2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control*

Gauthier Rousseau[1,2], Cristina Stoica Maniu[1], Sihem Tebbani[1], Mathieu Babel[2] and Nicolas Martin[2]

Abstract— **This paper proposes a receding waypoint horizon strategy generating a piecewise polynomial trajectory with minimum jerk and predictive tracking of camera references for quadrotors, in the context of autonomous aerial single-sequence shots in a static environment. In order to deal with the limited on-board computation resources, the camera control is performed with an undersampled model predictive controller generating a set-point trajectory and a feedforward control signal, both used by a larger frequency controller. The performance of the overall strategy is illustrated with a real flight, on a Parrot Bebop 2 drone.**

I. INTRODUCTION

With the spreading use of multirotor UAVs (Unmaned Aerial Vehicles) for aerial video making, research on trajectory generation and tracking has come closer to the concept of the flying camera, a high end solution for which the piloting aspect of the drone is completely transparent. Such a system would allow cinematographers to focus on artistic considerations and communicate high level instructions to their drones, which would autonomously take care of the trajectory generation and the tracking aspects.

The literature concerning the different areas of automatic control for multirotors (path planning, control etc.) is dense, as this has been a popular domain of research for the past few years. Different approaches have been suggested for the path planning problem. In [1] and [2] piecewise polynomial trajectories with minimum snap have been proposed, and successfully used in an obstructed environment in [3]. This strategy ensures constraints validation, such as flight corridor constraints, on a finite number of points of the trajectory that must be large enough in order to satisfy these constraints on the overall trajectory. In [4], an algorithm to generate feasible minimum jerk trajectories is described and applied to a ball catching quadrotor. B-Splines based methods have also proved their efficiency in [5], [6] and [7], and allow to enforce constraints on the overall trajectory rather than on a finite number of points, at the cost of more conservative trajectories. Flatness-based methods have also been studied for video making with multirotors with flight tests in [8]. Solutions with high end goals have been presented in [9] and [10]. Such a solution with a flatness-based method, adapted for filming static environments can be found in [11]. The method from [12] allows a drone to follow a trajectory intuitively drawn by hand with considerations on the jerk of the trajectory in order to ensure its smoothness. The authors of [13] and [14] investigate solutions for filming dynamic environment with a trajectory generation method which directly takes into account the video produced by one or several flying cameras, avoiding to have a drone in the field of view of another one.

Fig. 1. Parrot Bebop 2 drone

In this paper, a bilevel optimization approach is used to generate a visually satisfying trajectory from a given flight plan, specified by waypoints and flight corridors, speed references and camera behaviours between the considered waypoints. This optimal trajectory minimizes the jerk of the drone subject to feasibility constraints. Camera references are tracked by a Model Predictive Control (MPC) law that anticipates the future set-points in order to reduce framing errors while keeping the video smooth. A receding horizon strategy is used to reduce the computation load required by the proposed trajectory generation algorithm. One contribution of this paper is related to the used of a smooth, undersampled predictive control strategy for tracking camera references. A second contribution consists in the validation of the overall methodology with an outdoor flight, performed by a Parrot Bebop 2 drone (see Fig. 1).

Notation. Denote by $A^\top$ the transpose of a matrix A. For $i, j \in \mathbb{N}$, with $i < j$, the set $[\![i, j]\!]$ contains the consecutive integers $\{i, i+1, \ldots, j-1, j\}$. The 2-norm of $\mathbf{x} \in \mathbb{R}^n$ is denoted by $\|\mathbf{x}\|_2$, while $\|\mathbf{x}\|_{\mathbf{Q}}^2 = \mathbf{x}^\top \mathbf{Q}\, \mathbf{x}$, with $\mathbf{Q} \in \mathbb{R}^{n \times n}$.

II. FLIGHT PLAN SPECIFICATIONS

The considered flight plan consists in a series of $N+1$ consecutive 3D waypoints $\{W_0, W_1, \ldots, W_N\}$, i.e. a starting position W_0 followed by N waypoints to join, and flight corridors to be respected between each two consecutive waypoints. These corridors are considered as straight cylinders joining each pair of waypoints. Different types of waypoints are further considered

- *Stop waypoint:* reaching it with null speed and null acceleration. The first and last waypoints of the flight plan are usually stop waypoints, but stop points can

*This work was supported by Parrot Drones.

[1]Laboratoire des Signaux et Systèmes, CentraleSupélec-CNRS-Univ. Paris-Sud, Université Paris Saclay, Gif-sur-Yvette, France (e-mail: {gauthier.rousseau, cristina.maniu, sihem.tebbani}@centralesupelec.fr)

[2]Flight Control Department, Parrot Drones, Paris, France, (e-mail: {gauthier.rousseau, mathieu.babel, nicolas.martin}@parrot.com)

also be used during the mission for taking pictures, for standing at a given point in order to record a panorama.
- *Lock waypoint:* passing on the waypoint. This kind of waypoint can be used to impose precisely the position of the drone when passing through a window or a door for instance, or for a specific camera shot.
- *Autonext waypoint:* passing in a neighborhood of specified radius around the waypoint. The drone performs wider turns around the waypoint, while remaining in the flight corridors, leading to more natural trajectories.

A reference velocity v_i ($i \in [\![1, N]\!]$), a flight corridor radius r_i and a camera behaviour (i.e. type of camera heading and camera elevation reference) are specified between each pair of waypoints. The most common types of camera angle references (heading or elevation) are

- *Constant reference.* The reference is fixed between two waypoints. This is typically used for travelings.
- *Ramp reference.* The reference consists in a ramp with a constant slope. This kind of reference is mostly used for panoramas.
- *Tangent reference.* The direction of recording is given by the speed vector of the drone relatively to a ground fixed frame. This results in subjective, point-of-view like camera shots.
- *Point Of Interest (POI) reference.* The camera points toward a given target. This is a very popular type of camera behaviour among drone users.
- *Smooth reference.* This kind of reference consists in a smooth transition between the camera behaviours on the previous and the next pieces of trajectory. In this work, this kind of transitions are realized using a third order polynomial reference, with angle and rotation speed continuity constraints.

Notice that the camera heading and elevation can combine two different reference types on a same piece of trajectory, for instance a constant elevation and a ramp heading.

The drone used to perform the flight plan is a Parrot Bebop 2, which include a fixed, digitally stabilized camera. The camera can be considered as a virtual gimbal with limited elevation and heading, relatively to the drone.

III. TRAJECTORY GENERATION

In [1], piecewise polynomial trajectories with minimum snap are used for completing a flight plan under corridor constraints. The choice to minimize the snap is justified by its strong link to the control signals of the drone. The validation time of each waypoint must be specified and is chosen depending on the context, as imposed values or solution of an optimization problem. In the present paper, the quality of the video is the main factor to take into account. Thus, based on the strategy in [1], the jerk is minimized instead of the snap for several reasons. Firstly, the jerk quantifies the amplitude of the jolts of a mechanical system and should be as low as possible in order to ensure a good quality of the video. Secondly, the jerk of the drone is strongly linked to its rotation speed [4]. As the Parrot Bebop 2 drone used in this work has a digitally stabilized camera, fixed relatively to the drone, both the drone and the camera have the same rotation speed. This means that, though the video is stabilized, motion blur can still appear on the video when increasing too much the rotation speed of the drone. The way of choosing the times of flight is also adapted to the problem.

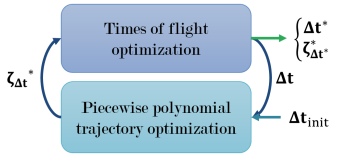

Fig. 2. Bilevel optimization strategy for generating the trajectory

The strategy used in this work for generating a feasible and visually satisfying trajectory can thus be resumed as follows. Firstly, a pre-processing of the flight plan is performed, to check that the velocity references are pertinent. Indeed, the reference velocity on each piece of trajectory is clamped so that the vertical component of the speed vector lies within admissible bounds and so that the lateral velocity does not exceed a given limitation. For $i \in [\![1, N]\!]$, we denote by $\widetilde{v}_i$ the clamped reference velocity for the i-th piece of trajectory. Secondly, a bilevel optimization method generates a visually satisfying feasible trajectory. This optimization is illustrated on Fig. 2: for a vector of times of flight between each pair of consecutive waypoints, $\mathbf{\Delta t}$, a trajectory $\boldsymbol{\zeta}^*_{\mathbf{\Delta t}}$ is generated by solving an optimization problem, as described in Section III-A. This vector of times of flight is then modified and the procedure is repeated until a satisfying trajectory is generated. A criterion for deciding which vector of times of flight should be chosen is described in Section III-B.

In Section III-C, a receding horizon strategy is proposed for dealing with large flight plans, containing numerous waypoints.

A. Minimum jerk trajectory

For a given vector $\mathbf{\Delta t} = \begin{pmatrix} \Delta t_1 & \Delta t_2 & \dots & \Delta t_N \end{pmatrix}^\top$ of times of flight between the waypoints, a minimum jerk piecewise polynomial trajectory $\boldsymbol{\zeta}^*_{\mathbf{\Delta t}}$ is generated as a solution of an optimization problem. The validation of *autonext* waypoints is ensured by an inequality constraint on the distance between the drone and the waypoint at the corresponding validation time. The position is constrained on the waypoints for *lock* and *stop* waypoints, with additional constraints of null speed and null acceleration for the *stop* waypoints. The continuity of the position, speed and acceleration is imposed on the connections between the different pieces of trajectory when not already ensured by the waypoint validation constraints. The possible discontinuities of jerk and snap are absorbed and smoothed by the controller (which is beyond the scope of this paper) in charge of trajectory tracking, which only requires $\mathcal{C}^2$ trajectories. Finally, flight corridor validation is ensured by inequality constraints on a finite number of checkpoints, based on [1]. The problem can be formulated as a Quadratic Programming (QP) problem, by optimizing, for instance, the vector of polynomial coefficients of $\boldsymbol{\zeta}_{\mathbf{\Delta t}}$ or the value of its derivatives on each waypoint, as described in [2]. Indeed, denoting by $\mathbf{x}$ the vector to optimize

(coefficients, derivatives etc.) and by $t_N = \sum_{k=1}^{N} \Delta t_k$, it is possible to write [1], [2]

$$\int_0^{t_N} \left\| \boldsymbol{\zeta}^{(3)}_{\boldsymbol{\Delta t}}(t) \right\|_2^2 \mathrm{d}t = \mathbf{x}^\top \mathbf{H} \mathbf{x}$$

with $\boldsymbol{\zeta}^{(3)}_{\boldsymbol{\Delta t}}$ the third derivative of $\boldsymbol{\zeta}_{\boldsymbol{\Delta t}}$ and $\mathbf{H}$ a positive semidefinite square matrix. Thus, the optimization problem has the following form [1], [2]

$$\mathbf{x}^* = \arg\min \mathbf{x}^\top \mathbf{H} \mathbf{x} \tag{1}$$

$$\text{s.t.} \begin{cases} \mathbf{A}_{\text{Stop}} \cdot \mathbf{x} = \mathbf{b}_{\text{Stop}} \\ \mathbf{A}_{\text{Lock}} \cdot \mathbf{x} = \mathbf{b}_{\text{Lock}} \\ \mathbf{A}_{\text{Autonext}} \cdot \mathbf{x} \leqslant \mathbf{b}_{\text{Autonext}} \\ \mathbf{A}_{\text{Continuity}} \cdot \mathbf{x} = \mathbf{b}_{\text{Continuity}} \\ \mathbf{A}_{\text{Corridor}} \cdot \mathbf{x} \leqslant \mathbf{b}_{\text{Corridor}} \end{cases}$$

Its solution gives the optimal trajectory $\boldsymbol{\zeta}^*_{\boldsymbol{\Delta t}}$ for the vector of times of flights $\boldsymbol{\Delta t}$.

B. Times of flight

The vector of times of flight $\boldsymbol{\Delta t}$ is chosen to minimize the time for completing the entire flight plan, such that the velocity on each piece of trajectory should not exceed the reference velocity and such that the acceleration and jerk norms should not exceed the maximum admissible values. Finally, camera heading and elevation excursions between each two consecutive waypoints are computed. Given a maximum camera rotation speed, these excursions give a lower bound on the times of flight on each piece of trajectory. For instance, if two waypoints are very close but a 120° heading panorama has to be performed between them at a maximum rotation speed of 5°/s, the time of flight between these two waypoints cannot be less than $24s$. The times of flight (considered positive) are hence reduced until at least one constraint is active solving

$$\boldsymbol{\Delta t}^* = \arg\min \sum_{i=1}^{N} \Delta t_i \tag{2}$$

$$\text{s.t.} \begin{cases} \Delta t_i \geqslant \Delta t_{i_{\min}}, \quad \forall i \in [\![1, N]\!] \\ \left\| \boldsymbol{\zeta}^*_{\boldsymbol{\Delta t}}(t) \right\|_2 \leqslant \widetilde{v}_i, \quad \forall i \in [\![1, N]\!], \quad \forall t \in [t_{i-1}, t_i] \\ \left\| \ddot{\boldsymbol{\zeta}}^*_{\boldsymbol{\Delta t}}(t) + \lambda \, \dot{\boldsymbol{\zeta}}^*_{\boldsymbol{\Delta t}}(t) \right\|_2 \leqslant a_{\max}, \quad \forall t \in [0, t_f] \\ \left\| \boldsymbol{\zeta}^{*(3)}_{\boldsymbol{\Delta t}}(t) \right\|_2 \leqslant j_{\max}, \quad \forall t \in [0, t_f] \end{cases}$$

with $\boldsymbol{\zeta}^*_{\boldsymbol{\Delta t}}$ the minimum jerk trajectory obtained by solving the problem (1), $a_{\max}$ the maximum admissible acceleration for a zero velocity, λ a friction coefficient, $j_{\max}$ the maximum admissible jerk for ensuring a good quality of the video as previously described. The constraint on the acceleration is derived from a simplified, first order friction model that translates the loss of acceleration capability when the drone velocity increases. When an estimation of the wind speed vector relatively to the ground $\mathbf{v}_{\text{wind}}$ is available, this constraint can be replaced by

$$\left\| \ddot{\boldsymbol{\zeta}}^*_{\boldsymbol{\Delta t}}(t) + \lambda \left(\dot{\boldsymbol{\zeta}}^*_{\boldsymbol{\Delta t}}(t) - \mathbf{v}_{\text{wind}} \right) \right\|_2 \leqslant a_{\max}, \quad \forall t \in [0, t_f]$$

in order to take into account the airspeed rather than the ground speed of the drone. However it implies that the wind must be low enough for a feasible solution to exist and may not be pertinent if the wind varies too much during the mission. This constraint is also a way to limit the drone angle while tracking the trajectory. More than improving the feasibility of the trajectory, it can also be justified by limitations on the camera orientation, since the tilt of the camera relatively to the drone is often limited. Situations where the camera cannot reach its reference because the drone angle is too high should be avoided.

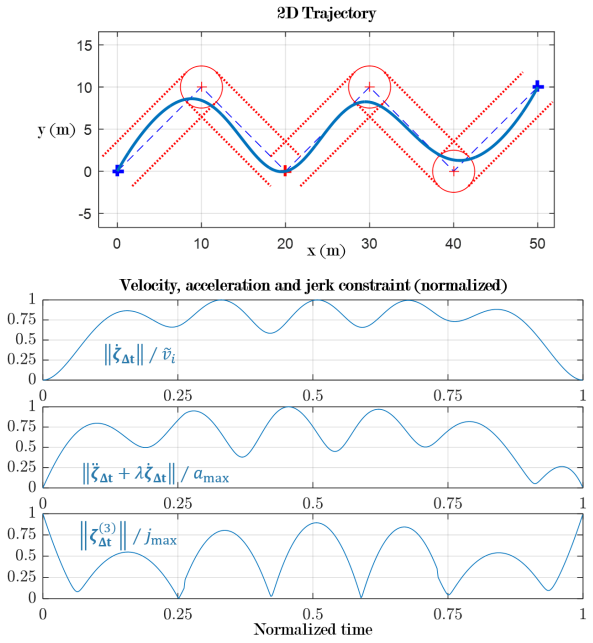

Fig. 3. Example of minimum jerk trajectory

Example 1. *An example of optimal trajectory is presented on Fig. 3. The flight plan contains* 6 *waypoints. It starts and ends on stop waypoints. The third waypoint is of type lock, while the remaining ones are autonext waypoints. Velocity, acceleration and jerk constraints are the limiting factor in decreasing the times of flight between the waypoints.*

C. Receding waypoints horizon

In order to reduce the computation load, especially for flight plans containing a large number of waypoints, this paper considers a receding waypoint horizon methodology. In this case, if the flight plan contains more waypoints than a given limit, the trajectory is not computed over the entire flight plan at once but in several steps, over truncated parts of the overall flight plan.

For a horizon N_H and a flight plan containing at least $N_H + 2$ waypoints, the trajectory is first computed between the first and the N_H following waypoints (i.e. the waypoints $W_0, W_1, \ldots, W_{N_H}$), using the strategy presented above. This results in a piecewise polynomial trajectory $\widetilde{\boldsymbol{\zeta}}_1$ containing N_H pieces. The first piece of this trajectory corresponds to the trajectory joining W_0 to W_1. As the trajectory does not necessarily pass on the waypoint (for *autonext* type), we call P_1 the end point of this first piece of trajectory (see Fig. 4). P_1 is reached with a speed vector $\mathbf{v}_{P_1}$ and an acceleration vector $\mathbf{a}_{P_1}$. Only this first piece of trajectory is kept and will constitute the first piece of the overall, final

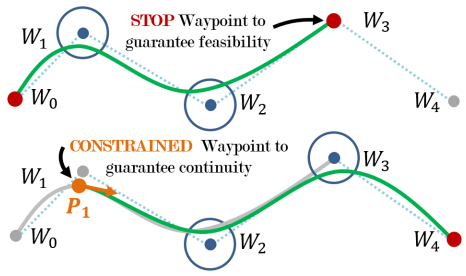

Fig. 4. Receding horizon strategy used for trajectory generation

trajectory $\boldsymbol{\zeta}$. According to the receding horizon strategy, the horizon is then moved by one waypoint and a new trajectory $\widetilde{\boldsymbol{\zeta}}_2$ is computed, but with a starting waypoint replaced by P_1 instead of W_1. This starting waypoint is of special type *constrained*, meaning that the position, speed and acceleration are imposed, in this case equal to P_1, $\mathbf{v}_{P_1}$ and $\mathbf{a}_{P_1}$, respectively. Again, only the first piece of this new trajectory $\widetilde{\boldsymbol{\zeta}}_2$ is kept and constitutes the second piece of the final trajectory $\boldsymbol{\zeta}$. The continuity at the connection between the two pieces is ensured by the constraints on the position and its derivatives on P_1. The process is repeated until the last waypoint of the flight plan is reached. In order to ensure the feasibility of the problem, the last waypoint of the horizon is always imposed to be of type *stop*, even though it was not in the initial flight plan.

Example 2. *A flight plan containing 5 waypoints* $\{W_0, \dots, W_4\}$ *and a horizon* $N_H = 3$ *is illustrated in Fig. 4. The trajectory is then computed in 2 steps: between the waypoints* W_0 *and* W_3 *first, and between* P_1 *and* W_4 *next.*

Step 1: a trajectory is generated between W_0 *and* W_3*, with* W_3 *replaced by a stop waypoint at the same position.*

Step 2: a trajectory is generated between the waypoints W_1 *and* W_4*, with the waypoint* W_1 *replaced by a constrained waypoint at the position* P_1*, ensuring the continuity with the trajectory computed at Step 1. Since the trajectory reaches the final waypoint* W_4*, there is no need to repeat the process.*

For online computation, the trajectory is updated each time a waypoint is validated. A constraint on the time of flight on the first piece of trajectory is added in the optimization problem (2), so that it is greater than the average trajectory computation time plus a security margin. In the event that a waypoint would still be validated before finding the optimal solution of the problem (2), the optimization process is interrupted and a feasible (yet suboptimal) trajectory is sent to the drone.

The loss of optimality of the final trajectory induced by this strategy is illustrated on Fig. 5, where the same flight plan as the one on Fig. 3 is processed with different horizons N_H. This loss is negligible for a sufficiently large horizon.

Notice that the systematic replacement of the last waypoint by a *stop* waypoint can reduce the speed of the trajectory for a series of close waypoints though, especially if they are aligned.

The strategy proposed in this section allows us to generate

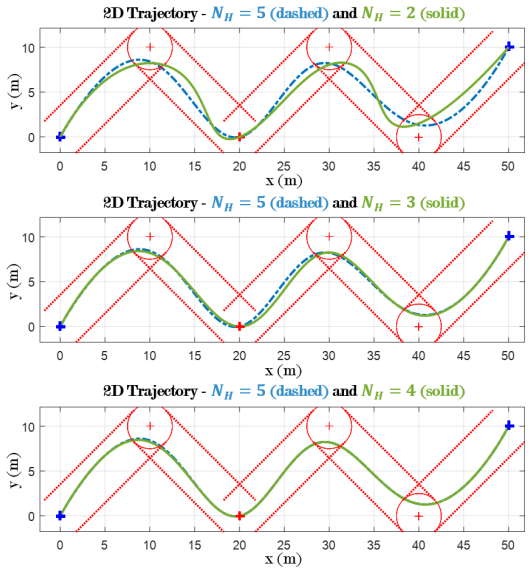

Fig. 5. Optimality loss induced by the receding horizon strategy when decreasing N_H

a smooth and feasible 3D trajectory for completing the flight plan. Camera references along this trajectory are presented in the next section.

IV. CAMERA REFERENCES TRACKING

As described in Section II, different types of camera behaviours can be specified over each piece of trajectory. A way to smoothly track the considered camera references while keeping the framing error low is further proposed.

A. Architecture

The drone used for this work is a Parrot Bebop 2, equipped with a fixed, digitally stabilized front camera, which acts as a virtual gimbal. This virtual gimbal can roll without restrictions but its heading and elevation relatively to the drone are limited into a cone. In order to keep the allowed camera elevation excursion as high as possible, the drone heading relatively to the ground is imposed to be the same as the one of the camera (still relatively to the ground). This way, without perturbations on the yaw axis, the virtual gimbal only has to roll and pitch during the flight.

This implies that the yaw axis behaviour of the attitude loop of the drone must meet some requirements in order to ensure a good video quality. These performance specifications are typically, in order of priority: a minimum phase behaviour, a smooth and slow time response and the absence of oscillations, undershoots and/or overshoots. The tracking errors result in bad framing of the video and thus should also be limited, which means that the disturbance rejection dynamics should be stiff (contrary to the reference tracking).

To achieve this, the Nominal Model Following Control (NMFC) architecture proposed in [15] is used in this work. This architecture consists in a 2-stage controller as illustrated Fig. 6. Its robustness regarding disturbances and its performances in decoupling reference tracking and disturbances rejection dynamics are discussed in [16].

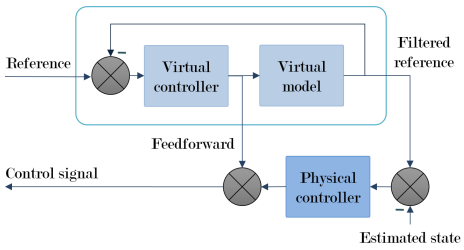

Fig. 6. Architecture of the heading controller

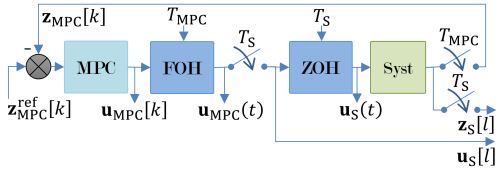

Fig. 7. Block diagram of the heading virtual controller

Heading references can vary significantly from a piece of trajectory to another, depending on the chosen camera behaviour. As a consequence, the heading reference over the entire mission can include steps, ramps or smooth parts. Due to its anticipating action, a model predictive controller is used as virtual controller, in order to efficiently track this reference. Furthermore, this type of controller can explicitly take constraints into account, e.g. in order to limit the rotation speed, acceleration or jerk.

B. Virtual model

The virtual heading dynamics is described by a continuous-time Linear Time Invariant (LTI) system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{z}(t) &= \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \end{aligned} \tag{3}$$

with $\mathbf{x}$ the state vector, $\mathbf{u}$ the control vector, $\mathbf{z}$ the output vector and the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ of appropriate dimensions. In the following, we choose

$$\mathbf{u} = u_\psi \text{ and } \mathbf{z} = \begin{pmatrix} \psi & r & \dot{r} \end{pmatrix}^\top$$

where u_ψ is the control signal on the yaw axis, ψ denotes the drone heading, r and $\dot{r}$ denote its rotation speed and rotation acceleration around the yaw axis, respectively. Though the real drone is not a linear system, it is well described in the literature [17] or [18] with decoupled roll, pitch, yaw and vertical acceleration linear models at low angles and rotation speed, corresponding to a common context for video making.

C. MPC controller

The virtual model (3) is discretized with a sampling period T_{MPC}. In the sequel, this equivalent discrete-time model is

$$\begin{aligned} \dot{\mathbf{x}}_{\mathrm{MPC}}[k+1] &= \mathbf{F}_{\mathrm{MPC}}\,\mathbf{x}_{\mathrm{MPC}}[k] + \mathbf{G}_{\mathrm{MPC}}\,\mathbf{u}_{\mathrm{MPC}}[k] \\ \mathbf{z}_{\mathrm{MPC}}[k] &= \mathbf{C}_{\mathrm{MPC}}\,\mathbf{x}_{\mathrm{MPC}}[k] + \mathbf{D}_{\mathrm{MPC}}\,\mathbf{u}_{\mathrm{MPC}}[k] \end{aligned} \tag{4}$$

This discretized model (4) is controlled by a model predictive control law. The controlled variables are the heading and its two first derivatives, i.e. the rotation speed and acceleration on the yaw axis. For a discrete signal s, we denote by $\hat{s}[k+i|k]$ the prediction of the value of s at instant $k+i$, computed using its value at instant k. The MPC controller generates at each step a control signal minimizing the following classical cost function (as described in [19] and [20])

$$\begin{aligned} J[k] = &\sum_{i=1}^{H_p} \left\| \mathbf{z}^{\mathrm{ref}}_{\mathrm{MPC}}[k+i] - \hat{\mathbf{z}}_{\mathrm{MPC}}[k+i-1|k] \right\|^2_{\mathbf{Q}} \\ &+ \sum_{i=0}^{H_u-1} \left\| \mathbf{u}_{\mathrm{MPC}}[k+i] - \mathbf{u}_{\mathrm{MPC}}[k+i-1] \right\|^2_{\mathbf{R}} \end{aligned} \tag{5}$$

where H_p and H_u define the prediction and control horizons, respectively, and $\hat{\mathbf{z}}_{\mathrm{MPC}}$ denotes the prediction of $\mathbf{z}_{\mathrm{MPC}}$. The weighting terms are the diagonal matrix $\mathbf{Q} = diag(\mu_\psi, \mu_r, \mu_{\dot{r}})$ and $\mathbf{R} = \mu_{u_\psi}$, with μ_ψ, μ_r, $\mu_{\dot{r}}$ adjustable weights on each controlled variable and μ_{u_ψ} a weight on the control signal variations. We can thus control the heading angle and speed tracking errors and reduce the acceleration for a smooth response. The jerk is also of great importance and should be reduced to prevent jolts in the video and is implicitly included in the cost function (5), since the variation of the control signals are linked to the jerk on the yaw axis.

D. MPC undersampling

To be pertinent, an MPC strategy must be able to predict the set-point and the system behaviour over a time horizon consistent with the desired time response of the closed-loop. In the case of the camera angle control, slow and smooth responses are desired, leading to a prediction horizon around one second or more. This is an issue as a low sampling period is required to efficiently control and reject disturbances. Typical orders of magnitude for this sampling period lie from 1ms to 10ms, which results in large prediction horizons of hundreds to thousands of steps (such as in [21]). The computation resources of the drone being restricted, a solution is to undersample the MPC to a lower frequency in order to get a more reasonable prediction horizon. The control signals sent by the MPC are then interpolated at higher frequency and sent as a feedforward to the real drone. Early simulations showed that undersampling the MPC to a reasonable sampling frequency prevented the use of a zero-order hold for this task, as the discontinuities of the control signal would produced small jolts in the video and that even smooth camera angles inputs from the user would still end looking like a sequence of steps.

In order to prevent this jerky behaviour, the MPC generates piecewise affine control signals at low frequency which can then be interpolated at higher frequency. To achieve this, two virtual models are used

- An oversampled virtual model, discretized at the drone sampling period T_{S}
- An undersampled virtual model, discretized at the MPC sampling period $T_{\mathrm{MPC}} = N\,T_{\mathrm{S}}$, with $N \in \mathbb{N}$.

Figure 7 illustrates the block diagram of the proposed heading virtual MPC controller. The reference generation then works as follows

- The undersampled virtual model is controlled by the MPC controller, which generates an undersampled control signal $\mathbf{u}_{\mathrm{MPC}}[k]$

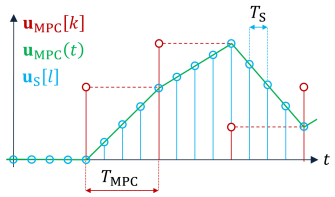

Fig. 8. Oversampling strategy for the MPC control of the heading: in red the undersampled signal, in green the undersampled signal filtered by the causal FOH and in blue the oversampled signal

- The undersampled control signal is interpolated using an affine law, leading to a continuous piecewise affine control signal $\mathbf{u}_{\mathrm{MPC}}(t)$
- This piecewise affine control signal is sampled at the drone sampling period, which gives an oversampled control signal $\mathbf{u}_{\mathrm{S}}[l]$, sent to the oversampled virtual model
- The output of the oversampled signal constitutes the reference to be tracked by the physical attitude controller, while the oversampled control signal is sent as a feedforward input
- The undersampled control signal keeps being interpolated until an entire sampling period T_{MPC} has passed, and a new undersampled control signal $\mathbf{u}_{\mathrm{MPC}}[k+1]$ is generated by the MPC controller.

An example of an undersampled signal filtered by a causal First Order Hold (FOH) filter before resampling at a higher frequency is presented on Fig. 8. Notice that the Zero-Order Hold (ZOH) discretization does not hold anymore for the undersampled virtual model. For a given time t such as $l\,T_{\mathrm{S}} \leqslant t < (l+1)T_{\mathrm{S}}$, with $l \in \mathbb{N}$, and $l\,T_{\mathrm{S}} = k\,T_{\mathrm{MPC}} + i\,T_{\mathrm{S}} = (k\,N+i)\,T_{\mathrm{S}}$, with $k \in \mathbb{N}$ and $i \in [\![0, N-1]\!]$, the following expression holds

$$\mathbf{u}_{\mathrm{S}}(t) = \mathbf{u}_{\mathrm{S}}[l] = \left(1 - \frac{i}{N}\right)\mathbf{u}_{\mathrm{MPC}}[k-1] + \frac{i}{N}\,\mathbf{u}_{\mathrm{MPC}}[k] \quad (6)$$

The continuous-time system is modeled by the LTI state-space representation (3). The ZOH discretization at the period T_{S} of this continuous-time system is given by

$$\begin{aligned} \mathbf{u}_{\mathrm{S}}[l+1] &= \mathbf{F}\,\mathbf{x}_{\mathrm{S}}[l] + \mathbf{G}\,\mathbf{u}_{\mathrm{S}}[l] \\ \mathbf{z}_{\mathrm{S}}[l] &= \mathbf{C}\,\mathbf{x}_{\mathrm{S}}[l] + \mathbf{D}\,\mathbf{u}_{\mathrm{S}}[l] \end{aligned} \quad (7)$$

with $\mathbf{F} = e^{\mathbf{A}T_{\mathrm{S}}}$ and $\mathbf{G} = \left(\int_0^{T_{\mathrm{S}}} e^{\mathbf{A}\theta}\,\mathrm{d}\theta\right)\mathbf{B}$.

The discretized expression of the undersampled system controlled by the MPC law is recursively computed

$$\begin{aligned} \mathbf{x}_{\mathrm{MPC}}[k+1] &= \mathbf{x}_{\mathrm{S}}\big[(k+1)N\big] \\ &= \mathbf{F}\,\mathbf{x}_{\mathrm{S}}\big[(k+1)N-1\big] + \mathbf{G}\,\mathbf{u}_{\mathrm{S}}\big[(k+1)N-1\big] \\ &= \mathbf{F}^N\,\mathbf{x}_{\mathrm{S}}[kN] + \sum_{i=0}^{N-1}\mathbf{F}^{N-1-i}\mathbf{G}\,\mathbf{u}_{\mathrm{S}}[kN+i] \end{aligned}$$

Using (6), it leads to

$$\begin{aligned} \mathbf{x}_{\mathrm{MPC}}[k+1] &= \mathbf{F}^N\,\mathbf{x}_{\mathrm{MPC}}[k] \\ &+ \left(\sum_{i=0}^{N-1}\left(1-\frac{i}{N}\right)\mathbf{F}^{N-1-i}\mathbf{G}\right)\mathbf{u}_{\mathrm{MPC}}[k-1] \\ &+ \left(\sum_{i=0}^{N-1}\frac{i}{N}\,\mathbf{F}^{N-1-i}\mathbf{G}\right)\mathbf{u}_{\mathrm{MPC}}[k] \end{aligned}$$

This leads to the new augmented discretized model

$$\begin{aligned} \widetilde{\mathbf{x}}_{\mathrm{MPC}}[k+1] &= \mathbf{F}_{\mathrm{MPC}}\,\widetilde{\mathbf{x}}_{\mathrm{MPC}}[k] + \mathbf{G}_{\mathrm{MPC}}\,\mathbf{u}_{\mathrm{MPC}}[k] \\ \mathbf{z}_{\mathrm{MPC}}[k] &= \mathbf{C}_{\mathrm{MPC}}\,\widetilde{\mathbf{x}}_{\mathrm{MPC}}[k] \end{aligned} \quad (8)$$

with $\mathbf{F}_{\mathrm{MPC}} = \begin{pmatrix} \mathbf{F}^N & \sum_{i=0}^{N-1}\left(1-\frac{i}{N}\right)\mathbf{F}^{N-1-i}\mathbf{G} \\ \mathbf{0}_{p\times n} & \mathbf{0}_{p\times p} \end{pmatrix}$, $\mathbf{G}_{\mathrm{MPC}} = \begin{pmatrix} \sum_{i=0}^{N-1}\frac{i}{N}\,\mathbf{F}^{N-1-i}\mathbf{G} \\ \mathbf{I}_p \end{pmatrix}$, $\widetilde{\mathbf{x}}_{\mathrm{MPC}}[k] = \begin{pmatrix} \mathbf{x}_{\mathrm{MPC}}[k] \\ \mathbf{u}_{\mathrm{MPC}}[k-1] \end{pmatrix}$, $\mathbf{C}_{\mathrm{MPC}} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \end{pmatrix}$. The same strategy as in Section IV-C can then be applied, but with the model (4) replaced by the model (8). Since the inputs and outputs of those two models are the same, the expression of the cost function (5) is unchanged (but model (8) is used for the prediction).

V. EXPERIMENTS

A. PSO-based MPC tuning

The model (8) is used for the MPC controller synthesis, which requires to tune 6 parameters: H_p, H_u μ_ψ, μ_r, $\mu_{\dot{r}}$ and μ_{u_ψ}. The sampling period for the MPC controller is considered $T_{\mathrm{MPC}} = 0.1$s. A time response around 1s is chosen for the closed-loop in this work, which means that a pertinent prediction horizon H_p would be around 10 steps. Finally, the control horizon H_u should be reduced in order to limit the computing resources requirement if constraints are to be added.

For given values of H_p and H_u, the tuning of the 4 remaining parameters is performed by a Particle Swarm Optimization (PSO) algorithm. First, an ideal response of the oversampled model to a heading step reference and a heading ramp reference are defined, as well as a template around these ideal responses. Then, the optimization program has to find the set of parameters of the undersampled MPC controller that leads to the closest oversampled response to the ideal one in the mean square sense. Penalties are added to the cost function should the closed-loop response exit the template, overshoots or undershoots, show a non-minimum phase behavior or oscillate.

The task is then repeated for different values of H_p and H_u and the best settings, meaning those that produce the lowest cost function value, are retained. This entire procedure is automatized for more convenience. Simulation results obtained with this method are shown on Fig. 9. For a ramp reference (red), the MPC generates an undersampled control signal (blue). This control signal is resampled at higher frequency (light blue) and sent to the oversampled virtual model, whose heading (light blue, serving as a filtered reference for the physical controller) is close to a previously defined ideal response (dash green).

The elevation of the camera does not have any dynamics since it is given by the virtual gimbal of the Bebop 2. The camera elevation references can then be either filtered by a non-causal low pass filter, which does not induce any phase distortion, or tracked using the same method as the heading, by conferring the camera elevation a virtual dynamics.

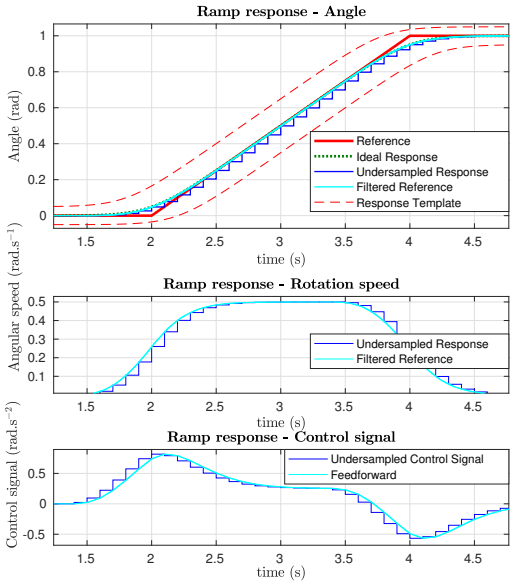

Fig. 9. Heading ramp response with the optimal settings

B. Outdoor flight

A outdoor flight has been performed on a Parrot Bebop 2 quadcopter in order to validate the overall strategy presented in this paper. The flight plan performed contains 5 waypoints, $\{W_0, \ldots, W_4\}$. All the waypoints are *autonext* except W_0 and W_4 which are *stop* waypoints and W_3 which is a *lock* one. The trajectory thus contains 4 pieces. The camera references are the following: Piece 1 - *tangent* heading and *constant* elevation, Piece 2 - *Smooth* transition to the next piece, both for the camera heading and elevation, Piece 3 - *POI* reference both for the camera heading and elevation, Piece 4 - *Smooth* transition from the previous piece to a final heading (south) and a final elevation (zero).

The video recorded by the drone is available at `https://youtu.be/a2tFIznBJ3I`. The produced video respects the specifications previously defined in terms of jerk, acceleration, and velocity, resulting in a smooth, continuous sequence. Notice that during the flight test, the digital stabilization of the camera of the Bebop 2 does not add any smoothing to the output of the MPC. However, due to the vibrations of the drone and the limitations of the system encoding the camera signals, the video still presents small jolts, independent of the proposed algorithm.

VI. CONCLUSIONS

A strategy for shooting aerial long takes with a quadrotor has been presented. The proposed strategy is based on the use of piecewise polynomial, minimum jerk trajectories and predictive tracking of camera references. In order to reduce the computation load for both the trajectory generation and camera reference tracking, a receding horizon method is proposed for the trajectory generation problem, and an undersampling strategy is adopted for the predictive tracking of camera references. The overall work is validated on an outdoor flight.

Including robustness issues with respect to uncertainties and bounded disturbances in the control strategy will be addressed in future work. Adding constraints in the MPC law for the camera references tracking so that the control signals and the drone rotation speed do not exceed maximum values would also be a way to improve the camera references tracking.

REFERENCES

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE ICRA*, 2011.

[2] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," *IEEE ICRA*, 2013.

[3] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *International Symposium of Robotics Research*, 2016.

[4] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, 2015.

[5] W. V. Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mechanical Sciences*, vol. 6, no. 2, 2015.

[6] R. Van Parys and G. Pipeleers, "Spline-based motion planning in an obstructed 3D environment," *20th IFAC World Congress*, 2017.

[7] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *Trans. on Control Systems Technology*, 2017.

[8] T. Engelhardt, T. Konrad, B. Schafer, and D. Abel, "Flatness-based control for a quadrotor camera helicopter using model predictive control trajectory generation," in *24th Mediterranean Conference on Control and Automation*, 2016.

[9] N. Joubert, L. E. Jane, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, and P. Hanrahan, "Towards a drone cinematographer: guiding quadrotor cameras using visual composition principles," *ACM Trans. Graph.*, 2016.

[10] N. Joubert, "Tools to facilitate autonomous quadrotor cinematography," Ph.D. dissertation, Stanford University, 2017.

[11] M. Roberts and P. Hanrahan, "Generating dynamically feasible trajectories for quadrotor cameras," *ACM Trans. Graph.*, vol. 35, no. 4, 2016.

[12] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," *Conf. on Human Factors in Computing Systems*, 2016.

[13] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, 2017.

[14] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Trans. Graph.*, vol. 36, no. 4, 2017.

[15] G. Li, K. M. Tsang, and S. L. Ho, "A novel model-following scheme with simple structure for electrical position servo systems," *International Journal of Systems Science*, vol. 29, no. 9, 1998.

[16] S. Skoczowski, "The robust control system with use of nominal model of controlled plant," *IFAC Symposium on Advanced Control of Chemical Processes*, vol. 33, no. 10, 2000.

[17] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2004.

[18] G. Rousseau, C. Stoica Maniu, S. Tebbani, and M. Babel, "Impact of propellers inertia and asymmetries on a V-shaped quadrotor," in *20th IFAC World Congress*, 2017.

[19] E. F. Camacho and C. Bordons, *Model Predictive Control, Second Edition*, Springer ed., 2008.

[20] J. M. Maciejowski, *Predictive control with constraints*, Prentice Hall ed., 2002.

[21] J. A. J. Ligthart, P. Poksawat, and L. Wang, "Experimentally validated Model Predictive Controller for a hexacopter," *20th IFAC World Congress*, 2017.