
ψ^h, ω^h	Human walking yaw and angular yaw speed
l_s^h, f_s^h	Human walking step length and step frequency
θ_y^h, ω_y^h	Human walking lateral phase and angular frequency
θ_z^h, ω_z^h	Human walking vertical phase and angular frequency
θ_ψ^h, ω_ψ^h	Human walking yaw phase and angular frequency
$\mathbf{x}^m$, $\mathbf{u}^m$	Imitation model states and inputs
$\mathbf{p}^m$, $\mathbf{v}^m$, $\mathbf{o}^m$	Imitation model position, velocity and orientation
θ^s	Smooth path progress parameter
$\boldsymbol{\Theta}^s$, a^s	Progress state and input
$\mathbf{A}^s$, $\mathbf{B}^s$	System matrices of progress
$\mathbf{r}(\theta^s)$	Reference spline ($\mathbb{R}^3$)
$\mathbf{a}^t(\theta^s)$	Normalized vector tangent to reference spline
$\mathbf{a}^n(\theta^s)$	Normalized vector orthogonal to reference spline
$\mathbf{a}^z(\theta^s)$	Normalized vector in z direction
c^l, c^c	Lag and contour cost
N	Prediction horizon length
T_s	Sampling time

Table 1. Summary of notation used in the body of the paper.

A NOTATION

For completeness and reproducibility of our method, Table 1 provides a summary of the notations used in the paper.

B PATH FOLLOWING

Path Following: The desired user-defined trajectory $\mathbf{r} \in \mathbb{R}^3$ is parameterized by $\theta^s \in [0, L]$, where L is the path length. We continuously optimize the drone path following cost to minimize the distance between the desired path and the drone. However, we cannot rely on a time-stamped reference path as is commonly done in MPC formulations [Nägeli et al. 2017a], since we want to give the user freedom in deciding the walking camera operator model parameters (e.g., the forward walking velocity of a camera operator that the drone should follow on the desired path). Similar to [Gebhardt et al. 2018; Nägeli et al. 2017b], we decompose the drone distance

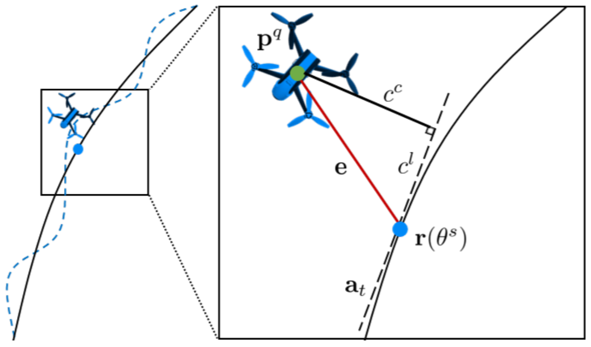

Fig. 13. Illustration of lag and contouring error decomposition.

to the closest point on the path into a contouring and lag error. In addition, we optimize the progress parameter θ^s so that $\mathbf{r}(\theta^s)$ returns a combination between the closest point and ensuring the drone progresses on the path during the imitation. We define $\mathbf{e}$ as the distance between the drone position $\mathbf{p}^q$ and a point $\mathbf{r}(\theta^s)$ on the desired path, and $\mathbf{a}_t(\theta^s)$ as the normalized tangent vector to the path at that point

$$\begin{aligned} \mathbf{e} &= \mathbf{r}(\theta^s) - \mathbf{p}^q, \\ \mathbf{a}_t(\theta^s) &= \frac{\mathbf{r}'(\theta^s)}{||\mathbf{r}'(\theta^s)||}, \end{aligned} \tag{11}$$

with $\mathbf{r}'(\theta^s) = \frac{\partial \mathbf{r}(\theta^s)}{\partial \theta^s}$. The vector $\mathbf{e}$ can now be decomposed into a lag error and a contour error (see Figure 13). The lag error is computed as the projection of $\mathbf{e}$ on the tangent of $\mathbf{r}(\theta^s)$ while the contour error is the component of $\mathbf{e}$ orthogonal to the normal:

$$\begin{aligned} c^l(\theta^s, \mathbf{p}^q) &= ||\langle \mathbf{e}, \mathbf{a}_t \rangle||^2, \\ c^c(\theta^s, \mathbf{p}^q) &= ||\mathbf{e} - \langle \mathbf{e}, \mathbf{a}_t \rangle \mathbf{a}_t||^2. \end{aligned} \tag{12}$$

Separating lag from contouring error allows us to differentiate how we penalize a deviation outside the path (c^c), from encouraging the drone to progress forward (c^l). The cost term $c^{\mathbf{a}_t}$ in Eq. (6) for drone imitation of the forward velocity of a walking camera operator encourages progress on the desired path.

Progress Along Path: We parameterize the user-defined camera path $\mathbf{r} \in \mathbb{R}^3$ by the path parameter θ^s from 0 to L. The path function $\mathbf{r}(\theta^s) : \mathbb{R} \to \mathbb{R}^3$ defines the desired 3D camera position w.r.t. the path parameter θ^s (e.g., $\mathbf{r}(L)$ is the last 3D point on the user-defined path $\mathbf{r}$). Given an initial path parameter at instant k (i.e., θ_k^s), the aim is to traverse forwards along the path from $\mathbf{r}(\theta_k^s)$ to $\mathbf{r}(\theta_{k+1}^s)$. We define the following linear discrete dynamics for θ^s:

$$\begin{aligned} \boldsymbol{\Theta}_{k+1}^s &= \mathbf{A}^s \boldsymbol{\Theta}_k^s + \mathbf{B}^s a_k^s, \\ \mathbf{A}^s &= \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B}^s = \begin{bmatrix} \frac{1}{2} T_s^2 \\ T_s \end{bmatrix}, \\ \mathbf{0} &\leq \boldsymbol{\Theta}_k^s \leq \boldsymbol{\Theta}_{max}^s, \\ a_{min}^s &\leq a_k^s \leq a_{max}^s, \end{aligned} \tag{13}$$

where $\boldsymbol{\Theta}^s = [\theta^s, \dot{\theta}^s]^T$ are the path progress states, T_s the sampling time, and $a^s = \ddot{\theta}^s$ the virtual input which determines the path evolution θ_{k+1}^s, and consequently $\mathbf{r}(\theta_{k+1}^s)$. The constraint $\dot{\theta}_k^s \geq 0$

enforces forward motion along the path while $0 \leq \theta^s_k \leq L$ prevents exceeding the user-defined path boundaries. Since we consider the path parameter's acceleration a^s as the input, it gives us one more degree of freedom to force a constraint on the path progress acceleration (i.e., $a^s_{min} \leq a^s_k \leq a^s_{max}$) and avoid sudden changes in the path progress.

Input Constraint: To avoid excessive use of the control inputs and limit progress acceleration on the desired spline, we define a cost term as:

$$c^{inp}(a^s, \mathbf{u}^q) = w_{a^s}||a^s||^2 + \mathbf{u}^{q^T}\mathbf{R}\mathbf{u}^q, \tag{14}$$

where w_{a_s} is a positive scalar weight parameter avoiding excessive acceleration on the progress of the desired smooth path, and $\mathbf{R} \in \mathbb{S}^2_+$ is a positive definite penalty matrix restricting control inputs.

C DRONE IMITATION OF A GENERAL DYNAMICAL SYSTEM

Let $\mathbf{p}^m \in \mathbb{R}^3$ and $\mathbf{o}^m \in \mathbb{R}^3$ denote the position and orientation of a non-linear model for imitation. This model is defined in its body-frame and can be written in the form of a differentiable function or a general memoryless non-linear model whose output in each instant just depends on its inputs at that moment. Let $\mathbf{v}^m = [v^m_x, v^m_y, v^m_z] \in \mathbb{R}^3$ be the velocity of the imitation model. Let us define the imitation model in the form of a discrete differentiable function $I : \mathbb{R}^{n_x \times n_u} \longrightarrow \mathbb{R}^{n_x}$ as

$$\mathbf{x}^m_{k+1} = I(\mathbf{x}^m_k, \mathbf{u}^m_k), \tag{15}$$

where n_x is the dimension of the desired imitation model states $\mathbf{x}^m \in \mathbb{R}^{n_x}$, and n_u is the dimension of its inputs $\mathbf{u}^m \in \mathbb{R}^{n_u}$. The imitation model input depends on each specific model defined for imitation, and its transitional and rotational states $[\mathbf{p}^m, \mathbf{o}^m, \mathbf{v}^m]^T$ are a subset of its states and inputs $\{\mathbf{x}^m, \mathbf{u}^m\}$. Our goal is to imitate the dynamics of this system with a drone while the drone follows a user-defined path. To this end, similar to the imitation of the human walking model (see Section 4.3), we use this imitation model Eq. (15) to predict its velocity $\mathbf{v}^m$ and orientation $\mathbf{o}^m$ in a prediction horizon, and then we use the same cost term as the drone imitation of the human walking model to imitate this dynamical system on a desired path. In our imitation cost term Eq. (6), we just need to change the human walking velocity v^h_x, v^h_y and v^h_z to the imitation model velocities v^m_x, v^m_y and v^m_z. To follow the rotational behavior of any dynamical model, we define the orientation cost term as

$$c^o(\mathbf{o}^q, \mathbf{o}^m) = ||\mathbf{o}^q - \mathbf{o}^m||^2. \tag{16}$$

Then, similar to following the human walking dynamical system, we construct our optimization problem (Eq. (10)).

D OPTIMIZATION WEIGHTS

The values for the weights of the objective function at Eq. (10) that we used in the user study and experiments are listed in Table 2. We empirically derived weights of our optimization problem based on both the visual feedback to imitate FPV shot style and the accuracy of our method to follow a desired path. $w_\mathbf{a}$ defines the penalizing rate for the imitation of the forward, lateral and vertical walking velocities. We set the value of this weight to 100. We use a single weight $w_\mathbf{a}$ to equally penalize violating the walking velocity constraint in all directions (lateral, vertical and tangent to the path). w_{ω_ψ} and w_ψ define the penalizing rate for imitating the walking yaw speed and following the desired camera yaw angle, respectively. We set a higher value to w_{ω_ψ} than w_ψ because our main focus is imitating the FPV style ($w_{\omega_\psi} = 600$), and the camera should smoothly rotate to the desired yaw angle ($w_\psi = 150$) to capture natural looking FPV shots. We tune all other weights similar to [Gebhardt et al. 2018; Nägeli et al. 2017b]. For example, we choose a high penalty on lag error ($w_l = 1000$) to improve the approximation quality of the contour error [Nägeli et al. 2017b]. For penalizing the contouring error, we allow some flexibility ($w_c = 300$) in order to account for the imitation of walking dynamics since it might be desirable to deviate locally from the desired path in favor of the imitation costs, i.e., the drone should locally move up-down and left-right around the desired path.

Weight	**Description**	**Value**
$w_\mathbf{a}$	velocity imitation	100
w_{ω_ψ}	angular velocity imitation	600
w_ψ	camera orientation	150
w_l	lag error	1000
w_c	contour error	300
w_{a^s}	restricting progress acceleration	0.1
$\mathbf{R}$	restricting control inputs	$diag$(0,10,10,0,0,0)
w_N	final stage weight	10

Table 2. Values of the weights used in Eq. (10).