Optimal trajectory planning and predictive control for cinematographic flight plans with quadrotors

Gauthier Rousseau

université PARIS-SACLAY

CentraleSupélec

Parrot

NNT : 2019SACLC086

Thèse de doctorat

Optimal trajectory planning and predictive control for cinematographic flight plans with quadrotors

Thèse de doctorat de l'Université Paris-Saclay
préparée à CentraleSupélec

Ecole doctorale n°580 Sciences et Technologies de l'information (STIC)
Spécialité de doctorat : Automatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 18/10/2019, par

GAUTHIER ROUSSEAU

Composition du Jury :

Nicolas Langlois Professeur, ESIGELEC/Université de Rouen Normandie	Président
Nicolas Petit Professeur, Mines ParisTech	Rapporteur
Didier Theilliol Professeur, Université de Lorraine	Rapporteur
Sylvain Bertrand Chercheur, ONERA	Examinateur
Pedro Castillo-García Professeur, Université de Technologie de Compiègne	Examinateur
Cristina Stoica Maniu Professeur, CentraleSupélec/L2S	Directrice de thèse
Sihem Tebbani Professeur, CentraleSupélec/L2S	Co-encadrante
Mathieu Babel Ingénieur, Parrot Drones	Co-encadrant
Nicolas Martin Ingénieur, Parrot Drones	Co-encadrant, invité

Remerciements

Je tiens à remercier Supélec et Parrot Drones pour m'avoir soutenu et aidé à monter ce projet de thèse CIFRE que je leur ai proposé à l'issue de mon stage de fin d'études au sein de Parrot Drones. Mes encadrants d'alors, Cristina Stoica Maniu, Professeur au département Automatique de CentraleSupélec, et Mathieu Babel, ingénieur automaticien à Parrot Drones, m'ont accordé leur confiance et leur temps pour concrétiser cette ambition de mêler recherche et industrie en approfondissant un sujet d'étude sur 3 ans. Je remercie les départements Automatique de Supélec et Parrot Drones ainsi que l'école doctorale STIC d'avoir rendu cela possible en m'accueillant et en m'intégrant à leurs équipes pendant cette période.

À ma directrice de thèse, Cristina Stoica Maniu, je tiens à manifester ma profonde reconnaissance pour son soutien, sa bienveillance et son enthousiasme à toute épreuve tout au long de cette aventure. Je remercie également Sihem Tebbani du département Automatique de CentraleSupélec, ma co-encadrante, pour ses conseils, son temps et son expertise.

Je tiens également à exprimer ma gratitude envers Mathieu Babel, co-encadrant de ma thèse à Parrot Drones pour avoir partagé avec moi sa maîtrise du domaine de la robotique et ses nombreuses qualités à la fois professionnelles et personnelles. Je remercie sincèrement Nicolas Martin, ingénieur automaticien et responsable de mon équipe à Parrot Drones pour son implication dans ce travail, ses conseils et ses aptitudes à la fois humaines et techniques.

Je remercie Nicolas Petit, Professeur à l'école des Mines de Paris, et Didier Theilliol, Professeur à l'Université de Lorraine, pour avoir accepté d'être rapporteurs de ma thèse, ainsi que pour leurs remarques et conseils avisés. Je remercie également Sylvain Bertrand et Pedro Castillo-García pour avoir accepté d'être membres de mon jury de thèse et Nicolas Langlois, pour l'avoir présidé.

Je remercie les membres des équipes Automatique de Supélec et Parrot Drones pour leur convivialité, leur disponibilité et leur expérience. J'ai eu l'occasion de vivre de nombreux moments précieux à leurs côtés et je leur en suis reconnaissant.

Je remercie également Olivier Deschamps, pilote professionnel de drones à Parrot Drones pour ses conseils sur la prise de vue aérienne.

Enfin, je tiens évidemment à remercier ma famille, mes amis, et Anne ma plus fidèle supportrice pour m'avoir accompagné et soutenu dans cette expérience.

Contents

Notation

Sets

$\mathbb{N}$, $\mathbb{N}^*$	The sets of natural and non-zero natural numbers, respectively
$\mathbb{R}$, $\mathbb{R}^*$, $\mathbb{R}_+$	The set of real number, non-zero real numbers and positive real numbers, respectively
$\mathbb{H}$	The set of quaternions
$\mathbb{R}_n[X]$	Vector space of polynomials with real coefficients, of degree inferior or equal to n
$[\![i, j]\!]$	For $(i, j) \in \mathbb{N}^2$, with $i \leqslant j$, the set of consecutive integers $\{i, i+1, \ldots, j-1, j\}$
$[a, b], [a, b[\,,]a, b[$	For $(a, b) \in \mathbb{R}^2$, with $a \leqslant b$, a closed interval, a semi-open interval and an open interval bounded by a and b, respectively
$\mathrm{Conv}(\mathcal{S})$	The convex hull of the set $\mathcal{S}$

Algebra

$a \in \mathbb{R}$	A scalar
$s \in \mathbb{C}$	The Laplace variable
$\mathbf{a} \in \mathbb{R}^n$	A vector of n elements
$\mathbf{A} \in \mathbb{R}^{n \times m}$	A matrix of n rows and m columns
$\mathbf{I}_n, \mathbf{I}_{n \times m}$	The identity matrices of size n-by-n and n-by-m, respectively
$\mathbf{0}_n, \mathbf{0}_{n \times m}$	The matrices filled with zeros of size n-by-n and n-by-m, respectively
$\mathbf{1}_n, \mathbf{1}_{n \times m}$	The matrices filled with ones of size n-by-n and n-by-m, respectively
$\mathbf{u}^\top, \mathbf{A}^\top$	the transpose of a vector $\mathbf{u}$ and a matrix $\mathbf{A}$, respectively
$\|\mathbf{u}\|_2$	The euclidean norm of a vector $\mathbf{u}$
$\|\mathbf{u}\|_\mathbf{Q}$	The $\mathbf{Q}$-norm of a vector $\mathbf{u}$, i.e. given by $\mathbf{u}^\top \mathbf{Q}\, \mathbf{u}$

$\mathbf{u} \times \mathbf{v}$	The cross product of 2 vectors $\mathbf{u} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$
$\hat{\mathbf{u}}$	The tensor such that, for $\mathbf{u} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$, $\hat{\mathbf{u}}\,\mathbf{v} = \mathbf{u} \times \mathbf{v}$
$\mathbf{u}^{\times\times}$	The tensor such that, for $\mathbf{u} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$, $\mathbf{u}^{\times\times}\,\mathbf{v} = \mathbf{u} \times (\mathbf{u} \times \mathbf{v})$
$\dot{f}, \dot{\mathbf{f}}, \dot{\mathbf{F}}$	The time-derivative of a scalar function f, a vector function $\mathbf{f}$, a matrix function $\mathbf{F}$, respectively
$\mathcal{C}^l$	The class of differentiability of a function having its first l derivatives defined and continuous

Trigonometric functions

$\cos, \sin, \tan$	The cosine, sine and tangent functions
$c_\theta, s_\theta, t_\theta$	Abbreviations for the cosine, sine and tangent of an angle θ
$\arctan_2(y, x)$	The 2 arguments arctangent function for computing $\arctan(y/x)$
$\tanh$	The hyperbolic tangent function

Rigid body mechanics

$\mathfrak{S}_\mathcal{B}, \mathfrak{S}_{\mathcal{P}_i}, \mathfrak{S}_\mathcal{D}$	The rigid bodies constituting the quadrotor body, the i-th propeller and the entire quadrotor, respectively
B, P_i, G	The centers of mass of the quadrotor body, the i-th propeller and the entire quadrotor, respectively
$m_\mathcal{B}, m_i, m$	The masses of the quadrotor body, the i-th propeller and the entire quadrotor, respectively
$\mathbf{J}_{\mathcal{B}/B}, \mathbf{J}_{\mathcal{P}_i/P_i}, \mathbf{J}_{\mathcal{D}/G}$	The inertia tensors of the quadrotor body, the i-th propeller and the entire quadrotor, respectively, at their centers of mass
$\mathfrak{R}_\mathcal{W}, \mathfrak{R}_\mathcal{B}, \mathfrak{R}_\mathcal{D}$	The reference frames attached to the ground, of origin O, to the drone body, of origin B, and to the entire quadrotor, of origin G, respectively
$\mathfrak{B}_\mathcal{W}, \mathfrak{B}_\mathcal{B}, \mathfrak{B}_\mathcal{D}$	The vector basis associated to the reference frames $\mathfrak{R}_\mathcal{W}, \mathfrak{R}_\mathcal{B}, \mathfrak{R}_\mathcal{D}$, respectively
$\mathbf{R}_{\mathcal{W}\to\mathcal{D}}$	The rotation operator transforming the vector basis $\mathfrak{B}_\mathcal{W}$ into the vector basis $\mathfrak{B}_\mathcal{D}$
$\mathbf{v}_{G/\mathcal{W}}$	The velocity of the point G relatively to the reference frame $\mathfrak{R}_\mathcal{W}$
$\mathbf{a}_{G/\mathcal{W}}$	The acceleration of the point G relatively to the reference frame $\mathfrak{R}_\mathcal{W}$
$\mathbf{p}_{\mathcal{D}/\mathcal{W}}$	The linear momentum of the solid $\mathfrak{S}_\mathcal{D}$ relatively to the reference frame $\mathfrak{R}_\mathcal{W}$

$\boldsymbol{\mu}_{\mathcal{D}/\mathcal{W}}$	The dynamic resultant of the solid $\mathfrak{S}_{\mathcal{D}}$ relatively to the reference frame $\mathfrak{R}_{\mathcal{W}}$
$\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}}$	The angular velocity of the vector basis $\mathfrak{B}_{\mathcal{D}}$ relatively to the vector basis $\mathfrak{B}_{\mathcal{W}}$
$\dot{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{W}}$	The angular acceleration of the vector basis $\mathfrak{B}_{\mathcal{D}}$ relatively to the vector basis $\mathfrak{B}_{\mathcal{W}}$
$\boldsymbol{\sigma}^{G}_{\mathcal{D}/\mathcal{W}}$	The angular momentum at the point G of the solid $\mathfrak{S}_{\mathcal{D}}$ relatively to the reference frame $\mathfrak{R}_{\mathcal{W}}$
$\boldsymbol{\delta}^{G}_{\mathcal{D}/\mathcal{W}}$	The dynamic moment at the point G of the solid $\mathfrak{S}_{\mathcal{D}}$ relatively to the reference frame $\mathfrak{R}_{\mathcal{W}}$
$\mathfrak{D}_{\mathcal{D}/\mathcal{W}}$	The dynamic torsor of the solid $\mathfrak{S}_{\mathcal{D}}$ relatively to the reference frame $\mathfrak{R}_{\mathcal{W}}$
$\mathbf{f}_{1\to 2}$	The mechanical resultant applied on 2 by 1
$\boldsymbol{\tau}^{G}_{1\to 2}$	The mechanical moment at the point G applied on 2 by 1
$\mathfrak{F}_{1\to 2}$	The mechanical torsor applied on 2 by 1
$\left\{ \begin{array}{c} \mathbf{r}_{\mathcal{T}} \\ \mathcal{T}(G) \end{array} \right\}_{G}$	The reduction at the point G of a torsor $\mathcal{T}$ of resultant $\mathbf{r}_{\mathcal{T}}$

Acronyms

AOA Angle Of Attack. 12, 32

BLDC Brushless Direct Current. 35, 51, 139

BOB Bang-Off-Bang. h, 76, 77, 103

CCW Counterclockwise. 43

COM Center Of Mass. 16, 17

CW Clockwise. 43

DOF Degrees Of Freedom. 4, 42, 119, 129, 130

EIS Electronic Image Stabilization. 4, 8

ESC Electronic Speed Control. 35, 40, 44, 139

FOH First-Order Hold. 126, 127, 129, 137, 140

FOM Figure Of Merit. 14

FOV Field of View. 4, 8, 11, 131, 135, 136

FPD Fundamental Principle of Dynamics. c, 18, 27, 34, 36, 39, 40, 42

GNC Guidance-Navigation-Control. 2, 14, 35, 37, 40

INDI Incremental Nonlinear Dynamics Inversion. 34

LTI Linear Time-Invariant. 48, 51, 124, 139

MIP Mixed Integer Programming. 83, 106

MPC Model Predictive Control. 54, 119, 122–130, 137, 140–142

MPCC Model Predictive Contouring Control. 54

NED North-East-Down. 16, 120, 130, 132

NLP Nonlinear Programming. h, 98, 103, 113, 114, 116, 117, 140, 142

NMFC Nominal Model Following Control. 122, 123, 137, 140

POI Point Of Interest. 6, 120–122

PSO Particle Swarm Optimization. 128, 129, 141

QP Quadratic Programming. 67–69, 113

SLAM Simultaneous Localization And Mapping. 3

SQP Sequential Quadratic Programing. 69, 77, 98, 116

TOF Time Of Flight. 62, 63, 67, 71, 79, 80

VRS Vortex Ring State. 32, 33, 56

ZOH Zero-Order Hold. 127, 137, 140

Figures

Tables

Chapter 1

Introduction

1.1 Quadrotors for aerial video making

In 2010, Parrot released the Parrot AR.Drone (figure 1.1), a smartphone piloted, fully stabilized quadrotor equipped with a front camera. While existing quadrotors were mostly oriented towards model aircraft enthusiasts and required advanced piloting skills, the Parrot AR.Drone was easily accessible to the general public thanks to its sensors and stabilization algorithm [19]. Without the barrier of advanced piloting skills, anyone could enjoy the possibilities of a flying camera which had been an expensive privilege reserved to professionals (from expert multirotor pilots to helicopter or aircraft mounted systems). Parrot's quadrotor peaked in popularity and the concept of low cost flying camera emerged. Though quadrotors were already present in literature before (see for instance [50], [23], [15], [48], [52], [56], [20], [56]), the release of the Parrot AR.Drone constituted a turning point in the democratization of the quadrotor to the general public.

Figure 1.1 – Parrot AR.Drone

Since then, several companies have joined the drone market and new generations of drones have been developed with new or improved flight and footage capacities. With their low cost of operation and their great maneuverability, they have become a valuable tool for aerial footage, allowing to conveniently and inexpensively perform high quality and visually rich footages. As a consequence, the system has found many applications, in a wide range of domains, such as monitoring of construction sites [33], civil infrastructure surveillance [49], forest fire prevention [129], urban traffic monitoring [63], quick deployment of search and rescue infrastructures after disasters [57], or, more and more commonly, 3D

mapping via photogrammetry [99], crops monitoring [28], [92] and cinematography [31]. In 2018, the use of drone for cinematography, photography and commercial video making represented 27% of the use cases of professional drone [112], placing it as the predominant use case before surveying and mapping (about 13%) and infrastructure inspections (about 8%) [112]. As the use of drones continues to spread in new applications and reinforces in existing ones, the global consumer drone market could be multiplied by more than 2.6 by 2020 compared to 2015, to reach a total of more than 3B$, while the total civilian drone market, including consumer, civilian and commercial applications, could rise up to 30B$ according to [45]. The research domain is not spared by the growing popularity of quadrotors and the number of publications involving such systems have grow exponentially since the early 2000s. The number of such publications has more than doubled between the 2009-2012 period and the 2013-2016 period according to [69], growing from around 150 papers to 377 publications among 8 of the most popular robotics journals or conferences.

As a consequence of this spiking popularity, it is now common to see amateurs using camera equipped quadrotors for video making. For these users, flying a quadrotor is not an end, as it used to be with model aircraft hobbyists, but only a mean to capture images. The piloting aspect of the drone thus tends to be put in the background while the general public looks for smarter drones, capable of accomplishing higher level tasks autonomously and allowing them to perform advanced footages. As the European leader in terms of consumer quadrotors, Parrot strives to develop its drones in this direction and today, the most recent products come with such features as automatic target framing, automatic target following or automatic flight plan completion in order to make the piloting aspect of the drone as transparent as possible.

Drone cinematography is one of the research domains for which the industry has been a precursor. While industrial companies were already advanced in this matter (released of the *FlightPlan* application by Parrot in 2015 for autonomous performance of static aerial single takes, released of the *Follow me* application by Parrot in 2016 for filming moving targets in a wide variety of contexts), it is only a recent domain of research in academic research and really emerged in the 2015-2016 period with such works as [60], [59], [38], [103], [35] or [43], which tackled static aerial single takes as well as moving actors filming. These works yielded impressive results and have been followed by numerous publications, such as [86], [87], [41], [58], [121], [130] in 2017, [62], [55], [44], [127], [26], [42], in 2018 or [14] and [47] at the beginning of 2019.

In order to produce such features, developments in a large field of research are required and concern domains such as computer sciences, electronics, mechanics, computer vision, machine learning and automatic control. In the case of automatic control, every aspect is concerned, from modeling and identification to state-estimation and controller design. These aspects are usually regrouped in 3 categories in the context of robotics, constituting the main blocks of what is usually referred to as the Guidance-Navigation-Control (GNC). These blocks are illustrated on figure 1.2 and can be defined as follows

- **Guidance.** The high level control layer of the drone is called guidance. The task of this stage is to generate feasible trajectories from high level indications such as waypoints to join, positions of obstacles to avoid etc. This block is often split into several sub-blocks dealing from higher to lower level tasks, leading to the generation of usable reference trajectory. For instance, in order to join a given position, this block can be divided into a path planner in charge of finding an obstacle-free path to the destination, a trajectory planner generating a trajectory to accomplish this path and a local planner that locally re-adjust this trajectory should unexpected obstacle

block the way.

- **Navigation.** In robotics, the real-time on-board estimation of the state of the system is the task of the navigation layer. For this aim, sensor data and control inputs sent by the control stage can be used. For tasks requiring high level of autonomy, this block can typically be split into a low level observer in charge of reconstructing the state of the robot by sensor fusion and a higher level layer in charge, for instance, of localizing the robot into a scene by Simultaneous Localization And Mapping (SLAM).

- **Control.** The control block is the low level control layer in charge of generating the control signals for the actuators, to track the trajectories provided by the guidance stage. To do so, this block can count on the navigation to get a feedback on the state of the system. In the case of the camera equipped quadrotor, the position and the attitude of the drone are controlled by the use of the four propellers while the orientation of the camera can be controlled by the gimbal motors for instance.

- **Supervision.** The overall supervision of the flight is operated by the supervision layer. This block is responsible of the tuning of the parameters of the three others, of the generation of the high level references sent to the guidance layer etc. This task can be assumed by a human or a program depending on the level of automation of the system.

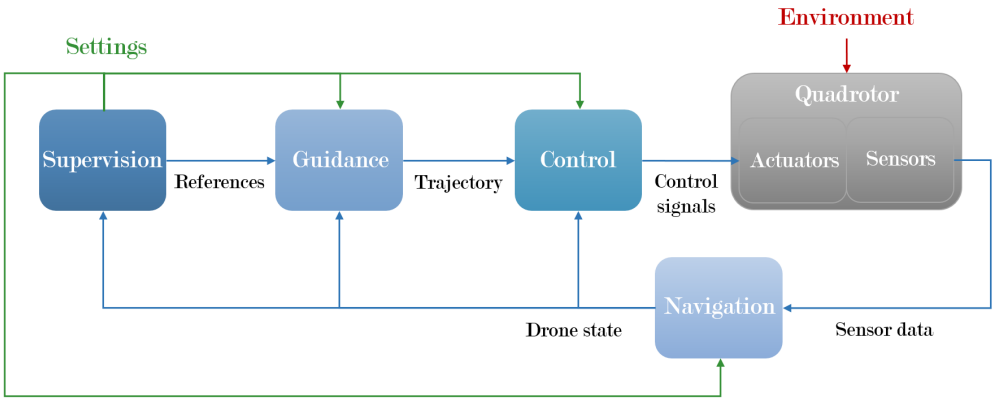

FIGURE 1.2 – GNC architecture

The goal of this PhD thesis is to contribute to the research on quadrotor guidance and control for the autonomous performance of static aerial single takes. Indeed, though it is one of the most complete of the current market, Parrot's application for completing autonomous cinematographic sequences, *FlightPlan*, has only known minor improvements since its release in 2015. In this work, we will then propose new ways to generate smooth and natural trajectories, both for the position of the drone and the angular orientation of the camera.

1.2 Problem statement

1.2.1 Aerial sequence

In this work, we seek to have a flying camera autonomously shoot an aerial sequence, specified by a list of actions to perform. More specifically, we consider as flying camera a

Parrot Bebop 2 quadrotor [91], illustrated on figure 1.3. It is equipped with 4 propellers and a fixed, digitally stabilized camera. This camera relies on a wide Field of View (FOV) to act as a virtual 3 axes gimbal, by recording images on only a part of its sensor at each frame. This is illustrated on figure 1.4, with the full FOV of the camera represented in cyan and the recorded FOV in blue. The active part of the sensor can be moved to compensate the rotation of the drone or change the direction of recording. This process is called Electronic Image Stabilization (EIS) and serves the same purpose as usual mechanical gimbals. For this reason, in the rest of this document, we consider that the drone is equipped with a virtual gimbal. The difference with a mechanical gimbal however is that the active part of the sensor can be arbitrarily reset for each frame, and the equivalent virtual gimbal is not limited by any dynamics.

FIGURE 1.3 – Parrot Bebop 2 quadrotor

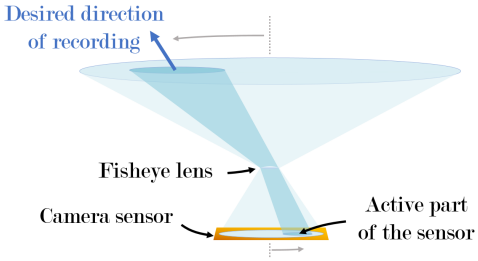

FIGURE 1.4 – Digital stabilization of the Bebop 2

A cinematographer could typically specify the sequence by the mean of a storyboard or a script. The camera has 7 Degrees Of Freedom (DOF): its 3D position, 3D attitude and magnification (i.e. zoom level). Each action is specified by a 3D position to join while rotating the camera and adjusting its magnification in a given way.

Example 1 *An example of simplistic cinematographic aerial sequence is illustrated on figure 1.5*

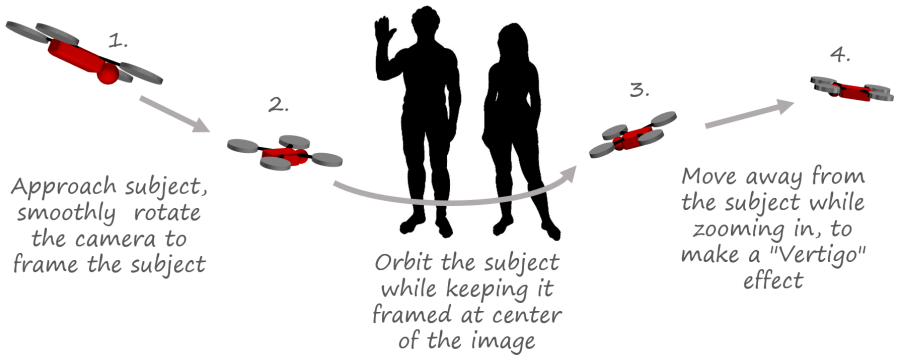

FIGURE 1.5 – Example of an aerial sequence

1. *The camera starts ahead of the target, at its left and slightly above it. The subject is framed at the bottom left corner of the image.*
2. *The camera moves toward the target and rotates in order to have the target framed at the center of the image.*
3. *The camera moves to the right while keeping the subject framed at the center of the image.*

4. *The camera moves away from the target while zooming into it, in order to make a "Vertigo" effect.*

This series of actions constitutes the high level user input and it is formalized as a *flight plan*, detailed in the following section.

1.2.2 Flight plan specifications

We consider flight plans consisting in series of $N+1$ consecutive 3D waypoints

$$\{\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_N\} \tag{1.1}$$

i.e. a starting position $\mathbf{w}_0$ followed by N waypoints to join, and flight corridors to be respected between each pair of consecutive waypoints. These corridors are considered as straight cylinders joining each pair of waypoints.

Each waypoint can be of 3 different types

- *Stop waypoint.* The drone stops on the waypoint. The first and last waypoints of the flight plan are usually *stop* waypoints, but they can also be used during the mission for taking pictures, for standing at a given point in order to record a panorama etc.
- *Lock waypoint.* The drone passes 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 specific camera shots.
- *Sphere waypoint.* The drone passes in a neighborhood of a specified radius $r_{\mathbf{w}_i}$ around the waypoint. This allows the drone to perform wider turns around the waypoint while remaining in the flight corridors, resulting in more natural trajectories.

These types are illustrated on figure 1.6 with a *stop* waypoint on the left, a *lock* waypoint on the middle and a *sphere* waypoint on the right.

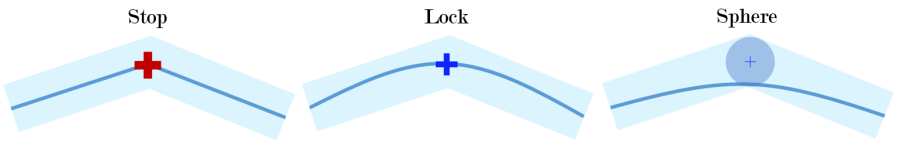

FIGURE 1.6 – Different types of waypoint validation

For each pair of consecutive waypoints $\{\mathbf{w}_{i-1}, \mathbf{w}_i\}$, $i \in [\![1, N]\!]$, a reference speed v_i, a flight corridor radius r_{corr_i} and a camera behavior are specified. The latter determines how the camera pans, tilts, rolls and zooms while joining a waypoint. One behavior can be specified for each one of the 3 rotation axes as well as the magnification. The available camera behaviors are the following

- *Constant.* The reference is fixed between two waypoints. This is typically used for travelings (see figure 1.7).

FIGURE 1.7 – Example of a constant pan reference

- *Ramp.* The reference consists in a ramp with a given constant slope. This kind of reference can be used for panoramas for instance (see figure 1.8).

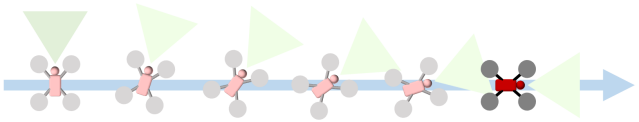

FIGURE 1.8 – Example of a ramp pan reference

- *Tangent.* This reference can only be specified for the yaw and pitch axes. The direction of recording is given by the velocity of the drone relatively to a ground fixed frame. This results in subjective, point-of-view like camera shots (see figure 1.9).

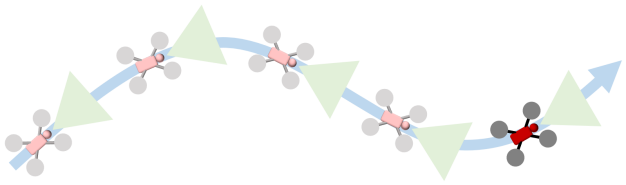

FIGURE 1.9 – Example of a tangent pan reference

- *Point Of Interest (POI).* This reference can only be specified for the yaw and pitch axes. The camera points toward a given target. This is a very popular type of camera behavior among drone users (see figure 1.10).

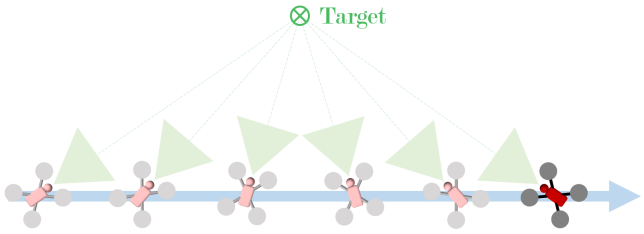

FIGURE 1.10 – Example of a POI pan reference

- *Vertigo.* This reference can only be specified for the magnification. Usually used simultaneously with POI behaviors on the pitch and yaw axes, the magnification varies such that the apparent diameter of a given target stays constant. This results in visual deformations of the background that adds intensity to a shot (see figure 1.11).

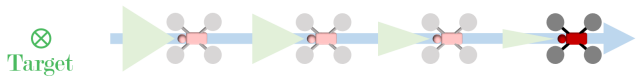

FIGURE 1.11 – Example of a vertigo magnification reference

- *Plane.* This reference can only be specified for the roll axis. Usually used with tangent behaviors on the pitch and yaw axes, the roll is linked to the rotation speed on the yaw axis to simulate the view from a fixed-wind aircraft(see figure 1.12).

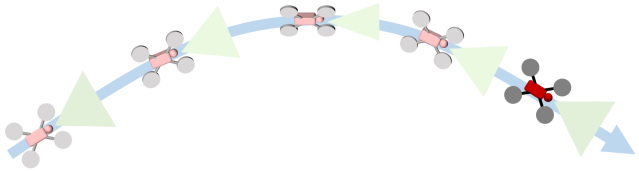

FIGURE 1.12 – Example of a plan roll reference

- *Smooth transition.* This kind of reference consists in a smooth transition between the previous and the next camera behaviors.

For instance, the sequence presented in example 1 can be formalized by the flight plan of figure 1.13 and table 1.1.

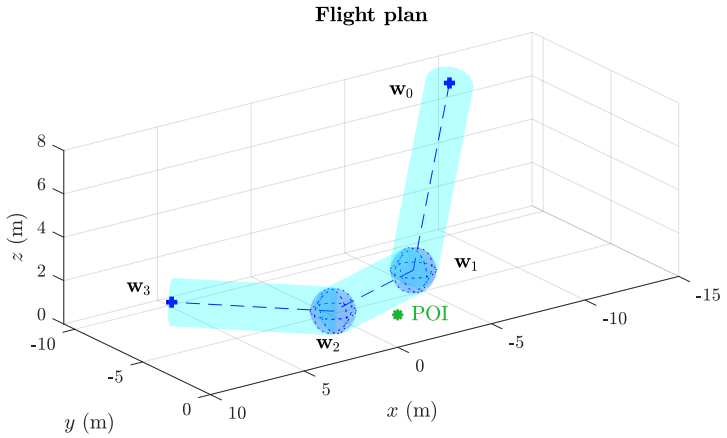

FIGURE 1.13 – Example of flight plan corresponding to the sequence of the example 1

	Type	Radius	Corridor	Speed	Yaw	Pitch	Roll	Magnification
$\mathbf{w}_0$	*stop*	-	-	-	*cst.* 0°	*cst.* 0°	*cst.* 0°	*cst.* 1×
$\mathbf{w}_1$	*sphere*	1 m	1 m	2 m·s^{-1}	*smooth*	*smooth*	*cst.* 0°	*cst.* 1×
$\mathbf{w}_2$	*sphere*	1 m	1 m	2 m·s^{-1}	*POI*	*POI*	*cst.* 0°	*cst.* 1×
$\mathbf{w}_3$	*stop*	-	1 m	2 m·s^{-1}	*POI*	*POI*	*cst.* 0°	*vertigo*

TABLE 1.1 – Detail of the flight plan on figure 1.13

Some aesthetic requirements have also been identified to guarantee a certain quality of the video when completing a flight plan and are further detailed.

1.2.3 Requirements for aesthetic aerial shots

Since the aim of this work is to improve the autonomous flight capabilities of the Parrot Bebop 2, in the context of cinematographic aerial shots, it is useful to define the *autonomous cinematographic behavior* the drone should adopt in order to ensure a good video quality. Several factors have been identified to characterize the quality of a video on

these drones, especially for the Parrot Bebop 2 for which the camera is fixed relatively to the drone and the video stabilization is entirely digital.

- First, the position, attitude and magnification should vary smoothly and naturally, without jolts, discontinuities, oscillations or overshoots. An example of inadequate position trajectory for an aerial shot is presented on figure 1.14. As both the acceleration and the *jerk* (time derivative of the acceleration) quantify the smoothness and the jolts in a motion, they should be limited for both the translational and rotational motions of the camera view.

- Secondly, following basic cinematographic principles, the reference angle of the camera should not vary too quickly, otherwise, the video would look chaotic and would not be visually satisfying. After discussion and outdoor tests with one of Parrot's professional pilots, a maximum rotation speed of $20°{\cdot}s^{-1}$ of the camera references has been judged adequate.

- As the Parrot Bebop 2 uses a fully digital stabilization of the camera, its angle relatively to the ground should also stay below 30° as much as possible, because of the way the EIS works. This is partly due to the limited FOV of the front camera, which prevents the camera to record in some directions (at the vertical of the drone typically) when the drone angle relatively to the ground reaches too high values.

- Furthermore, since the camera is fixed relatively to the drone, the rotation speed of the drone should remain limited in order to prevent motion blur to degrade the quality of the video too much.

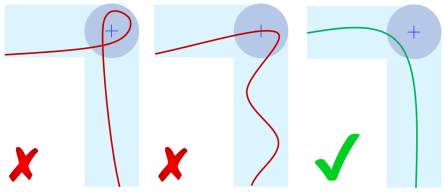

FIGURE 1.14 – Examples of inadequate and satisfying cinematographic trajectories

The respect of these conditions is a good start for making smooth and aesthetic videos.

1.3 Plan

This manuscript presents the work accomplished during this PhD thesis to achieve the goal specified in section 1.2. It is organized as follows.

First, we propose a model of the drone in chapter 2, by using the tools of rigid body mechanics. Unlike what is usually done in the literature, we build the model using the Fundamental Principle of the Dynamics along with screw theory rather than Lagrange's formalism. This study leads to a nonlinear model of the quadrotor, which is then linearized around the hovering equilibrium. The contribution of this chapter is twofold. First, by including the inertia of the propellers in the linearized model, we highlight the apparition of a non-minimum phase behavior in the rotation dynamics when tilting the propellers. The method we propose to bring out this phenomenon from the model also allows to highlight the coupling terms related to the propellers tilt between the different rotation

and translation axes of the drone. This method and its results were published in [105]. Then, we propose a state-space representation of the dynamics of the quadrotor near the hovering equilibrium which takes these phenomena into account.

Chapter 3 presents a first attempt at generating cinematographic position trajectories by using a bi-level optimization algorithm. To this aim, we first propose a new characterization of the feasibility of a 3D position trajectory, suited to quadrotor cinematography. Then, we use a bi-level optimization procedure to minimize the jerk and the duration of the trajectory. The minimization of the jerk results in a smooth and natural trajectory, while the minimization of the duration makes the speed profile getting closer to the speed references requested by the user. This results in a new cinematographic trajectory generation algorithm that computes both the shape and the speed profile of the trajectory at once, rather than in two different steps as it can be found in several works in the literature. This new algorithm can notably take into account the drag of the quadrotor, resulting in a better feasibility of the trajectory. The strategy is then validated both in simulation and on an outdoor flight with a Parrot Bebop 2 drone. The results obtained in this chapter have been published in [106].

As an alternative to the bi-level optimization algorithm, we then propose in chapter 4 a new technique to generate a B-spline trajectory with minimum duration and we apply it to aerial cinematography with quadrotors. This chapter proposes several contributions to the existing literature, starting with the use of non-uniform B-spline curves rather than uniform ones, along with a new, compact representation of piecewise, non-uniform B-spline trajectories satisfying the waypoint validation criteria defined in section 1.2.2 and the continuity of a specified amount of derivatives at the connection between each piece of trajectory. While other metrics than the duration are usually minimized when generating B-spline trajectories, the use of non-uniform B-spline curves along with this new compact representation allows us to directly minimize the duration of the trajectory. This novel method is then applied to the case of aerial cinematography and confronted to simulations and on an outdoor experiment. It has been published in [107].

Then, chapter 5 deals with the generation of the magnification trajectory and rotation motion of the camera. After detailing a way to compute the different references for each camera behavior defined in section 1.2.2, we combine a Nominal Model Following Control architecture with a Model Predictive Control law to smoothly track the references while keeping the disturbance rejection stiff. The results have been published in [106]. At last, we study the reconstruction of the 3D attitude reference of the drone from a heading and a thrust direction references. In particular, we propose a new method to achieve this and compare it with the usual method in the literature, for different Parrot quadrotors.

Finally, concluding remarks and perspectives are given in chapter 6.

1.4 Publications

In addition to the present document, the work provided during this thesis also includes the following publications.

1.4.1 Peer-reviewed journal paper

- **Rousseau, G.**, Stoica Maniu, C., Tebbani, S., Babel, M. and Martin, N. (2019). Minimum-time B-spline trajectories with corridor constraints. Application to cine-

matographic quadrotor flight plans. *Control Engineering Practice*, 89:190-203.

1.4.2 Peer-reviewed conferences papers

- **Rousseau, G.**, Stoica Maniu, C., Tebbani, S., Babel, M. and Martin, N. (2018). Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control. *Proceedings of the European Control Conference*, pages 10686-10690, Limassol, Cyprus.
- **Rousseau, G.**, Stoica Maniu, C., Tebbani, S. and Babel, M. (2017). Impact of propellers inertia and asymmetries on a V-shaped quadrotor. *Preprints of the 20th World Congress of the International Federation of Automatic Control*, pages 10686-10690, Toulouse, France.

1.4.3 Other publications

- **Rousseau, G.**, Stoica Maniu, C., Tebbani, S., Babel, M. and Martin, N. (2018). Making aerial single-sequence shots with minimum jerk trajectories and predictive camera tracking. *Nonlinear predictive control workshop of the workgroup GDR-CPNL*, Chatillon, France.
- **Rousseau, G.** (2019). Introduction to Bézier and B-spline curves. Intervention in the massive open online course *DroMOOC - A Massive Open Online Course on Drones and Aerial Multi Robot Systems* created by ONERA, CentraleSupélec and ENSTA.
- Stoica Maniu, C., Vlad, C., Chevet, T., **Rousseau, G.**, Bertrand, S., Olaru, S. (2019). Modernizing teaching through experimentation on UAVs formations. *Proceedings of the IFAC Advances on Control Education Symposium*, Philadelphia, Pennsylvania, USA.
- Wehbe, A., Arsaoui, M., Bouallou, R., Chevet, T., Castagna, F., **Rousseau G.**, Vlad, C., Stoica Maniu, C. (2019) Mini-drone modeling & control design. *Matlab-Expo*, Paris, France.

Chapter 2

Quadrotor modeling

2.1 Introduction

2.1.1 Goal

In cinematography, both the position and the attitude of the camera are controlled and can be considered as the outputs of the flying camera system, i.e. in this thesis a camera equipped quadrotor. In the case of the Bebop 2, both the pitch and roll of the camera are digitally stabilized and do not require modeling. The position of the camera however is directly linked to the one of the drone itself and, due to the limited FOV of the front camera (see figure 1.4), the yaw of the camera can not differ too much from the one of the drone as well. The goal of this chapter is then to connect the position and the yaw of the drone, considered as its outputs, to some inputs that will be specified in the following section.

To this aim, we apply the tools of rigid bodies mechanics to build a quadrotor model that will be used in the design of guidance and control laws in the next chapters. Typically, this model will allow characterizing the feasibility of trajectories and synthesizing smooth, stable controllers. Along with the construction of the model, remarks about existing results will be given in order to elaborate the state-of-the-art of quadrotor modeling.

2.1.2 Vocabulary

First and foremost, in order to avoid confusions, it can be useful to clarify the main part of the dedicated vocabulary used in this thesis.

Yaw, pitch and roll. The *attitude* of a rigid body, i.e. its 3D angular position, is often described through rotations around the *yaw*, *pitch* and *roll* axes, as illustrated on figure 2.1. The angles of each of these rotation are usually called the *heading*, *elevation* and *bank* angles, which generally correspond to the ZYX Euler angles, as illustrated on figure 2.1. Often, these 3 angles are simply called yaw, pitch and roll, respectively.

Notice that the yaw, pitch and roll rotation axes respectively correspond to the *pan*, *tilt* and *roll* rotation axes that are often referred to in the cinematographic vocabulary.

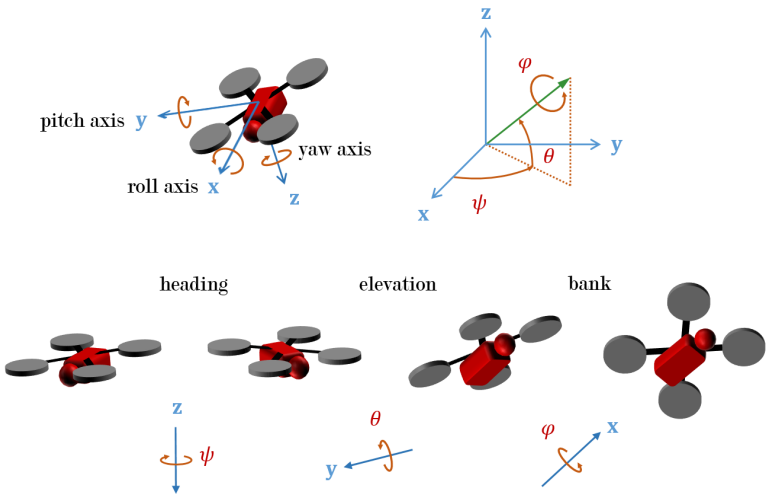

FIGURE 2.1 – Rotation of a rigid body

Ground angle. In this document, we call *ground angle* the angle between the vertical (z axis) of the drone and the vertical of the ground (denoted α on figure 2.2).

Air speed. The vector representing the velocity of the drone relatively to its surrounding air mass is called the *air velocity* and its norm is called the *air speed*.

Angle of attack. The angle between a reference line of a body (its longitudinal axis for instance) and the air velocity is called the Angle Of Attack (AOA) (see figure 2.3).

Lift & Drag. The resultant of the aerodynamic actions applied on a solid is usually split into two components, applied on the *center of pressure*. The *lift* is the component orthogonal to the velocity of the solid with respect to the surrounding air mass (air velocity). The *drag* is the component collinear to the air velocity. These notions are illustrated on figure 2.3.

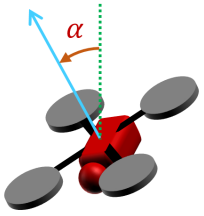

FIGURE 2.2 – Ground angle

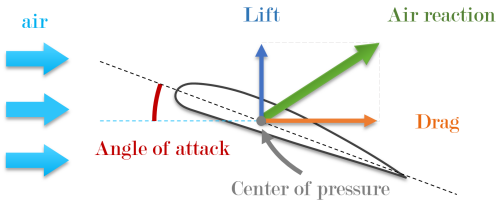

FIGURE 2.3 – Air reaction on a solid

We can now tackle the quadrotor model construction. We start by presenting the general method that we use in this chapter to build the model, in the following section.

2.1.3 Strategy

Different levels of modeling. As suggested by their name, quadrotor drones are actuated through 4 propellers, usually powered by electric motors. Technically then, the duty cycle or the voltage applied to each electric motor could be considered as the inputs of the

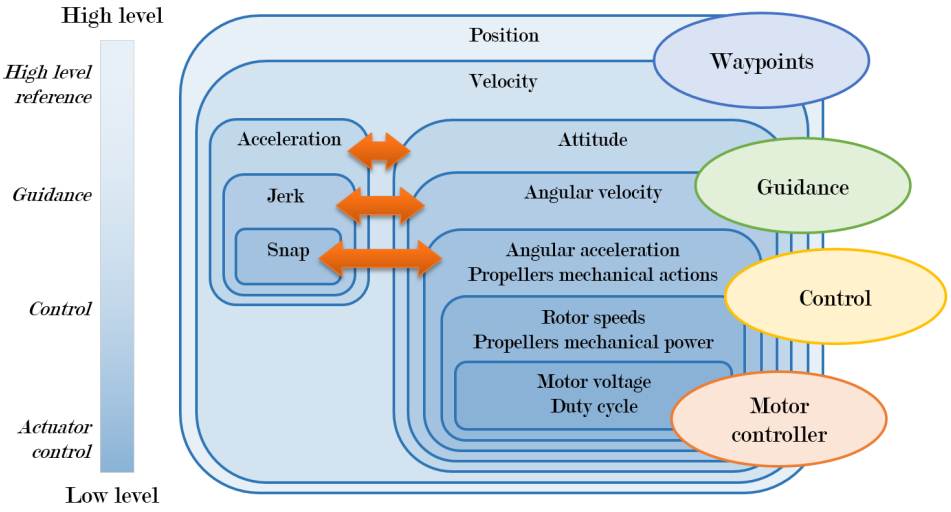

FIGURE 2.4 – Different levels of references and inputs

drone. However, the desired level of details can typically change depending on whether it is used for control or guidance, or on the complexity of the algorithm to design.

As a consequence, depending on the use of the model, other quantities can be considered as control inputs, such as the angular velocity of the drone, that would then be sent as reference to a low level controller in charge of tracking angular velocity trajectories. Such a low level controller is often supposed ideal or approximated by a simple dynamics by a higher level algorithm. Furthermore, flatness properties can be used to link some quantities such as the mechanical actions of the propellers or the angular velocity of the drone, to some derivatives of the outputs of the drone (position and yaw), which are then considered as inputs by high level algorithms (guidance algorithms typically).

As an example, in [88], the inputs of the drone are considered to be the mechanical actions applied by the propellers on the drone. These actions can be linked to the *snap* of the drone (the 4-th time derivative of the position) and the angular acceleration of the drone on the yaw axis, through a flatness analysis. Then a high level guidance algorithm can generate a snap trajectory with constraints on the time-derivatives of the trajectory (determined through the flatness analysis) for guaranteeing its feasibility. The obtained trajectory is then tracked by a lower level control algorithm that considers the mechanical actions of the propellers as inputs. The latter are finally controlled by an even lower algorithm, supposed ideal.

More generally, the higher level the algorithm, the higher level the input. This is for instance the case in [83], where a jerk trajectory is generated, or in [75], were snap trajectories are used. Notice that it is not always the case though and that some guidance algorithms can, on the contrary, make use of lower level inputs; e.g. in [72] a guidance algorithm generates advanced collision-free trajectories by taking into account the attitude required to follow the trajectory.

In a similar fashion, different levels of modeling can be considered for the control stage. Figure 2.4 provides a non-exhaustive, but representative list of the different levels of inputs that can be found in the literature, which mainly depends on the choice of the physical quantity considered as control input (motor speeds input, angular speed input etc.). These inputs are ordered from higher to lower; orange double arrows indicate links between quantities that can be highlighted through flatness analysis.

Example 2 *In order to better illustrate the classification proposed on figure 2.4, here is an example of decomposition of the different inputs for each layer of the GNC*

- *A high level client may use very simplified model of the drone dynamics, linking a high level reference (such as a waypoint to join or a velocity to reach), to high levels quantities (such as the translational acceleration of the drone) through a simplified low-pass filter model for example. This high level user lies at the top of the classification and it is represented by the blue patch.*
- *Then, a guidance block generates a trajectory based on a model considering the angular velocity of the drone as physical input. Thanks to a flatness analysis, the position of the drone appears as a flat output of the system, with its third derivative, the jerk, given as a function of the angular velocity. This allows the guidance block to work at a kinematic level for the trajectory generation, rather than using the modeled dynamics of the drone. This guidance layer is represented by the green patch.*
- *This trajectory is then tracked by a controller synthesized by considering the rotor speeds as the physical inputs and represented by the orange patch. This controller returns rotor speed references which are finally sent to a low level motor controller computing duty-cycle or voltage control signals to the actuators and represented by the red patch.*

Remark 1 *As it will be seen later, the mechanical actions provided by the propellers are given by a function of their rotation speeds and the angular acceleration is a function of these mechanical actions. As a consequence, the rotor speed and the angular acceleration could be considered as same level inputs. However, compared to a model considering these mechanical actions as inputs, a model with the rotor speed as inputs required an additional level of modeling, describing the link between the mechanical actions and the rotor speed. As it will be seen, this additional level of modeling is not trivial and these two models are then placed at two different levels in figure 2.4. The same goes for the electrical mechanical power of the propellers, which is linked by there aerodynamic power through a Figure Of Merit (FOM), which also constitutes an additional level of the model.*

Building the model. In order to design guidance and control algorithms, we seek to build a model of the drone taking as inputs some quantities among those defined in the previous paragraph and as outputs some state variables, for instance. To this aim, the tools of rigid body mechanics are used in this document. Yet a quadrotor does not constitute such a body

- The drone is not monolithic, it is constituted of several components that can move relatively to each other (rotation of the propellers, movements of the payloads etc.).
- The components of the drone have a given elasticity and plasticity and can be subject to deformations during the flight. This can lead to the apparition of vibration modes, for instance.

The first assumption for the construction of the model is to consider the drone components to be rigid, by neglecting their deformations. However, the first point still prevents to use the tools of rigid body mechanics directly on the entire quadrotor as a single solid. The drone is thus divided into several components, that can reasonably be considered as

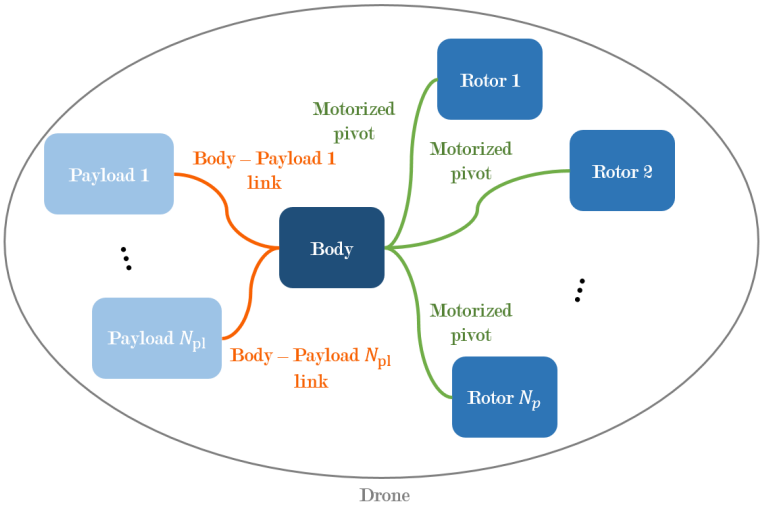

FIGURE 2.5 – Joints diagram of the drone system

rigid parts. This decomposition is illustrated on figure 2.5 and consists in the drone body (containing the camera, the battery, the sensors the arms, the antennas, etc.), the rotors and the potential payloads. The latter are neglected in this work though, since the Parrot Bebop 2 has a fixed camera and no payload.

However, in this work, the goal of the guidance or control algorithms is to control the position of the center of mass of the entire drone (body, rotors, etc.), not only of one of its components. Therefore, in order to build a model describing the evolution of the entire drone, we adopt the following method

- Apply the fundamental principle of dynamics to the drone *body* in order to get a model describing the evolution of its attitude and the position of its center of mass. This step is described in sections 2.2 to 2.4.
- Deduce from the equations obtained for the drone body a model describing the evolution of the entire drone. This step is described in section 2.5.

In the next section, we introduce the reference frames and the physical quantities relative of the drone component and the environment, used for constructing the model.

2.2 General formulation

The model is constructed under the following assumptions

Assumption 1 ***Locally flat ground.*** *Since the quadrotor is not supposed to fly over further than a few kilometers, for no longer than half an hour, the rotation and curvature of Earth are neglected. Since in addition the drone is not supposed to reach altitudes higher than a few hundreds of meters above take-off, gravity is assumed uniform during the entire flight.*

Assumption 2 ***Rigid bodies*** *Every component of the drone is assumed ideally rigid. This means that the vector joining two points of a same solid is invariant in any basis attached to this solid.*

Drone body. The *drone body* (without any rotor or payload), denoted by $\mathfrak{S}_\mathcal{B}$, is considered as a rigid body of mass $m_\mathcal{B}$, Center Of Mass (COM) B and inertia tensor $\mathbf{J}_{\mathcal{B}/B}$ at the center of mass. The mass and the inertia of the drone body are considered invariant. Attached to this solid is the reference frame *Body*, denoted by $\mathfrak{R}_\mathcal{B}$, of origin B and vector basis $\mathfrak{B}_\mathcal{B} = (\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B})$.

Propellers. The drone is equipped with N_p propellers denoted by $\mathfrak{S}_{\mathcal{P}_i}$ with $i \in [\![1, N_\text{p}]\!]$ ($N_\text{p} = 4$ for a quadrotor such as the Bebop 2). For $i \in [\![1, N_\text{p}]\!]$, we denote by $\mathfrak{S}_{\mathcal{P}_i}$ the i-th propeller, of mass m_i, center of mass P_i, and inertia $\mathbf{J}_{\mathcal{P}_i/P_i}$. Each propeller is motorized by an electrical motor whose stator belongs to the drone body $\mathfrak{S}_\mathcal{B}$, and whose rotor belongs to the propeller $\mathfrak{S}_{\mathcal{P}_i}$.

Inertial reference frame. We study the movement of the drone body $\mathfrak{S}_\mathcal{B}$ relatively to the inertial reference frame *World*, denoted by $\mathfrak{R}_\mathcal{W}$, attached to the ground, of origin O and basis $\mathfrak{B}_\mathcal{W} = (\mathbf{x}_\mathcal{W}, \mathbf{y}_\mathcal{W}, \mathbf{z}_\mathcal{W})$. The origin of $\mathfrak{R}_\mathcal{W}$, O, is chosen as the position of take-off. The North-East-Down (NED) basis, local at O, is chosen for $\mathfrak{B}_\mathcal{W}$ (as illustrated on figure 2.6). Earth rotation is neglected, hence $\mathfrak{R}_\mathcal{W}$ can be considered inertial.

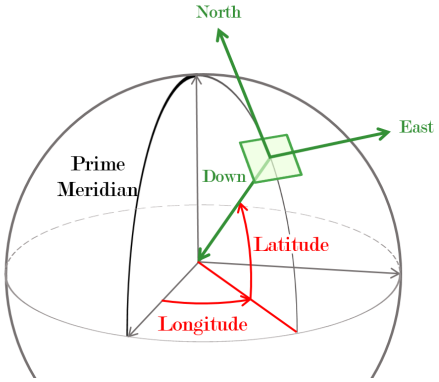

FIGURE 2.6 – NED reference frame

Drone. As explained in section 2.1.3, the study of the motion of the drone body, in interaction with its propellers and the surrounding environmnent, will be used to deduce the motion of the entire drone, *with* its propellers. To this aim, we also define G the center of mass of the entire drone (body and propellers), as well as the reference frame *Drone*, denoted by $\mathfrak{R}_\mathcal{D}$, of origin G and vector basis $\mathfrak{B}_\mathcal{D} = (\mathbf{x}_\mathcal{D}, \mathbf{y}_\mathcal{D}, \mathbf{z}_\mathcal{D}) = (\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B})$. In section 2.5, it will be shown that the entire drone can be assimilated to an equivalent fictitious rigid body $\mathfrak{S}_\mathcal{D}$, of mass m, center of mass G and inertia tensor $\mathbf{J}_{\mathcal{D}/G}$.

The rigid bodies are represented on figure 2.7 while the ground and drone frames are represented on figure 2.8.

Tables 2.1 and 2.2 also give a summary of the notation used for each solid and frame.

Position. The position of the center of mass of the drone body B and of the center of mass of the entire drone G relatively to the ground-fixed frame $\mathfrak{R}_\mathcal{W}$ are respectively given

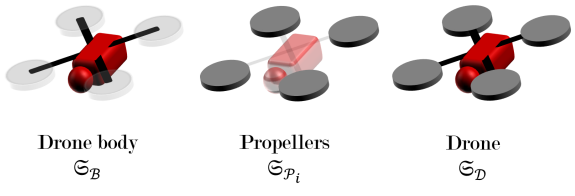

FIGURE 2.7 – Drone system and its rigid components

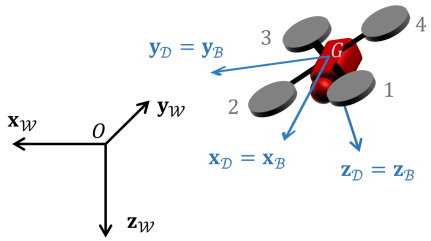

FIGURE 2.8 – Ground and drone frames

	Solid	COM	Mass	Inertia
Body	$\mathfrak{S}_{\mathcal{B}}$	B	$m_{\mathcal{B}}$	$\mathbf{J}_{\mathcal{B}/B}$
Propeller i	$\mathfrak{S}_{\mathcal{P}_i}$	P_i	m_i	$\mathbf{J}_{\mathcal{P}_i/P_i}$
Drone	$\mathfrak{S}_{\mathcal{D}}$	G	$m_{\mathcal{B}}$	$\mathbf{J}_{\mathcal{D}/G}$

TABLE 2.1 – Notation for different rigid components involved in the model construction

	Frame	Basis	Origin
Ground	$\mathfrak{R}_{\mathcal{B}}$	$\mathfrak{B}_{\mathcal{B}}$	B
Body	$\mathfrak{R}_{\mathcal{P}_i}$	$\mathfrak{B}_{\mathcal{P}_i}$	P_i
Drone	$\mathfrak{R}_{\mathcal{D}}$	$\mathfrak{B}_{\mathcal{D}}$	G

TABLE 2.2 – Notation for the different frames involved in the model construction

by the vectors

$$\boldsymbol{\zeta}_{\mathcal{B}} \triangleq \overrightarrow{OB} = \begin{pmatrix} x_{\mathcal{B}} \\ y_{\mathcal{B}} \\ z_{\mathcal{B}} \end{pmatrix}_{\mathcal{W}}, \quad \boldsymbol{\zeta} \triangleq \overrightarrow{OG} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\mathcal{W}}$$

Velocity. The velocities of the centers of mass B and G relatively to $\mathfrak{R}_{\mathcal{W}}$ are respectively given by the vectors

$$\mathbf{v}_{B/\mathcal{W}} = \begin{pmatrix} \dot{x}_{\mathcal{B}} \\ \dot{y}_{\mathcal{B}} \\ \dot{z}_{\mathcal{B}} \end{pmatrix}_{\mathcal{W}} = \begin{pmatrix} u_{\mathcal{B}} \\ v_{\mathcal{B}} \\ w_{\mathcal{B}} \end{pmatrix}_{\mathcal{B}}, \quad \mathbf{v}_{G/\mathcal{W}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_{\mathcal{W}} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}_{\mathcal{D}} \tag{2.1}$$

Attitude. The attitude of the drone relatively to the basis $\mathfrak{B}_{\mathcal{W}}$ is described by the rotation matrix $\mathrm{R}_{\mathcal{W}\to\mathcal{B}} = \mathrm{R}_{\mathcal{W}\to\mathcal{D}}$ (both the drone body and the entire drone have the same vector basis, thus the same attitude). It represents the rotation operator $\mathbf{R}_{\mathcal{W}\to\mathcal{B}}$ transforming the basis $\mathfrak{B}_{\mathcal{W}}$ into the basis $\mathfrak{B}_{\mathcal{B}}$. This allows writting

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \mathrm{R}_{\mathcal{W}\to\mathcal{B}} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{2.2}$$

for instance.

Angular velocity. The angular velocity of the drone relatively to $\mathfrak{B}_{\mathcal{W}}$ is given by

$$\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}_{\mathcal{B}}$$

It is linked to the attitude of the drone by the relation [126], [25]

$$\dot{\mathrm{R}}_{\mathcal{W}\to\mathcal{B}} = \mathrm{R}_{\mathcal{W}\to\mathcal{B}}\, \hat{\Omega}^{\mathcal{B}}_{\mathcal{W}/\mathcal{B}} \tag{2.3}$$

with $\hat{\Omega}^{\mathcal{B}}_{\mathcal{W}/\mathcal{B}} = \begin{pmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{pmatrix}$. Since the drone body $\mathfrak{S}_{\mathcal{B}}$ and the entire drone $\mathfrak{S}_{\mathcal{D}}$ share the same vector basis, we have $\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} = \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$.

Fundamental Principle of Dynamics. The model is constructed by applying the Fundamental Principle of Dynamics (FPD) to the drone body $\mathfrak{S}_{\mathcal{B}}$

$$\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \mathfrak{F}_{ext\to\mathcal{B}} \tag{2.4}$$

with

- $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$ the torsor representing the dynamics of the drone body relatively to the ground fixed frame $\mathfrak{R}_{\mathcal{W}}$, $\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \begin{Bmatrix} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} \\ \boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} \end{Bmatrix}_{B}$
- $\boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}}$ the resultant of $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$, equal to the time derivative of the linear momentum of the drone body relatively to $\mathfrak{R}_{\mathcal{W}}$
- $\boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}}$ the moment (i.e. the evaluation) of $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$ at B, equal to te time derivative of the angular momentum of the drone body relatively to $\mathfrak{R}_{\mathcal{W}}$

and with

- $\mathfrak{F}_{ext\to\mathcal{B}}$ the torsor representing the mechanical actions experienced by the drone body, $\mathfrak{F}_{ext\to\mathcal{B}} = \begin{Bmatrix} \mathbf{f}_{ext\to\mathcal{B}} \\ \boldsymbol{\tau}^{B}_{ext\to\mathcal{B}} \end{Bmatrix}_{B}$
- $\mathbf{f}_{ext\to\mathcal{B}}$ the resultant of $\mathfrak{F}_{ext\to\mathcal{B}}$
- $\boldsymbol{\tau}^{B}_{ext\to\mathcal{B}}$ the moment of $\mathfrak{F}_{ext\to\mathcal{B}}$ at B, the center of mass of the drone body

Remark 2 *A short reminder about torsors and screw theory is given in appendix A.*

The form of the drone model then depends on how these two physical quantities are written and modeled. The expression of the dynamic resultant and moment of the drone body typically depend on the choice of the representation of its attitude and the potential approximations made in this choice. The expression of the mechanical actions experienced by the drone body depends on the choice of which action to take into account and on how they are modeled. In the next section, we focus on the left term of (2.4), i.e. the expression of the dynamic torsor.

2.3 Drone body dynamics – $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$

General expression. The linear momentum of the drone body relatively to the ground fixed frame $\mathfrak{R}_{\mathcal{W}}$ is

$$\mathbf{p}_{\mathcal{B}/\mathcal{W}} = m_{\mathcal{B}}\,\mathbf{v}_{B/\mathcal{W}} \tag{2.5}$$

Its mass being supposed invariant, the time derivative of this momentum is given by

$$\boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} = m_{\mathcal{B}}\,\mathbf{a}_{B/\mathcal{W}} \tag{2.6}$$

with $\mathbf{a}_{B/\mathcal{W}} = \dot{\mathbf{v}}_{\mathcal{B}/\mathcal{W}}$ the acceleration of the drone body in $\mathfrak{R}_{\mathcal{W}}$.

The angular momentum of the drone body relatively to $\mathfrak{R}_{\mathcal{W}}$, at B, is given by

$$\boldsymbol{\sigma}^{B}_{\mathcal{B}/\mathcal{W}} = \mathbf{J}_{\mathcal{B}/B}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \tag{2.7}$$

Its inertia being supposed invariant, the time derivative of this angular momentum is given by

$$\boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} = \mathbf{J}_{\mathcal{B}/B}\,\dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{J}_{\mathcal{B}/B}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \tag{2.8}$$

This gives the expressions of the resultant and the moment at B of the dynamic torsor $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$. Denoting by $\mathrm{J}^{\mathcal{B}}_{\mathcal{B}/B}$ the inertia matrix at B, representing the inertia tensor at B, $\mathbf{J}_{\mathcal{B}/B}$, in the vector basis attached to the drone body $\mathfrak{B}_{\mathcal{B}}$, such as

$$\mathrm{J}^{\mathcal{B}}_{\mathcal{B}/B} = \begin{pmatrix} J_x & J_{xy} & J_{xz} \\ J_{xy} & J_y & J_{yz} \\ J_{xz} & J_{yz} & J_z \end{pmatrix}$$

we can detail the expressions (2.6) and (2.7)

$$\begin{cases} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} = \begin{pmatrix} m_{\mathcal{B}}\,\ddot{x}_{\mathcal{B}} \\ m_{\mathcal{B}}\,\ddot{y}_{\mathcal{B}} \\ m_{\mathcal{B}}\,\ddot{z}_{\mathcal{B}} \end{pmatrix}_{\mathcal{W}} = \begin{pmatrix} m_{\mathcal{B}}\,(\dot{u}_{\mathcal{B}} + q\,w_{\mathcal{B}} - r\,v_{\mathcal{B}}) \\ m_{\mathcal{B}}\,(\dot{v}_{\mathcal{B}} + r\,u_{\mathcal{B}} - p\,w_{\mathcal{B}}) \\ m_{\mathcal{B}}\,(\dot{w}_{\mathcal{B}} + p\,v_{\mathcal{B}} - q\,u_{\mathcal{B}}) \end{pmatrix}_{\mathcal{B}} \\ \boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} = \begin{pmatrix} J_x\dot{p} + J_{xy}\dot{q} + J_{xz}\dot{r} + q\,(J_{xz}p + J_{yz}q + J_z r) - r\,(J_{xy}p + J_y q + J_{yz}r) \\ J_{xy}\dot{p} + J_y\dot{q} + J_{yz}\dot{r} + r\,(J_x p + J_{xy}q + J_{xz}r) - p\,(J_{xz}p + J_{yz}q + J_z r) \\ J_{xz}\dot{p} + J_{yz}\dot{q} + J_z\dot{r} + p\,(J_{xy}p + J_y q + J_{yz}r) - q\,(J_x p + J_{xy}q + J_{xz}r) \end{pmatrix}_{\mathcal{B}} \end{cases} \tag{2.9}$$

Simplifications. The vector basis of the drone body, $\mathfrak{B}_{\mathcal{B}}$, is supposed to be a principal basis of inertia of the drone body (if not, a fixed change of basis suffices to make verify this assumption, since the drone body inertia is supposed invariant). In this case, the inertia matrix $\mathrm{J}^{\mathcal{B}}_{\mathcal{B}/\mathcal{B}}$ is diagonal

$$\mathrm{J}^{\mathcal{B}}_{\mathcal{B}/B} = \begin{pmatrix} J_x & & \\ & J_y & \\ & & J_z \end{pmatrix} \tag{2.10}$$

which simplifies the expression of the dynamic moment

$$\boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} = \begin{pmatrix} J_x\dot{p} + qr\,(J_z - J_y) \\ J_y\dot{q} + pr\,(J_x - J_z) \\ J_z\dot{r} + pq\,(J_y - J_x) \end{pmatrix}_{\mathcal{B}} \tag{2.11}$$

Coupling terms in $\boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}}$ can thus be canceled if there are symmetries in the geometry of the drone body. For instance, if $J_x = J_y$, the term $pq\,(J_y - J_x)$ is canceled, which allows (in some cases) to decouple the yaw dynamics from the roll and pitch dynamics.

For small angular speeds, (2.9) can be linearized and, if (2.10) is verified, this gives the very simple expression

$$\begin{cases} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} \approx \begin{pmatrix} m_{\mathcal{B}}\,\dot{u}_{\mathcal{B}} \\ m_{\mathcal{B}}\,\dot{v}_{\mathcal{B}} \\ m_{\mathcal{B}}\,\dot{w}_{\mathcal{B}} \end{pmatrix}_{\mathcal{B}} \\ \boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} \approx \begin{pmatrix} J_x\,\dot{p} \\ J_y\,\dot{q} \\ J_z\,\dot{r} \end{pmatrix}_{\mathcal{B}} \end{cases} \tag{2.12}$$

Expression at G. The expression of the dynamic torsor of the drone body at the center of mass of the entire drone, G, can be obtained from its expression at the drone body center of mass by

$$\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \left\{ \begin{matrix} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} \\ \boldsymbol{\delta}^{B}_{\mathcal{B}/\mathcal{W}} + \overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} \end{matrix} \right\}_{G} \tag{2.13}$$

The expression (2.13) dictates the evolution of the drone position and angular velocity and (2.3) gives the evolution of the attitude of the drone body. Injecting (2.3) into (2.9) results in a generic expression linking the mechanical actions experienced by the drone body to the evolution of its position and attitude. The obtained expression depends on the method used to parameterize the attitude.

Choice of attitude representation. The orientation of the drone (vector basis *Body*, $\mathfrak{B}_{\mathcal{B}}$) relatively to the ground (vector basis *World*, $\mathfrak{B}_{\mathcal{W}}$) can be described in several ways. In all cases, these representations rest on

- The choice of a set of parameters for characterizing the angular position of the drone at a given time instant. This choice should typically offer 3 degrees of freedom.
- The characterization of the evolution of these parameters in time, in order to describe the angular dynamics of the drone.

The following section gives an overview on three classical representations of the attitude: Euler angles, unit quaternions and rotation matrices.

2.3.1 Attitudes representation: Euler angles

Very intuitive, Euler angles are one candidate to describe the attitude of the drone. This representation consists in decomposing the attitude into 3 successive rotations.

The decomposition illustrated on figure 2.9 is used in this document. It consists in

- A rotation R_{ψ}, around the z-axis, of angle $\psi \in]-\pi, \pi]$ called *yaw* (or *heading*)
- A rotation R_{θ}, around the y-axis, of angle $\theta \in [0, \pi]$ called *pitch* (or *elevation*)
- A rotation R_{φ}, around the x-axis, of angle $\varphi \in]-\pi, \pi]$ called *roll* (or *bank*)

The reader will notice that θ is represented with a negative value on figure 2.9.

This constitutes the ZYX Euler angles convention commonly used in aerospace and corresponds to the *pan*, *tilt* and *roll* rotations in cinematography. In the literature, these Euler angles are also referred to as the Tait-Bryan angles when, as in this thesis, the decomposition is operated along 3 different axes (ZYX and not ZYZ for instance). The attitude of the drone body relatively to the ground is then

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} = \mathrm{R}_{\psi}\,\mathrm{R}_{\theta}\,\mathrm{R}_{\varphi} = \begin{pmatrix} c_\theta c_\psi & s_\varphi s_\theta c_\psi - c_\varphi s_\psi & c_\varphi s_\theta c_\psi + s_\varphi s_\psi \\ c_\theta s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi \\ -s_\theta & s_\varphi c_\theta & c_\varphi c_\theta \end{pmatrix} \tag{2.14}$$

Remark 3 *The ground angle is linked to the pitch and roll angles by the relation*

$$c_\alpha = c_\varphi c_\theta \tag{2.15}$$

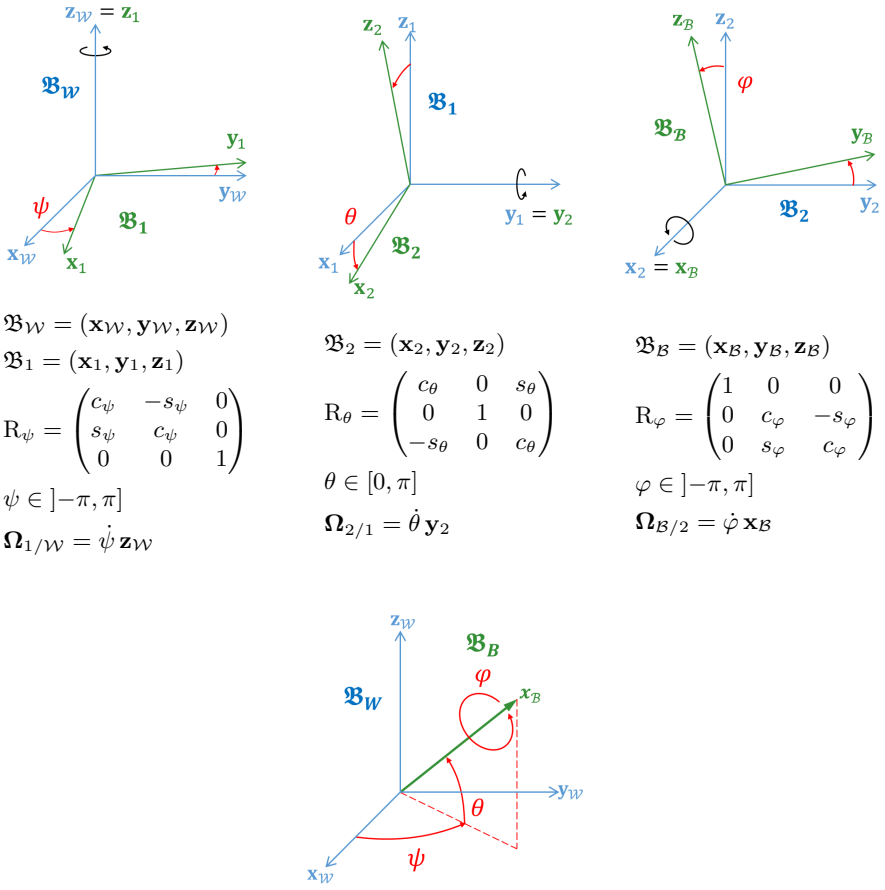

FIGURE 2.9 – Euler parameterization of the attitude

Decomposing the angular velocity $\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ along the axes of each of the rotations R_φ, R_θ and R_ψ highlights the derivatives of the 3 parameters φ, θ and ψ

$$\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} = \dot{\varphi}\, \mathbf{x}_{\mathcal{B}} + \dot{\theta}\, \mathbf{y}_2 + \dot{\psi}\, \mathbf{z}_{\mathcal{W}} \tag{2.16}$$

leading to

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & 0 & -s_\theta \\ 0 & c_\varphi & s_\varphi c_\theta \\ 0 & -s_\varphi & c_\varphi c_\theta \end{pmatrix} \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{2.17}$$

and

$$\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & s_\varphi t_\theta & c_\varphi t_\theta \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi / c_\theta & c_\varphi / c_\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2.18}$$

Remark 4 *Equation* (2.18) *highlights two singularities at* $\theta = 0$ *and* $\theta = \pi$. *They correspond to the situation of gimbal lock, for which the yaw and the roll axes are equal, resulting in the loss of one degree of freedom of the representation.*

Derivating the expression (2.18) and injecting the result into (2.11) leads to

$$\boldsymbol{\delta}^B_{\mathcal{B}/\mathcal{W}} = \begin{pmatrix} \mathrm{k}_{\mathrm{md}_1} \\ \mathrm{k}_{\mathrm{md}_2} \\ \mathrm{k}_{\mathrm{md}_3} \end{pmatrix}_{\mathcal{B}}$$

with

$$\mathrm{k}_{\mathrm{md}_1} = \ddot{\varphi}J_x - \ddot{\psi}J_x s_\theta + \dot{\theta}\dot{\psi}\left(\frac{1}{2}(J_z - J_y)c_{2\varphi}c_\theta - J_x c_\theta\right) - \frac{1}{2}\dot{\theta}^2(J_z - J_y)s_{2\varphi} + \frac{1}{2}\dot{\psi}^2(J_z - J_y)s_{2\varphi}{c_\theta}^2$$

$$\mathrm{k}_{\mathrm{md}_2} = -\ddot{\theta}J_y c_\varphi + \ddot{\psi}J_y s_\varphi c_\theta - \dot{\varphi}\dot{\theta}(J_y s_\theta + (J_x - J_z)s_\varphi) + \dot{\varphi}\dot{\psi}(J_x + J_y - J_z)c_\psi c_\theta + \dot{\theta}\dot{\psi}(J_x + J_y - J_z)s_\varphi s_\theta - \frac{1}{2}\dot{\psi}^2(J_x - J_z)s_{2\theta}c_\varphi$$

$$\mathrm{k}_{\mathrm{md}_3} = -\ddot{\theta}J_z s_\varphi + \ddot{\psi}J_z c_\varphi c_\theta \dot{\varphi}\dot{\theta}((J_y - J_x)c_\varphi - J_z s_\varphi c_\theta) + \dot{\varphi}\dot{\psi}(J_x - J_y - J_z)s_\varphi c_\theta + \dot{\theta}\dot{\psi}(J_x - J_y - J_z)c_\varphi s_\theta + \frac{1}{2}\dot{\psi}^2(J_x - J_y)s_\varphi s_{2\theta}$$

The expression (2.18) can be approximated at order 0 for $\varphi \approx 0$ and $\theta \approx 0$, leading to

$$\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} \approx \begin{pmatrix}p\\ q\\ r\end{pmatrix} \tag{2.19}$$

and thus

$$\begin{pmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{pmatrix} \approx \begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} \tag{2.20}$$

Injecting these two results into (2.11) returns a simpler expression of $\boldsymbol{\delta}^G_{\mathcal{B}/\mathcal{W}}$

$$\boldsymbol{\delta}^B_{\mathcal{B}/\mathcal{W}} \approx \begin{pmatrix}J_x\,\ddot{\varphi} + \dot{\theta}\,\dot{\psi}\,(J_z - J_y)\\ J_y\,\ddot{\theta} + \dot{\varphi}\,\dot{\psi}\,(J_x - J_z)\\ J_z\,\ddot{\psi} + \dot{\varphi}\,\dot{\theta}\,(J_y - J_x)\end{pmatrix}_{\mathcal{B}} \tag{2.21}$$

If, in addition, the rotation speed is small, the expression (2.11) can be approximated at order 1 (in p, q, r, φ, θ and ψ), which gives

$$\boldsymbol{\delta}^B_{\mathcal{B}/\mathcal{W}} \approx \begin{pmatrix}J_x\,\ddot{\varphi}\\ J_y\,\ddot{\theta}\\ J_z\,\ddot{\psi}\end{pmatrix}_{\mathcal{B}} \tag{2.22}$$

This approximation is acceptable in a sufficient large domain around the hovering equilibrium to control a quadrotor at low speed. In [1], this approximation is considered to be valid for a ground angle below 15° and the controller is forbidden to exceed this value in order to maintain the fidelity of the model.

It can also be noticed that the rotation matrix $\mathrm{R}_{\mathcal{W}\to\mathcal{B}}$ can also be approximated at order 1 in φ, θ and ψ, leading to

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} \approx \begin{pmatrix}1 & -\psi & \theta\\ \psi & 1 & -\varphi\\ -\theta & \varphi & 1\end{pmatrix}$$

This corresponds to the expression of an infinitesimal rotation, as the sum of the identity and a skew-symmetric matrix.

Though this representation has the advantage to be intuitive, it is neither unique nor regular. In order to make it unique, it is necessary to bound the parameters ψ, θ and φ in intervals such as the ones given at the beginning of this section, which introduces discontinuities in the evolution of the parameters. This representation also exhibits two singularities at $\theta = 0$ and $\theta = \pi$. Some care is required for dealing with these singularities, as well as their neighborhood which are numerically sensitive. However, in a small domain around the hovering equilibrium, this representation can be linearized, leading to very simplified expressions of $\boldsymbol{\delta}^G_{\mathcal{B}/\mathcal{W}}$.

2.3.2 Attitudes representation: Unit quaternions

Unit quaternions constitute another way to parametrize the attitude of a rigid body, by the mean of 4 parameters. Though it is not an unique representation, it has the advantage to be singularity-free.

A rotation of angle ν around the unit vector $\mathbf{u} = a\,\mathbf{x}_\mathcal{W} + b\,\mathbf{y}_\mathcal{W} + c\,\mathbf{z}_\mathcal{W}$, with $(a, b, c) \in \mathbb{R}^3$ and $\sqrt{a^2 + b^2 + c^2} = 1$, can be represented by the two unit quaternions $\mathbf{q} \in \mathbb{H}$ and $-\mathbf{q}$ with

$$\mathbf{q} = c_{\nu/2} + a\,s_{\nu/2}\,\mathbf{i} + b\,s_{\nu/2}\,\mathbf{j} + c\,s_{\nu/2}\,\mathbf{k} \tag{2.23}$$

where $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1$.

More generally, a unit quaternion $\mathbf{q} = q_0 + q_1\,\mathbf{i} + q_2\,\mathbf{j} + q_3\,\mathbf{k}$, with $\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$, can be used to represent the rotation $\mathrm{R}_{\mathcal{B}\to\mathcal{W}}$. The expression of $\mathrm{R}_{\mathcal{B}\to\mathcal{W}}$ from the components of $\mathbf{q}$ is given by [32]

$$\mathrm{R}_{\mathcal{B}\to\mathcal{W}} = \begin{pmatrix} 1 - 2\left({q_2}^2 + {q_3}^2\right) & 2\left(q_1 q_2 - q_0 q_3\right) & 2\left(q_0 q_2 + q_1 q_3\right) \\ 2\left(q_0 q_3 + q_1 q_2\right) & 1 - 2\left({q_1}^2 + {q_3}^2\right) & 2\left(q_2 q_3 - q_0 q_1\right) \\ 2\left(q_1 q_3 - q_0 q_2\right) & 2\left(q_0 q_1 + q_2 q_3\right) & 1 - 2\left({q_1}^2 + {q_2}^2\right) \end{pmatrix} \tag{2.24}$$

which can also be written in the following manner

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} = \left(q_0^2 - \mathbf{q}_v^\top \mathbf{q}_v\right)\mathbf{I}_3 + 2\,\mathbf{q}_v\mathbf{q}_v^\top + 2\,q_0\hat{\mathbf{q}}_v \tag{2.25}$$

with

$$\mathbf{q}_v = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix}^\top$$
$$\hat{\mathbf{q}}_v = \begin{pmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{pmatrix} \tag{2.26}$$

The evolution of this quaternion is given by

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}\boldsymbol{\varpi} \tag{2.27}$$

with $\boldsymbol{\varpi}$ the pure imaginary quaternion representing $\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ in $\mathfrak{B}_\mathcal{B}$, i.e. $\boldsymbol{\varpi} = p\,\mathbf{i} + q\,\mathbf{j} + r\,\mathbf{k}$. This expression can be written in the following matrix representation

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\begin{pmatrix} 0 \\ p \\ q \\ r \end{pmatrix} \tag{2.28}$$

Remark 5 *A numerical integration of the relation* (2.27) *(for propagating a model for instance) requires care as the norm of the quaternion should remain unitary.*

As for the Euler angles, it is possible to linearize this expression near the hovering equilibrium. For a small heading and a small ground angle, $\mathbf{q}$ is close to the identity quaternion ($q_0 \approx 1$ and $q_1 \approx q_2 \approx q_3 \approx 0$). If the rotation speed is small too, then we can approximate equation (2.28) at order 1 by the following expression

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} \approx \frac{1}{2} \begin{pmatrix} 0 \\ p \\ q \\ r \end{pmatrix} \tag{2.29}$$

which allows approximating the expression of the time derivative of the angular momentum

$$\boldsymbol{\delta}^G_{\mathcal{B}/\mathcal{W}} \approx \begin{pmatrix} J_x\, \ddot{q}_1 \\ J_y\, \ddot{q}_2 \\ J_z\, \ddot{q}_3 \end{pmatrix}_{\mathcal{B}} \tag{2.30}$$

The expression of the rotation matrix (2.24) can be approximated the same way, leading to

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} \approx \mathbf{I}_3 + 2\,\hat{\mathbf{q}}_v$$

or

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} \approx \begin{pmatrix} 1 & -2\,q_3 & 2\,q_2 \\ 2\,q_3 & 1 & -2\,q_1 \\ -2\,q_2 & 2\,q_1 & 1 \end{pmatrix} \tag{2.31}$$

Remark 6 *It can be noticed that the assumptions for this linearization are stronger than the assumptions used for the linearization with the Euler angles representation. Indeed, the heading is supposed small for the quaternion representation, while it could take any value with Euler angles. For this reason, the linearization of equation* (2.11) *is less common when unit quaternion are used for representing the attitude of the drone.*

2.3.3 Attitudes representation: Rotation matrices

Finally, the attitude can directly be parameterize via the 9 components of the rotation matrix $\mathrm{R}_{\mathcal{W}\to\mathcal{B}}$. This representation is both regular and unique, but requires 9 parameters and particular care regarding numerical errors. The matrix $\mathrm{R}_{\mathcal{W}\to\mathcal{B}}$ must indeed be orthogonal with a determinant equal to 1, which constitutes a total of 6 constraints on the components of $\mathrm{R}_{\mathcal{W}\to\mathcal{B}}$

$$\mathrm{R}_{\mathcal{W}\to\mathcal{B}} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \tag{2.32}$$

with the evolution of the rotation matrix $\mathrm{R}_{\mathcal{W}\to\mathcal{B}}$ given by equation (2.3).

2.3.4 Attitudes representation: Comparison

Very intuitive, the Euler angles representation involves only 3 parameters and is largely adopted. This leads to a nonlinear model due to the presence of trigonometric functions. It

can be used in its general formulation, as in [81] or [39], or it can be simplified close to the hovering state, by linearizing the relation linking the rotation velocity to the derivatives of the Euler angles. Numerous work opt for the latter solution, e.g. [2], [16], [36], [128], [68], [98]. The model is even more simplified if the linearized expressions of the derivative of the angular momentum (2.22) is used, such as in [15] or [1].

Unit quaternions are also quite spread for the control of rigid bodies. This approach rests on 4 parameters, linked by one constraint (the quaternion must be unitary). A linearization of the expression of the dynamic moment can also be performed with unit quaternions (2.30), leading to a simpler expression. However, this is only possible for a domain around a fixed heading, while the latter can vary freely with Euler angles. Such a linearized quaternion model can still be found in the literature, e.g. [100]. Nonetheless, the nonlinear model is more common when quaternions are used for parameterizing the attitude, as in [18], [119] or [71].

Though the 4 parameters of the quaternion-based representation do not allow to visualize the attitude of the drone as intuitively as the Euler angles, it is still reasonably easy to extract a rotation angle and a rotation axis from a unit quaternion to have a better idea of the rotation it represents. In addition, though the Euler angles representation only uses 3 parameters, it can be more computationally expensive as it requires the evaluation of trigonometric functions. It also presents 2 singularities (gimbal lock) while the unit quaternions are regular. Nevertheless, both these representations are not unique, which can lead to undesirable behavior of control algorithms if this has not been taken into account during their design, as explained in [12] (unwinding phenomenon). It is however possible to make these representations unique at the price of introducing discontinuities: e.g. by bounding the 3 Euler angles into adequate intervals (see figure 2.9) or by imposing the real part of the unit quaternions to be positive, as in [18]).

Finally, the representation of the attitude directly with rotation matrices is both unique and regular, but requires 9 parameters linked by 6 constraints (characterizing the orthogonality and unit determinant). The consequence is that it can be heavier to propagate a numerical model based on rotation matrices (with methods such as the Crouch-Grossman method [27], [74], as used in [61] for instance). However, the use of this parameterization leads to compact and elegant models, that have been used to design popular control laws [65], [66], [96], [5], [46], [64], [6], [61] or [22]. Such laws have been successively used in different applications, such as indoor flight [75], [102].

Table 2.3 summarizes the main differences between each representation. The computation requirements for each of these representations are discussed in [34].

Euler angles	Unit quaternions	Rotation matrices
3 parameters	4 parameters	9 parameters
No constraint	1 constraint	6 constraints
Singular	Regular	Regular
Intuitive	Unintuitive	Unintuitive
Trigonometric operations	Matrix operations	Matrix operations

Table 2.3 – Comparison of 3 attitude representations

In this section, we detailed the expression of the dynamic torsor $\mathfrak{D}_{\mathcal{B}/\mathcal{W}}$ and presented several ways to represent and link the attitude of the drone to the dynamic torsor. In the following section, we tackle the expression of the right term of (2.4), i.e. the modeling of the mechanical actions experienced by the drone.

2.4 Drone mechanical actions – $\mathfrak{F}_{ext\to\mathcal{B}}$

With no payload nor deformations (rigid Bebop 2 assumption), the drone body experiences 3 predominant mechanical actions

- *Gravity.* The action of gravity on the drone body, represented by the torsor $\mathfrak{F}_{\text{grav}\to\mathcal{B}}$
- *Propellers actions.* The action of the propellers on the drone body, represented by the torsor $\mathfrak{F}_{\text{prop}\to\mathcal{B}}$
- *Aerodynamic actions.* The aerodynamic action of the surrounding air mass on the drone body, represented by the torsor $\mathfrak{F}_{\text{air}\to\mathcal{B}}$

Any other mechanical action can reasonably be neglected. The total (cumulative) mechanical action experienced by the drone body is then given by

$$\mathfrak{F}_{ext\to\mathcal{B}} = \mathfrak{F}_{\text{grav}\to\mathcal{B}} + \mathfrak{F}_{\text{prop}\to\mathcal{B}} + \mathfrak{F}_{\text{air}\to\mathcal{B}} \tag{2.33}$$

In the next sections, we detail each one of these mechanical actions applied on the drone body and represent them by torsors. In order to facilitate the construction of the entire drone model later, we reduce these torsors at the center of mass of the entire drone, G, rather than at the center of mass of the drone body, B.

2.4.1 Action of gravity on the drone body – $\mathfrak{F}_{\text{grav}\to\mathcal{B}}$

Unlike what can typically be found in space applications, gravity is usually assimilated to an uniform force field for consumer drones. The drone weight is thus represented by an invariant force applied to the center of mass of the drone body, B, of magnitude $m_{\mathcal{B}}\, g$ and direction $\mathbf{z}_{\mathcal{W}}$. It can thus be represented by the following torsor

$$\mathfrak{F}_{\text{grav}\to\mathcal{B}} = \left\{ \begin{matrix} m_{\mathcal{B}}\, g\, \mathbf{z}_{\mathcal{W}} \\ \mathbf{0} \end{matrix} \right\}_B = \left\{ \begin{matrix} m_{\mathcal{B}}\, g\, \mathbf{z}_{\mathcal{W}} \\ m_{\mathcal{B}}\, g\, \overrightarrow{GB} \times \mathbf{z}_{\mathcal{W}} \end{matrix} \right\}_G \tag{2.34}$$

The expression of this torsor in the drone body basis depends on the parameterization of the attitude (e.g. Euler angles, unit quaternion or rotation matrix)

$$m_{\mathcal{B}}\, g\, \mathbf{z}_{\mathcal{W}} = \begin{pmatrix} -m_{\mathcal{B}}\, g s_\theta \\ m_{\mathcal{B}}\, g s_\varphi c_\theta \\ m_{\mathcal{B}}\, g c_\varphi c_\theta \end{pmatrix}_{\mathcal{B}} = \begin{pmatrix} 2m_{\mathcal{B}}\, g(q_1 q_3 - q_0 q_2) \\ 2m_{\mathcal{B}}\, g(q_0 q_1 + q_2 q_3) \\ m_{\mathcal{B}}\, g\big(1 - 2\big({q_1}^2 + {q_2}^2\big)\big) \end{pmatrix}_{\mathcal{B}} = \left(m_{\mathcal{B}}\, g\, \mathrm{R}_{\mathcal{W}\to\mathcal{B}}^{\top} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right)_{\mathcal{B}} \tag{2.35}$$

If a linearized attitude is used (2.3.1), (2.31), we get the approximation

$$m_{\mathcal{B}}\, g\, \mathbf{z}_{\mathcal{W}} \approx \begin{pmatrix} -m_{\mathcal{B}}\, g\, \theta \\ m_{\mathcal{B}}\, g\, \varphi \\ m_{\mathcal{B}}\, g \end{pmatrix}_{\mathcal{B}} \approx \begin{pmatrix} -2m_{\mathcal{B}}\, g\, q_2 \\ 2m_{\mathcal{B}}\, g\, q_1 \\ m_{\mathcal{B}}\, g \end{pmatrix}_{\mathcal{B}} \tag{2.36}$$

2.4.2 Actions of the propellers – $\mathfrak{F}_{\text{prop}\to\mathcal{B}}$

The mechanical actions of the propellers on the drone body involve several possibly complex and nonlinear phenomena. Nonetheless, depending on the context, simple models

of these actions can be sufficient to design guidance or control algorithms within acceptable flight domains.

The drone is equipped with N_p propellers numbered from 1 to N_p ($N_\mathrm{p} = 4$ for a quadrotor such as the Bebop 2). Each propeller is motorized by an electrical motor of rotation axis $\mathbf{z}_{\mathcal{P}_i}$, whose stator belongs to the drone body $\mathfrak{S}_\mathcal{B}$, and whose rotor belongs to the propeller $\mathfrak{S}_{\mathcal{P}_i}$. For $i \in [\![1, N_\mathrm{p}]\!]$, we denote by $\mathfrak{S}_{\mathcal{P}_i}$ the i-th propeller, of mass m_i, center of mass P_i and inertia tensor $\mathbf{J}_{\mathcal{P}_i P_i}$. The position and orientation of the i-th propeller relatively to the drone is parameterized as illustrated on figure 2.10

$$\overrightarrow{GP_i} = \begin{pmatrix} l_{xi} \\ l_{yi} \\ l_{zi} \end{pmatrix}_\mathcal{B} , \quad \mathbf{z}_{\mathcal{P}_i} = \begin{pmatrix} s_{\theta_i} c_{\psi_i} \\ s_{\theta_i} s_{\psi_i} \\ c_{\theta_i} \end{pmatrix}_\mathcal{B} \tag{2.37}$$

The angular velocity of the i-th propeller relatively to the drone body is given by $\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}} = \omega_i \, \mathbf{z}_{\mathcal{P}_i}$. Often, the propellers can only rotate in one direction. For this reason, we fix $\omega_i \geqslant 0$, which leaves only one choice for $\mathbf{z}_{\mathcal{P}_i}$. Furthermore, given the shape of the propeller, the axis $\mathbf{z}_{\mathcal{P}_i}$ can be considered as a principal axis of inertia of the i-th propeller (though it not exactly correct in practice).

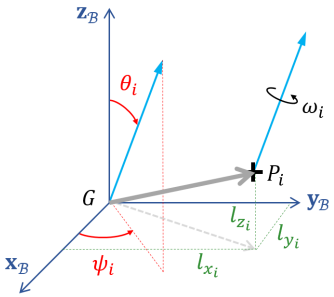

FIGURE 2.10 – Pose of the i-th propeller relatively to the drone body

The general expression of the mechanical action applied to the drone body by the i-th propeller is given by the FPD applied to $\mathfrak{S}_{\mathcal{P}_i}$ in the ground fixed frame $\mathfrak{R}_\mathcal{W}$

$$\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} = \mathfrak{F}_{\mathcal{B}\to\mathcal{P}_i} + \mathfrak{F}_{\mathrm{grav}\to\mathcal{P}_i} + \mathfrak{F}_{\mathrm{air}\to\mathcal{P}_i} \tag{2.38}$$

with

- $\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}}$ the torsor describing the dynamics of the i-th propeller at P_i
- $\mathfrak{F}_{\mathcal{B}\to\mathcal{P}_i}$ the torsor describing the mechanical actions applied to i-th propeller by the drone body
- $\mathfrak{F}_{\mathrm{grav}\to\mathcal{P}_i}$ the torsor describing the mechanical actions of gravity on the i-th propeller
- $\mathfrak{F}_{\mathrm{air}\to\mathcal{P}_i}$ the torsor describing the mechanical actions applied to i-th propeller by the surrounding air mass

From this expression, we can deduce the action of the i-th propeller on the drone body, knowing that $\mathfrak{F}_{\mathcal{B}\to\mathcal{P}_i} = -\mathfrak{F}_{\mathcal{P}_i\to\mathcal{B}}$ given Newton's 3$^\mathrm{rd}$ law

$$\mathfrak{F}_{\mathcal{P}_i\to\mathcal{B}} = -\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} + \mathfrak{F}_{\mathrm{grav}\to\mathcal{P}_i} + \mathfrak{F}_{\mathrm{air}\to\mathcal{P}_i} \tag{2.39}$$

The actions of all the propellers on the drone body are then given by the sum of the action of each propeller

$$\mathfrak{F}_{\text{prop}\to\mathcal{B}} = \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\mathcal{P}_i\to\mathcal{B}} \tag{2.40}$$

The modeling of each one of these quantities thus determines the model of the action of the propellers on the drone body and is presented in sections 2.4.2.1 to 2.4.2.3.

2.4.2.1 Propeller dynamics – $\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}}$

The expression of the torsor describing the dynamics of the i-th propeller, reduced at G, can be obtained from its expression reduced at P_i, as follows

$$\begin{aligned}\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} &= \begin{Bmatrix} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \\ \boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} \end{Bmatrix}_{P_i} = \begin{Bmatrix} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \\ \boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} + \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \end{Bmatrix}_{G} \\ &= \begin{Bmatrix} m_i\, \mathbf{a}_{P_i/\mathcal{W}} \\ \mathbf{J}_{\mathcal{P}_i/P_i}\, \dot{\boldsymbol{\Omega}}_{\mathcal{P}_i/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{W}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\, \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{W}} + m_i\, \overrightarrow{GP_i} \times \mathbf{a}_{P_i/\mathcal{W}} \end{Bmatrix}_{G}\end{aligned} \tag{2.41}$$

with $\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}$ the dynamic resultant of the propeller, $\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}}$ its dynamic moment and $\mathbf{a}_{P_i/\mathcal{W}}$ the acceleration of its center of mass relatively to the inertial frame $\mathfrak{R}_\mathcal{W}$.

The time derivative of the linear momentum of the propeller relatively to the ground can be expressed by

$$\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} = m_i\, \mathbf{a}_{\mathcal{P}_i/\mathcal{W}} = m_i(\mathbf{a}_\text{r} + \mathbf{a}_\text{d} + \mathbf{a}_\text{c}) \tag{2.42}$$

with

$$\begin{aligned}\mathbf{a}_\text{r} &= \mathbf{a}_{P_i/\mathcal{B}} \\ \mathbf{a}_\text{d} &= \mathbf{a}_{B/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \left(\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i}\right) + \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i} \\ \mathbf{a}_\text{c} &= 2\, \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{v}_{P_i/\mathcal{B}}\end{aligned}$$

the relative, drive and Coriolis accelerations, respectively. Since each component of the drone is supposed rigid, $\mathbf{v}_{P_i/\mathcal{B}} = \mathbf{0}$ and $\mathbf{a}_{P_i/\mathcal{B}} = \mathbf{0}$, the relative and Coriolis accelerations of the propeller are null and the expression of the dynamic resultant becomes

$$\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} = m_i\, \mathbf{a}_{B/\mathcal{W}} + m_i\, \boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}}\, \overrightarrow{BP_i} + m_i\, \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i} \tag{2.43}$$

where $\boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}}$ is the tensor such that $\boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}}\, \overrightarrow{BP_i} = \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \left(\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i}\right)$.

A similar decomposition can be operated on the angular moment derivative, by injecting $\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{W}} = \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ into equation (2.41). The detailed calculations are provided in appendix B. Following either the assumption

Assumption 3 *The i-th propeller behaves as a flat cylinder of axis $\mathbf{z}_{\mathcal{P}_i}$ (especially adapted for propellers with more than 3 blades)*

or the assumption

Assumption 4 *The derivative of the angular momentum of the propeller, $\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}}$, can be approximated by its mean value for one full rotation of the propeller around its axis*

and under the 3 following assumptions

Assumption 5 *The propeller is well balanced and, relatively to the drone body, only rotates around its axis* $\mathbf{z}_{\mathcal{P}_i}$

Assumption 6 *The inertia of the propeller is significantly smaller than the inertia of the drone body*

Assumption 7 *The rotation speed of the propeller relatively to the drone body,* $\left\|\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}}\right\|_2$*, is much higher than the rotation speed of the drone body relatively to the ground* $\left\|\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}}\right\|_2$

we can rewrite the dynamic torsor of the i-th propeller as the sum of 3 fictitious mechanical actions

$$\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} \approx -\mathfrak{F}_{\text{inertia}_i\to\mathcal{B}} - \mathfrak{F}_{\text{gyro}_i\to\mathcal{B}} - \mathfrak{F}_{\text{lever}_i\to\mathcal{B}} \tag{2.44a}$$

with the terms detailed in appendix B

$$\mathfrak{F}_{\text{inertia}_i\to\mathcal{B}} = -\left\{\begin{matrix}\mathbf{0}\\ \mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\mathbf{\Omega}}_{\mathcal{P}_i/\mathcal{B}}\end{matrix}\right\}_G \tag{2.44b}$$

$$\mathfrak{F}_{\text{gyro}_i\to\mathcal{B}} = -\left\{\begin{matrix}\mathbf{0}\\ \mathbf{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}}\end{matrix}\right\}_G \tag{2.44c}$$

$$\mathfrak{F}_{\text{lever}_i\to\mathcal{B}} = -\left\{\begin{matrix}\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}\\ \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}\end{matrix}\right\}_G \tag{2.44d}$$

and

$$\mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\mathbf{\Omega}}_{\mathcal{P}_i/\mathcal{B}} = \begin{pmatrix}\dot{\omega}_i J_{zi} s_{\theta_i} c_{\psi_i}\\ \dot{\omega}_i J_{zi} s_{\theta_i} s_{\psi_i}\\ \dot{\omega}_i J_{zi} c_{\theta_i}\end{pmatrix}_{\mathcal{B}} \tag{2.44e}$$

$$\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}} = \begin{pmatrix}\omega_i J_{zi}(q c_{\theta_i} - r s_{\theta_i} s_{\psi_i})\\ -\omega_i J_{zi}(p c_{\theta_i} - r s_{\theta_i} c_{\psi_i})\\ -\omega_i s_{\theta_i} J_{zi}(q c_{\psi_i} - p s_{\psi_i})\end{pmatrix}_{\mathcal{B}} \tag{2.44f}$$

where J_{zi} denotes the inertia of the i-th propeller on its rotation axis.

2.4.2.2 Propeller weight – $\mathfrak{F}_{\text{grav}\to\mathcal{P}_i}$

The action of gravity on the propellers can be represented by a force applied at the center of mass P_i of the propeller i

$$\mathfrak{F}_{\text{grav}\to\mathcal{P}_i} = \left\{\begin{matrix}m_i\,\mathbf{g}\\ \mathbf{0}\end{matrix}\right\}_{P_i} = \left\{\begin{matrix}m_i\,\mathbf{g}\\ m_i\,\overrightarrow{GP_i} \times \mathbf{g}\end{matrix}\right\}_G \tag{2.45}$$

2.4.2.3 Propeller thrust – $\mathfrak{F}_{\text{air}\to\mathcal{P}_i}$

As the propellers rotate, each of their blades receives a reaction of the surrounding air mass. This reaction is represented on figure 2.11 and is used to actuate the drone. The aerodynamic actions experienced by each blade of the propeller i adds to each other, resulting in a force, the *thrust* ($\mathbf{t}_i$ on figure 2.11), predominantly produced by the lift

of each blade ($\mathbf{l}_1$ and $\mathbf{l}_2$ on figure 2.11), and an *aerodynamic torque* ($\boldsymbol{\tau}_i$ on figure 2.11), notably produced by the drag of the blades ($\mathbf{d}_1$ and $\mathbf{d}_2$ on figure 2.11). The deflection of the surrounding air by the blades of a propeller generates a lift which adds to the lift of the other blades and constitute the thrust of the propeller (in blue on the figure). This deflection induces vortices, especially at the tip of each blade, that dissipate energy and generate a force opposing the movement of each blade called the induced drag. In the meantime, as the air flows around the blades, the air generates a force that tends to draw the blade with it due to its viscosity. This contribute to what is called the profile drag. The sum of these two drag forces is represented in red on the the figure, for each blade. The drags experienced by the blades compensate each other and the leverages they generate at the center of mass of the propeller add to generate a torque.

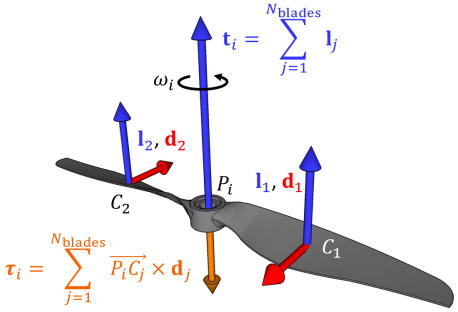

FIGURE 2.11 – Lift and drag of a propeller

Thus, each propeller experiences the air reaction in the following general form

$$\mathfrak{F}_{\text{air}\to\mathcal{P}_i} = \begin{Bmatrix} \mathbf{t}_i \\ \boldsymbol{\tau}_i \end{Bmatrix}_{P_i} = \begin{Bmatrix} \mathbf{t}_i \\ \boldsymbol{\tau}_i + \overrightarrow{GP_i} \times \mathbf{t}_i \end{Bmatrix}_{G} \tag{2.46}$$

with $\mathbf{t}_i$ the thrust and $\boldsymbol{\tau}_i$ the aerodynamic torque.

The two quantities $\mathbf{t}_i$ and $\boldsymbol{\tau}_i$ remain to be modeled. Most of the time, a quadratic model in ω_i is derived from the blade element theory

$$\begin{aligned} \mathbf{t}_i &= \rho\, a_i\, \omega_i^2\, \mathbf{z}_{\mathcal{P}_i} \\ \boldsymbol{\tau}_i &= \rho\, b_i\, \omega_i^2\, \mathbf{z}_{\mathcal{P}_i} \end{aligned} \tag{2.47}$$

with ρ the air density and a_i and b_i constants, with the coefficients a_i and b_i identified on a test bench. Though this model is quite spread and can be suitable at low speed and in a confined environment with no wind, it remains a simplistic model. Indeed, this model is mostly acceptable in static flight (i.e. with no air speed).

The thrust and torque experienced by the propeller is usually non trivial as numerous phenomena occur on the propeller in dynamic flight. Some of these phenomena are presented in the following paragraphs. The aim is not to detail the physics and models behind all these effects but to give some typical examples and their impact on the behavior of the propellers. For detailed theory and models of the aerodynamics of helicopter and multicopter rotors, one can consult [108], [3].

Dissymmetry of lift If the surrounding air mass has a non zero lateral speed relatively to the drone, the blade moving upstream (the advancing blade) has a higher air speed than the blade moving downstream (the retreating blade).

This results in a higher lift and a higher drag on the advancing blade than on the retreating one. As a consequence, the leverages induced by the lift of each blade at the center of mass the propeller, P_i, do not compensate each other anymore, and the lift now induces a torque on the propeller, as illustrated on figure 2.12, with this additional torque represented in orange. In addition, the resultant of the drag of each blade do not compensate anymore as well, and the sum of each of these drag adds a lateral component to the thrust, along $\mathbf{v}_{\text{air}/\mathcal{B}}$ and behave as a drag action on the entire propeller, represented in purple on figure 2.13.

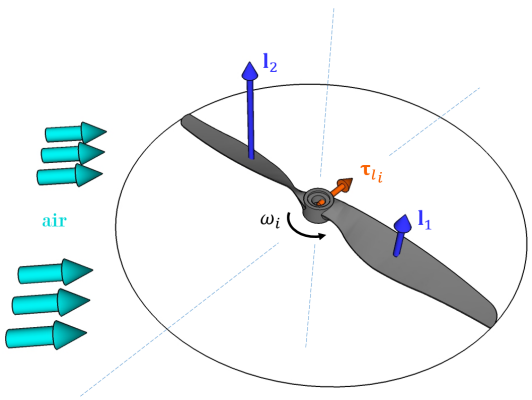

FIGURE 2.12 – Dissymmetry of lift

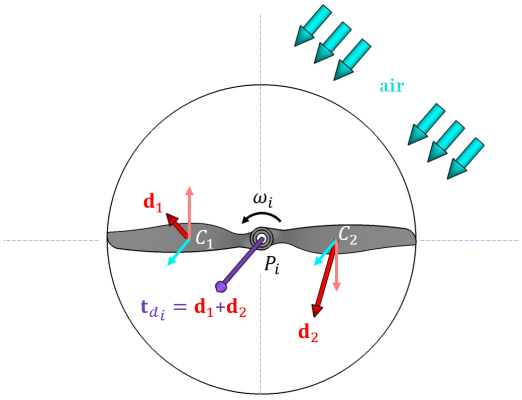

FIGURE 2.13 – Dissymmetry of drag

The impact of this additional lift-induced torque and drag-induced force on the aerodynamic actions experienced by the propeller is illustrated on figure 2.14, with the total force experienced by the propeller in purple and the total torque in orange, and can be compared to the aerodynamic actions with no lateral air speed illustrated on figure 2.11.

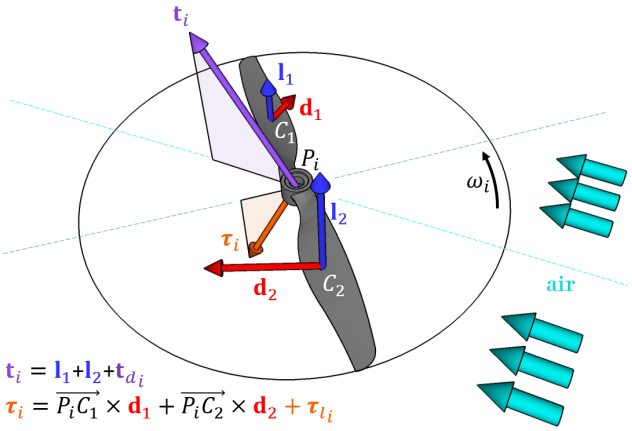

FIGURE 2.14 – Dissymmetry of lift and drag

Blade flapping Though it was assumed that all the components of the drone are rigid, this is not completely accurate in practice, especially for the rotor blades which tend to deform under the action of the lift and drag they experience. In the absence of lateral air speed, all the blades deform the same way (the rotor thus has the shape of flattened cone), and both the thrust and the torque generated by the propeller are oriented along its rotation axis.

However, with a non-zero lateral air speed, because of the dissymmetry of lift and drag,

the advancing blade experiences more mechanical stress and deforms more than the retreating one, as illustrated on figure 2.15. Because of its high rotation speed, the rotor undergoes a gyroscopic torque in response to this mechanical stress, that tends to tilt the rotor backward (figure 2.15).

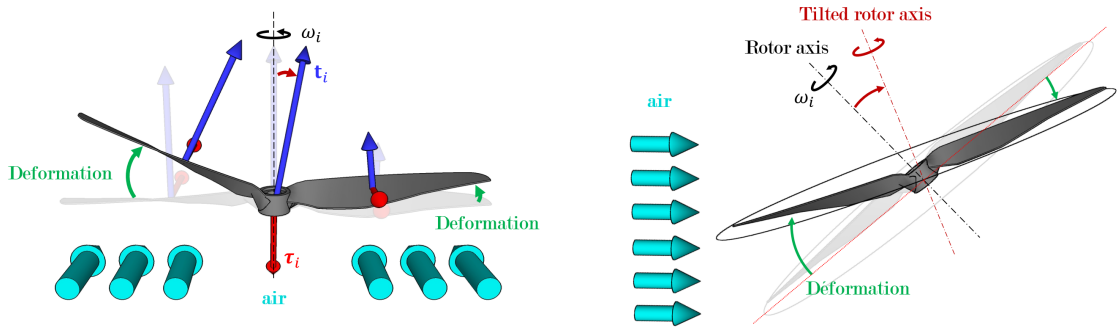

FIGURE 2.15 – Blade flapping

It should be noticed that the deformations have been exaggerated on figure 2.15 for a better illustration of the phenomenon. This effect tends to compensate a part of the dissymmetry of lift and drag, but not entirely. This deformation of the rotor disk is called *blade flapping* and results in what could be assimilated to a virtual tilt of the rotor axis. In [89], the result of this phenomenon is modeled as an additional drag term on the drone, proportional to the air speed.

Angle of attack A change of AOA induces a change of lift and drag of each blade of the propellers. As a consequence, the thrust is lower in ascending flight than in hovering flight [90]. In lateral flight, for a given rotation speed of the propeller, this typically leads to thrust pikes when the drone is braking even though the rotation speed of the propeller is unchanged (see figure 2.16).

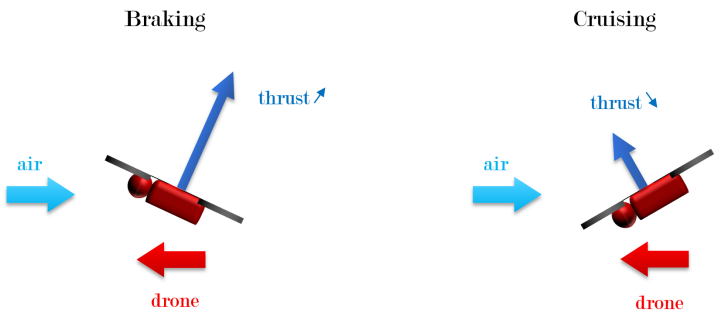

FIGURE 2.16 – Impact of the AOA on the thrust for a fixed rotation speed of the propellers

Turbulence The air stream near the rotor is turbulent and contains vortices, which propagate in the propeller wake. They are in part due to pressure difference between the upper skin (*extrados*) and the lower skin (*intrados*) of the blade. These turbulences have an impact on the propeller thrust and are linked to the other phenomena listed here. The Vortex Ring State (VRS) corresponds to an extreme case of sustained vortex between the upstream and downstream air flows of the rotor (see figure 2.17), arising for a certain range of ascending air speed. This phenomenon can degrade the thrust to such a point that the drone becomes unable to counter gravity.

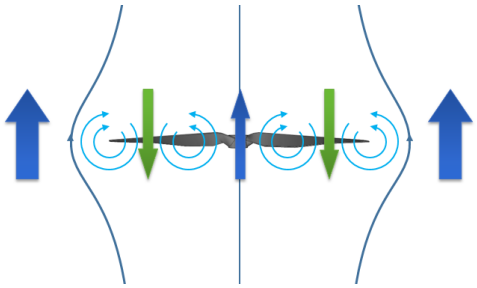

FIGURE 2.17 – Vortex Ring State

Altitude Since the lift and drag of the blades are proportional to the air density, the thrust decreases as the altitude increases (see figure 2.18). Though a drone will usually not ascend high enough during a flight to notice any significant difference in air density (regulations limits the ground altitude of drone to 150 m or less depending on the country), it still means that a given drone will behave differently when taking off at sea level or at the top of a mountain.

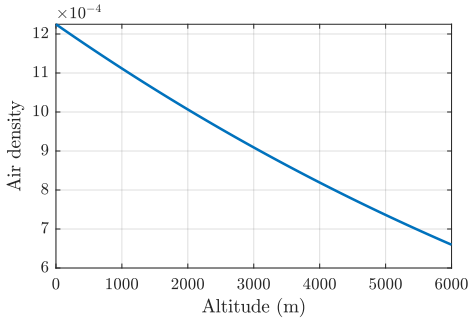

FIGURE 2.18 – International standard atmosphere model for air density

Environment In addition to all these phenomena, the environment (ground, walls, obstacles etc.) also impact the air flow around the propellers. The ground effect for instance designates the increase of thrust close to the ground, as the air deflection by the ground causes an overpressure under the propellers [10], [97], [4].

Conclusion The physics behind some of these phenomena can typically be found in aerodynamics and helicopter theory [108]. However, the small, fast, fixed-pitch rotors of multicopters do not behave exactly as helicopter rotors. In-depth studies of these phenomena and their modeling in the more specific context of multirotor propellers has been lead in [20], [5], [8] and [3]. Though much more realistic, the resulting model is also more complex than the simplified quadratic, invariant one (2.47). In [7] and [3], this more complete model is successfully used for control design.

Other works successfully included some of these phenomena in the model, through simpler, approximated models, for control synthesis and led to significant improvements of the position and the attitude control [52], [56]. However, these model-based approaches can require extensive efforts in the identification of the numerous parameters of the model. They also lead to models that are very specific to a given drone. A significant dispersion of the parameters can occur among a series of mass product drones such as the Bebop 2, so these parameters may vary too much for an identified model to be transposed to every

drone even of a same series. In addition, these parameter are expected to vary during the life cycle of a drone, through damage or alterations of its components (blade damage or replacement, change of camera, addition of accessories etc.).

An alternative to such model based approaches is to adjust a simplified model in real-time via adaptive design for instance. In particular, the use of *Incremental Nonlinear Dynamics Inversion* (INDI, [110]) has proven to be capable to efficiently deal with these phenomena even in the presence of non trivial flight environments (wind gusts [113], payload transportation [30] etc.). In [120], online identification is used to additionally take into account physical interactions with the surrounding environment.

In this manuscript, the action of air on the propellers is approximated by the sum of 2 mechanical actions

- The on-axis rotor torque and thrust modeled by a quadratic function of the propeller speed and with parameters depending of the air velocity (derived from [20])

$$\begin{aligned}\mathfrak{F}_{\text{thrust}_i\to\mathcal{P}_i} &= \begin{Bmatrix}\mathbf{t}_i\\ \boldsymbol{\tau}_i\end{Bmatrix}_{P_i} = \begin{Bmatrix} t_i\,\mathbf{z}_{\mathcal{P}_i}\\ -\tau_i\,\mathbf{z}_{\mathcal{P}_i}\end{Bmatrix}_{P_i} = \begin{Bmatrix} t_i\,\mathbf{z}_{\mathcal{P}_i}\\ -\tau_i\,\mathbf{z}_{\mathcal{P}_i} + t_i\,\overrightarrow{GP_i}\times\mathbf{z}_{\mathcal{P}_i}\end{Bmatrix}_{G}\\ &= \left\{\begin{array}{c}\begin{pmatrix} t_i\,s_{\theta_i}c_{\psi_i}\\ t_i\,s_{\theta_i}s_{\psi_i}\\ t_i\,c_{\theta_i}\end{pmatrix}_{\mathcal{B}}\\ \begin{pmatrix}\tau_i\,s_{\theta_i}c_{\psi_i} + t_i\left(l_{yi}\,c_{\theta_i} - l_{zi}\,s_{\theta_i}s_{\psi_i}\right)\\ \tau_i\,s_{\theta_i}s_{\psi_i} + t_i\left(l_{zi}\,s_{\theta_i}c_{\psi_i} - l_{xi}\,c_{\theta_i}\right)\\ \tau_i\,c_{\theta_i} + t_i\,s_{\theta_i}\left(l_{xi}\,s_{\psi_i} - l_{yi}\,c_{\psi_i}\right)\end{pmatrix}_{\mathcal{B}}\end{array}\right\}_{G}\end{aligned} \tag{2.48}$$

  with

$$\begin{cases} t_i = \rho\left(\alpha_{0i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big) + \alpha_{1i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big)\,\omega_i + \alpha_{2i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big)\,\omega_i^2\right)\\ \tau_i = \rho\left(\beta_{0i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big) + \beta_{1i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big)\,\omega_i + \beta_{2i}\big(\mathbf{v}_{\mathcal{B}/\text{air}}\big)\,\omega_i^2\right)\end{cases} \tag{2.49}$$

  and the coefficients α_{ji}, β_{ji} (scalar functions), $i \in [\![0,2]\!]$, $j \in [\![1,N_\text{p}]\!]$, identified on test bench for different air speeds for instance. In this work, this action is considered as the actuation of the drone.

- The remaining actions, considered as disturbances and approximated as follows

$$\mathfrak{F}_{\text{dist}_i\to\mathcal{P}_i} = \begin{Bmatrix}\mathbf{f}_{\text{dist}_i\to\mathcal{P}_i}\\ \boldsymbol{\tau}^{G}_{\text{dist}_i\to\mathcal{P}_i}\end{Bmatrix}_{G} \tag{2.50}$$

The action of air on the propeller is then, given by summing these 2 actions

$$\mathfrak{F}_{\text{air}\to\mathcal{P}_i} = \mathfrak{F}_{\text{thrust}_i\to\mathcal{P}_i} + \mathfrak{F}_{\text{dist}_i\to\mathcal{P}_i} \tag{2.51}$$

2.4.2.4 Motor – $\mathfrak{F}_{\mathcal{B}\to\mathcal{P}_i}$

Often neglected, the dynamics of the electric motor driving the propeller can be taken into account to improve the synthesis of control algorithms. The FPD applied to the i-th propeller (2.39) can be projected along the rotor axis $\mathbf{z}_{\mathcal{P}_i}$, leading to

$$\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}}\cdot\mathbf{z}_{\mathcal{P}_i} = \left(\boldsymbol{\tau}^{P_i}_{\mathcal{B}\to\mathcal{P}_i} + \boldsymbol{\tau}^{P_i}_{\text{air}\to\mathcal{P}_i} + \boldsymbol{\tau}^{P_i}_{\text{grav}\to\mathcal{P}_i}\right)\cdot\mathbf{z}_{\mathcal{P}_i} \tag{2.52}$$

Gravity does not induce any torque on the propeller. The variation of angular momentum has been detailed section 2.4.2.3. The term $\boldsymbol{\tau}^{P_i}_{\text{air}\to\mathcal{P}_i}$ depends on the model of aerodynamic actions on the propeller. For the quadratic thrust model (2.48), we have $\boldsymbol{\tau}^{P_i}_{\text{air}\to\mathcal{P}_i} \cdot \mathbf{z}_{\mathcal{P}_i} = \tau_i$.

Injecting (2.44a) into (2.52) leads to

$$J_{zi}\,\dot{\omega}_i = \boldsymbol{\tau}^{P_i}_{\mathcal{B}\to\mathcal{P}_i} \cdot \mathbf{z}_{\mathcal{P}_i} + \boldsymbol{\tau}^{P_i}_{\text{air}\to\mathcal{P}_i} \cdot \mathbf{z}_{\mathcal{P}_i} \tag{2.53}$$

The motors are either brushed electric motor, such as on Parrot AR.Drone or the Parrot Mambo drones, or Brushless Direct Current (BLDC) motors such as on the Parrot Bebop 2 drone. An electric motor can be represented by the following electro-mechanical equations

$$\begin{cases} V_i = R_i\,I_i + L_i\,\dot{I}_i + k_{\omega_i}\,\omega_i \\ \boldsymbol{\tau}^{P_i}_{\mathcal{B}\to\mathcal{P}_i} \cdot \mathbf{z}_{\mathcal{P}_i} = k_{i_i}\,i_i - f_i\,\omega_i \end{cases} \tag{2.54}$$

with V_i the input voltage, I_i the input current, R_i the resistance, L_i the inductance, k_{ω_i} the counter electromotive force gain, k_{i_i} the electromotive torque gain and f_i the friction coefficient of the stator/rotor pivot link.

The GNC is not always in charge of controlling the speed of each propeller. Often, a separate low level controller called the Electronic Speed Control (ESC) is in charge of this task. In this case, the control stage of the GNC sends, as control input, rotor speeds or rotor thrust references that are then tracked by the ESC. The dynamics of the system {ESC + rotor} is often neglected in the synthesis of guidance or control algorithms, however it will be shown in section 2.6.3 that this dynamics can have significant impact on the dynamics of the drone, especially for some particular configurations of the propellers.

If the GNC is in charge of controlling the rotor speed, then it can be worth including the electric motor model into the drone model, as in [6]. Otherwise, the behavior of the dynamics of the controlled system {ESC + rotor} can for instance be approximated by a low pass filter, linking the propellers speed or thrust to their actual value [76], i.e.

$$\begin{aligned} &\dot{\omega}_i + f_{ci}\,\omega_i = \omega_{\text{ref}\,i} && \text{for a first order low pass filter} \\ &\ddot{\omega}_i + 2\xi\varpi_i\,\dot{\omega}_i + \varpi_i^2\,\omega_i = \varpi_i^2\,\omega_{\text{ref}\,i} && \text{for a second order low pass filter} \end{aligned}$$

These models are limited though, as they do not include the saturation of speed and acceleration of the propellers, as well as the quadratic nature of the drag they experience which tends to augment their characteristic response time at high speed, for instance.

2.4.3 Action of the surrounding air mass – $\mathfrak{F}_{\text{air}\to\mathcal{B}}$

The aerodynamic reaction of the air on the drone body is not trivial to model. This reaction can be divided into a drag (parasitic drag) in the same direction as the air speed and a lift produced by the drone body form factor. In forward flight, this lift can take the form of a down-force due to the attitude of the drone body (see figure 2.19). These actions depend on the air speed of the drone body but are also influenced by the airflow generated by the propellers. As a consequence it is not completely accurate to separate on the one hand the action of air on the drone body, and on the other hand its action on the propellers.

This action can still be approximated by a linear [116] or quadratic function of the speed [5], [81], or neglected at low speed as the propeller blade flapping and drag are predominant

(less than $10\text{m}\cdot\text{s}^{-1}$, [5])

$$\mathbf{f}_{\text{air}\to\mathcal{B}} = -\rho \left\|\mathbf{v}_{\mathcal{B}/\text{air}}\right\|_2 \mathbf{C}_{\text{par}}\, \mathbf{v}_{\mathcal{B}/\text{air}} \tag{2.55}$$

with ρ the air density and $\mathbf{C}_{\text{par}}$ a tensor depending on the air speed $\mathbf{v}_{\text{air}/\mathcal{B}}$ that act as lift and drag coefficients and can be identified in a wind tunnel for instance.

Furthermore, the surrounding air also resists any rotational movement, by generating a torque that can be approximated by a quadratic function

$$\boldsymbol{\tau}^{B}_{\text{air}\to\mathcal{B}} = -\rho \left\|\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}\right\|_2 \boldsymbol{\Gamma}_{\text{par}}\, \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \tag{2.56}$$

and the complete mechanical action of air on the drone body is

$$\mathfrak{F}_{\text{air}\to\mathcal{B}} = \left\{ \begin{matrix} \mathbf{f}_{\text{air}\to\mathcal{B}} \\ \boldsymbol{\tau}^{G}_{\text{air}\to\mathcal{B}} \end{matrix} \right\}_G \tag{2.57}$$

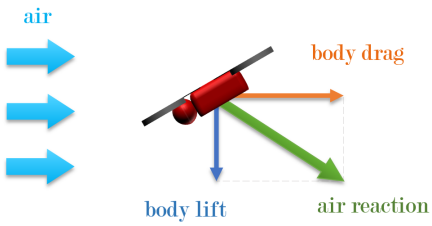

FIGURE 2.19 – Body lift acting as a down force in lateral flight

Remark 7 *Often, all the aerodynamics phenomena (both on the propellers and the drone body) are regrouped as one mechanical action. This will be done in section 2.5 when building the model of the entire drone.*

The FPD applied to the drone body (2.4) allows building a model of the drone body, in interaction with the propellers and the environment. However, the goal of this chapter is to build a model of the entire drone. Such a model can be deduced from the study of the drone body dynamics proposed in section 2.3 and the mechanical actions experienced by the drone body, presented in this section 2.4. This is the topic of the next section.

2.5 Equivalent full quadrotor system – $\mathfrak{S}_{\mathcal{D}}$

Fundamental Principle of Dynamics. Given the model of the mechanical actions experienced by the drone body, detailed in section 2.4, the FPD applied to the drone body (2.4) can expanded as follows

$$\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \sum_{i=1}^{N_{\text{p}}} \left(\mathfrak{F}_{\text{inertia}_i\to\mathcal{B}} + \mathfrak{F}_{\text{gyro}_i\to\mathcal{B}} + \mathfrak{F}_{\text{lever}_i\to\mathcal{B}} + \mathfrak{F}_{\text{grav}\to\mathcal{P}_i} + \mathfrak{F}_{\text{thrust}_i\to\mathcal{P}_i} + \mathfrak{F}_{\text{dist}_i\to\mathcal{P}_i} \right) + \mathfrak{F}_{\text{grav}\to\mathcal{B}} + \mathfrak{F}_{\text{air}\to\mathcal{B}} \tag{2.58}$$

It is not convenient to use a model of the drone body, interacting with the propellers as separate mechanical bodies. Indeed, most of the parameters involved in the model are

identified on the drone *with* its propellers (center of mass, inertia, aerodynamics coefficients, thrust coefficients, etc.). Furthermore, the trajectory of the entire drone (position of the center of mass and attitude) matters for the GNC rather than the one of the drone body. Equation (2.58) can thus be rewritten to describe the entire drone as one equivalent mechanical system {drone body + propellers}, denoted by $\mathfrak{S}_\mathcal{D}$

$$\underbrace{\mathfrak{D}_{\mathcal{B}/\mathcal{W}} - \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\text{lever}_i \to \mathcal{B}}}_{\mathfrak{D}_{\mathcal{D}/\mathcal{W}}} = \underbrace{\mathfrak{F}_{\text{grav}\to\mathcal{B}} + \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\text{grav}\to\mathcal{P}_i}}_{\mathfrak{F}_{\text{grav}\to\mathcal{D}}} + \underbrace{\mathfrak{F}_{\text{air}\to\mathcal{B}} + \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\text{dist}_i\to\mathcal{P}_i}}_{\mathfrak{F}_{\text{air}\to\mathcal{D}}} + \sum_{i=1}^{N_\text{p}} \underbrace{\left(\mathfrak{F}_{\text{inertia}_i\to\mathcal{B}} + \mathfrak{F}_{\text{gyro}_i\to\mathcal{B}} + \mathfrak{F}_{\text{thrust}_i\to\mathcal{P}_i}\right)}_{\mathfrak{F}_{\mathcal{P}_i\to\mathcal{D}}} \tag{2.59}$$

We now detail each term in this expression. Particularly, we will identify the mass, the center of mass and the inertia of this equivalent solid $\mathfrak{S}_\mathcal{D}$, such that the left terms in (2.59) correspond to its dynamic tensor. We start by focusing on the total action of gravity of the system and the identification of the mass and the center of mass of the entire drone.

Drone gravity. From (2.34) and (2.45), the term $\mathfrak{F}_{\text{grav}\to\mathcal{D}}$ is given by

$$\mathfrak{F}_{\text{grav}\to\mathcal{D}} \triangleq \mathfrak{F}_{\text{grav}\to\mathcal{B}} + \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\text{grav}\to\mathcal{P}_i} = \left\{ \begin{array}{c} \left(m_\mathcal{B} + \sum_{i=1}^{N_\text{p}} m_i\right) g\, \mathbf{z}_\mathcal{W} \\ g\left(m_\mathcal{B}\, \overrightarrow{GB} + \sum_{i=1}^{N_\text{p}} m_i\, \overrightarrow{GP_i}\right) \times \mathbf{z}_\mathcal{W} \end{array} \right\}_G \tag{2.60}$$

Defining

$$m = m_\mathcal{B} + \sum_{i=1}^{N_\text{p}} m_i \tag{2.61a}$$

$$\overrightarrow{BG} = \frac{1}{m} \sum_{i=1}^{N_\text{p}} m_i\, \overrightarrow{BP_i} \tag{2.61b}$$

the 2 terms of the moment of $\mathfrak{F}_{\text{grav}\to\mathcal{D}}$ at G cancel each other and we get the action of gravity on the entire drone, as a force applied on its overall center of mass

$$\mathfrak{F}_{\text{grav}\to\mathcal{D}} = \left\{ \begin{array}{c} m g\, \mathbf{z}_\mathcal{W} \\ \mathbf{0} \end{array} \right\}_G \tag{2.62}$$

We now focus on the left term of (2.59) to identify the inertia of the equivalent entire drone system.

Drone dynamic torsor. From equations (2.13) and (2.44d), the term $\mathfrak{D}_{\mathcal{D}/\mathcal{W}}$ is given by

$$\mathfrak{D}_{\mathcal{D}/\mathcal{W}} \triangleq \mathfrak{D}_{\mathcal{B}/\mathcal{W}} - \sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\text{lever}_i\to\mathcal{B}} = \left\{ \begin{array}{c} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\text{p}} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \\ \boldsymbol{\delta}^B_{\mathcal{B}/\mathcal{W}} + \overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\text{p}} \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \end{array} \right\}_G \tag{2.63}$$

In the following, we seek to show that the resultant and the moment at G of this torsor can be written in a similar fashion as in (2.6) and (2.7), by using the the obtained expression of the mass m and the center of mass G of the equivalent drone system, as well as its inertia tensor $\mathbf{J}_{\mathcal{D}/G}$, that we will identify in the process.

We start with the resultant of the torsor (2.63). Knowing that $\overrightarrow{OG} = \overrightarrow{OB} + \overrightarrow{BG}$, we have

$$\mathbf{v}_{G/\mathcal{W}} = \mathbf{v}_{B/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BG} \tag{2.64}$$

and

$$\mathbf{a}_{G/\mathcal{W}} = \mathbf{a}_{B/\mathcal{W}} + \boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}} \overrightarrow{BG} + \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BG} \tag{2.65}$$

Using equation (2.61b), we thus have

$$\mathbf{a}_{G/\mathcal{W}} = \mathbf{a}_{B/\mathcal{W}} + \frac{1}{m} \sum_{i=1}^{N_\mathrm{p}} m_i \left[\boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}} \overrightarrow{BG} + \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BG} \right] \tag{2.66}$$

Multiplying this expression by m and identifying the obtained terms with equations (2.6) and (2.43) leads to

$$\boldsymbol{\mu}_{\mathcal{D}/\mathcal{W}} \triangleq \frac{\mathrm{d}}{\mathrm{d}t}\left(m\, \mathbf{v}_{G/\mathcal{W}}\right) = \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \tag{2.67}$$

and the resultant of the the torsor (2.63) corresponds to the dynamic resultant of a rigid body of mass m and center of mass m.

We now focus on the moment at G of the torsor (2.63). Since $\overrightarrow{GP_i} = \overrightarrow{GB} + \overrightarrow{BP_i}$, the two right terms in the expression of the moment of (2.63) can be rewritten

$$\overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} = -\overrightarrow{BG} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} - \overrightarrow{BG} \times \sum_{i=1}^{N_\mathrm{p}} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \overrightarrow{BP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \tag{2.68}$$

By replacing the dynamic resultant of the drone body $\boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}}$ and of the dynamic resultant of propellers $\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}$ by their expressions (2.43), (2.6), this equation becomes

$$\begin{aligned}
\overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} &= -\overrightarrow{BG} \times m_\mathcal{B}\, \mathbf{a}_{B/\mathcal{W}} \\
&\quad - \overrightarrow{BG} \times \sum_{i=1}^{N_\mathrm{p}} m_i\, \mathbf{a}_{B/\mathcal{W}} - \overrightarrow{BG} \times \boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}} \sum_{i=1}^{N_\mathrm{p}} m_i \overrightarrow{BP_i} - \overrightarrow{BG} \times \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i} \\
&\quad + \sum_{i=1}^{N_\mathrm{p}} m_i \overrightarrow{BP_i} \times \mathbf{a}_{B/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} m_i \overrightarrow{BP_i} \times \boldsymbol{\Omega}^{\times\times}_{\mathcal{B}/\mathcal{W}} \overrightarrow{BP_i} + \sum_{i=1}^{N_\mathrm{p}} m_i \overrightarrow{BP_i} \times \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}} \times \overrightarrow{BP_i}
\end{aligned} \tag{2.69}$$

Using (2.61b) and the fact that, for two vectors $\mathbf{u} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$, $\mathbf{u} \times \mathbf{v}^{\times\times}\mathbf{u} = -\mathbf{v} \times \mathbf{u}^{\times\times}\mathbf{v}$, expression (2.69) can be simplified as follows

$$\begin{aligned}
\overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} = \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \left(-m\, \overrightarrow{BG}^{\times\times} - \sum_{i=1}^{N_\mathrm{p}} m_i\, \overrightarrow{BP_i}^{\times\times} \right) \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \\
+ \left(-m\, \overrightarrow{BG}^{\times\times} - \sum_{i=1}^{N_\mathrm{p}} m_i\, \overrightarrow{BP_i}^{\times\times} \right) \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}}
\end{aligned} \tag{2.70}$$

Defining

$$\mathbf{J}_{\mathcal{D}/G} = \mathbf{J}_{\mathcal{B}/B} - m\,\overrightarrow{BG}^{\times\times} - \sum_{i=1}^{N_\mathrm{p}} m_i\,\overrightarrow{BP_i}^{\times\times} \tag{2.71}$$

and using both (2.7) and (2.70) we can finally rewrite (2.70)

$$\boldsymbol{\delta}^B_{\mathcal{B}/\mathcal{W}} + \overrightarrow{GB} \times \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} + \sum_{i=1}^{N_\mathrm{p}} \overrightarrow{GP_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} = \mathbf{J}_{\mathcal{D}/G}\,\dot{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} \times \mathbf{J}_{\mathcal{D}/G}\,\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} \tag{2.72}$$

We obtained the desired expression

$$\mathfrak{D}_{\mathcal{D}/\mathcal{W}} = \begin{Bmatrix} m\,\mathbf{a}_{G/\mathcal{W}} \\ \mathbf{J}_{\mathcal{D}/G}\,\dot{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{W}} + \boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} \times \mathbf{J}_{\mathcal{D}/G}\,\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} \end{Bmatrix}_G = \begin{Bmatrix} \boldsymbol{\mu}_{\mathcal{D}/\mathcal{W}} \\ \boldsymbol{\delta}^G_{\mathcal{D}/\mathcal{W}} \end{Bmatrix}_G \tag{2.73}$$

with $\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} = \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$. The expression of $\mathbf{J}_{\mathcal{D}/G}$ corresponds to the inertia obtained with the Huygens formula for a system consisting of the drone body along with each propeller assimilated to point masses. The torsor (2.73) corresponds to a dynamic torsor, allowing to define an equivalent system {drone body + rotors}.

Drone system. We define the equivalent, fictional rigid solid *drone*, $\mathfrak{S}_\mathcal{D}$, constituted of the drone body and the propellers, the latter being assimilated to N_p punctual masses m_i, located at P_i, with no inertia. This system is of mass m, center of mass G and inertia $\mathbf{J}_{\mathcal{D}/G}$, defined by (2.61a), (2.61b) and (2.71), respectively.

To this solid is attached the frame $\mathfrak{R}_\mathcal{D}$ of origin G and basis $\mathfrak{B}_\mathcal{D}$. The latter can be chosen to be principal of inertia for instance.

The FPD applied to the drone $\mathfrak{S}_\mathcal{D}$ is given by equation (2.59)

$$\mathfrak{D}_{\mathcal{D}/\mathcal{W}} = \mathfrak{F}_{\mathrm{grav}\to\mathcal{D}} + \mathfrak{F}_{\mathrm{air}\to\mathcal{D}} + \sum_{i=1}^{N_\mathrm{p}} \mathfrak{F}_{\mathcal{P}_i\to\mathcal{D}} \tag{2.74}$$

Simplified nonlinear model. For guidance typically, the mechanical actions applied on the drone by the propellers are directly used as input for the model (i.e. the force and the torque they apply on the drone). As a consequence, neither the inertial actions of the rotors (reaction and gyroscopic torques) nor their aerodynamic model appear in the drone model. Thus, the expression (2.74) can be written as the following nonlinear model

$$\begin{cases} \ddot{\boldsymbol{\zeta}} = \mathbf{g} + u_1\,\mathbf{R}_{\mathcal{W}\to\mathcal{D}}\,\mathbf{z}_\mathcal{W} + \mathbf{f}_\mathrm{dist} \\ \dot{\mathbf{R}}_{\mathcal{W}\to\mathcal{B}} = \mathbf{R}_{\mathcal{W}\to\mathcal{B}}\,\hat{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{W}} \\ \mathbf{J}_{\mathcal{D}/G}\,\dot{\boldsymbol{\Omega}}_{\mathcal{D}/\mathcal{W}} = u_2\,\mathbf{x}_\mathcal{D} + u_3\,\mathbf{y}_\mathcal{D} + u_4\,\mathbf{z}_\mathcal{D} - \boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} \times \mathbf{J}_{\mathcal{D}/G}\,\boldsymbol{\Omega}_{\mathcal{D}/\mathcal{W}} + \boldsymbol{\tau}_\mathrm{dist} \end{cases} \tag{2.75}$$

with u_1, u_2, u_3 and u_4 the inputs of the model, and $\mathbf{f}_\mathrm{dist}$ and $\boldsymbol{\tau}_\mathrm{dist}$ disturbances that can be considered or not into the model. A model of this kind has lead to the design of popular control laws, without disturbances ([66]) or with disturbances ([64], [84]) as well as guidance algorithms ([75]) since the outputs of the drone can easily be linked to the inputs of this model through a flatness analysis.

2.6 Linear model near hovering

Often, ESC are used to control the speed of each propeller. These rotor speeds thus constitute good control input candidates for the control stage of the GNC system. It it then reasonable to use a model that incorporates the dynamics of the propellers for the synthesis of a control law. By doing so, it appears that the inertia of the rotors can have a significant impact on the attitude dynamics of the drone. This section emphasizes several dynamical aspects of a quadrotor induced by propellers inertia, especially when its rotors are tilted and/or asymmetrically distributed around its body, as for the Bebop 2. To achieve this, we derive a linear model from the one obtained in section 2.5 and apply it to different configurations of the propellers.

2.6.1 Linear model

For small pitch and roll angles as well as small speeds and angular speeds, the FPD-based model (2.74) can be linearized near the hovering equilibrium. We first detail the linearization of each term of the model, for a ZYX Euler angles representation of the drone attitude.

Drone dynamics. We suppose the vector basis of the drone $\mathfrak{B}_{\mathcal{D}}$ principal of inertia, and thus its inertia matrix in the drone vector basis diagonal

$$\mathrm{J}_{\mathcal{D}/G}^{\mathcal{D}} = \begin{pmatrix} J_x & & \\ & J_y & \\ & & J_z \end{pmatrix} \tag{2.76}$$

Then, the linearization of the dynamic torsor of the drone near the hovering equilibrium gives

$$\mathfrak{D}_{\mathcal{D}/\mathcal{W}} \approx \left\{ \begin{pmatrix} m\,\dot{u} \\ m\,\dot{v} \\ m\,\dot{w} \end{pmatrix}_{\mathcal{D}} \right\}_{G} \left\{ \begin{pmatrix} J_x\,\dot{p} \\ J_y\,\dot{q} \\ J_z\,\dot{r} \end{pmatrix}_{\mathcal{D}} \right\}_{G} \tag{2.77}$$

Gravity The expression of the action of gravity is linearized around the hovering equilibrium, leading to

$$\mathfrak{F}_{\mathrm{grav}\to\mathcal{D}} \approx \left\{ \begin{pmatrix} -mg\,\theta \\ mg\,\varphi \\ mg \end{pmatrix}_{\mathcal{D}} \right\}_{G} \left\{ \begin{pmatrix} \mathbf{0} \end{pmatrix} \right\}_{G} \tag{2.78}$$

Air reaction The action of the surrounding air on the drone (parasitic drag, blade flapping, etc.) is approximated by a linear model near the hovering equilibrium

$$\mathfrak{F}_{\text{air}\to\mathcal{D}} \approx \left\{ \begin{pmatrix} -c_u\, u \\ -c_v\, v \\ -c_w\, w \end{pmatrix}_{\mathcal{D}} \right\}_G \qquad \left\{ \begin{pmatrix} -c_p\, p \\ -c_q\, q \\ -c_r\, r \end{pmatrix}_{\mathcal{D}} \right\}_G \tag{2.79}$$

with c_u, c_v, c_w, c_p, c_q, c_r scalar constants.

Propellers By denoting ω_{hi} the speed of the i-th propeller when hovering, we introduce the rotation speed increment $\delta\omega_i$ such that the speed of this propeller near the hovering equilibrium can be expressed

$$\omega_i = \omega_{hi} + \delta\omega_i \tag{2.80}$$

with $\delta\omega_i \ll \omega_{hi}$. Supposing the air density invariant during the flight, the thrust and torque of the propellers (2.48) can be simplified near the hovering equilibrium

$$\begin{aligned} \mathbf{t}_i &= \left(\alpha_{0i} + \alpha_{1i}\,\omega_i + \alpha_{2i}\,\omega_i^2\right) \mathbf{z}_{\mathcal{P}_i}, \\ \boldsymbol{\tau}_i &= \left(\beta_{0i} + \beta_{1i}\,\omega_i + \beta_{2i}\,\omega_i^2\right) \mathbf{z}_{\mathcal{P}_i} \end{aligned} \tag{2.81}$$

with α_{0i}, α_{1i}, α_{2i}, β_{0i}, β_{1i} and β_{2i} scalar constants. The linearization of this quadratic thrust model near the hovering equilibrium gives

$$\mathfrak{F}_{\text{thrust}_i\to\mathcal{D}} \approx \left\{ \begin{matrix} (t_{hi} + a_i\,\delta\omega_i)\,\mathbf{z}_{\mathcal{P}_i} \\ -(\tau_{hi} + b_i\,\delta\omega_i)\,\mathbf{z}_{\mathcal{P}_i} \end{matrix} \right\}_{P_i} \tag{2.82a}$$

with

$$\begin{aligned} t_{hi} &= \alpha_{0i} + \alpha_{1i}\,\omega_{hi} + \alpha_{2i}\,{\omega_{hi}}^2 \\ a_i &= \alpha_{1i} + 2\,\alpha_{2i}\,\omega_{hi} \\ \tau_{hi} &= \beta_{0i} + \beta_{1i}\,\omega_{hi} + \beta_{2i}\,{\omega_{hi}}^2 \\ b_i &= \beta_{1i} + 2\,\beta_{2i}\,\omega_{hi} \end{aligned} \tag{2.82b}$$

The reaction torque of the i-th propeller given by (2.44b) has a linear form

$$\mathfrak{F}_{\text{inertia}_i\to\mathcal{D}} \approx \left\{ \begin{matrix} \mathbf{0} \\ -J_{zi}\,\delta\dot{\omega}_i\,\mathbf{z}_{\mathcal{P}_i} \end{matrix} \right\}_G \tag{2.83}$$

with J_{zi} the inertia of the propeller along its rotation axis. Since the rotation speed of the drone relatively to the ground has been considered small, the gyroscopic torques are neglected.

Given the parameterization of $\mathbf{z}_{\mathcal{P}_i}$ and $\overrightarrow{GP_i}$ in the drone fixed basis (2.37), the sum of the actions of all the propellers can be expressed as follows

$$\sum_{i=1}^{N_\text{p}} \mathfrak{F}_{\mathcal{P}_i\to\mathcal{D}} = \sum_{i=1}^{N_\text{p}} \left(\mathfrak{F}_{\text{thrust}_i\to\mathcal{D}} + \mathfrak{F}_{\text{inertia}_i\to\mathcal{D}}\right) = \left\{ \begin{matrix} \sum_{i=1}^{N_\text{p}} t_{hi}\,\mathbf{z}_{\mathcal{P}_i} \\ \sum_{i=1}^{N_\text{p}} \left(\tau_{hi} + t_{hi}\,\overrightarrow{GP_i} \times \mathbf{z}_{\mathcal{P}_i}\right) \end{matrix} \right\}_G + \left\{ \begin{matrix} \begin{pmatrix} u_u \\ u_v \\ u_w \end{pmatrix}_{\mathcal{D}} \\ \begin{pmatrix} u_p \\ u_q \\ u_r \end{pmatrix}_{\mathcal{D}} \end{matrix} \right\}_G \tag{2.84}$$

with $(u_u, u_v, u_w, u_p, u_q, u_r) \in \mathbb{R}^6$ obtained using (2.48)

$$\begin{pmatrix} u_u \\ u_v \\ u_w \\ u_p \\ u_q \\ u_r \end{pmatrix} = \mathrm{B}_{\omega \to u} \cdot \begin{pmatrix} \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \\ \delta\omega_4 \end{pmatrix} + \mathrm{B}_{\dot{\omega} \to u} \cdot \begin{pmatrix} \delta\dot{\omega}_1 \\ \delta\dot{\omega}_2 \\ \delta\dot{\omega}_3 \\ \delta\dot{\omega}_4 \end{pmatrix} \tag{2.85a}$$

and

$$\mathrm{B}_{\omega \to u} = \begin{pmatrix} B_1 & B_2 & B_3 & B_4 \end{pmatrix}, \quad \mathrm{B}_{\dot{\omega} \to u} = \begin{pmatrix} C_1 & C_2 & C_3 & C_4 \end{pmatrix}$$

$$\forall i \in [\![1, N_\mathrm{p}]\!] \quad B_i = \begin{pmatrix} a_i\, s_{\theta_i} c_{\psi_i} \\ a_i\, s_{\theta_i} s_{\psi_i} \\ a_i\, c_{\theta_i} \\ -b_i\, s_{\theta_i} c_{\psi_i} + a_i \left(l_{y_i}\, c_{\theta_i} - l_{zi}\, s_{\theta_i} s_{\psi_i} \right) \\ -b_i\, s_{\theta_i} s_{\psi_i} + a_i \left(l_{zi}\, s_{\theta_i} c_{\psi_i} - l_{xi}\, c_{\theta_i} \right) \\ -b_i\, c_{\theta_i} + a_i\, s_{\theta_i} \left(l_{xi}\, s_{\psi_i} - l_{y_i}\, c_{\psi_i} \right) \end{pmatrix}, \quad C_i = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -J_{zi}\, s_{\theta_i} c_{\psi_i} \\ -J_{zi}\, s_{\theta_i} s_{\psi_i} \\ -J_{zi}\, c_{\theta_i} \end{pmatrix} \tag{2.85b}$$

Fundamental Principle of Dynamics The application of the FPD (2.74) at the hovering equilibrium then gives

$$\begin{aligned} \mathbf{0} &= m g\, \mathbf{z}_W + \sum_{i=1}^{4} t_{hi}\, \mathbf{z}_{\mathcal{P}_i} \\ \mathbf{0} &= \sum_{i=1}^{4} \left(\tau_{hi} + t_{hi}\, \overrightarrow{GP_i} \times \mathbf{z}_{\mathcal{P}_i} \right) \end{aligned} \tag{2.86}$$

Its linearization around the hovering equilibrium gives, in the drone fixed basis $\mathfrak{B}_\mathcal{D}$

$$\begin{aligned} m\, \dot{u} &= -m g\, \theta - c_u\, u + u_u, & J_x\, \dot{p} &= -c_p\, p + u_p, \\ m\, \dot{v} &= m g\, \varphi - c_v\, v + u_v, & J_x\, \dot{q} &= -c_q\, q + u_q, \\ m\, \dot{w} &= -c_w\, w + u_w, & J_x\, \dot{r} &= -c_r\, r + u_r \end{aligned} \tag{2.87}$$

Ideally, each of the 6 DOF of the quadrotor would be decoupled and controlled independently. However, with only 4 propellers, the drone is underactuated and this is not possible. Nevertheless, it is usually possible to decouple the action of the propellers on the three rotation axes and the z translation axis. One way to achieve this is to introduce the matrix

$$\mathrm{B}_{\omega \to v} = \begin{pmatrix} \mathbf{0}_{4\times 2} & \mathbf{I}_4 \end{pmatrix} \cdot \mathrm{B}_{\omega \to u} \tag{2.88}$$

and the decoupled control inputs $(v_w, v_p, v_q, v_r) \in \mathbb{R}^4$

$$\begin{pmatrix} v_w \\ v_p \\ v_q \\ v_r \end{pmatrix} = \mathrm{B}_{\omega \to v} \cdot \begin{pmatrix} \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \\ \delta\omega_4 \end{pmatrix} \tag{2.89}$$

We then define the *mix* matrix

$$\mathrm{B}_{v \to \omega} = \left(\mathrm{B}_{\omega \to v} \right)^{-1} \tag{2.90}$$

defining the speed increment required for each rotor to achieve a given force on the drone z axis, and given torques on the 3 drone axes. It can then be written

$$\begin{pmatrix} u_u \\ u_v \\ u_w \\ u_p \\ u_q \\ u_r \end{pmatrix} = \mathrm{B}_{\omega \to u} \cdot \mathrm{B}_{v \to \omega} \cdot \begin{pmatrix} v_w \\ v_p \\ v_q \\ v_r \end{pmatrix} + \mathrm{B}_{\dot{\omega} \to u} \cdot \mathrm{B}_{v \to \omega} \cdot \begin{pmatrix} \dot{v}_w \\ \dot{v}_p \\ \dot{v}_q \\ \dot{v}_r \end{pmatrix} \tag{2.91}$$

This relation will help us highlighting the impact of the configuration of the propellers in the next sections.

Motor model Each brushless motor is controlled by a ESC which is assumed to confer the motors a second order low-pass filter behavior near the hovering equilibrium state

$$\delta\ddot{\omega}_i + 2\xi_i \varpi_i \, \delta\dot{\omega}_i + \varpi_i^2 \, \delta\omega_i = \varpi_i^2 \, \delta\omega_{\mathrm{ref}\, i} \tag{2.92}$$

In the next section, we apply this linear model to the case of an X4 quadrotor.

2.6.2 X4 quadrotor

The most common quadrotor configuration is called the X4 configuration and is represented on figure 2.8. It is composed of four identical fan propellers, symmetrically disposed around the center of mass of the drone, alternating clockwise and anti-clockwise directions of rotation and pointing toward the vertical axis of the drone. The plans $(G, \mathbf{x}_\mathcal{D}, \mathbf{y}_\mathcal{D})$ and $(G, \mathbf{x}_\mathcal{D}, \mathbf{z}_\mathcal{D})$ are symmetry plans of the quadrotor and the 4 propellers are oriented toward $\pm\mathbf{z}_\mathcal{D}$.

If the propeller 1 rotates Counterclockwise (CCW configuration, as the Parrot Bebop 2), the axes of the propellers are given by

$$\mathbf{z}_{\mathcal{P}_i} = -\mathbf{z}_\mathcal{D}, \;\; \mathbf{z}_{\mathcal{P}_i} = \mathbf{z}_\mathcal{D}, \;\; \mathbf{z}_{\mathcal{P}_i} = -\mathbf{z}_\mathcal{D}, \;\; \mathbf{z}_{\mathcal{P}_i} = \mathbf{z}_\mathcal{D} \tag{2.93}$$

Otherwise, if the propeller 1 rotates Clockwise (CW configuration, Parrot Bebop 1), then the signs should be inverted. In the following, we focus on the CCW configuration as it corresponds to the Parrot Bebop 2. For an X4-CCW, the propellers rotate along the axes defined in (2.93) and the angular velocity of each propeller relatively to the drone is then

$$\boldsymbol{\Omega}_{\mathcal{P}_1/\mathcal{D}} = -\omega_1 \, \mathbf{z}_\mathcal{D} \;\; \boldsymbol{\Omega}_{\mathcal{P}_2/\mathcal{D}} = \omega_2 \, \mathbf{z}_\mathcal{D} \;\; \boldsymbol{\Omega}_{\mathcal{P}_3/\mathcal{D}} = -\omega_3 \, \mathbf{z}_\mathcal{D} \;\; \boldsymbol{\Omega}_{\mathcal{P}_4/\mathcal{D}} = \omega_4 \, \mathbf{z}_\mathcal{D} \tag{2.94}$$

The propellers poses relatively to the drone are parameterized as follows

$$\begin{pmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \\ \theta_1 & \theta_2 & \theta_3 & \theta_4 \\ l_{x1} & l_{x2} & l_{x3} & l_{x4} \\ l_{y1} & l_{y2} & l_{y3} & l_{y4} \\ l_{z1} & l_{z2} & l_{z3} & l_{z4} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \pi & 0 & \pi & 0 \\ l_x & l_x & -l_x & -l_x \\ -l_y & l_y & l_y & -l_y \\ l_z & l_z & l_z & l_z \end{pmatrix} \tag{2.95}$$

with $l_x > 0$ and $l_y > 0$. Since the 4 propellers are identical, we also have

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ J_{z1} & J_{z2} & J_{z3} & J_{z4} \end{pmatrix} = \begin{pmatrix} a & -a & a & -a \\ b & b & b & b \\ J_p & J_p & J_p & J_p \end{pmatrix} \tag{2.96}$$

This results in the following expressions for $\mathrm{B}_{\omega\to u}$ and $\mathrm{B}_{\dot{\omega}\to u}$

$$\mathrm{B}_{\omega\to u} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -a & -a & -a & -a \\ l_y\, a & -l_y\, a & -l_y\, a & l_y\, a \\ l_x\, a & l_x\, a & -l_x\, a & -l_x\, a \\ b & -b & b & -b \end{pmatrix}, \quad \mathrm{B}_{\dot{\omega}\to u} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ J_p & -J_p & J_p & -J_p \end{pmatrix} \tag{2.97}$$

The propellers actions on the x and y translation axes are null and their action on the z translation axis and the three rotation axes are fully decoupled

$$\mathrm{B}_{\omega\to u} \cdot \mathrm{B}_{v\to\omega} = \begin{pmatrix} \mathbf{0}_{2\times 4} \\ \mathbf{I}_4 \end{pmatrix}, \quad \mathrm{B}_{\dot{\omega}\to u} \cdot \mathrm{B}_{v\to\omega} = \begin{pmatrix} \mathbf{0}_{5\times 3} & \mathbf{0}_{5\times 1} \\ \mathbf{0}_{1\times 3} & \frac{J_p}{b} \end{pmatrix} \tag{2.98}$$

and we can write

$$\begin{aligned} &u_u(s) = 0, && u_v(s) = 0, && u_w(s) = v_w(s), \\ &u_p(s) = v_p(s), && u_q(s) = v_q(s), && u_r(s) = \left(1 + \frac{J_p}{b}\, s\right) v_r(s), \end{aligned} \tag{2.99}$$

with s the Laplace variable. Injecting these expressions into (2.87) gives

$$\begin{aligned} w(s) &= \frac{1}{c_z + m\, s}\, v_w(s), \quad p(s) = \frac{1}{c_p + J_x\, s}\, v_p(s), \\ q(s) &= \frac{1}{c_q + J_y\, s}\, v_q(s), \quad r(s) = \frac{1 + \frac{J_p}{b}\, s}{c_r + J_z\, s}\, v_r(s), \end{aligned} \tag{2.100a}$$

and

$$\begin{aligned} u(s) &= \frac{-m\, g}{s\,(c_q + J_y\, s)(c_v + m\, s)}\, v_q(s) \\ v(s) &= \frac{m\, g}{s\,(c_p + J_x\, s)(c_u + m\, s)}\, v_p(s) \end{aligned} \tag{2.100b}$$

The inertia of the propellers induces a zero in the yaw dynamics. The latter is usually significant, and allows reducing and smoothing the control signals on this axis during dynamic flight phases. This also means that a drone model taking the rotation speed of the propellers as inputs should include the dynamics of the electric motor or the motor controlled by the ESC, as they act as low-pass filters. Otherwise the transfer of the drone rotation speed on the yaw axis would not be strictly proper anymore. Indeed, any discontinuity of the propeller rotation speeds would lead to an infinite inertia torque on the yaw axis and would allow a discontinuous behavior of the yaw axis, which contradicts reality and degrades the fidelity of the model.

Usually, industrial drone are not exactly in X4 configuration. Indeed, the propellers are often tilted in order to improve the actuation on the yaw axis. In the next section, we study the impact of tilting the propellers in the dynamics of the translational and rotational axes. Inn particular, we focus on the effects due to the inertia of the propellers.

2.6.3 V4 quadrotor

On an X4 configuration, the rotation on the yaw axis is controlled using the propellers drag and the reaction torques on their axis. These tend to be one or more orders of

magnitude smaller than the torques induced by the propellers lift on the roll and pitch axes. The pitch and roll axes are critical as they are used to control the lateral position of the drone. However, in the case of the Bebop 2, the yaw axis of the drone is used to actuate the yaw axis of its camera, fixed on the drone body. As a consequence, the yaw axis is also critical, for the quality of the video.

A solution to balance these dynamics is to tilt the propeller axes in order to have a leverage induced by the lift on the yaw axis, too. This can be achieved by actuating the orientation of the propellers, as in [82], or by using non-X4 configurations. One way to improve a quadrotor actuation on its yaw axis is then to tilt its propellers in a "V" shape, as illustrated on figure 2.20. Sometimes only the rear propellers are tilted (called a *V-tail* or Y4 configuration). Such an Y4 configuration is studied in [54], and a Lynxmotion Hunter V400 frame, a popular V-tail frame among hobbyists, is used in [9].

In the following, we suppose that both the front and rear propellers are tilted as on figure 2.20. We call this configuration the *V4 configuration*.

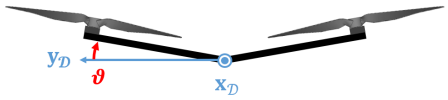

FIGURE 2.20 – V4 configuration

The four propellers are supposed identical. The pose of the propellers is similar to (2.95) with the following difference

$$\begin{pmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \\ \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{pmatrix} = \begin{pmatrix} \pi/2 & \pi/2 & -\pi/2 & -\pi/2 \\ \pi-\vartheta & \vartheta & \pi-\vartheta & \vartheta \end{pmatrix} \tag{2.101}$$

with $\vartheta \in \left[0, \frac{\pi}{2}\right]$ the V-angle, illustrated on figure 2.20. Such a configuration leads to the following $\mathrm{B}_{\omega \to u}$ matrix

$$\mathrm{B}_{\omega \to u} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ a\,s_\vartheta & -a\,s_\vartheta & -a\,s_\vartheta & a\,s_\vartheta \\ -a\,c_\vartheta & -a\,c_\vartheta & -a\,c_\vartheta & -a\,c_\vartheta \\ a\,(l_x c_\vartheta - l_z s_\vartheta) & -a\,(l_x c_\vartheta - l_z s_\vartheta) & -a\,(l_x c_\vartheta - l_z s_\vartheta) & a\,(l_x c_\vartheta - l_z s_\vartheta) \\ -b\,s_\vartheta + a\,l_x c_\vartheta & -b\,s_\vartheta + a\,l_x c_\vartheta & b\,s_\vartheta - a\,l_x c_\vartheta & b\,s_\vartheta - a\,l_x c_\vartheta \\ b\,c_\vartheta + a\,l_x s_\vartheta & -b\,c_\vartheta - a\,l_x s_\vartheta & b\,c_\vartheta + a\,l_x s_\vartheta & -b\,c_\vartheta - a\,l_x s_\vartheta \end{pmatrix} \tag{2.102}$$

A part of the propellers lift is now added to their drag torques on the yaw axis, improving the actuation of the drone on this axis. However, these drag torques also appear on the pitch axis now, and counter a part of the propellers lift, degrading the actuation on this axis.

Inversing the direction of rotation of the propellers, i.e. using a V4-CW configuration leads to

$$\mathrm{B}_{\omega \to u} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ a\,s_\vartheta & -a\,s_\vartheta & -a\,s_\vartheta & a\,s_\vartheta \\ -a\,c_\vartheta & -a\,c_\vartheta & -a\,c_\vartheta & -a\,c_\vartheta \\ a\,(l_x c_\vartheta - l_z s_\vartheta) & -a\,(l_x c_\vartheta - l_z s_\vartheta) & -a\,(l_x c_\vartheta - l_z s_\vartheta) & a\,(l_x c_\vartheta - l_z s_\vartheta) \\ b\,s_\vartheta + a\,l_x c_\vartheta & b\,s_\vartheta + a\,l_x c_\vartheta & -b\,s_\vartheta - a\,l_x c_\vartheta & -b\,s_\vartheta - a\,l_x c_\vartheta \\ b\,c_\vartheta - a\,l_x s_\vartheta & -b\,c_\vartheta + a\,l_x s_\vartheta & b\,c_\vartheta - a\,l_x s_\vartheta & -b\,c_\vartheta + a\,l_x s_\vartheta \end{pmatrix} \tag{2.103}$$

The propellers lift now counters the propellers drag on the yaw axis, degrading the yaw dynamics. Since the propellers have been tilted in order to increase the yaw dynamics, this

solution is not relevant here. For this reason, hereafter, we keep the V4-CCW configuration for the propellers.

Using the expressions

$$\begin{aligned} \mathrm{B}_{\omega\to u}\cdot\mathrm{B}_{v\to\omega} &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & k_{pv} & 0 & 0 \\ \hdashline & & \mathbf{I}_4 & \end{pmatrix} \\ \mathrm{B}_{\dot\omega\to u}\cdot\mathrm{B}_{v\to\omega} &= \begin{pmatrix} & & \mathbf{0}_4 & \\ \hdashline 0 & 0 & -\tau_q & 0 \\ 0 & 0 & 0 & \tau_r \end{pmatrix} \end{aligned} \tag{2.104a}$$

with

$$\begin{aligned} k_{pv} &= \frac{s_\vartheta}{l_y c_\vartheta - l_z s_\vartheta} \\ \tau_q &= \frac{J_p\, s_\vartheta}{a\, l_x c_\vartheta - b\, s_\vartheta} \\ \tau_r &= \frac{J_p\, c_\vartheta}{b\, c_\vartheta + a\, l_x s_\vartheta} \end{aligned} \tag{2.104b}$$

This leads to the following transfer functions

$$\begin{array}{lll} u_u(s) = 0, & u_v(s) = k_{pv}\, v_p(s), & u_w(s) = v_w(s), \\ u_p(s) = v_p(s), & u_q(s) = (1-\tau_q\, s)\, v_q(s), & u_r(s) = (1+\tau_r\, s)\, v_r(s), \end{array} \tag{2.105}$$

The differences with respect to (2.99) are the following: u_v and u_p are now coupled since both depends on v_v and a zero appears on the pitch axis, as a part of the torques induced by the inertia of the propellers. For realistic values of the V-angle ϑ, thrust coefficients a, b and distance l_x, the propellers lift action on the pitch axis is still much stronger than the action of the propellers drag torques, and $a\, l_x c_\vartheta > b\, s_\vartheta$. This zero on the pitch axis is thus a positive real scalar, inducing a non-minimum phase behavior on the pitch axis. However, this is usually a high frequency zero that only affects the very beginning of the transient response of the pitch axis.

These additional terms in the transfer functions (2.105) lead to the following drone dynamics

$$\begin{aligned} w &= \frac{1}{c_z + m\, s}\, v_w \quad & p &= \frac{1}{c_p + J_x\, s}\, v_p \\ q &= \frac{1-\tau_q\, s}{c_q + J_y\, s}\, v_q, \quad & r &= \frac{1+\tau_r\, s}{c_r + J_z\, s}\, v_r, \end{aligned} \tag{2.106a}$$

and

$$\begin{aligned} u &= -m\, g\, \frac{1-\tau_q\, s}{s\,(c_q + J_y\, s)(c_v + m\, s)}\, v_q \\ v &= \frac{m\, g + k_{pv} c_p\, s + k_{pv} J_x\, s^2}{s\,(c_p + J_x\, s)(c_u + m\, s)}\, v_p \end{aligned} \tag{2.106b}$$

It can be noticed that zeros are introduced in the transfer functions of the speeds u and v, in comparison to the X4 configuration (2.100).

A simulation of the linear model was performed for a symmetric quadrotor in V4 configuration, for different values of the V-angle ϑ, and with a realistic set of parameters (mass, inertia, thrust and drag coefficients, propellers positions etc.), similar to the parameters of a Bebop 2. For a V angle $\vartheta = 10°$ on each propeller, the required propeller rotation

speed increments $\delta\omega_i$ required to achieve a given torque on the yaw axis was divided by 2 in steady state compared to an X4 configuration (see figure 2.21).

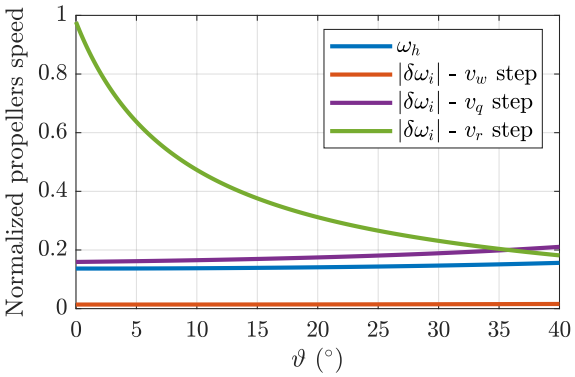

FIGURE 2.21 – Impact of the V angle on each axis

In the meantime, the propeller rotation speed increments required to obtain a given torque on the pitch axis and a given force on the z axis were nearly unchanged (less than 5% higher than an X4), up to a V angle $\vartheta = 20°$ ($\sim 10\%$ higher than an X4).

Figure 2.22 shows the evolution of the total propellers torque on the pitch axis u_q in response to a step of decoupled torque control input v_q. The impact of the V-angle on a PID-controlled pitch angle closed-loop was also simulated. The PID controller was tuned for a 0° V angle, and its robustness regarding the positive real zero introduced by the V-angle is presented figure 2.23. As expected, the larger the V-angle ϑ, the stronger the non-minimum phase behavior in the pitch dynamics. However, the step response of u_q/v_q typically reaches its minimum in less than 10 ms (figure 2.22). The overshoot increases by 20% for a 40° V-angle on figure 2.23 and could be reduced by retuning the PID controller. Hence, the non-minimum phase behavior on the pitch axis is hardly visible for a V-angle ϑ that could reasonably be encountered on a potential industrial quadrotor.

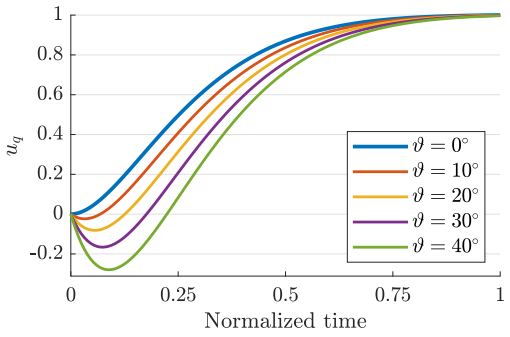

FIGURE 2.22 – u_q/v_q step response

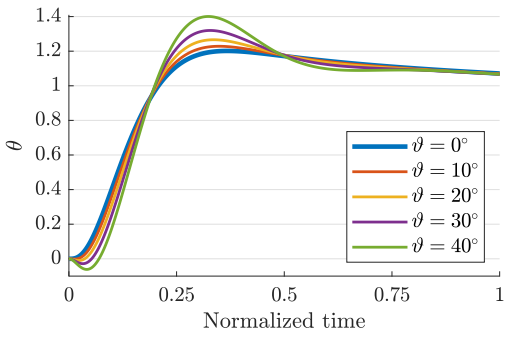

FIGURE 2.23 – $\theta/\theta_{\text{ref}}$ step response

In the meantime, figure 2.24 illustrates that when ϑ increases, the action of the inertia of the propellers on the yaw axis relatively to the action of the lift and drag torques dramatically decreases ($\sim$ halved for $\vartheta = 20°$). As a consequence, if the V4 configuration improves the static gain between propellers speeds and propellers torques on $\mathbf{z}_{\mathcal{D}}$, it could degrade the bandwidth on the yaw axis.

The impact of the coupling gain k_{pv} in (2.105) is illustrated by figure 2.25. The force generated by the term $mg\,\varphi$ (in blue) behaves in a similar way as a double integrator

regarding v_p (and does not depend on ϑ). The green line stands for the evolution of the term $k_{pv}\, v_p$. Only the motor dynamics remains (also present in the term $mg\,\varphi$), giving here a second order low-pass filter behavior. The red curve represents the sum of these two terms, and can be compared to the blue curve to emphasize the extra dynamics added by the term $k_{pv}\, v_p$ on the V4 configuration.

Consequently, for a quadrotor designed to achieve high velocities along $\mathbf{x}_\mathcal{D}$, it could be more interesting to tilt the propellers around $\mathbf{y}_\mathcal{D}$, instead of $\mathbf{x}_\mathcal{D}$. This way, the roll dynamics would be degraded, leading to a non-minimum phase behavior, in favor of a better yaw dynamics, while the pitch dynamics would have a slightly higher bandwidth due to a coupling term between v_q and u_u.

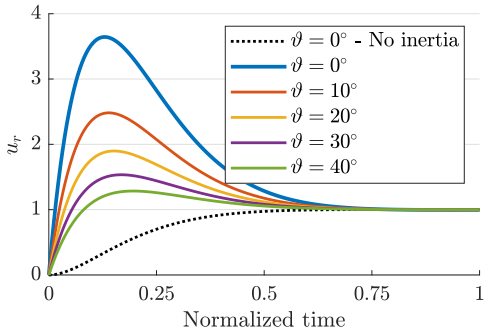

FIGURE 2.24 – u_r/v_r step response

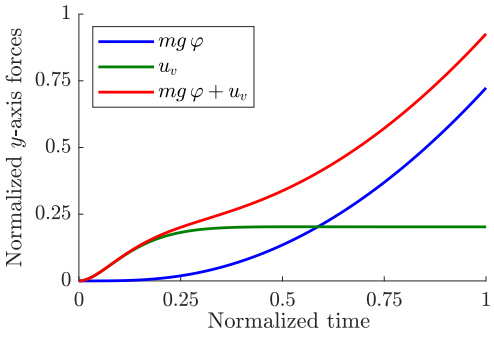

FIGURE 2.25 – Impact of the V4 configuration on the y translation axis

Remark 8 *The V4 configuration is useful for understanding the effects of tilting a propellers, but a better way to make the propellers lift contribute to the actuation of the yaw axis is to tilt the propellers along the axis joining their center of mass to the center of mass of the drone* $\overrightarrow{GP_i}$*. We call this the W4 configuration and it can be found on many consumer drones, such as the Bebop 2.*

In a more general way, [118] proposed a method to characterize the dynamics of the 3 *rotation axes for any propeller configuration. This method is then used to optimize the configuration in order to reach a desired balance between the dynamics of the* 3 *rotation axes, for instance, having the propeller act equally on the* 3 *axes rather than having a weaker yaw axis, as for the X4 configuration.*

However, this work does not take the inertia of the propeller into account, as we did in this section. As for the V4 configuration, the computation of the matrix $\mathrm{B}_{\dot{\omega}\to u}.\mathrm{B}_{v\to\omega}$ *we introduced in this section reveals that the W4 configuration also induces a non-minimum phase behavior but on both the pitch and roll axes. The effect of the inertia of the propellers is then present on the* 3 *rotation axes but the real positive zeros on the pitch and roll axes are smaller than the one on the pitch axis of the V4 configuration, for a same rotation angle of the propeller axes.*

In the following section, we write the linear model described above as a Linear Time-Invariant (LTI) state-space model.

2.6.4 General LTI state-space model

The motor model (2.92) linking the rotor speed references $\delta\omega_{\text{ref}\,i}$ to their actual speed $\delta\omega_i$ can be written

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\boldsymbol{\Omega}\\ \dot{\boldsymbol{\Omega}}\end{pmatrix} = \mathbf{A}_\Omega \begin{pmatrix}\boldsymbol{\Omega}\\ \dot{\boldsymbol{\Omega}}\end{pmatrix} + \mathbf{B}_\Omega\, \boldsymbol{\Omega}_{\text{ref}} \tag{2.107}$$

with

$$\begin{aligned}
\boldsymbol{\Omega} &= \begin{pmatrix}\delta\omega_1 & \delta\omega_2 & \delta\omega_3 & \delta\omega_4\end{pmatrix}^\top\\
\boldsymbol{\Omega}_{\text{ref}} &= \begin{pmatrix}\delta\omega_{\text{ref}\,i} & \delta\omega_{\text{ref}\,i} & \delta\omega_{\text{ref}\,i} & \delta\omega_{\text{ref}\,i}\end{pmatrix}^\top\\
\mathbf{A}_\Omega &= \begin{pmatrix} \multicolumn{4}{c}{\mathbf{0}_4} & \multicolumn{4}{c}{\mathbf{I}_4}\\
-\varpi_1^2 & 0 & 0 & 0 & -2\xi_1\varpi_1 & 0 & 0 & 0\\
0 & -\varpi_2^2 & 0 & 0 & 0 & -2\xi_2\varpi_2 & 0 & 0\\
0 & 0 & -\varpi_3^2 & 0 & 0 & 0 & -2\xi_3\varpi_3 & 0\\
0 & 0 & 0 & -\varpi_4^2 & 0 & 0 & 0 & -2\xi_4\varpi_4
\end{pmatrix}\\
\mathbf{B}_\Omega &= \begin{pmatrix} \multicolumn{4}{c}{\mathbf{0}_4}\\
\varpi_1^2 & 0 & 0 & 0\\
0 & \varpi_2^2 & 0 & 0\\
0 & 0 & \varpi_3^2 & 0\\
0 & 0 & 0 & \varpi_4^2
\end{pmatrix}
\end{aligned} \tag{2.108}$$

In order to use the decoupled control inputs v_i as model inputs rather than the propellers rotation speed increments $\delta\omega_i$ we can operate the following change of variable on the model (2.108). Introducing the reference decoupled control inputs $\left(v_{w\text{ref}}, v_{p_{\text{ref}}}, v_{q_{\text{ref}}}, v_{r\text{ref}}\right) \in \mathbb{R}^4$, the model (2.108) can be re-written by using the mix matrix (see (2.90) and (2.89))

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} = \mathbf{A}_V \begin{pmatrix}\mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} + \mathbf{B}_V\, \mathbf{V}_{\text{ref}} \tag{2.109}$$

with

$$\begin{aligned}
\mathbf{V} &= \begin{pmatrix}v_w & v_p & v_q & v_r\end{pmatrix}^\top\\
\mathbf{V}_{\text{ref}} &= \begin{pmatrix}v_{w\text{ref}} & v_{p_{\text{ref}}} & v_{q_{\text{ref}}} & v_{r\text{ref}}\end{pmatrix}^\top\\
\mathbf{A}_V &= \begin{pmatrix}\mathrm{B}_{\omega\to v} & \mathbf{0}_4\\ \mathbf{0}_4 & \mathrm{B}_{\omega\to v}\end{pmatrix} \mathbf{A}_\Omega \begin{pmatrix}\mathrm{B}_{v\to\omega} & \mathbf{0}_4\\ \mathbf{0}_4 & \mathrm{B}_{v\to\omega}\end{pmatrix}\\
\mathbf{B}_V &= \begin{pmatrix}\mathrm{B}_{\omega\to v} & \mathbf{0}_4\\ \mathbf{0}_4 & \mathrm{B}_{\omega\to v}\end{pmatrix} \mathbf{B}_\Omega\, \mathrm{B}_{v\to\omega}
\end{aligned} \tag{2.110}$$

with $\mathrm{B}_{\omega\to v}$ defined in (2.88) and $\mathrm{B}_{v\to\omega}$ defined in (2.90). and we can now link the reference decoupled control inputs $v_{i\text{ref}}$ to the mechanical action generated by the propellers u_i by using (2.91)

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{U}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} = \mathbf{A}_U \begin{pmatrix}\mathbf{U}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} + \mathbf{B}_U\, \mathbf{V}_{\text{ref}} \tag{2.111}$$

with

$$\begin{aligned}
\mathbf{U} &= \begin{pmatrix}u_u & u_v & u_w & u_p & u_q & u_r\end{pmatrix}^\top\\
\mathbf{A}_U &= \begin{pmatrix}\mathbf{0}_6 & \left(\mathrm{B}_{\omega\to u}\mathrm{B}_{v\to\omega} \quad \mathrm{B}_{\dot\omega\to u}\mathrm{B}_{v\to\omega}\right)\mathbf{A}_V\\ \mathbf{0}_{8\times 6} & \mathbf{A}_V\end{pmatrix}\\
\mathbf{B}_U &= \begin{pmatrix}\left(\mathrm{B}_{\omega\to u}\mathrm{B}_{v\to\omega} \quad \mathrm{B}_{\dot\omega\to u}\mathrm{B}_{v\to\omega}\right)\mathbf{B}_V\\ \mathbf{B}_V\end{pmatrix}
\end{aligned} \tag{2.112}$$

and $\mathrm{B}_{\omega\to u}$ and $\mathrm{B}_{\dot{\omega}\to u}$ introduced in (2.85).

Finally, using FPD, linking between the mechanical actions of the propellers to the system outputs (2.87), we can write the state space model representation of the system

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{X}\\ \mathbf{U}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} = \mathbf{A}_X \begin{pmatrix}\mathbf{X}\\ \mathbf{U}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} + \mathbf{B}_X\,\mathbf{V}_{\mathrm{ref}} \tag{2.113}$$

with

$$\begin{aligned}
\mathbf{X} &= \begin{pmatrix} u & v & w & \varphi & \theta & \psi & p & q & r\end{pmatrix}^\top\\
\mathbf{A}_X &= \begin{pmatrix} \mathbf{C}_{\mathrm{trans}} & \mathbf{G} & \mathbf{0}_3 & \frac{1}{m}\mathbf{I}_3 & \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_{9\times 11}\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{C}_{\mathrm{rot}} & \mathbf{J}^{-1} & \\ & \mathbf{0}_{14\times 9} & & & \mathbf{A}_U \end{pmatrix}\\
\mathbf{B}_X &= \begin{pmatrix}\mathbf{0}_{9\times 4}\\ \mathbf{B}_U\end{pmatrix}
\end{aligned} \tag{2.114}$$

with

$$\begin{aligned}
\mathbf{C}_{\mathrm{trans}} &= \frac{1}{m}\begin{pmatrix}-c_u & 0 & 0\\ 0 & -c_v & 0\\ 0 & 0 & -c_w\end{pmatrix}\\
\mathbf{G} &= \begin{pmatrix}0 & -g & 0\\ g & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\\
\mathbf{C}_{\mathrm{rot}} &= \mathbf{J}^{-1}\begin{pmatrix}-c_p & 0 & 0\\ 0 & -c_q & 0\\ 0 & 0 & -c_r\end{pmatrix}\\
\mathbf{J} &= \begin{pmatrix}J_x & 0 & 0\\ 0 & J_y & 0\\ 0 & 0 & J_z\end{pmatrix}
\end{aligned} \tag{2.115}$$

We can actually reduce this model by removing the mechanical action of the propellers as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\mathbf{X}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} = \mathbf{A}'_X \begin{pmatrix}\mathbf{X}\\ \mathbf{V}\\ \dot{\mathbf{V}}\end{pmatrix} + \mathbf{B}'_X\,\mathbf{V}_{\mathrm{ref}} \tag{2.116}$$

with

$$\begin{aligned}
\mathbf{A}'_X &= \begin{pmatrix} \mathbf{C}_{\mathrm{trans}} & \mathbf{G} & \mathbf{0}_3 & \frac{1}{m}\begin{pmatrix}\mathbf{I}_3 & \mathbf{0}_3\end{pmatrix}\mathrm{B}_{\omega\to u}\,\mathrm{B}_{v\to\omega} & \frac{1}{m}\begin{pmatrix}\mathbf{I}_3 & \mathbf{0}_3\end{pmatrix}\mathrm{B}_{\dot{\omega}\to u}\,\mathrm{B}_{v\to\omega}\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_4 & \mathbf{0}_4\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{C}_{\mathrm{rot}} & \mathbf{J}^{-1}\begin{pmatrix}\mathbf{0}_3 & \mathbf{I}_3\end{pmatrix}\mathrm{B}_{\omega\to u}\,\mathrm{B}_{v\to\omega} & \mathbf{J}^{-1}\begin{pmatrix}\mathbf{0}_3 & \mathbf{I}_3\end{pmatrix}\mathrm{B}_{\dot{\omega}\to u}\,\mathrm{B}_{v\to\omega}\\ & \mathbf{0}_{8\times 12} & & \mathbf{0}_4 & \mathbf{I}_4\\ & & & \mathbf{A}_V & \mathbf{0}_4 \end{pmatrix}\\
\mathbf{B}'_X &= \begin{pmatrix}\mathbf{0}_{13\times 4}\\ \mathbf{B}_V\end{pmatrix}
\end{aligned} \tag{2.117}$$

2.7 Conclusion

In this chapter, a model of a quadrotor has been derived from the equations of rigid body mechanics. One contribution of this chapter consists in using the screw theory rather the than Lagrange formalism to build the model. In order to build the model, we started by detailing the dynamics of the drone body and we presented 3 representations of the attitude of the drone. We also detailed the main mechanical actions experienced by the body of the quadrotor. Then, we proposed one method to rewrite the obtained equations in order to build a model of the entire quadrotor.

In a second part, we linearized this model around the hovering equilibrium. As another contribution, we emphasized the impact of the inertia of the propellers in the drone dynamics for different configurations of the propellers, more specifically the apparition of a non-minimum phase behavior and coupling terms. This result was presented at the IFAC World Congress 2017 [105].

Finally, we proposed a new Linear Time-Invariant (LTI) state-space model representation that takes into account the BLDC motors and the inertia of the propellers into account, for any quadrotor configuration.

The obtained nonlinear and linearized models will be used in the following chapters for designing guidance and control laws.

Chapter 3

Guidance – bi-level optimization

3.1 Introduction

Performing autonomous aerial footage requires having a flying camera execute a smooth and natural 7D motion (3D position, 3D rotation and magnification) meeting the requirements of cinematography in terms of shape, speed profile and smoothness, such as the ones specified in section 1.2. This is a wide problem; its different aspects have been and are still largely studied in the research community. In this work, in order to simplify the problem, it is split into, firstly, the 3D position motion and, secondly, the 4D rotation and zoom motion. This chapter deals with the generation of a 3D position motion suitable for cinematography.

Two opposing approaches are classically proposed in the literature to tackle the problem of motion generation, including the area of aerial cinematography.

- On the one hand, the paradigm of *trajectory tracking* suggests to compute a trajectory allowing to complete the mission, giving the position of the drone as a function of time. This trajectory constitutes a reference that the drone has to track by the mean of a controller. At a given time instant t, this controller compares the evaluation of the reference trajectory at t with the current estimated state of the drone and sends control signals to the actuators to cancel the error between the reference and the estimate.

- On the other hand, the paradigm of *path following* suggests to remove the time dependency of the motion by designing a *policy* from which emerges the completion of the mission. At a given time instant, the behavior of the drone is then typically given by its position relatively to the path to follow, rather than the evaluation of a timed reference as it is the case for the *trajectory tracking*.

The advantage of the path following strategy is obvious; free of time dependency, it is much more robust to deviations from the expected path than the trajectory tracking, when encountering obstacles or heavy disturbances, for example. This is illustrated on figure 3.1. A path in light green is followed by the drone, whose motion is represented in blue. On the left, a timing law is added to the path and a trajectory tracking strategy is adopted to complete the mission. At each time instant, the evaluation of the reference trajectory corresponds to a reference position to join and is represented by a green circle. On the right, a path following strategy is adopted. At the second time instant, a wind gust makes

the drone deviate from the path. On the left, the evaluation of the reference trajectory goes on, unaware of the deviation, and actually prevents the drone from clinging back to the path. On the right, the drone first rejoins the path before going on. The trajectory tracking strategy leads to a higher overall deviation from the path than the path following method. On the other side, by rejoining the path in priority, the path following method leads to delay in the performance of the mission. To summarize, the trajectory tracking could seem less robust than the path following strategy, but gives better control on the state of the drone at a given time instant and allows taking into account time-domain constraints in the generation of the motion.

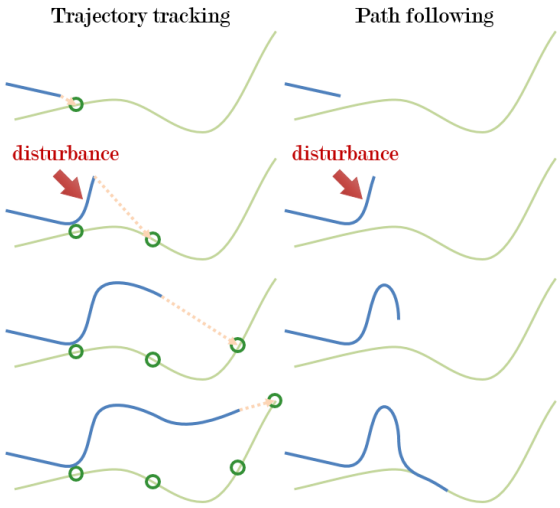

FIGURE 3.1 – Robustness of path following over trajectory tracking

In addition, various methods exist to mitigate this weakness of the trajectory tracking method, such as *local re-planning*, which consists in modifying the reference trajectory locally in order to avoid an obstacle for instance [51], [122], or the *dynamic time-scaling*, for which the time is dilated (the trajectory is slowed or accelerated) when the drone deviates from the reference trajectory [117].

The domain of aerial cinematography is not spared by the confrontation of these two approaches. Predictive control techniques constitutes good candidate policy for a path following method. Indeed, Model Predictive Control (MPC) has been successively applied to quadrotors in [17], [1], [6] and [61] and could be a candidate to simultaneously perform both the feedback control and the smooth trajectory generation. A promising path following strategy currently developed for cinematography thus consists in the use of a Model Predictive Contouring Control (MPCC) law to follow a given flight path, along with camera references, as suggested in [35] for quadrotor cameras. This strategy has also been applied to the case of target following with the drone motion constrained on a virtual rail and has yield impressive results, as in [86], [87], where an MPC strategy is used to generate motions directly through the optimization of cinematographic criteria, such as framing or occlusion, with obstacle avoidance solutions.

Unfortunately, this involves to solve a nonlinear optimization problem in real-time, usually achieved by using quadrotors with a high on-board computation power or by making the computations on a deported computer. This kind of decentralized computation is not suited to the case studied in this thesis, and the Bebop 2 is far from having the required computation resources to solve the problem on-board, in real-time. Furthermore, as mentioned in the paragraphs above, the path following strategy leaves little control on

the duration of the motion. This is an issue as some camera references used in this thesis require precise timing to be convincing, such as a constant speed panorama from one waypoint to another.

For these reasons, the trajectory tracking approach has been chosen in the present work. Several precursor works have already been lead with similar high level inputs as the flight plan presented in section 1.2.2. The use of smooth piecewise polynomials to generate smooth trajectories is quite popular. In [75] trajectories are generated through the minimization of the root mean square of one or several derivatives of the position, for a given duration of the overall trajectory. In this method, flight corridors were satisfied on a finite number of points (gridding). It was successfully developed and applied on real systems [101], [102], where a bi-level optimization is used to optimize the trajectory duration and to ensure its feasibility. In [60], a similar method is used for designing a smooth path which evaluation is given by a discretized timing law, in order to deal with the dynamics of the quadrotor and its payload, and applied to cinematography. A continuous-time alternative to this timing law is proposed in [103], in which it was used to make a cinematographic trajectory feasible without altering its shape. It was also used in [59], as part of a method proposed to perform scripted aerial shots with quadrotors.

The trajectory generation is usually divided into 2 steps: first the computation of a smooth path completing the flight plan, with a suitable shape, and then the computation of a timing law to perform this path, i.e. the conversion of this path into a reference trajectory respecting smoothness and feasibility constraints. As an example, in [38] a cinematographic path is first generated by interpolation of two camera poses. Then, a steering method (such as [40]) is used to complete the path. A similar strategy is applied in [41] with a smooth path generated using a method similar to [75].

In this chapter, we propose an algorithm merging the two steps, i.e. that directly syntheses a cinematographic trajectory from an input flight plan as detailed in section 1.2.2. To this aim, we first propose in sections 3.2 and 3.3 a novel way to guarantee the feasibility of the trajectory regarding the dynamics of the drone along with the respect of the cinematographic requirements detailed in section 1.2.3. Then, in sections 3.4 and 3.6 we propose a method inspired by the work of [75] and [101], adapted and extended to be suitable for cinematography. Finally, some improvements of the methods are explored in section 3.7.

We first begin with the suggestion of a quick preprocessing of the input flight plan in order to check the relevancy of the inputs and to cover a part of the feasibility issue.

3.2 Flight plan preprocessing

Before generating any trajectory from the input flight plan, a quick preprocessing is performed to check the relevance of the references and to split the trajectory generation problem into simpler ones.

Splitting the flight plan. First, the flight plan is split at each *stop* waypoint other than the first (i.e. $\mathbf{w}_0$) and the last (i.e. $\mathbf{w}_N$) waypoints. Whenever two waypoints are superimposed, they are both replaced by stop waypoints and the flight plan is split. This results in several flight plans, containing only *lock* and *sphere* waypoints, except for the first and the last ones. Trajectories are separately generated for each flight plan and are then laid end to end.

As a consequence, from this point, we only consider a flight plan containing two *stop* waypoints $\mathbf{w}_0$ and $\mathbf{w}_N$ at the beginning and at the end, and between them only *lock* and *sphere* waypoints.

Colliding waypoints. In order to prevent the validation of more than one *sphere* waypoint at a time, and to prevent any singular camera behavior, the validation radius of each *sphere* waypoint is limited to a third of the distance with its previous and next waypoints, as illustrated on figure 3.2.

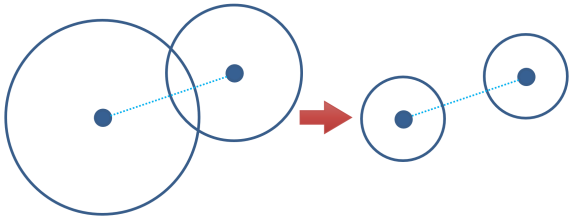

FIGURE 3.2 – Limitation of the waypoints validation radii

Clamping the speed references. From the identification of the parameters of the drone model, and from the flight experience acquired with flight tests, a simplified flight envelop in terms of velocity is determined. This envelop is given by the maximum descending speed of the drone before entering VRS state (see section 2.4.2.3), with a security margin, its maximum admissible ground angle and a limiting collective thrust (sum of the truths of each propeller, giving the vertical force applied on the drone) before degrading the attitude control. The last two factors give a maximum reachable lateral speed and a maximum reachable ascending speed, which are used to characterize the flight envelope. The set of admissible velocities is thus given by a cylinder

$$\mathcal{S}_v = \left\{ (v_x, v_y, v_z) \in \mathbb{R}^3 \;\middle|\; \begin{cases} \sqrt{v_x^2 + v_y^2} \leqslant v_{\text{lat}} \\ v_z \leqslant v_{\text{descent}} \\ v_z \geqslant -v_{\text{ascent}} \end{cases} \right\} \tag{3.1}$$

with v_{lat}, v_{descent} and v_{ascent} the lateral, descending and ascending velocities, respectively. The VRS threshold speed is given by [109], [5]

$$v_{\text{VRS}} = \frac{1}{2} v_h \tag{3.2}$$

with v_h the mean induced velocity of the air over the rotor surface during undisturbed hovering (approximately $5\,\text{m}\cdot\text{s}^{-1}$ for a Bebop 2). The maximum descending speed is then chosen with the security margin $\varepsilon_{\text{VRS}} \in]0, v_{\text{VRS}}[$

$$v_{\text{descent}} = v_{\text{VRS}} - \varepsilon_{\text{VRS}} \tag{3.3}$$

In addition, a minimum admissible speed reference v_{min} is added in order to prevent the user to plan an overly long flight. Finally, camera yaw, pitch and roll excursions between each pair of consecutive waypoints are approximated (see chapter 5). Given a maximum camera rotation speed, these excursions give an additional upper bound on the speed reference $v_{i,\text{cam}}$ (that can be lower than v_{min}). For instance, if two waypoints are very close but a 120° yaw panorama has to be performed between them at a maximum

rotation speed of $5°.s^{-1}$, the speed reference between these 2 waypoints should not exceed $d \cdot 120/5\,\mathrm{m{\cdot}s^{-1}}$ (with d the distance between the 2 waypoints).

The third step of the preprocessing, after splitting the flight plan and preventing waypoints collision, is thus to clamp the reference velocities so that they lie within the admissible set. From the original speed reference v_i, $i \in [\![1, N]\!]$, between the waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ we define the clamped reference speed $\nu_i \in \mathbb{R}$. Defining

$$\mathbf{u}_i = \begin{pmatrix} u_{i,x} & u_{i,y} & u_{i,z} \end{pmatrix}^\top = \frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2} \tag{3.4}$$

with $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|$ always strictly positive (otherwise the flight plan would have been split on this waypoint as explained above), we compute the clamped reference speed as follows

$$\begin{aligned} \nu_{i,z} &= \max\left\{v_{\text{ascent}}, \min\left\{v_{\text{descent}}, v_i\, u_{i,z}\right\}\right\} \\ \nu_{i,xy} &= \min\left\{v_{\text{lat}}, v_i \sqrt{u_{i,x}^2 + u_{i,y}^2}\right\} \\ \nu_i &= \min\left\{v_{i,\text{cam}}, \max\left\{\sqrt{\nu_{i,xy}^2 + \nu_{i,z}^2}, v_{\min}\right\}\right\} \end{aligned} \tag{3.5}$$

This preprocessing of the flight plan alone does not ensure the feasibility of the trajectory nor the respect of aesthetic requirements detailed in section 1.2.3. To complete these requirements, different criteria are highlighted in the next section.

3.3 Feasibility of the trajectory

In [83], a simplified drone model is used for the trajectory generation, with the total thrust and the angular velocity of the drone considered as inputs of the drone. The actuation limitations are then reformulated into constraints on the time derivatives of the position that ensure the feasibility of the trajectory. In this chapter, we use the same model as in [83], which can be derived from the model (2.75) by neglecting the disturbances and considering the angular velocity ideally controlled (and thus as an input).

The inputs are the total thrust normalized by the drone mass, f, and the 3 components of the angular velocity of the drone $\mathbf{\Omega} = \begin{pmatrix} p & q & r \end{pmatrix}^\top$. The drone model is then

$$\ddot{\boldsymbol{\zeta}} = g\,\mathbf{z}_{\mathcal{W}} - f\,\mathbf{R}\,\mathbf{z}_{\mathcal{W}} \tag{3.6a}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,\widehat{\mathbf{\Omega}} \tag{3.6b}$$

with $\widehat{\mathbf{\Omega}} = \begin{pmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{pmatrix}$. Notice that, compared to (2.75), the indexes of the rotation operator $\mathbf{R}_{\mathcal{W}\to\mathcal{D}}$ and angular velocity $\mathbf{\Omega}_{\mathcal{W}\to\mathcal{D}}$ have been droped to improve readability, as there is no possible frame confusion anymore.

Camera yaw speed constraint. As explained in section 1.2.3, from purely aesthetic considerations, the camera should not rotate too quickly. With the fully digital stabilization of the camera on a Bebop 2, the pitch and roll of the camera can rotate freely, independently of the drone motion. Nevertheless, the camera is fixed relatively to the drone and does not have a sufficient field of view to record at an arbitrary heading. On the contrary, it is limited to record around the drone heading. To simplify, this means that the camera

heading is approximatively equal to the drone heading. This aesthetic criterion on the camera rotation motion thus applies to the drone rotation around its yaw axis.

As a consequence, the time variation of the drone heading is limited to a maximum value

$$\dot{\psi} \leqslant \dot{\psi}_{\max} \tag{3.7}$$

Actuation constraints. The inputs have limited range and must verify

$$f \geqslant f_{\min} \tag{3.8a}$$
$$f \leqslant f_{\max} \tag{3.8b}$$
$$\|\boldsymbol{\Omega}\|_2 \leqslant \Omega_{\max} \tag{3.8c}$$

with $f_{\min} < g$, $f_{\max} > g$ and $\Omega_{\max} > 0$. The limitation on the norm of the angular velocity (3.8c), $\Omega_{\max}$, is given by the capabilities of the angular velocity control loop (dynamics of the controller and actuators, saturation of the actuators, estimation accuracy, sensors noises and saturations etc.).

In addition, the ground angle of the drone α must also remain below a maximum value for 3 reasons

- There is a maximum ground angle above which the drone is no longer able to counter gravity at maximum horizontal speed and starts to fall. Though this would not be an issue for a descending motion for instance, it was desired to keep the capacity to counter gravity at all time as a safety measure.
- Above a certain ground angle, the efficiency of the vertical camera of the drone (which takes part in the drone velocity estimation) and the ultrasound sensors begin to degrade.
- As it was mentionned in section 1.2.3, the ground angle should be limited in order to guaratee that the camera can record in the desired direction.

The lowest of these limiting values is chosen as upper bound on the ground angle

$$\alpha \leqslant \alpha_{\max} \tag{3.8d}$$

with $\alpha_{\max} \in]0, \pi/2[$.

In a similar way to [83], we reformulate these limitations into constraints on the time derivatives of the position of the drone. The additional difficulty in our work is the non zero rotational motion around the yaw axis of the drone and the maximum ground angle, which complexifies a bit the computation of the feasibility constraints.

Acceleration constraint. Given the model (3.6a), the acceleration of the drone is given by the sum of the gravity, of direction $\mathbf{z}_{\mathcal{W}}$ and magnitude g, and the thrust, of direction $\mathbf{R}\,\mathbf{z}_{\mathcal{W}}$ and magnitude t. This is illustrated on figure 3.3, with the thrust and the acceleration in thick orange arrows. The limitations induced by the constraints (3.8a), (3.8b) and (3.8d) are also represented on figure 3.3. Given the limitations (3.8), the set of admissible acceleration can thus be represented, by a white domain on figure 3.3. However, this would lead to an asymmetrical behavior of the flying camera on the translational axes.

In the context of video making, this was judged not satisfying to have different acceleration capabilities whether the drone is ascending or descending and wether the drone is moving horizontally or vertically. It was thus chosen to get a bound on the magnitude of the acceleration that would ensure the feasibility of the trajectory, as illustrated on figure 3.4. Though it introduces some conservatism, it confers a homogeneous behavior of the translational motion along the 3 axes.

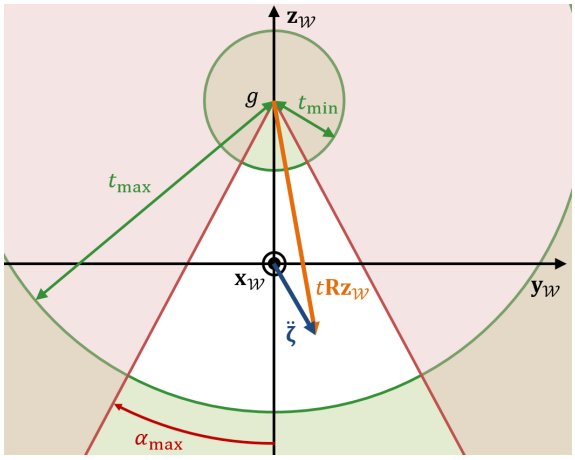

FIGURE 3.3 – Admissible set for the acceleration

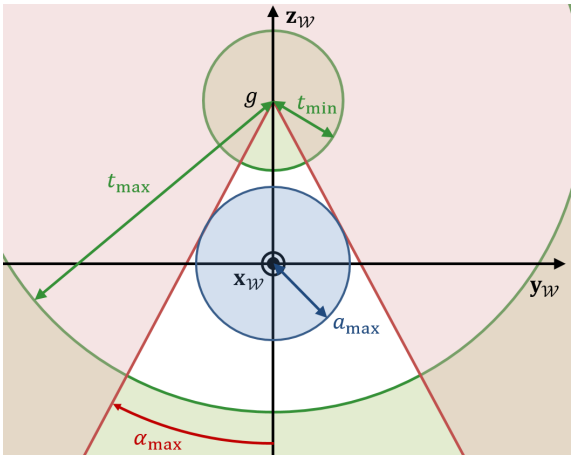

FIGURE 3.4 – Acceleration bound

Using figure 3.4, we propose the following new bounds on the acceleration

Proposition 1

$$\left(\left\|\ddot{\boldsymbol{\zeta}}\right\|_2 \leqslant a_{\max} \triangleq \min\left\{g - f_{\min},\ f_{\max} - g,\ g\, s_{\alpha_{\max}}\right\}\right) \Rightarrow \begin{cases} f \geqslant f_{\min} \\ f \leqslant f_{\max} \\ \alpha \leqslant \alpha_{\max} \end{cases} \tag{3.9}$$

We now show that this result can directly be obtained from (3.6a), (3.8a), (3.8b) and (3.8d).

First, let us define

$$\ddot{\boldsymbol{\zeta}} = a_x\, \mathbf{x}_\mathcal{W} + a_y\, \mathbf{y}_\mathcal{W} + a_z\, \mathbf{z}_\mathcal{W} \tag{3.10a}$$

$$a_\perp = \sqrt{a_x^2 + a_y^2} \tag{3.10b}$$

$$a = \sqrt{a_\perp^2 + a_z^2} = \left\|\ddot{\boldsymbol{\zeta}}\right\|_2 \tag{3.10c}$$

Using (3.6a) we have

$$f^2 = a_\perp^2 + (g - a_z)^2 \tag{3.11}$$

Replacing $a_\perp^2$ from (3.10c) in (3.11) gives

$$f^2 = a^2 - 2g\, a_z + g^2 \tag{3.12}$$

Since $a_z^2 = a^2 - a_\perp^2$ (3.10c), we have $|a_z| \leqslant a$ and

$$\underbrace{a^2 - 2g\, a + g^2}_{b_1(a)} \leqslant f^2 \leqslant \underbrace{a^2 + 2g\, a + g^2}_{b_2(a)} \tag{3.13}$$

Solving $b_1(a) \geqslant f_{\min}$ leads to

$$(a \leqslant g - f_{\min}) \Rightarrow (t \geqslant f_{\min}) \tag{3.14}$$

while solving $b_2(a) \leqslant f_{\max}$ leads to

$$(a \leqslant f_{\max} - g) \Rightarrow (t \leqslant f_{\max}) \tag{3.15}$$

Furthermore, for a given acceleration magnitude a, we have

$$\tan(\alpha) = \frac{a_\perp}{g - a_z} = \frac{\sqrt{a^2 - a_z^2}}{g - a_z} \tag{3.16}$$

where $\tan(\alpha)$ reaches its maximum for $a_z = a^2/g$. We thus have

$$\tan(\alpha) \leqslant \underbrace{\frac{a}{\sqrt{g^2 - a^2}}}_{b_3(a)} \tag{3.17}$$

Solving $b_3(a) \leqslant \tan(\alpha_{\max})$ leads to

$$(a \leqslant g s_{\alpha_{\max}}) \Rightarrow (\alpha \leqslant \alpha_{\max}) \tag{3.18}$$

Equations (3.14), (3.15) and (3.18) lead to (3.9).

Jerk constraint. The constraint on the angular speed (3.8c) remains to be dealt with. In order to guarantee its respect, we propose the following new constraint on the jerk

Proposition 2

$$\begin{cases} \dot{\psi}_{\max} < \Omega_{\max} \\ j \leqslant j_{\max} \triangleq \dfrac{\sqrt{\Omega_{\max}^2 \left(1 + {s_{\alpha_{\max}}}^2\right) - \dot{\psi}_{\max}^2} - s_{\alpha_{\max}} \dot{\psi}_{\max}}{1 + {s_{\alpha_{\max}}}^2} f_{\min} \end{cases} \Rightarrow (\|\boldsymbol{\Omega}\|_2 \leqslant \Omega_{\max}) \tag{3.19}$$

We now give the proof of this proposition. Derivating (3.6a) and injecting (3.6b) into the obtained expression give

$$\boldsymbol{\zeta}^{(3)} = -\dot{f}\, \mathbf{R}\, \mathbf{z}_{\mathcal{W}} - f\, \mathbf{R} \widehat{\boldsymbol{\Omega}}\, \mathbf{z}_{\mathcal{W}} \tag{3.20}$$

or, since $f > 0$ and $\widehat{\boldsymbol{\Omega}}\, \mathbf{z}_{\mathcal{W}} = q\, \mathbf{x}_{\mathcal{W}} - p\, \mathbf{y}_{\mathcal{W}}$,

$$q\, \mathbf{x}_{\mathcal{W}} - p\, \mathbf{y}_{\mathcal{W}} = \frac{\dot{f}}{f}\, \mathbf{z}_{\mathcal{W}} - \frac{1}{f}\, \mathbf{R}^\top\, \boldsymbol{\zeta}^{(3)} \tag{3.21}$$

The thrust f is considered as an input with no slew rate hence the term $\dot{f}$ is unbounded and there is no limitation on the component of the jerk along $\mathbf{R}\, \mathbf{z}_{\mathcal{W}}$. Projecting (3.21) along $\mathbf{x}_{\mathcal{W}}$ and $\mathbf{y}_{\mathcal{W}}$ allows us to write

$$\sqrt{p^2 + q^2} = \frac{\sqrt{\left(\mathbf{R}^\top\, \boldsymbol{\zeta}^{(3)} \cdot \mathbf{x}_{\mathcal{W}}\right)^2 + \left(\mathbf{R}^\top\, \boldsymbol{\zeta}^{(3)} \cdot \mathbf{y}_{\mathcal{W}}\right)^2}}{f} \leqslant \frac{j}{f_{\min}} \tag{3.22}$$

with $j = \left\|\boldsymbol{\zeta}^{(3)}\right\|_2$. This bound is typically used in the literature [83] but it does not take into account the vertical component of the angular velociy, r. As a consequence, the latter is often imposed null ([83], [88]) which does not impact the position control of the drone which is decoupled from its yaw movement. Here, however, the drone has to follow heading references between the waypoints, as detailed in section 1.2.2. This implies that the vertical

component of the angular velocity is non zero in the general case. In this work, we use the constraint on the heading varitation given by (3.7) to get a bound on r, allowing to get a new, extended bound on the jerk ensuring the feasibility of the trajectory.

From (2.17) and (2.15) we get

$$r = -s_\varphi \, \dot{\theta} + c_\alpha \, \dot{\psi} \tag{3.23}$$

and from (2.18)

$$\dot{\theta} = c_\varphi \, p - s_\varphi \, q \tag{3.24}$$

If both p and q are null, then $\dot{\theta}$ is null too. Otherwise, the expression (3.24) can be written as follows

$$\dot{\theta} = c_{\varphi+\gamma} \sqrt{p^2 + q^2} \tag{3.25}$$

with

$$\gamma = \arctan_2 \left(\frac{q}{\sqrt{p^2+q^2}}, \frac{p}{\sqrt{p^2+q^2}} \right) \tag{3.26}$$

which leads to the following bound on $\dot{\theta}$

$$\left| \dot{\theta} \right| \leqslant \sqrt{p^2 + q^2} \leqslant \frac{j}{f_{\min}} \tag{3.27}$$

Furthermore, it can be deduced from (2.15) that $|\varphi| \leqslant \alpha$ and $|s_\varphi| \leqslant s_\alpha \leqslant s_{\alpha_{\max}}$. Given (3.7), we then have

$$|r| \leqslant s_{\alpha_{\max}} \frac{j}{f_{\min}} + \dot{\psi}_{\max} \tag{3.28}$$

As a consequence we can finally write

$$\|\mathbf{\Omega}\|_2^2 = p^2 + q^2 + r^2 \leqslant \underbrace{\frac{1 + {s_{\alpha_{\max}}}^2}{f_{\min}^2} j^2 + \frac{2 s_{\alpha_{\max}} \dot{\psi}_{\max}}{f_{\min}} \, j + \dot{\psi}_{\max}^2}_{b_4(j)} \tag{3.29}$$

Solving $b_4(j) \leqslant \Omega_{\max}^2$ leads to (3.19).

The feasibility of the trajectory can thus be characterized by one constraint on the acceleration and one constraint on the jerk. These constraints are then used for the trajectory generation process, which is summmarized in the next section.

3.4 Bi-level optimization

In [75], 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, revealed by a flatness analysis ([75]). The validation time of each waypoint must be specified and is chosen depending on the context, as imposed values or as solution of an optimization problem, for instance. A similar approach is used in this work, adapted to the cinematographic context for generating a feasible and visually satisfying trajectory.

Piecewise polynomial trajectory. Piecewise polynomial trajectories are commonly used in robotics as they allow to parameterize rich trajectories from a reasonable number of parameters. The trajectory is divided into N pieces numbered from 1 to N, each one joining a pair of consecutive waypoints. For $i \in [\![1, N]\!]$, the i-th piece joins the waypoint $\mathbf{w}_{i-1}$ to the waypoint $\mathbf{w}_i$.

Each piece of the trajectory is parameterized as a polynomial curve of degree $n \in \mathbb{N}$ and polynomial coefficients $\mathbf{C}_i \in \mathbb{R}^{3\times(n+1)}$

$$\mathbf{C}_i = \begin{pmatrix} \mathbf{c}_{i,0} & \mathbf{c}_{i,1} & \dots & \mathbf{c}_{i,n} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{C}_i^x \\ \mathbf{C}_i^y \\ \mathbf{C}_i^z \end{pmatrix} \tag{3.30}$$

The i-th polynomial piece, of coefficients $\mathbf{C}_i$, is denoted by $\boldsymbol{\varrho}_{\mathbf{C}_i}$ and its evaluation at a given time instant is given by

$$\forall t \in \mathbb{R} \quad \boldsymbol{\varrho}_{\mathbf{C}_i}(t) = \sum_{j=0}^{n} \mathbf{c}_{i,j}\, t^j \tag{3.31}$$

We denote by $\mathbf{C} \in \mathbb{R}^{3N\times(n+1)}$ the matrix containing the coefficients of all the polynomial pieces

$$\mathbf{C} = \{\mathbf{C}_i\}_{i\in[\![1,N]\!]} \tag{3.32}$$

Finally, for $i \in [\![1, N]\!]$ we define $\Delta t_i \in \mathbb{R}_+^*$, the duration of the piece of trajectory number i, as well as the vector of durations $\boldsymbol{\Delta t} \in \mathbb{R}^{1\times N}$

$$\boldsymbol{\Delta t} = \begin{pmatrix} \Delta t_1 & \Delta t_2 & \dots & \Delta t_N \end{pmatrix} \tag{3.33}$$

In the following, this vector is designated as the Time Of Flight (TOF) vector. These durations allow computing the time instants for which the waypoints are validated and where a connection between two pieces of trajectory occurs. For a given initial time t_0, these connections occur at the time instants $\{t_i\}_{i\in[\![1,N]\!]}$ such as

$$\forall i \in [\![1, N]\!] \quad t_i = t_{i-1} + \Delta t_i \tag{3.34}$$

and correspond to the instants for which the overall trajectory switches between two polynomial representations. In the following, we consider that $t_0 = 0$. The total duration of the trajectory is denoted by T

$$T = \sum_{i=1}^{N} \Delta t_i \tag{3.35}$$

This overall piecewise trajectory is denoted by $\boldsymbol{\zeta}_{\boldsymbol{\Delta t},\mathbf{C}}$ and its evaluation is given by

$$\boldsymbol{\zeta}_{\boldsymbol{\Delta t},\mathbf{C}}(t) = \begin{cases} \boldsymbol{\varrho}_{\mathbf{C}_0}(t_0) & \text{if } t < t_0 \\ \boldsymbol{\varrho}_{\mathbf{C}_i}(t) & \text{if } t_{i-1} \leqslant t < t_i,\ i \in [\![1, N]\!] \\ \boldsymbol{\varrho}_{\mathbf{C}_N}(t_N) & \text{if } t \geqslant t_N \end{cases} \tag{3.36}$$

Bi-level optimization. In this work, a similar scheme as in [75] is adapted to the cinematographic context and used for generating a feasible and visually satisfying trajectory. After the preprocessing of the flight plan, we use a bi-level optimization to generate a visually satisfying trajectory. This optimization is illustrated on figure 3.5: for a given vector of durations between each pair of consecutive waypoints, $\boldsymbol{\Delta t}$, a set of polynomial

coefficients $\mathbf{C}^*(\mathbf{\Delta t})$ is generated by solving an optimization problem on the polynomial coefficients. These coefficients are the optimal coefficients for *this* vector of durations. The vector of duration is then modified and the procedure is repeated until a satisfying trajectory $\boldsymbol{\zeta}_{\mathbf{\Delta t}^*,\mathbf{C}^*}$ has been generated, for an optimal vector of durations $\mathbf{\Delta t}^*$.

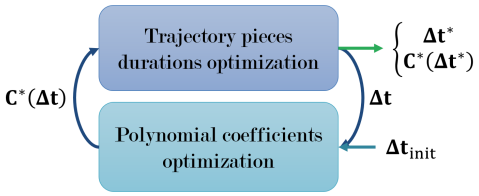

FIGURE 3.5 – Bi-level optimization principle

The reason for adopting this scheme of optimization is that the computation of the polynomial coefficient can be expressed as a simple quadratic programming problem if the durations of the pieces of trajectory are fixed, at least for the cost functions and constraints used in this work. On the contrary, the choice of the durations of the pieces of trajectory is a nonlinear problem. Knowing that the polynomial coefficients represent many more parameters than the durations of the pieces of trajectories, this scheme allows splitting the problem into

- A "large" but reasonable to solve quadratic problem returning the polynomial coefficients
- A nonlinear but "small" optimization problem returning the durations of the pieces of trajectories

which seems less time consuming that solving only one "large" nonlinear problem returning both the polynomial coefficients and the durations.

Given a TOF vector $\mathbf{\Delta t}$, the problem solved for generating a piecewise polynomial trajectory is of the general form

$$\begin{aligned} \mathbf{C}^*(\mathbf{\Delta t}) = \arg \min_{\mathbf{C}\in\mathbb{R}^{3\times(n+1)}} \; & J_{\mathbf{C}}(\mathbf{\Delta t}, \mathbf{C}) \\ \text{s.t.} \begin{cases} f_{\mathbf{C}}(\mathbf{\Delta t}, \mathbf{C}) = 0 \\ g_{\mathbf{C}}(\mathbf{\Delta t}, \mathbf{C}) \leqslant 0 \end{cases} & \end{aligned} \tag{3.37}$$

In a similar way, the optimization problem solved for choosing a TOF vector can be formalized as follows

$$\begin{aligned} \mathbf{\Delta t}^* = \arg \min_{\mathbf{\Delta t}\in\mathbb{R}^{1\times N}} \; & J_{\mathbf{\Delta t}}(\mathbf{\Delta t}) \\ \text{s.t.} \begin{cases} f_{\mathbf{\Delta t}}(\mathbf{\Delta t}) = 0 \\ g_{\mathbf{\Delta t}}(\mathbf{\Delta t}) \leqslant 0 \end{cases} & \end{aligned} \tag{3.38}$$

Remark 9 *It should be noticed that it is usually easier and better suited for numerical stability to parameterize each i-th piece of trajectory as a polynomial function of* $\tau = (t - t_i)/(t_{i+1} - t_i) \in [0, 1]$ *rather than* t *directly [75]. In addition, this problem can be formulated as an optimization problem on the derivatives of the trajectories at the connections between the pieces (i.e. at the time instants* t_i*) rather than on the polynomial coefficients [102], which reduces the number of decision variables and increases the numerical robustness of the algorithm. Neither of these recommendations are followed in this*

chapter in order to better understand the general strategy used for generating the trajectory. However, the proposed optimization problem can be reformulated to comply with these recommendations, which were followed for the actual implementation of the algorithm and the flight experiment presented at the end of this chapter.

The following sections deal with the choice of criteria for the two levels of optimization as well as the details of the constraints.

3.5 Smooth speed and contouring optimization

In the following, we detail our first attempt at generating cinematographic trajectories, using generic criteria for the two levels of optimization. The aim is to generate trajectories similar to the ones generated by the Parrot Flight Plan feature of the Bebop 2 (i.e. a series of straight lines with turns on the waypoints) but smoother and with proof of feasibility.

3.5.1 Lower level – polynomial coefficients

The optimization problem for the lower stage of the bi-level optimization is defined as follows.

Cost function. Three derivatives are included in the cost function, each one with a given weight

- **Snap**. Following the reasoning in [75], the snap is included in the cost function as a way to reduce the control signals required to track it.
- **Jerk**. As mentioned previously, the jerk quantifies the jolts in the drone motion and it is also strongly linked to its rotation speed, which should be limited to prevent the motion blur from degrading the video quality. It is hence included in the cost function, too.
- **Acceleration**. The acceleration is linked to the drone angle and it can thus be reduced to keep a margin relatively to its maximum admissible value.

Furthermore, two additional terms are also included in the objective function

- **Velocity**. Since a desired speed is specified by the user on each piece of trajectory, the mean square of the error between the velocity and its reference is also added to the cost function.
- **Position**. Finally, a contouring term, the mean square of the distance of the drone to the straigth line joining the previous and the next waypoint, is included too (figure 3.6). The reasons for this is to produce trajectories that look like the ones returned by the already implemented algorithm in the Parrot Bebop 2, i.e. a sequence of straight lines between waypoints and turns on waypoints. This term also constitutes an alternative to the corridor hard constraints as it tends to make the trajectory stick to the straight lines connecting the waypoints.

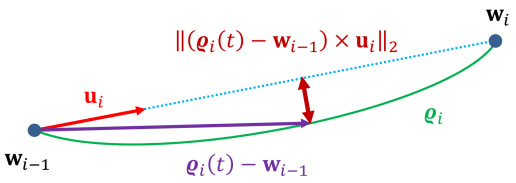

FIGURE 3.6 – Contouring error

The cost function in (3.37) is then given by

$$\begin{aligned} J_{\mathbf{C}}(\mathbf{\Delta t}, \mathbf{C}) &= \int_0^T \Big(\mu_c \left\| (\boldsymbol{\zeta}_{\mathbf{\Delta t},\mathbf{C}}(t) - \mathbf{w}_{i-1}) \times \mathbf{u}_i \right\|_2^2 + \mu_v \left\| v_i \, \mathbf{u}_i - \dot{\boldsymbol{\zeta}}_{\mathbf{\Delta t},\mathbf{C}}(t) \right\|_2^2 \\ &\quad + \mu_a \left\| \ddot{\boldsymbol{\zeta}}_{\mathbf{\Delta t},\mathbf{C}}(t) \right\|_2^2 + \mu_j \left\| \boldsymbol{\zeta}^{(3)}_{\mathbf{\Delta t},\mathbf{C}}(t) \right\|_2^2 + \mu_s \left\| \boldsymbol{\zeta}^{(4)}_{\mathbf{\Delta t},\mathbf{C}}(t) \right\|_2^2 \Big) \, \mathrm{d}t \\ &= \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \Big(\mu_c \left\| (\boldsymbol{\varrho}_{\mathbf{C}_i}(t) - \mathbf{w}_{i-1}) \times \mathbf{u}_i \right\|_2^2 + \mu_v \left\| v_i \, \mathbf{u}_i - \dot{\boldsymbol{\varrho}}_{\mathbf{C}_i}(t) \right\|_2^2 \\ &\quad + \mu_a \left\| \ddot{\boldsymbol{\varrho}}_{\mathbf{C}_i}(t) \right\|_2^2 + \mu_j \left\| \boldsymbol{\varrho}^{(3)}_{\mathbf{C}_i}(t) \right\|_2^2 + \mu_s \left\| \boldsymbol{\varrho}^{(4)}_{\mathbf{C}_i}(t) \right\|_2^2 \Big) \, \mathrm{d}t \end{aligned} \tag{3.39}$$

Constraints. The trajectory is subject to the following constraints

- **Waypoints validation**. The waypoints are validated on the connections between the different trajectory pieces, i.e. the waypoint $\mathbf{w}_i$ is validated at time t_i. Hence the trajectory must satisfy the waypoint validation criteria described in section 1.2 for each time t_i depending on the type of the waypoint $\mathbf{w}_i$.
- $\mathcal{C}^L$ **continuity**. Each piece of trajectory is a polynomial and is thus infinitely derivable on its interval. However, the entire trajectory being piecewise polynomial, its derivatives may not be continuous on the connections between the different pieces. In order to have a smooth and feasible trajectory, the continuity of the derivatives should be imposed on the connections up to a given order L.
- **Flight corridor**. The trajectory should not exit cylinders of given radius joining each pair of waypoints.

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

$$\forall i \in [\![1, N]\!] \begin{cases} \boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i-1}) = \mathbf{w}_{i-1} & \text{if } \mathbf{w}_{i-1} \text{ is of type } \textit{stop} \text{ or } \textit{lock} \\ \forall l \in [\![1, L]\!] \quad \boldsymbol{\varrho}^{(l)}_{\mathbf{C}_i}(t_{i-1}) = 0 & \text{if } \mathbf{w}_{i-1} \text{ is of type } \textit{stop} \\ \boldsymbol{\varrho}_{\mathbf{C}_i}(t_i) = \mathbf{w}_i & \text{if } \mathbf{w}_i \text{ is of type } \textit{stop} \text{ or } \textit{lock} \\ \forall l \in [\![1, L]\!] \quad \boldsymbol{\varrho}^{(l)}_{\mathbf{C}_i}(t_i) = 0 & \text{if } \mathbf{w}_i \text{ is of type } \textit{stop} \end{cases}$$

The validation of *sphere* waypoints is ensured by an inequality constraint on the distance between the drone and the waypoint at the corresponding validation time

$$\forall i \in [\![1, N]\!] \begin{cases} \left\| \boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i-1}) - \mathbf{w}_{i-1} \right\|_2 \leqslant r_{\mathbf{w}_{i-1}} & \text{if } \mathbf{w}_{i-1} \text{ is of type } \textit{sphere} \\ \left\| \boldsymbol{\varrho}_{\mathbf{C}_i}(t_i) - \mathbf{w}_i \right\|_2 \leqslant r_{\mathbf{w}_i} & \text{if } \mathbf{w}_i \text{ is of type } \textit{sphere} \end{cases}$$

with $r_{\mathbf{w}_i}$ the radius of the validation sphere centered on $\mathbf{w}_i$, after the preprocessing (see section 3.2). These quadratic constraints are approximated by linear constraints, by constraining the trajectory in a box of side length $r_{\mathbf{w}_i}/\sqrt{3}$ centered on the waypoint, rather than in a sphere.

When not already guaranteed by the waypoints validation constraints, the continuity of the position and its L first time derivatives is imposed on the connections between the different pieces of trajectory

$$\forall i \in [\![1, N-1]\!] \begin{cases} \boldsymbol{\varrho}_{\mathbf{C}_i}(t_i) = \boldsymbol{\varrho}_{\mathbf{C}_{i+1}}(t_i) & \text{if } \mathbf{w}_i \text{ is of type } \textit{sphere} \\ \forall l \in [\![1, L]\!] \quad \boldsymbol{\varrho}^{(l)}_{\mathbf{C}_i}(t_i) = \boldsymbol{\varrho}^{(l)}_{\mathbf{C}_{i+1}}(t_i) & \text{if } \mathbf{w}_i \text{ is of type } \textit{sphere} \text{ or } \textit{lock} \end{cases}$$

Gridding is used for formulating the last constraint as in [75], meaning that the constraint is imposed on a finite number n_{grid} of points on the trajectory via an inequality constraint (figure 3.7). As there is no guarantee that the trajectory would not exit the corridors between these points, the amount of points n_{grid} must not be too low.

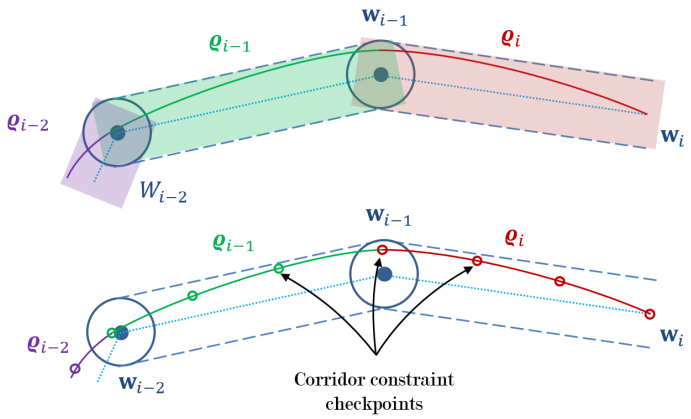

FIGURE 3.7 – Corridor constraints and griding strategy

For each point of the grid, 3 constraints ensure that this point lies inside the cylinder corresponding to the flight corridor between two waypoints

- 2 longitudinal constraints ensuring that the point lies within the two waypoints of the piece of trajectory
- 1 lateral constraint ensuring that the point lies within the cross section of the flight corridor

The 2 longitudinal constraints are written using a simple projection of the point on the axis joining the two waypoints and the lateral constraint is obtained using the contouring error of the point (see figure 3.6)

$$\forall i \in [\![1, N-1]\!], \quad \forall j \in [\![0, n_{\text{grid}} - 1]\!] \begin{cases} \mathbf{u}_i^\top \left(\boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i,j}) - \mathbf{w}_{i-1}\right) \geqslant -\varepsilon_{\text{corr}} \\ \mathbf{u}_i^\top \left(\boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i,j}) - \mathbf{w}_{i-1}\right) \leqslant \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 + \varepsilon_{\text{corr}} \\ \left\|\left(\boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i,j}) - \mathbf{w}_{i-1}\right) \times \mathbf{u}_i\right\|_2 \leqslant r_{\text{corr}\,i} \end{cases}$$

with $\varepsilon_{\text{corr}}$ a positive tolerance that can be set to zero and $t_{i,j}$ computed as follows

$$\forall i \in [\![1, N]\!], \quad \forall j \in [\![0, n_{\text{grid}} - 1]\!] \quad t_{i,j} = t_{i-1} + \frac{j}{n_{\text{grid}} - 1}(t_i - t_{i-1})$$

As for the *sphere* waypoints validation, the lateral corridor constraints are linearized by constraining the vector $(\boldsymbol{\varrho}_{\mathbf{C}_i}(t_{i,j}) - \mathbf{w}_{i-1}) \times \mathbf{u}_i$ inside a box of side length $r_{\text{corr}i}/\sqrt{3}$ and centered on the origin O, resulting in 6 linear lateral constraints. This is illustrated on figure 3.8.

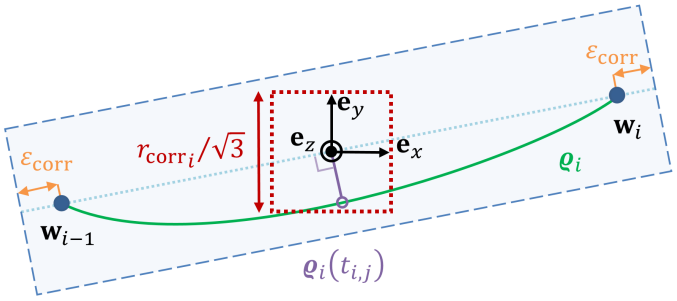

FIGURE 3.8 – Linearized corridor constraints

Optimization problem. As a result, for a given TOF vector, a piecewise polynomial trajectory can be generated by solving a Quadratic Programming (QP) problem under linear constraints. We define the optimization vector $\mathbf{x}_{\mathbf{C}} \in \mathbb{R}^{3N(n+1)\times 1}$

$$\mathbf{x}_{\mathbf{C}} = \begin{pmatrix} \mathbf{C}_1^x & \mathbf{C}_1^y & \mathbf{C}_1^z & \mathbf{C}_2^x & \mathbf{C}_2^y & \mathbf{C}_2^z & \dots & \mathbf{C}_N^x & \mathbf{C}_N^y & \mathbf{C}_N^z \end{pmatrix}^\top \tag{3.40}$$

The cost function to minimize (3.39) can be expressed as a quadratic function of $\mathbf{x}_{\mathbf{C}}$ [75], [101]. The problem to solve then has the form

$$\mathbf{x}_{\mathbf{C}}^*(\boldsymbol{\Delta t}) = \arg \min_{\mathbf{x}_{\mathbf{C}} \in \mathbb{R}^{3N(n+1)\times 1}} \mathbf{x}_{\mathbf{C}}^\top \mathbf{H}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} + \mathbf{f}^\top(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} + u(\boldsymbol{\Delta t})$$

$$\text{s.t.} \begin{cases} \mathbf{A}_{\text{stop}}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{stop}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{lock}}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{lock}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{sphere}}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} \leqslant \mathbf{b}_{\text{sphere}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{continuity}}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{continuity}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{corridor}}(\boldsymbol{\Delta t})\, \mathbf{x}_{\mathbf{C}} \leqslant \mathbf{b}_{\text{corridor}}(\boldsymbol{\Delta t}) \end{cases} \tag{3.41}$$

Notice that the constant scalar term $u(\boldsymbol{\Delta t})$ in the cost function in (3.41) is not required for solving the QP problem. It will however be needed for the optimization of the TOF vector, which is the task of the upper level of the bi-level optimization procedure. This upper level is now detailed.

3.5.2 Upper level – durations

On the one had, the lower level of the bi-level optimization algorithm returns a set of optimal polynomial coefficients for a given TOF vector. On the other hand, the upper level is in charge of find the TOF vector minimizing the cost function (3.39).

Cost function. As the total duration of the flight plan does not matter in this context, the vector $\boldsymbol{\Delta t}$ for which the corresponding optimal trajectory $\boldsymbol{\zeta}_{\boldsymbol{\Delta t}, \mathbf{C}^*(\boldsymbol{\Delta t})}$ has the lowest cost $J_{\mathbf{C}}(\boldsymbol{\Delta t}, \mathbf{C})$ is chosen. This means that, looking back at (3.38), we have

$$J_{\boldsymbol{\Delta t}}(\boldsymbol{\Delta t}) = J_{\mathbf{C}}(\boldsymbol{\Delta t}, \mathbf{C}^*(\boldsymbol{\Delta t})) \tag{3.42}$$

which is a nonlinear cost function of the TOF vector.

Constraints. The duration of each piece of trajectory, Δt_i, is imposed strictly positive via a simple linear inequality constraint

$$\boldsymbol{\Delta t} \geqslant \boldsymbol{\varepsilon}_{\boldsymbol{\Delta t}}$$

with $\boldsymbol{\varepsilon}_{\boldsymbol{\Delta t}} = \varepsilon \, \mathbf{1}N \times 1$ and $\varepsilon \in \mathbb{R}_+^*$.

Furthermore, in order to guarantee the feasibility of the trajectory, constraints on the norms of the acceleration and the jerk are added. Gridding is used for this, meaning that these constraints are enforced on a finite set of n_{cstrt} checkpoints along the trajectory

$$\forall i \in [\![1, N]\!], \quad \forall j \in [\![0, n_{\text{cstrt}} - 1]\!] \quad \begin{cases} \left\| \ddot{\boldsymbol{\varrho}}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) \right\|_2 \leqslant a_{\max} \\ \left\| \boldsymbol{\varrho}^{(3)}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) \right\|_2 \leqslant j_{\max} \end{cases} \tag{3.43}$$

with $a_{\max}$ and $j_{\max}$ defined in (3.9) and (3.19) respectively. These constraints on the acceleration and the jerk can be formulated as quadratic constraints on the polynomial coefficients, which are the solution of a QP problem (giving the optimal polynomial coefficients for a given vector $\boldsymbol{\Delta t}$). This makes the constraints on the acceleration and the jerk nonlinear.

Optimization problem. The optimization problem to solve for the choice of the duration of each piece of trajectory can thus be formulated as follows

$$\begin{aligned} \boldsymbol{\Delta t}^* &= \arg \min_{\boldsymbol{\Delta t} \in \mathbb{R}^{1 \times N}} J_{\mathbf{C}}(\boldsymbol{\Delta t}, \mathbf{C}^*(\boldsymbol{\Delta t})) \\ \text{s.t.} & \begin{cases} -\boldsymbol{\Delta t} \leqslant -\boldsymbol{\varepsilon}_{\boldsymbol{\Delta t}} \\ \forall j \in [\![1, n_{\text{cstrt}}]\!] \quad {\mathbf{x}_{\mathbf{C}}^*}^{\top}(\boldsymbol{\Delta t}) \, \mathbf{Q}_{\text{acc},j} \, \mathbf{x}_{\mathbf{C}}^*(\boldsymbol{\Delta t}) \leqslant \mathbf{b}_{\text{acc}} \\ \forall j \in [\![1, n_{\text{cstrt}}]\!] \quad {\mathbf{x}_{\mathbf{C}}^*}^{\top}(\boldsymbol{\Delta t}) \, \mathbf{Q}_{\text{jerk},j} \, \mathbf{x}_{\mathbf{C}}^*(\boldsymbol{\Delta t}) \leqslant \mathbf{b}_{\text{jerk}} \end{cases} \end{aligned} \tag{3.44}$$

The bi-level optimization method for generating the overall trajectory is then summarized on figure 3.9, with the hilgh level at the top and the low level at the bottom.

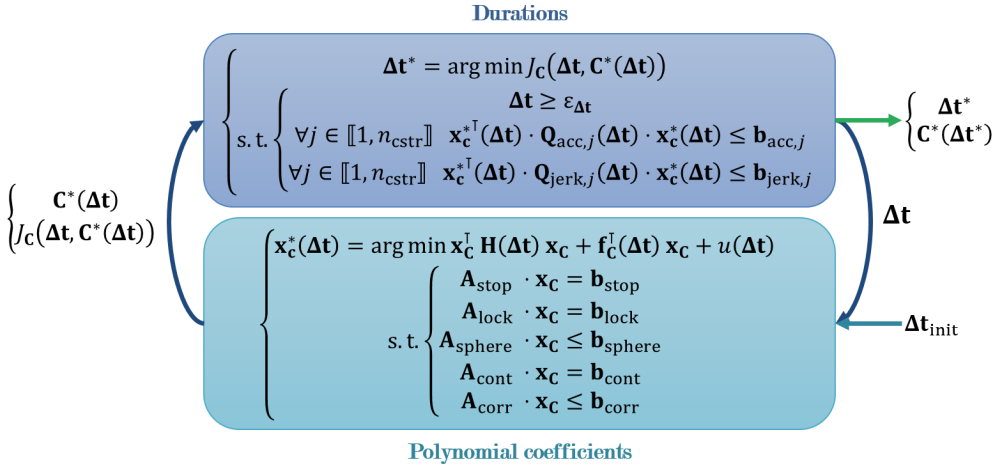

FIGURE 3.9 – Smooth speed and contouring bi-level optimization formulation

3.5.3 Numerical simulations and discussion

The proposed bi-level optimization algorithm is further tested on simple benchmarks.

Optimization parameters. A trajectory is generated using the presented bi-level strategy, for both benchmarks, with the following parameters

- **Feasibility bounds.** The thrust is bounded in the interval $[t_{\min}, t_{\max}] = [1.5, 25]\,\mathrm{m{\cdot}s^{-2}}$. The ground angle is limited to $\alpha_{\max} = 30^\circ$ and the angular speed is bounded by $\Omega_{\max} = 200^\circ{\cdot}\mathrm{s}^{-1}$. The bound on the heading variation is $\dot{\psi}_{\max} = 20^\circ{\cdot}\mathrm{s}^{-1}$.
- **Differentiability class.** Since the angular speed is bounded, so is the derivative of the attitude of the drone (see (3.6b)). The attitude of the drone is thus continuous and, given (3.6a), the acceleration of the drone should be continuous too. The trajectory is thus chosen to be $\mathcal{C}^2$, i.e. $L = 2$.
- **Polynomial degree.** Each piece of trajectory is a polynomial of degree n and is thus an element of $\mathbb{R}_n[X]$ (the vector space of polynomials with real coefficients, of degree inferior or equal to n) which is of dimension $n + 1$. The waypoints validation and/or continuity constraints impose the $L + 1$ first and last time derivatives of the position (position to L-th derivative). Each of these constraints is a linear equality constraint on the coefficients and consitutes a hyperplan of $\mathbb{R}_n[X]$. The intersection of these $2(L + 1)$ hyperplans must be non empty for the problem to be feasible. As a consequence, a sufficient condition to ensure the feasibility of the lower level optimization (polynomial coefficients) is to choose the polynomial degree of the trajectory $n \geqslant 2(L + 1)$. In order to avoid any case of unique feasible solution (and thus a non optimizable trajectory) and to keep the polynomial degree reasonably low to avoid numerical issues, a polynomial degree $n = 2(L + 1) + 1 = 7$ is chosen.
- **Weights.** The weighting terms of the cost function (3.39) are chose as follows $\mu_c = 10$, $\mu_v = 10$, $\mu_a = 1$, $\mu_j = 5$, $\mu_s = 10$.
- **Grid density.** On each piece we use 10 checkpoints for both the corridor and feasibility constraints.

A convex interior-point algorithm is used to solve lower level QP problem corresponding to the optimization of the polynomial coefficients, while a Sequential Quadratic Programming algorithm (SQP, [37]) is used to solve the nonlinear times of flight optimization.

Benchmarks. The first benchmark is a small 2D flight plan (constant altitude) containing 3 pieces, to be completed at low speed ($2\,\mathrm{m.s^{-1}}$, $3\,\mathrm{m.s^{-1}}$ and $1\,\mathrm{m.s^{-1}}$). The start and ending waypoints $\mathbf{w}_0$ and $\mathbf{w}_3$ are *stop* waypoints, the second one $\mathbf{w}_1$ is a *lock* waypoint and the third one $\mathbf{w}_2$ is an *autonext* waypoint. This benchmark is illustrated on figure 3.10.

The second benchmark is a 3D flight plan containing 9 pieces, extracted from a real flight plan. This benchmark is illustrated on figure 3.11.

Results. The results for both benchmarks are respectively presented on figures 3.12 and 3.13.

Though the trajectory generated by the cascade is smooth and verifies the acceleration and jerk constraints, this formulation of the problem appears unsatisfactory. Indeed, this first formulation was designed in a way to resemble the actual trajectories used by Parrot: a sequence of straight lines and turns on wapyoints. However, three problems made us understand that this design was not exactly adapted to our context

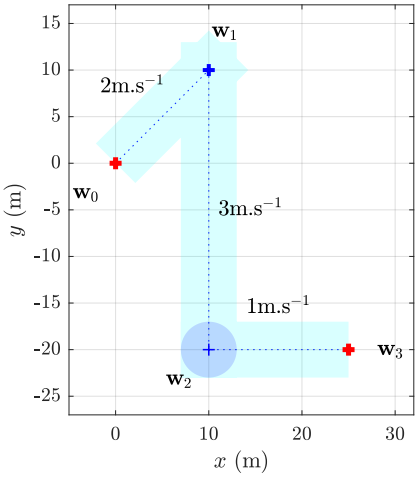

FIGURE 3.10 – Benchmark 1

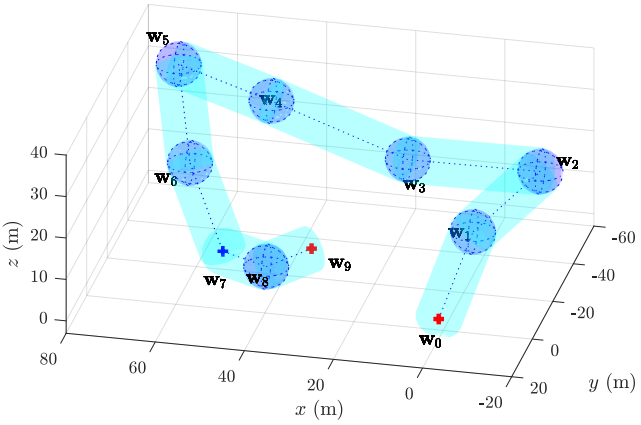

FIGURE 3.11 – Benchmark 2

- Firstly, the optimization depends on many parameters, making it difficult to find one set of these parameters giving satisfactory results for a wide variety of flight plans.
- Secondly, the reference speeds are approximately respected but the speed tends to exceed and oscillate around its reference values.
- Finally, the trajectory, even produced with an adequate set of parameters, is too "robotic", meaning that if the sequence of straigth lines and turns is easy to predict and to understand from the user point of view, it does not suit the context of video making. When comparing to human made quadrotor trajectories for video making, it appears obvious that, on the contrary, the trajectory has to be smooth and as far as possible from a robotic sequence.

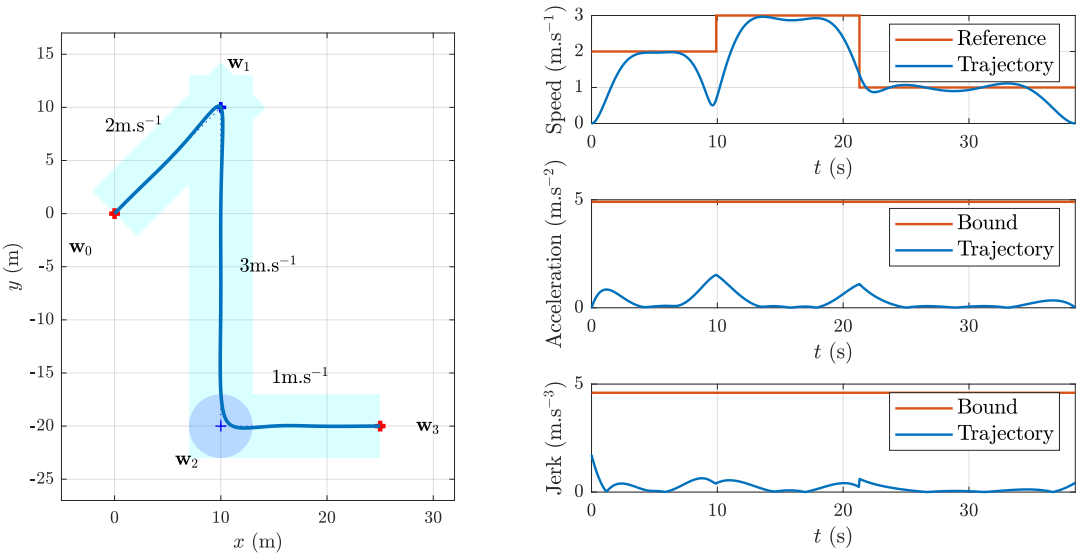

FIGURE 3.12 – Optimal trajectory for benchmark 1

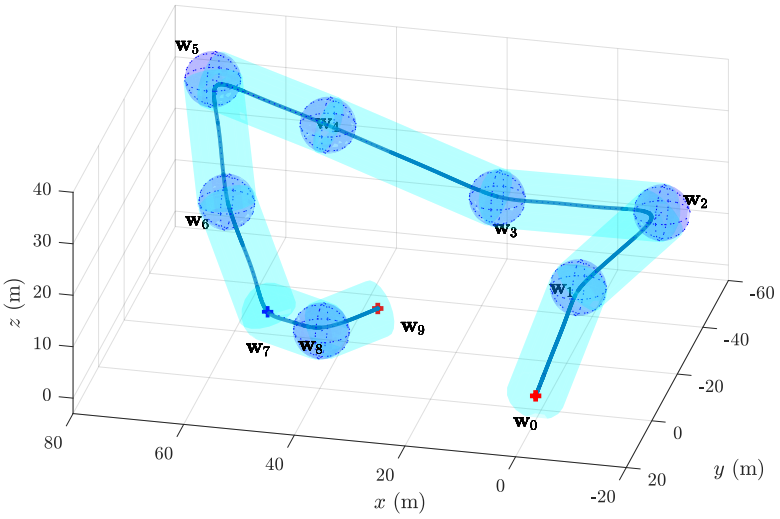

FIGURE 3.13 – Optimal trajectory for benchmark 2

In order to address these issues, we reformulate the bi-level optimization to generate minimum-time/minimum-jerk trajectories.

3.6 Minimum-time minimum-jerk trajectory

The idea behind this strategy is to seek the fastest feasible cinematographic trajectory respecting the velocity references, via a similar cascade optimization as previously

- For a given TOF vector, a minimum jerk trajectory is generated under waypoint validation, continuity and corridor constraints. By minimizing the jerk, this trajectory has minimum jolts and reduces the drone rotation speed required for the trajectory tracking (and thus minimizes the motion blur).

- The TOF vector is chosen so that it minimizes the overall duration of the mission, under the constraint that the drone velocity should not exceeds the velocity references and that the drone acceleration, angle and jerk should not exceed given limits.

3.6.1 Lower level – polynomial coefficients

Since only the jerk is minimized, the cost function is simplified to the following expression

$$J_{\mathbf{C}}(\mathbf{\Delta t}, \mathbf{C}) = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \left\| \boldsymbol{\zeta}^{(3)}_{\mathbf{\Delta t},\mathbf{C}}(t) \right\|_2^2 \mathrm{d}t \tag{3.45}$$

and constitutes a quadratic form. The same constraints of waypoint validation, continuity and flight corridor as for the previous formulation are used, leading to the following

optimization problem for the lower level of the procedure

$$\mathbf{x}_{\mathbf{C}}^{*}(\boldsymbol{\Delta t}) = \arg \min_{\mathbf{x}_{\mathbf{C}} \in \mathbb{R}^{3\,N\,(n+1)\times 1}} \mathbf{x}_{\mathbf{C}}^{\top}\,\mathbf{H}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}}$$
$$\text{s.t.} \begin{cases} \mathbf{A}_{\text{stop}}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{stop}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{lock}}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{lock}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{sphere}}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}} \leqslant \mathbf{b}_{\text{sphere}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{continuity}}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}} = \mathbf{b}_{\text{continuity}}(\boldsymbol{\Delta t}) \\ \mathbf{A}_{\text{corridor}}(\boldsymbol{\Delta t})\,\mathbf{x}_{\mathbf{C}} \leqslant \mathbf{b}_{\text{corridor}}(\boldsymbol{\Delta t}) \end{cases} \tag{3.46}$$

3.6.2 Upper level – durations

Cost function and constraints. Since the lower level optimization loop does not take into account the reference velocities, the upper level loop will try to reduce the duration of each piece of trajectory as much as possible, but such that the drone speed does not exceed its references. The overall duration of the trajectory is used as cost function

$$J_{\boldsymbol{\Delta t}}(\boldsymbol{\Delta t}) = \sum_{i=1}^{N} \Delta t_i \tag{3.47}$$

which is linear. The respect of the reference speed is added as a hard constraint

$$\forall i \in [\![1, N]\!], \quad \forall j \in [\![0, n_{\text{cstrt}} - 1]\!] \quad \left\| \dot{\boldsymbol{\varrho}}_{\mathbf{C}_i^{*}(\boldsymbol{\Delta t})}(t_{i,j}) \right\|_2 \leqslant \nu_i \tag{3.48}$$

It can be noticed that this is a better formulation as it consider the speed rather than velocity, as in the previous formulation.

Optimization problem. The optimization problem for the choice of the duration of each piece of trajectory can thus be formulated as follows

$$\boldsymbol{\Delta t}^{*} = \arg \min_{\boldsymbol{\Delta t} \in \mathbb{R}^{1\times N}} \mathbf{1}_{1\times N} \cdot \boldsymbol{\Delta t}$$
$$\text{s.t.} \begin{cases} -\,\boldsymbol{\Delta t} \leqslant -\boldsymbol{\varepsilon}_{\boldsymbol{\Delta t}} \\ \forall j \in [\![1, n_{\text{cstrt}}]\!] \quad \mathbf{x}_{\mathbf{C}}^{*}{}^{\top}(\boldsymbol{\Delta t})\,\mathbf{Q}_{\text{speed},j}\,\mathbf{x}_{\mathbf{C}}^{*}(\boldsymbol{\Delta t}) \leqslant \mathbf{b}_{\text{speed}} \\ \forall j \in [\![1, n_{\text{cstrt}}]\!] \quad \mathbf{x}_{\mathbf{C}}^{*}{}^{\top}(\boldsymbol{\Delta t})\,\mathbf{Q}_{\text{acc},j}\,\mathbf{x}_{\mathbf{C}}^{*}(\boldsymbol{\Delta t}) \leqslant \mathbf{b}_{\text{acc}} \\ \forall j \in [\![1, n_{\text{cstrt}}]\!] \quad \mathbf{x}_{\mathbf{C}}^{*}{}^{\top}(\boldsymbol{\Delta t})\,\mathbf{Q}_{\text{jerk},j}\,\mathbf{x}_{\mathbf{C}}^{*}(\boldsymbol{\Delta t}) \leqslant \mathbf{b}_{\text{jerk}} \end{cases} \tag{3.49}$$

The bi-level optimization method for generating the overall trajectory is then summarized on figure 3.14.

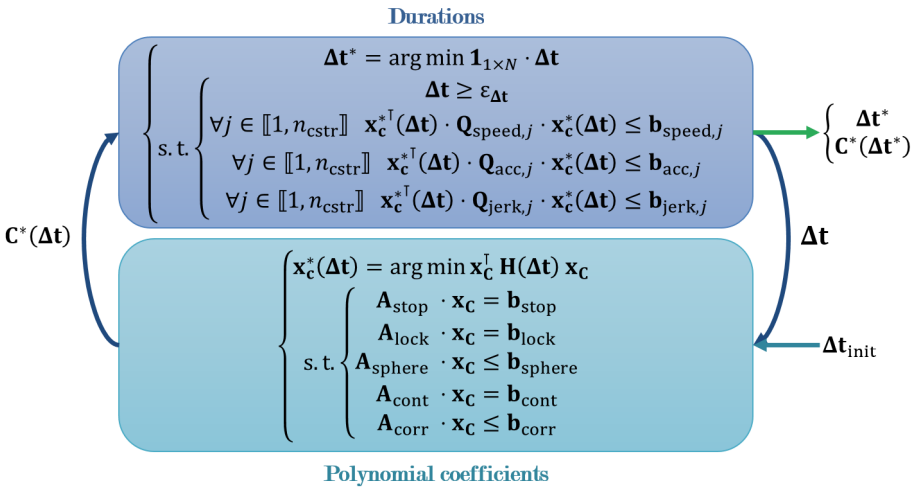

FIGURE 3.14 – Minimum-time/minimum-jerk bi-level optimization formulation

3.6.3 Simulations and discussion

The strategy is confronted to the 2 benchmarks presented in section 3.5.3. The results are presented on figures 3.15 and 3.16. In both cases the obtained trajectory is much smoother and seems much more natural and less robotic than with the previous formulation.

The results still show some direction of improvement

- The trajectory on benchmark 1 is quite slow, while the bounds on acceleration and jerk are far to be (see figure 3.15, right). In other words, the trajectory appears suboptimal. Actually, it can be seen that the speed is the limiting factor as a speed constraint is active on the 3rd piece of trajectory. This is inherent to this formulation of the problem. Indeed, the lower level optimization (polynomial coefficient) only minimizes the jerk and does not contain any consideration on the speed.

- The size of the problem to solve is given by the number of pieces of trajectory, and can be significant for large flight plans.

- At high speed, the air drag can significantly decrease the acceleration capacity of the drone, and the proof of feasibility of the trajectory given in section 3.3 might not hold.

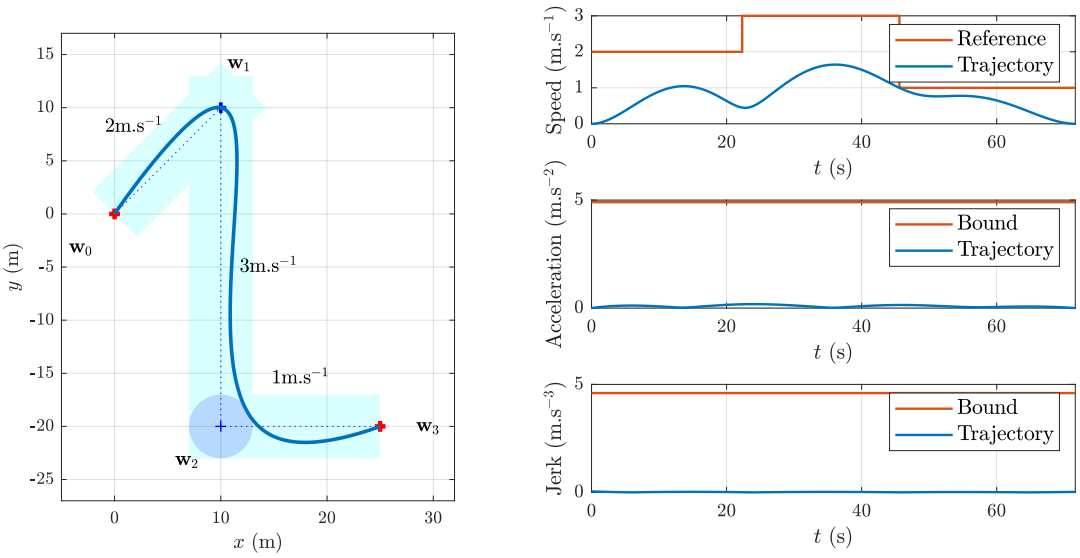

FIGURE 3.15 – Minimum-time/minimum-jerk trajectory for benchmark 1

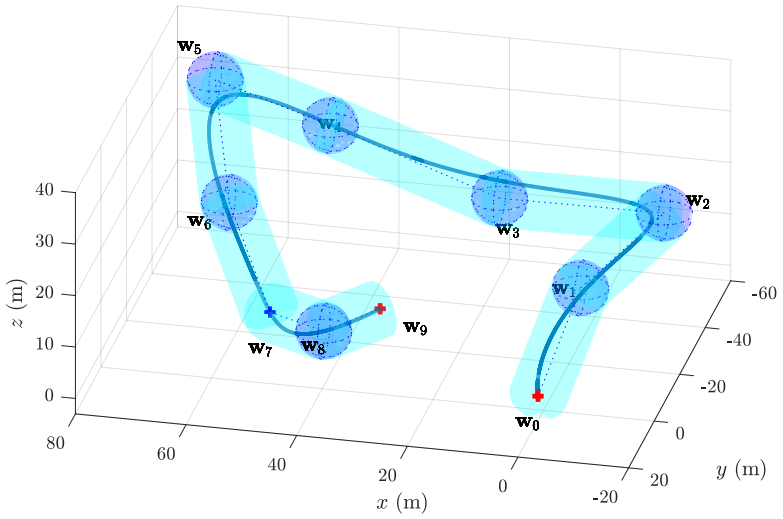

FIGURE 3.16 – Minimum-time/minimum-jerk trajectory for benchmark 2

In order to overcome this weakness, the next section proposes some improvements to this strategy.

3.7 Improvements

3.7.1 Linear drag model

In order to improve the feasibility of the trajectory at high speed, the model (3.6) is augmented with a linear drag action, in order to represent the loss of acceleration capacity

when the speed increases. Equation (3.6a) is thus replaced by

$$\ddot{\boldsymbol{\zeta}} = g\,\mathbf{z}_{\mathcal{W}} - t\,\mathbf{R}\,\mathbf{z}_{\mathcal{W}} - \lambda\,\dot{\boldsymbol{\zeta}} \tag{3.50}$$

and the constraint on the acceleration in (3.49) becomes

$$\forall i \in [\![1, N]\!], \quad \forall j \in [\![0, n_{\text{cstrt}} - 1]\!] \quad \left\| \ddot{\boldsymbol{\varrho}}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) + \lambda \dot{\boldsymbol{\varrho}}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) \right\|_2 \leqslant a_{\max} \tag{3.51}$$

Furthermore, if a measurement of the wind velocity is available, the constraint can even be rewritten to include a more accurate representation of the drag

$$\forall i \in [\![1, N]\!], \quad \forall j \in [\![0, n_{\text{cstrt}} - 1]\!] \quad \left\| \ddot{\boldsymbol{\varrho}}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) + \lambda \left(\dot{\boldsymbol{\varrho}}_{\mathbf{C}_i^*(\boldsymbol{\Delta t})}(t_{i,j}) - \mathbf{v}_{\text{wind}} \right) \right\|_2 \leqslant a_{\max} \tag{3.52}$$

with $\mathbf{v}_{\text{wind}}$ the mean value of the wind velocity relatively to the ground. This supposes that the wind does not vary too much and is low enough for a feasible trajectory to exists

Benchmark 3. In order to illustrate the impact of this augmented formulation, it is tested on a simple benchmark, presented on figure 3.17. The flight plan is such that the acceleration is the limiting constraint rather than the reference speed or the jerk limitation. The trajectory and its derivatives are shown on figure 3.17, the blue line corresponds to the result obtained without considering the drag, by solving the problem (3.49), while the green line corresponds to the result including the drag model (3.51).

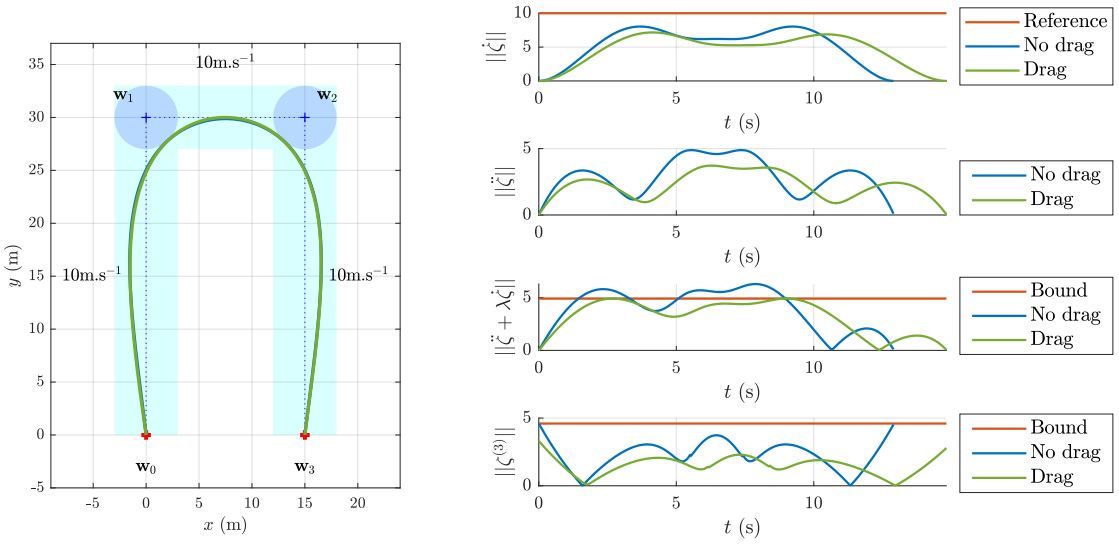

FIGURE 3.17 – Minimum-time/minimum-jerk trajectory with and without drag for benchmark 3

For the green curve, the acceleration constraint is given by (3.50), while it is only given by (3.43) for the blue one. This leads to an excessive value of the acceleration at high speed for the blue curve, when the drag is considered, which can threaten the feasibility of the trajectory.

A wind of $5\,\text{m}\cdot\text{s}^{-1}$ oriented along $\mathbf{y}_{\mathcal{W}}$ is now added to the benchmark 3. The result is illustrated on figure 3.18. The two trajectories represented include the drag but only the green one includes the wind velocity measurement (3.52).

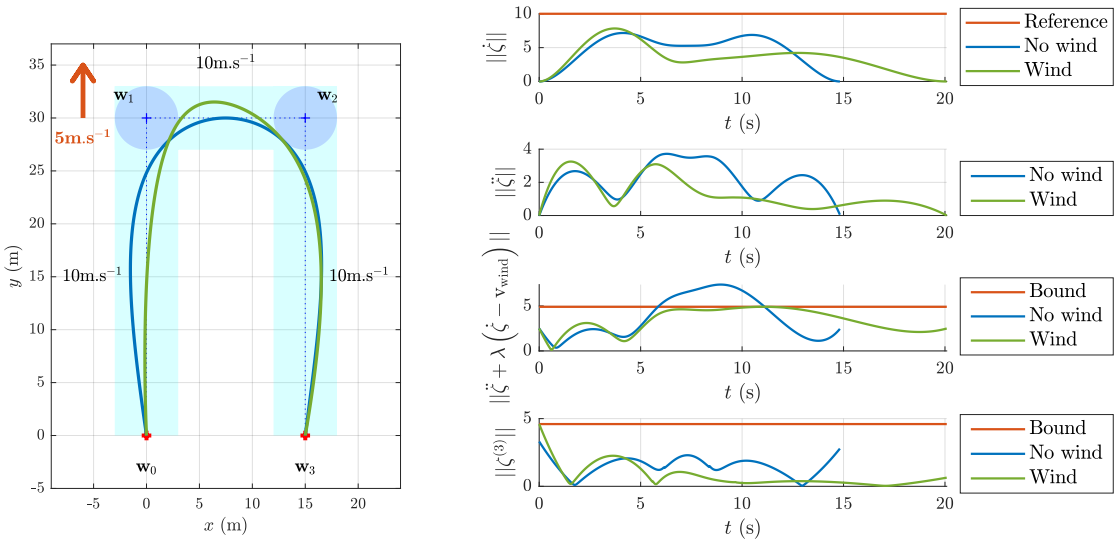

FIGURE 3.18 – Minimum-time/minimum-jerk trajectory with and without wind for benchmark 3

At the beginning of the flight plan, the wind is actually helping the drone to accelerate, which is taken into account on the green curve, with a higher peak acceleration and speed, while respecting the constraints. On the contrary, at the end of the flight plan, the drone has to fight the wind which leads to a lower achievable acceleration for a given speed. For the blue curve, not taking the wind into account leads to an infeasible trajectory.

3.7.2 Initialization

Usually, a starting point is required for solving the optimization problem (3.49), i.e. a starting value for each time of flight Δt_i. The three following methods to initialize the algorithm have been proposed and tested

- **Arbitrary value.** A fixed, arbitrary value

$$\forall i \in [\![1, N]\!] \quad \Delta t_i = cst \tag{3.53}$$

- **Constant speed.** An estimated obtained by supposing the speed constant equal to the reference on each piece of trajectory

$$\forall i \in [\![1, N]\!] \quad \Delta t_i = \frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{\nu_i} \tag{3.54}$$

  This initial guess is expected to be lower than the optimal value, as any accelerating and braking phases are neglected.

- **Bang-off-bang acceleration.** An estimated obtained with a rest-to-rest, Bang-Off-Bang (BOB) acceleration profile between each waypoint, i.e. a trajectory that stops at each waypoint and with a trapezoidal speed profile

$$\forall i \in [\![1, N]\!] \quad \Delta t_i = \begin{cases} 2\sqrt{\frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{a_{\max}}} & \text{if } \frac{\nu_i^2}{a_{\max}} > \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 \\ \frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{\nu_i} + \frac{\nu_i}{a_{\max}} & \text{otherwise} \end{cases} \tag{3.55}$$

This initial guess is expected to be larger than the optimal value, as the trajectory stops at each waypoint.

The results in terms of number of iterations to solve the problem (3.49) with a SQP as well as the final cost of the trajectory (overall duration) are presented in table 3.1 for each initialization method, for benchmarks 1 and 2.

Method	Benchmark 1		Benchmark 2	
	Iterations	Cost	Iterations	Cost
Arbitrary	36	70.82	31	121.18
Constant speed	16	71.64	19	121.18
BOB acceleration	27	71.64	45	121.18

TABLE 3.1 – Comparison of the initialization methods

As expected, the arbitrary method does not yield the best results in terms of number of iterations, obtained with the constant speed model. The latter result can be explained by the conservatism of the BOB acceleration method, which supposes that the trajectory stops at each waypoints, leading to largely overestimated times of flights. However, the arbitrary initialization returns the trajectory with a slightly better cost for the benchmark 1 ($< 1.2\%$ better than with the two other methods), which could be explained by a local minimum of the cost function. This difference does not appear on the benchmark 2.

3.7.3 Waypoint horizon

In order to reduce the size of the optimization problem (3.49), we suggest a receding horizon methodology. 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 pieces of the overall flight plan.

For a horizon $N_{\mathbf{w}}$ and a flight plan containing at least $N_{\mathbf{w}}+2$ waypoints, the trajectory is first computed between the first and the $(N_{\mathbf{w}}+1)$-th waypoints (i.e. $\mathbf{w}_0$ to $\mathbf{w}_{N_{\mathbf{w}}}$). This results in a piecewise trajectory $\boldsymbol{\zeta}_1$ containing $N_{\mathbf{w}}$ pieces. The first piece of this trajectory corresponds to the trajectory joining $\mathbf{w}_0$ to $\mathbf{w}_1$. As the trajectory does not necessarily pass on the waypoint $\mathbf{w}_1$ (for *sphere* waypoints), we call κ_1 the ending position of this first piece of trajectory (see figure 3.19) and $\kappa_1^{(l)}$ the l-th time derivative of the position on this point ($l \in [\![1, L]\!]$). Only this first piece of trajectory is kept and will constitute the first piece of the overall, final trajectory $\boldsymbol{\zeta}$. According to the receding horizon strategy, the horizon is then shifted by one waypoint and a new trajectory $\boldsymbol{\zeta}_2$ is computed, having κ_1 as starting waypoint $\mathbf{w}_1$. This new starting waypoint is of special type *constrained*, meaning that the position and its first L time derivatives are imposed, in this case equal to κ_1, $\kappa_1^{(1)}$, ..., $\kappa_1^{(L)}$. Again, only the first piece of this new trajectory $\boldsymbol{\zeta}_2$ is kept and it constitutes the second piece of the final trajectory $\boldsymbol{\zeta}$. The continuity at the connection between the two trajectory pieces is ensured by the constraints on the position and its derivatives on κ_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. The next step is then initialized with the previous solution, plus a feasible rest-to-rest trajectory for the new piece of trajectory (there always exists one).

Example 3 *A flight plan containing 5 waypoints* $\{\mathbf{w}_0, \ldots, \mathbf{w}_4\}$ *and a horizon* $N_{\mathbf{w}} = 3$ *is*

illustrated in figure 3.19. The trajectory is computed in 2 steps: between the waypoints $\mathbf{w}_0$ *and* $\mathbf{w}_3$ *first, and between* $\mathbf{w}_1$ *and* $\mathbf{w}_4$ *next.*

Step 1: A trajectory is generated between $\mathbf{w}_0$ *and* $\mathbf{w}_3$*, with* $\mathbf{w}_3$ *replaced by a stop waypoint at the same position.*

Step 2: A trajectory is generated between the waypoints $\mathbf{w}_1$ *and* $\mathbf{w}_4$*, with the waypoint* $\mathbf{w}_3$ *of type sphere, as it was originally, and the waypoint* $\mathbf{w}_1$ *replaced by a constrained waypoint at the position* κ_1*, ensuring the continuity with the trajectory computed at Step 1.*

Since the new trajectory reaches the final waypoint $\mathbf{w}_4$*, there is no need to repeat the process.*

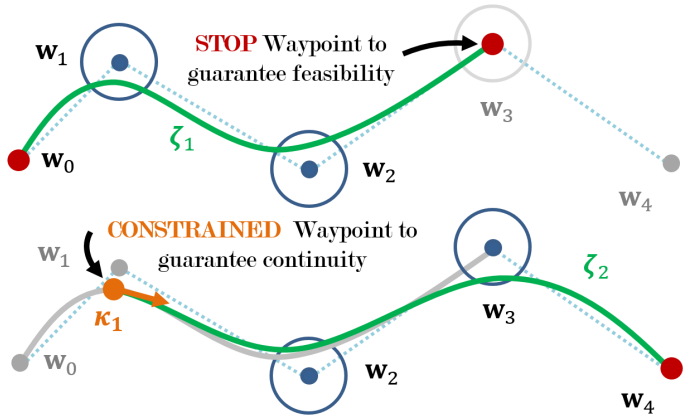

FIGURE 3.19 – Receding waypoint horizon

For on-line computation, the trajectory could for instance be updated each time a waypoint is validated. A constraint on the time of flight on the first piece of trajectory can also be added in the optimization problem (3.49), so that it is greater than the average trajectory computation time plus a security margin.

The gain in terms of computation time provided by this receding horizon strategy was evaluated by generating a trajectory for the benchmark 2, first without waypoint horizon and then with a waypoint horizon $N_{\mathbf{w}} = 3$. The test was repeated several times and the mean values of the time to compute the overall trajectory and the times to compute each step were saved. These values are given in table 3.2, first the mean value to compute each step is given and then the mean time to compute the overall trajectory. These values are normalized by the computation time without horizon. Not only the waypoint horizon allows computing small steps much faster than the overall trajectory, but the overall computation time is also reduced.

	No horizon, 1 step		Horizon $N_{\mathbf{w}} = 3$, 7 steps	
	step	total	step	total
computation time	1	1	0.034	0.24

TABLE 3.2 – Computation time with and without waypoint horizon on benchmark 2

This comes at a cost however, in terms of optimality of the overall trajectory. The loss of optimality of the final trajectory induced by this strategy is illustrated on figure 3.20, where a given flight plan is processed with different horizons $N_{\mathbf{w}} \in \{2, 3, 4\}$. This loss is negligible for a sufficiently large horizon.

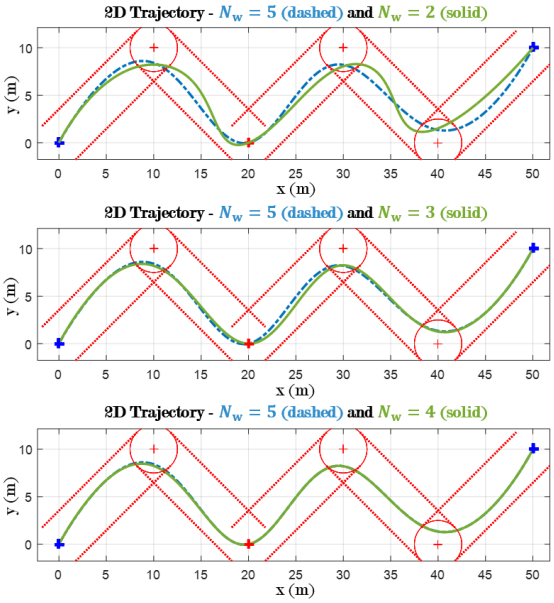

FIGURE 3.20 – Impact of the waypoint horizon on the optimality of the trajectory

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, especially if they are aligned.

3.7.4 Suboptimal trajectory

The trajectory obtained with this bi-level algorithm is smooth and feasible. However, as it can be seen on figure 3.15, the trajectory appears largely suboptimal in term of duration: the acceleration and jerk constraint are far from being active anywhere, meaning that the drone could have completed the flight plan much faster. Actually, in the example on figure 3.15, only one constraint on the speed is active, at the connection between the second and the third pieces of trajectory, i.e. at the time instant $t = t_2$. This is due to the lower level optimization, on the polynomial coefficients, having no information on the bounds on the derivatives, it only generates a continuous trajectory with minimum jerk on its overall duration. On figure 3.15, for the same TOF vector, the lower level optimization could adjust the polynomial coefficients so that the speed is smaller at t_2 and larger at $t = 60\,\text{s}$ for instance. This would move the time at which the peak speed is reached on pieces 2 and/or 3 so that the trajectory has the same duration as on figure 3.15 but with no active constraint. This would let the upper level optimization reducing the times of flight, leading to a better solution. Unfortunately, with this minimum-time/minimum-jerk bi-level formulation of the optimization problem, the lower level optimization has no reason to do so as it does not take such consideration into account.

In this work, the goal is not to generate a trajectory with minimum duration as it could be the case for racing or delivery missions for instance, but a trajectory suited for cinematography. The minimization of the duration is only one way to have the speed get closer to its reference during the flight. As a consequence, the issue of suboptimal duration explained above could be ignored. After all, the trajectory on figure 3.15 is smooth, feasible, respects the requirements identified in section 1.2.3 for cinematography and though duration could be smaller, it is still very acceptable.

Unfortunately, this suboptimality can become an issue for waypoints separated by a large distance, to be joined at low speed. Such a situation can be encountered for some timelapses[1] for instance. This issue is illustrated on figure 3.21, where a trajectory is generated to complete a trivial flight plan containing 2 waypoints separated by 100 m. The speed reference on the unique piece of trajectory to compute is $1\,\mathrm{m{\cdot}s^{-1}}$. Neither the drag nor the wind are taken into account for this test.

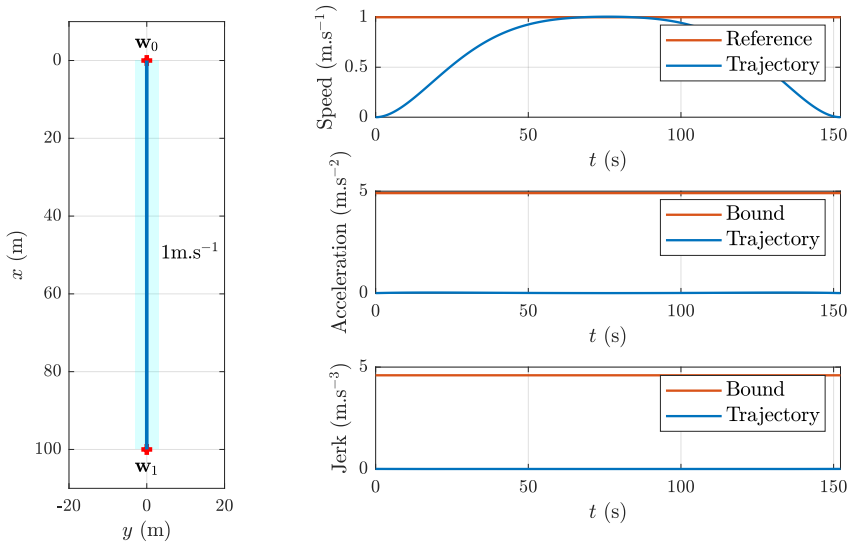

FIGURE 3.21 – Intrinsic limitation of the bi-level formulation

The drone takes more than 50 s to reach its peak speed, of only $1\,\mathrm{m{\cdot}s^{-1}}$, for a total mission duration of more than 150 s. At $1\,\mathrm{m{\cdot}s^{-1}}$, it would have been expected to see the 100 m covered in approximately 100 s with the reference speed reached in a few seconds. This large suboptimality is caused by the reasons given above, and can especially become an issue for large flight plans, since the duration of the mission is limited by the battery.

In response to this limitation of the proposed algorithm, an alternative approach has been tested, with the bounds on the derivatives also included in the lower level optimization, on the polynomial coefficients. In order to prevent any infeasibility of the problem, these constraints have been formulated as soft constraint, should the upper level optimization (durations) try a TOF vector that would require to exceed the bounds on the derivatives. The results have shown that this modification can improve the speed profile, but at the expanse of a less smooth and natural curve, less suited for cinematography, as illustrated on figure 3.22. In this figure, the soft constraint strategy is applied to the benchmark 1, leading to a better respect of the speed references as well as a decrease of almost 10 s of the total duration of the mission when compared to figure 3.15.

Furthermore, this extension of the algorithm has revealed less robust to numerical issues and the penalty on the soft constraints arduous to tune. For these reasons, this soft constraint formulation has been abandoned. A solution to overcome this suboptimal duration issue is presented in the next chapter, through a single-level optimization of a B-spline trajectory.

[1] A timelapse is a footage performed at low frame rate, which is then read with a normal framerate. This results in an accelerated video.

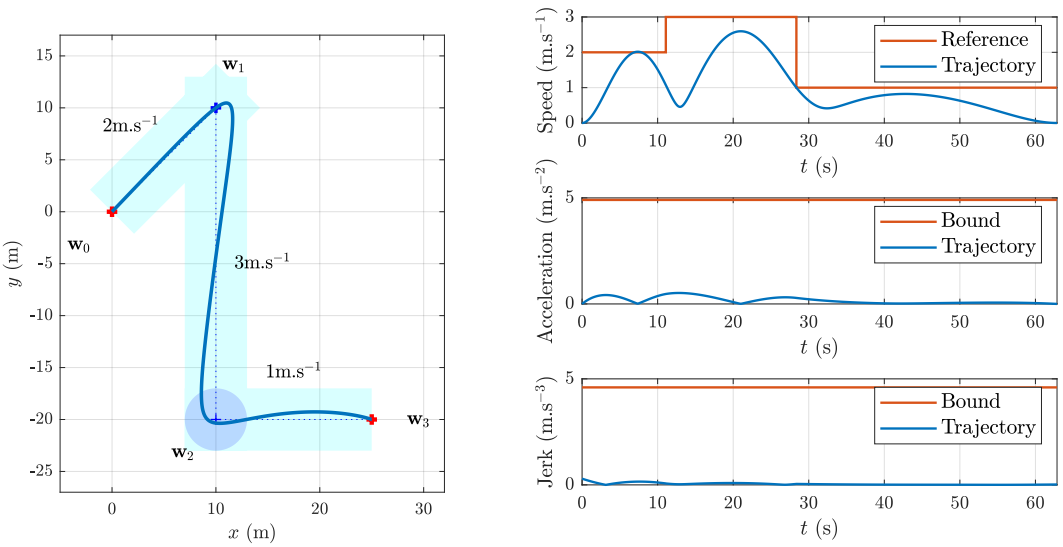

FIGURE 3.22 – Improvement of the speed profile with soft constraints in the lower level optimization for benchmark 1

3.8 Conclusion

In this chapter, a novel method to generate cinematographic quadrotor trajectories have been proposed, by improving and extending the bi-level trajectory generation procedure [75], [101], [102], to the case of cinematography. The trajectory is parameterized as a piecewise polynomial, with given polynomial coefficients and a given duration for each piece.

Our first contribution lies in a new approach to characterize the feasibility of a quadrotor trajectory, suited for cinematography.

Then, the bi-level optimization strategy for generating piecewise polynomial trajectories is adapted to the specific requirements of this work. The proposed trajectory generation method thus consists in solving a bi-level optimization on the polynomial coefficients and the durations of the polynomial pieces, under the constraints of waypoints validation, flight corridor satisfaction, and feasibility guarantee. In particular, we proposed to minimize the duration of the mission by optimizing the duration of each piece of trajectory independantly. The cinematographic requirements are met through the minimization of the jerk as well as constraints on the duration of each polynomial piece.

Another contribution lies in the suggestion of new improvements of the technique, such as different methods of initilization, the inclusion of the drag and the wind into the model or the addition of soft constraint into the lower level of the bi-level procedure for improving the speed profile.

Overall, the method returns smooth, natural and feasible trajectories. The method performed well confronted to an outdoor experiment and the results were published in [105]. The video of the flight is available at https://www.youtube.com/watch?v=IR9Ablo-ryI. However, singular cases have been highlighted for which the bi-level strategy tends to return overly slow trajectories which can be an issue. Another weakness of this method is the use of gridding to enforce the corridor and feasibility constraints, which does not

guarantee their respect over the entire trajectory.

For these reasons, we propose in the next chapter a new strategy which fixes both these issues.

Chapter 4

Guidance – minimum-time B-spline trajectories

4.1 Introduction

Though it has been successfully applied on a Parrot Bebop 2, the bi-level formulation presented in chapter 3 still lacks some guarantees in the respect of the constraints (corridor and feasibility constraints) and a speed profile closer to the reference speeds. We were also able to highlight some numerical robustness issues of the bi-level optimization algorithm on singular flight plans. As explained in section 3.7.4, the tendency of the bi-level formulation of chapter 3 to produce too slow trajectory is inherent to this formulation. This is also the case of the numerical issues that have been encountered. One way to overcome both inconvenients is to merge the 2 optimization levels, i.e. to abandon the bi-level formulation.

Concerning the guarantee on the constraints satisfaction, one elegant way to achieve this is to parameterize the trajectory by the mean of B-spline curves. Due to their intrinsic property to constrain the curve and its derivatives inside convex regions, the use of B-spline curves to parameterize the trajectory is popular for trajectory generation, in a wide variety of applications [125], [114], [94], [80], [78], [79]. These curves have also recently spread in the domain of aerial robotics [123], [115], [124], [77], [88]. They can be seen as an extension of the Bézier curves, which have already been used for trajectory generation [24], [53] and are piecewise polynomials parameterized by a set of control points and a vector of knots, defining when the curve switches between two polynomial representations. The piecewise nature of these curve is another advantage as it allows us to keep the polynomial degree low, while still being able to parameterize rich trajectories.

Most of the time, a particular kind of B-spline is used, i.e. uniform B-spline (also called cardinal B-spline), for which the knots are equally spaced. Their advantage is that they can simplify some of the calculation and allow some of them to be preformed off-line. This can significantly speed up the B-spline generation process which is useful for embedded systems. However, one drawback is that it is difficult to generate time-optimal trajectories using uniform B-splines, and other metrics are usually optimized rather than the duration for their generation (e.g. the length or the root mean square of a derivative). This can complexify the generation of trajectories with both the shape and speed profile suited for cinematography. In [115] it is suggested to optimize the number of control points, which can help improving the results for cinematography, but requires the resolution of a Mixed Integer Programming (MIP) problem.

In this chapter, we propose a new method to generate minimum-time B-spline trajectories for aerial cinematography which avoids the use of a bi-level optimization and only minimizes the duration of the trajectory. This allows generating trajectories with better speed profiles and improves the numerical stability of the algorithm. Secondly, the framework of B-splines is used in order to guarantee the validation of the constraints over the entire trajectory, rather than on a finite set of points (gridding).

In order to facilitate the comprehension of trajectory generation algorithm proposed in this chapter, we first introduce useful notations and preliminary results about B-splines and clamped B-splines in section 4.2. We then present a novel, compact way to represent a piecewise clamped B-spline trajectory in section 4.3, used in section 4.4 to formulate the trajectory generation as an optimization problem. The algorithm is then confronted to the same benchmarks as in chapter 3 to compare its performance with the bi-level formulation in section 4.5. Finally, a study on the initialization of the of the optimization problem is proposed in section 4.6.

4.2 Overview on B-splines

Basis-spline curves (commonly called B-spline curves) are often considered as an extension of the Bézier curves. The latter can be seen as one way to parameterize polynomials as convex combinations of control points, weighted by Bernstein polynomials. In a similar fashion, B-spline curves can be seen as one way to parameterize piecewise polynomials as convex combinations of control points, weighted by B-spline functions. This is useful since piecewise polynomials allow to parameterize rich trajectories while keeping the polynomial degree low, thus avoiding potential issues such as oscillations ("wiggling") or numerical instability.

B-spline curves are piecewise polynomial curves described by

- A set of $(n+1)$ control points, which have a similar role as the polynomial coefficients
- A polynomial degree k
- A set of $(m+1)$ knots defining where the curve switches between two polynomial representations

An example of such a B-spline curve of degree 2 is presented on figure 4.1, with the control points $\{\mathbf{p}_i\}_{i\in[\![0,4]\!]}$ in black and each polynomial piece of the curve in a different color.

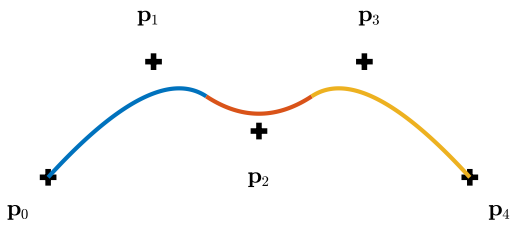

FIGURE 4.1 – Example of B-spline curve

In order to facilitate the comprehension of this chapter, this section gives a quick overview about B-spline curves and introduces notations and preliminary results that will be used

in the other sections. It starts with notions on B-spline functions, used as weights on the control points. For more details, [29] and [95] provide in-depth presentations of these objects.

4.2.1 B-spline functions

Given a polynomial degree $k \geqslant 0$ and a vector of $(m+1) > k+1$ increasing knots, represented by the matrix

$$\boldsymbol{\tau} = (\tau_0 \ \tau_1 \ \dots \ \tau_m) \in \mathbb{R}^{1\times(m+1)} \tag{4.1}$$

a set of $(n+1) = m-k$ B-spline functions $\left\{N^{\boldsymbol{\tau}}_{i,k}\right\}_{i\in[\![0,n]\!]}$ can be defined, using the following recurrence ([29], [95])

$\forall i \in [\![0,n]\!], \quad \forall t \in \mathbb{R}$

If $k = 0$

$$N^{\boldsymbol{\tau}}_{i,k}(t) = \begin{cases} 1 \ \text{ if } \tau_i \leqslant t < \tau_{i+1} \\ 0 \ \text{ otherwise} \end{cases} \tag{4.2a}$$

Else

$$N^{\boldsymbol{\tau}}_{i,k}(t) = \omega^{\boldsymbol{\tau}}_{i,k}(t)\ N^{\boldsymbol{\tau}}_{i,k-1}(t) \ + \ \left(1-\omega^{\boldsymbol{\tau}}_{i+1,k}(t)\right) N^{\boldsymbol{\tau}}_{i+1,k-1}(t) \tag{4.2b}$$

with

$$\omega^{\boldsymbol{\tau}}_{i,k}(t) = \begin{cases} \dfrac{t-\tau_i}{\tau_{i+k}-\tau_i} & \text{if } \ \tau_{i+k} > \tau_i \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Each B-spline function is then a piecewise polynomial that switches between 2 polynomial representations at each knot. Notice that, following the definition, the support of $N^{\boldsymbol{\tau}}_{i,k}$ is $[\tau_i, \tau_{i+k+1}[$. Outside of this domain, $N^{\boldsymbol{\tau}}_{i,k}$ is null.

These B-spline functions define a partition of unity over the interval $[\tau_k, \tau_{n+1}[$ ([29], [95])

$$\begin{gathered} \forall i \in [\![0,n]\!] \ \ N^{\boldsymbol{\tau}}_{i,k} \geqslant 0 \\ \forall t \in [\tau_k, \tau_{n+1}[\ \ \sum_{i=0}^{n} N^{\boldsymbol{\tau}}_{i,k}(t) = 1 \end{gathered} \tag{4.4}$$

Figure 4.2 illustrates an example of B-spline functions of degree 2 with 8 knots. The thick line represents the sum of the basis functions and the light blue rectangle represents the domain over which the B-spline functions form a partition of unity.

The B-spline functions can be used to build curves, as explained in the following paragraph.

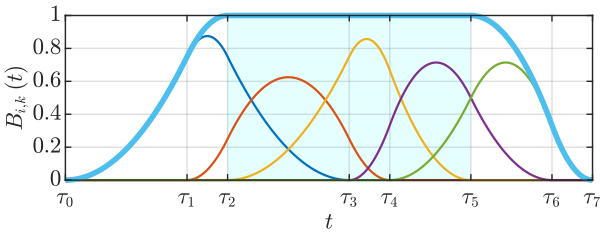

FIGURE 4.2 – Example of B-spline functions

4.2.2 B-spline curves

By introducing a set of $(n+1)$ control points $\{\mathbf{p}_i\}_{i\in[\![0,n]\!]}$ represented by

$$\begin{aligned} &\forall i \in [\![0,n]\!] \quad \mathbf{p}_i = \begin{pmatrix} p_i^x & p_i^y & p_i^z \end{pmatrix}^\top \in \mathbb{R}^{3\times 1} \\ &\mathbf{P} = \begin{pmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \dots & \mathbf{p}_n \end{pmatrix} \in \mathbb{R}^{3\times(n+1)} \end{aligned} \tag{4.5}$$

a B-spline curve $\boldsymbol{\mathcal{B}}$, of knots $\boldsymbol{\tau}$ and control points $\mathbf{P}$, can be defined as follows

$$\boldsymbol{\mathcal{B}} = \sum_{i=0}^{n} N_{i,k}^{\boldsymbol{\tau}}\, \mathbf{p}_i \tag{4.6}$$

Remark 10 *Notice that a knot vector and a set of control points suffice to define a B-spline curve as its polynomial degree k is then set by $k = m - n - 1$ (see section 4.2.1).*

Remark 11 *Though 3 dimensional control points are used in this chapter, as they will be used to parameterize the position of a quadrotor, their definition is not restricted to $\mathbb{R}^3$ in the general case.*

The evaluation of the curve at $t \in \mathbb{R}$ is then the combination of the control points weighted by the evaluation of the B-spline functions at t. Since these functions are piecewise polynomials, the B-spline curve $\boldsymbol{\mathcal{B}}$ is a piecewise polynomial curve, switching between two polynomial representations at each knot.

Due to (4.4), in the interval $[\tau_k, \tau_{n+1}[$, a B-spline curve verifies the so-called convex hull property ([29], [95]).

Property 1 *For $i \in [\![k,n]\!]$ and $t \in [\tau_i, \tau_{i+1}[$*

$$\boldsymbol{\mathcal{B}}(t) \in \text{Conv}(\ \{\mathbf{p}_j \mid j \in [\![i-k, i]\!]\}\ ) \tag{4.7}$$

As a consequence, the curve lies within the convex hull of the entire set of control points in $[\tau_k, \tau_{n+1}[$. Though this consequence is weaker than property 1, it will be used in section 4.4 to constrain B-spline curves into convex regions. It can then be convenient to separate the knot vector into

- k external knots at the beginning of the knot vector and k external knots at the end
- A starting knot τ_k and an ending knot τ_{n+1}, which define the domain over which the convex hull property holds

- $n-k$ internal knots, which are the instants for which the B-spline curve switches between two polynomial representations, inside the interval $[\tau_k, \tau_{n+1}[$

$$\underbrace{(\tau_0 \ \dots \ \tau_{k-1}}_{k \text{ knots}} \ \underbrace{\tau_k}_{\substack{\downarrow \\ \text{starting} \\ \text{knot}}} \ \underbrace{\tau_{k+1} \ \dots \ \tau_n}_{\substack{n-k \\ \text{internal knots}}} \ \underbrace{\tau_{n+1}}_{\substack{\downarrow \\ \text{ending} \\ \text{knot}}} \ \underbrace{\tau_{n+2} \ \dots \ \tau_m)}_{k \text{ knots}} \tag{4.8}$$

Figure 4.3 illustrates an example of a B-spline curve of degree 2, with the same knot vector as figure 4.2. The thick black dots are the control points and the light blue polygon is the interior of their convex hull. The green line is the B-spline curve, evaluated inside (continuous) and outside (dashed) the domain $[\tau_k, \tau_{n+1}[$.

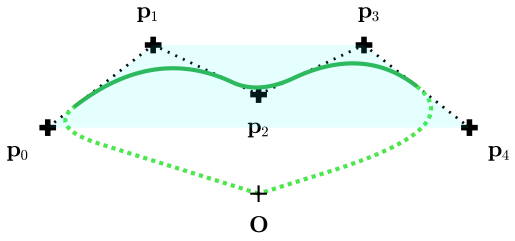

FIGURE 4.3 – Convex hull of a B-spline of degree 2

The following paragraph details a specific case of B-spline curves, that is later used to parameterize trajectories.

4.2.3 Clamped B-spline

In this work the trajectory is parameterized as a clamped B-spline, i.e. a B-spline whose first k knots are equal to the starting knot τ_k, and last k knots are equal to the ending knot τ_{n+1}. The knot vector thus has the following form

$$\underbrace{(\tau_k \ \dots \ \tau_k}_{\substack{k+1 \\ \text{equal knots}}} \ \underbrace{\tau_{k+1} \ \dots \ \tau_n}_{n-k \text{ knots}} \ \underbrace{\tau_{n+1} \ \dots \ \tau_{n+1})}_{\substack{k+1 \\ \text{equal knots}}} \tag{4.9}$$

The impact of this specificity is illustrated on figure 4.4, with an example of B-spline functions of degree 2, with a knot vector of the same form as (4.9) and containing 2 internal knots (τ_3 and τ_4).

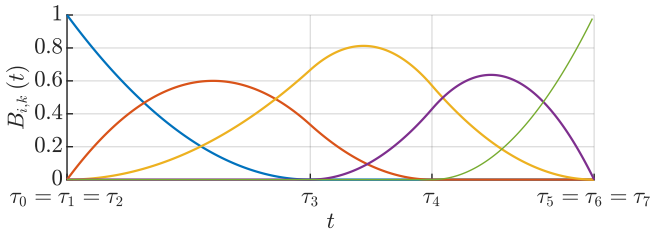

FIGURE 4.4 – Example of clamped B-spline functions

The consequence of the particular form of the knot vector (4.9) is that, when evaluated over $[\tau_k, \tau_{n+1}[$, the B-spline curve starts on the first control point and ends on the last one

$$\boldsymbol{\mathcal{B}}(\tau_k) = \mathbf{p}_0, \qquad \lim_{t \to \tau_{n+1}^-} \boldsymbol{\mathcal{B}}(t) = \mathbf{p}_n \tag{4.10}$$

Remark 12 *To be more accurate, the curve tends to* $\mathbf{p}_n$ *when approaching* τ_{n+1} *by the left. The value of each B-spline function at* τ_{n+1} *being zero for a clamped B-spline curve, given* (4.2)*, the evaluation of the curve then jumps to the origin when reaching* τ_{n+1}*.*

An example of a clamped B-spline curve of degree 2 is given on figure 4.5. The knot vector is the same as the one on figure 4.4 and the control points are the same as on figure 4.3.

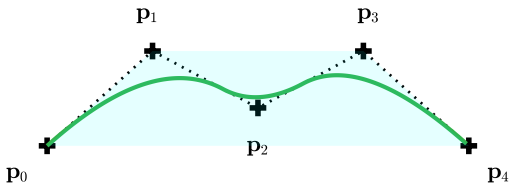

FIGURE 4.5 – Example of clamped B-spline curve

Furthermore, given the form of this knot vector, it can be written differently. In this work, a vector of knot steps is thus introduced, represented by the vector

$$\boldsymbol{\Delta\tau} = \begin{pmatrix} \Delta\tau_0 & \Delta\tau_1 & \dots & \Delta\tau_{n-k} \end{pmatrix} \in \mathbb{R}^{1\times(n-k+1)} \tag{4.11}$$

Given an initial knot τ_0, the knot vector can be reconstructed from the knot step vector as follows

$$\forall j \in [\![0, m]\!] \quad \tau_j = \begin{cases} \tau_0 & \text{if } j \leqslant k \\ \tau_0 + \displaystyle\sum_{i=0}^{j-k-1} \Delta\tau_i & \text{if } k < j \leqslant n \\ \tau_0 + \displaystyle\sum_{i=0}^{n-k} \Delta\tau_i & \text{otherwise} \end{cases} \tag{4.12}$$

In this work, the knot steps are imposed to be strictly positive, which implies that a B-spline curve of degree k is at least $\mathcal{C}^{k-1}$ continuous on $[\tau_k, \tau_{n+1}[$. However these knot steps can differ from each other, unlike for *uniform* clamped B-splines, for which they are all equal.

4.2.4 Clamped B-spline derivatives

The derivatives of a clamped B-spline curve are also clamped B-spline curves, which share the same knot steps as the original curve. The control points of the B-spline curve derivatives are given by linear combination of the original ones. For the l-th derivative, $l \in [\![0, k]\!]$, each of these control points $\mathbf{p}_i^{(l)}$ can be obtained as follows ([29], [95])

$$\mathbf{p}_i^{(l)} = \begin{cases} \mathbf{p}_i & \text{if } l = 0 \\ \dfrac{k-l+1}{\tau_{i+k+1} - \tau_{i+l}} \left(\mathbf{p}_{i+1}^{(l-1)} - \mathbf{p}_i^{(l-1)} \right) & \text{otherwise} \end{cases} \tag{4.13}$$

In this work, this expression is reformulated using the knot step vector introduced above. For $l \in [\![1, k]\!]$, we define the matrix $\mathbf{D}_l(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(n+2-l)\times(n+1-l)}$ such that $\forall (i,j) \in [\![0, n+1-l]\!] \times [\![0, n-l]\!]$

$$D_{l_{i,j}}(\boldsymbol{\Delta\tau}) = \begin{cases} -\rho_j & \text{if } i = j \\ \rho_j & \text{if } i = j+1 \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

with

$$\rho_j = \frac{k-l+1}{\sum\limits_{q=\max(0, j+l-k)}^{\min(n-k, j)} \Delta\tau_q} \tag{4.15}$$

given by injecting (4.12) into (4.13). Then (4.13) can be re-written

$$\mathbf{P}^{(l)} = \begin{cases} \mathbf{P} & \text{if } l = 0 \\ \mathbf{P}^{(l-1)}\, \mathbf{D}_l(\boldsymbol{\Delta\tau}) & \text{otherwise} \end{cases} \tag{4.16}$$

Defining, for $l \in [\![1, k]\!]$

$$\mathbf{D}_{0\to l}(\boldsymbol{\Delta\tau}) = \prod_{q=1}^{l} \mathbf{D}_q(\boldsymbol{\Delta\tau}) \tag{4.17}$$

where $\mathbf{D}_q$ is given by (4.14), it comes that

$$\mathbf{P}^{(l)} = \mathbf{P}\ \mathbf{D}_{0\to l}(\boldsymbol{\Delta\tau}) \tag{4.18}$$

This derivative has a polynomial degree $k-l$ and thus possesses $n+1-l$ control points. Since it is also a clamped B-spline, it verifies (4.10)

$$\lim_{t\to\tau_k^+} \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}(t) = \mathbf{p}_0^{(l)}, \quad \lim_{x\to\tau_{n+1}^-} \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}(t) = \mathbf{p}_{n-l}^{(l)} \tag{4.19}$$

In order to improve the readability, the rigorous limits symbols are dropped in the following. It should nonetheless be remembered that, due to their piecewise nature, some derivatives of a B-spline curve are only one-sided or not defined at all.

4.2.5 Clamped B-spline integrals

In the same manner, for $l \in \mathbb{N}^*$, we define the matrix $\mathbf{D}_{-l}(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(n+l)\times(n+1+l)}$ such that $\forall (i,j) \in [\![0, n+l]\!] \times [\![0, n+1+l]\!]$

$$D_{-l_{i,j}}(\boldsymbol{\Delta\tau}) = \begin{cases} \sigma_i & \text{if } j \geqslant i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

with

$$\sigma_i = \frac{\sum\limits_{q=\max(0, i-k-l)}^{\min(n-k, i-1)} \Delta\tau_q}{k+l} \tag{4.21}$$

The control points of the l-th integral ($l \geqslant 0$) can then be obtained as follows

$$\mathbf{P}^{(-l)} = \begin{cases} \mathbf{P} & \text{if } l = 0 \\ \mathbf{P}^{(1-l)}\, \mathbf{D}_{-l}(\boldsymbol{\Delta\tau}) + \mathbf{p}_0^{(-l)}\, \mathbf{1}_{1\times(n+1+l)} & \text{otherwise} \end{cases} \tag{4.22}$$

with $\mathbf{p}_0^{(-l)}$ the initial value of the l-th integral, at the starting knot, and $\mathbf{1}_{a\times b}$ a a-by-b matrix filled with ones. This integral has a polynomial degree $k+l$.

As a consequence, it is easy to constrain a B-spline curve and its derivatives (or integrals) into convex polytopes, thanks to the convex hull property.

In this section, clamped B-spline curves were presented along with some of their properties. In the following, they are used to parameterize trajectories, composed of several clamped B-spline curves laid end to end. Considering the specifications to be met by these trajectories for validating the waypoints (detailed in section 1.2.2), we now propose a novel, compact way to represent them.

4.3 Compact representation of the trajectory

The flight plan to complete is preprocessed with the methode described in section 3.2. In particular, we consider a flight plan with only *lock* or *sphere* waypoints except for the first waypoint that can be either *stop* or *constrained* and the last one that can only be *stop*.

4.3.1 Piecewise clamped B-spline trajectory

Similarly to chapter 3, for a flight plan containing $N+1$ waypoints, the trajectory is divided into N pieces, each one joining a pair of consecutive waypoints. For $i \in [\![1, N]\!]$, the i-th piece joins the waypoint $\mathbf{w}_{i-1}$ to the waypoint $\mathbf{w}_i$. Each i-th piece of the trajectory is parameterized by a clamped B-spline curve $\boldsymbol{\mathcal{B}}_i$, defined by the knot steps $\boldsymbol{\Delta\tau}_i \in \mathbb{R}^{1\times(n-k+1)}$

$$\boldsymbol{\Delta\tau}_i = \begin{pmatrix} \Delta\tau_{i,0} & \Delta\tau_{i,1} & \dots & \Delta\tau_{i,n-k} \end{pmatrix} \tag{4.23}$$

and the control points $\mathbf{P}_i \in \mathbb{R}^{3\times(n+1)}$

$$\mathbf{P}_i = \begin{pmatrix} \mathbf{p}_{i,0} & \mathbf{p}_{i,1} & \dots & \mathbf{p}_{i,n} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{P}_i^x \\ \mathbf{P}_i^y \\ \mathbf{P}_i^z \end{pmatrix} \tag{4.24}$$

This choice to parameterize each piece of trajectory by a clamped B-spline is motivated by the two following reasons

- The extremities of each piece of trajectory correspond to the validation of a waypoint. The properties of clamped B-splines (4.19) and (4.10) are especially convenient to adjust the position and its time derivatives at the extremities of the pieces of trajectory, in order to meet the validation requirements of each waypoint.

- Each piece of trajectory should lie within its flight corridors and the magnitude of the time derivatives of the position should verify given upper bounds. The fact that the derivatives of a clamped B-spline are also clamped B-spline and the property (4.7) of these curves allows to efficiently formulate these constraints.

In a similar way as in chapter 3, these trajectory pieces are connected so that the position and its L first time derivatives coincide at the connections, with $L < k$. The overall

trajectory is then $\mathcal{C}^L$ continuous. Given an initial time t_0, these connections occur at the time instants $\{t_i\}_{i\in[\![1,N]\!]}$ such as

$$\forall i \in [\![1, N]\!] \;\; t_i = t_{i-1} + \sum_{j=0}^{n-k} \Delta\tau_{i,j} \tag{4.25}$$

and correspond to the time instants where a waypoint is validated.

The evaluation of the overall trajectory $\boldsymbol{\zeta}$ at a time instant $t \in \mathbb{R}$ is obtained in a similar fashion as in chapter 3

$$\boldsymbol{\zeta}(t) = \begin{cases} \mathbf{w}_0 & \text{if } t < t_0 \\ \boldsymbol{\mathcal{B}}_i(t) & \text{if } t_{i-1} \leqslant t < t_i,\ i \in [\![1, N]\!] \\ \mathbf{w}_N & \text{if } t \geqslant t_N \end{cases} \tag{4.26}$$

In order to validate the waypoints while remaining smooth, this trajectory has to satisfy several constraints at the connections between its different pieces, that are now detailed.

4.3.2 Trajectory derivatives at the connections

In order to get an overall trajectory that is $\mathcal{C}^L$, $L < k$, and to respect the criteria of validation of each waypoint, boundary conditions on the time derivatives must be met at the beginning and at the end of each piece of trajectory. Depending on the type of waypoints, these derivatives can be imposed or set free, in which case they must coincide with the ones of the previous and the following pieces of trajectory. Considering that the first waypoint is either a *stop* or a *constrained* waypoint and that the last one is always a *stop* waypoint, these constraints on the time derivatives can be formulated in a similar way as in chapter 3

$\forall i \in [\![1, N]\!]$

If $\mathbf{w}_{i-1}$ is a *stop* waypoint

$$\begin{cases} \boldsymbol{\mathcal{B}}_i(t_{i-1}) = \mathbf{w}_{i-1} \\ \forall l \in [\![1, L]\!] \quad \dfrac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_{i-1}) = \mathbf{0} \end{cases} \tag{4.27a}$$

Else, if $\mathbf{w}_{i-1}$ is a *constrained* waypoint

$$\forall l \in [\![0, L]\!] \quad \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_{i-1}) = \mathbf{w}_{i-1}^{(l)} \tag{4.27b}$$

Else, if $\mathbf{w}_{i-1}$ is a *lock* waypoint

$$\begin{cases} \boldsymbol{\mathcal{B}}_i(t_{i-1}) = \mathbf{w}_{i-1} \\ \forall l \in [\![1, L]\!] \quad \dfrac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_{i-1}) = \dfrac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_{i-1}(t_{i-1}) \end{cases} \tag{4.27c}$$

Else, if $\mathbf{w}_{i-1}$ is a *sphere* waypoint

$$\forall l \in [\![0, L]\!] \quad \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_{i-1}) = \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_{i-1}(t_{i-1}) \tag{4.27d}$$

If $\mathbf{w}_i$ is a *stop* waypoint

$$\forall l \in [\![0, L]\!] \quad \frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_i) = \mathbf{0} \tag{4.27e}$$

Else, if $\mathbf{w}_i$ is a *lock* waypoint

$$\boldsymbol{\mathcal{B}}_i(t_i) = \mathbf{w}_i \tag{4.27f}$$

Else, if $\mathbf{w}_i$ is a *sphere* waypoint

$$\|\boldsymbol{\mathcal{B}}_i(t_i) - \mathbf{w}_i\|_2 \leqslant r_{\mathbf{w}_i} \tag{4.27g}$$

where $r_{\mathbf{w}_i}$ is the validation radius of the waypoint $\mathbf{w}_i$ and $\mathbf{w}_{i-1}^{(l)}$ is value of the l-th derivative of the trajectory imposed at the validation of the waypoint $\mathbf{w}_i$. Since clamped B-splines are considered, these constraints can directly be written as constraints on the first and the last control points of each piece of trajectory and their derivatives

$$\frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_{i-1}) = \mathbf{p}_{i,0}^{(l)}$$
$$\frac{\mathrm{d}^l}{\mathrm{d}t^l}\boldsymbol{\mathcal{B}}_i(t_i) = \mathbf{p}_{i,n-l}^{(l)}$$

Whatever the type of waypoint, the $L+1$ first time derivatives (including the position) of a piece of trajectory are then constrained at its beginning. Some derivatives can also be imposed at the end of this piece, depending on the type of waypoint.

In the following we represent the time derivatives of the overall trajectory $\boldsymbol{\zeta}$ at the beginning of a piece of trajectory, i.e. at the time instant t_{i-1}, with $i \in [\![1, N]\!]$ by the matrix $\boldsymbol{\Gamma}_i \in \mathbb{R}^{3\times(L+1)}$

$$\boldsymbol{\Gamma}_i = \begin{pmatrix} \boldsymbol{\zeta}(t_{i-1}) & \frac{\mathrm{d}^1}{\mathrm{d}t^1}\boldsymbol{\zeta}(t_{i-1}) & \dots & \frac{\mathrm{d}^L}{\mathrm{d}t^L}\boldsymbol{\zeta}(t_{i-1}) \end{pmatrix} \tag{4.28}$$

These constraints allow reducing the number of parameters necessary for the trajectory generation, as explained in the next paragraph.

4.3.3 Reduced set of control points

A consequence of the constraints (4.27) is that the $L+1$ first control points of each piece of trajectory are imposed and can thus be removed from the trajectory generation problem. Furthermore, depending on the type of waypoint at the end of the trajectory piece, the $L+1$ last control points (*stop* waypoint) or the last control point (*lock* waypoint) can also be removed.

As another contribution of this thesis, we thus introduce a compact way to represent a piecewise B-spline trajectory satisfying both the continuity constraints and the *stop* and *lock* waypoints validation constraints detailed in (4.27). For each i-th piece of trajectory, we introduce a *reduced set of control points*, equal to the set of control points of the i-th piece minus the control points imposed by the continuity constraints. For $i \in [\![1, N]\!]$, the number of reduced control points of the i-th piece of trajectory is denoted by $\widetilde{n}_i \leqslant n - L$ such that

$$\widetilde{n}_i = \begin{cases} n - 2\,L - 1 & \text{if } \mathbf{w}_i \text{ is } \textit{stop} \\ n - L - 1 & \text{if } \mathbf{w}_i \text{ is } \textit{lock} \\ n - L & \text{if } \mathbf{w}_i \text{ is } \textit{sphere} \end{cases} \tag{4.29}$$

and the reduced set of control points is represented by the matrix $\widetilde{\mathbf{P}}_i \in \mathbb{R}^{3\times\widetilde{n}_i}$

$$\widetilde{\mathbf{P}}_i = \begin{pmatrix} \mathbf{p}_{L+1} & \mathbf{p}_{L+2} & \cdots & \mathbf{p}_{\widetilde{n}_i+L} \end{pmatrix} \triangleq \begin{pmatrix} \widetilde{\mathbf{P}}_i^x \\ \widetilde{\mathbf{P}}_i^y \\ \widetilde{\mathbf{P}}_i^z \end{pmatrix} \tag{4.30}$$

The rest of this section details the reconstruction of the full set of control points from the reduced one, starting with the $L+1$ first control points.

4.3.4 Reconstruction of the $L+1$ first control points

For the i-th piece of trajectory, $i \in [\![1, N]\!]$, the $L+1$ first time derivatives at the beginning of the piece of trajectory, $\mathbf{\Gamma}_i$, are either

- Explicitly given by the waypoint $\mathbf{w}_{i-1}$ (*stop* or *constrained* waypoint)
- Imposed equal to the $L+1$ first time derivatives at the end of the previous piece of trajectory (piece $i-1$, if $i>1$) by the continuity constraints (*sphere* waypoint). These derivatives are given by the last $L+1$ control points of the $(i-1)$-th piece of trajectory
- Given both by the waypoint $\mathbf{w}_{i-1}$ and the time derivatives at the end of the $(i-1)$-th piece of trajectory (*lock* waypoint).

The reconstruction of the $L+1$ first control points of the i-th piece of trajectory can then be performed in two steps, illustrated on figure 4.6, for $L=2$

- First, the time derivatives $\mathbf{\Gamma}_i$ (green arrow on figure 4.6) are determined from $\mathbf{w}_{i-1}$ and the last $L+1$ control points of the previous piece (piece $i-1$, if $i>1$), using the derivative properties (4.16). The latter are represented by the matrix $\mathbf{P}_{i-1,\text{end}}$, equal to the last $L+1$ columns of $\mathbf{P}_{i-1}$ (control points in solid red on figure 4.6).
- Then, the first $L+1$ control points of the i-th piece are reconstructed from $\mathbf{\Gamma}_i$, using the integral properties (4.22). These control points are represented by the matrix $\mathbf{P}_{i,\text{start}}$ equal to the first $L+1$ columns of $\mathbf{P}_i$ (control points in solid blue on figure 4.6).

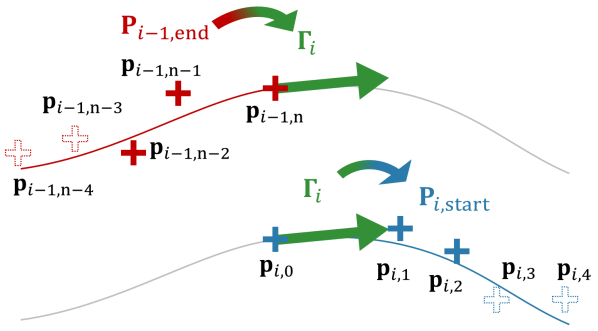

FIGURE 4.6 – Reconstruction of the first $L+1$ control points

These two steps are now detailed, starting with the determination of $\mathbf{\Gamma}_i$.

4.3.4.1 Derivatives at the end of the previous piece of trajectory

Since clamped B-splines are used to parameterize each piece of trajectory, the $L+1$ first time derivatives at the end of a piece are given by the last control points of each of its derivatives (4.19). These can be obtained from the knot steps and the last $L+1$ control points as follows.

For a given knot step vector $\boldsymbol{\Delta\tau}$, using (4.16), we define the matrix $\mathbf{O}_l(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(L+1)\times(L+1)}$ such that, for $l \in [\![1, L]\!]$ and $\forall (i,j) \in [\![0, L]\!]^2$

$$O_{l_{i,j}}(\boldsymbol{\Delta\tau}) = \begin{cases} 1 & \text{if } j < l \text{ and } i = j \\ -\gamma_j & \text{if } j \geqslant l \text{ and } i = j \\ \gamma_j & \text{if } j \geqslant l \text{ and } i = j-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

where

$$\gamma_j = \frac{k-l+1}{\sum\limits_{q=n-k-j+l}^{n-k} \Delta\tau_q} \tag{4.32}$$

For $i \in [\![2, N]\!]$ we define the *control points to derivatives* matrix $\mathbf{M}_{\mathrm{p}\to\mathrm{d}}(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(L+1)\times(L+1)}$, giving the first $L+1$ derivatives at the end of the $(i-1)$-th piece of trajectory from its last $L+1$ control points

$$\mathbf{M}_{\mathrm{p}\to\mathrm{d}}(\boldsymbol{\Delta\tau}) = \mathbf{J}_{L+1} \prod_{l=1}^{L} \mathbf{O}_l(\boldsymbol{\Delta\tau}) \tag{4.33}$$

where $\mathbf{J}_a$ is the a-by-a anti-diagonal matrix with every anti-diagonal term equal to 1. The derivatives are then obtained as follows

$$\begin{aligned} &\begin{pmatrix} \mathbf{p}_{i-1,n} & \mathbf{p}^{(1)}_{i-1,n-1} & \cdots & \mathbf{p}^{(L)}_{i-1,n-L} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{p}_{i-1,n-L} & \mathbf{p}_{i-1,n-L+1} & \cdots & \mathbf{p}_{i-1,n} \end{pmatrix} \mathbf{M}_{\mathrm{p}\to\mathrm{d}}(\boldsymbol{\Delta\tau}_{i-1}) \end{aligned} \tag{4.34}$$

4.3.4.2 Derivatives of the beginning of the current piece of trajectory

As explained before, some derivatives of the trajectory can be imposed on a waypoint, depending on its type. A generic expression of the first $L+1$ derivatives at the beginning of the i-th piece of trajectory is then

$\forall i \in [\![1, N]\!]$

If $i = 1$

$$\boldsymbol{\Gamma}_i = \begin{cases} \begin{pmatrix} \mathbf{w}_{i-1} & \mathbf{0}_{3\times L} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{stop} \\ \begin{pmatrix} \mathbf{w}_{i-1} & \mathbf{w}^{(1)}_{i-1} & \cdots & \mathbf{w}^{(L)}_{i-1} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{constrained} \end{cases} \tag{4.35a}$$

Else

$$\boldsymbol{\Gamma}_i = \mathbf{P}_{i-1,\mathrm{end}}\, \mathbf{M}_{\mathrm{p}\to\mathrm{d}}(\boldsymbol{\Delta\tau}_{i-1}) \tag{4.35b}$$

where

$$\forall \in [\![1, N-1]\!] \quad \mathbf{P}_{i,\text{end}} = \begin{cases} \begin{pmatrix} \mathbf{p}_{i,n-L} & \mathbf{p}_{i,n-L+1} & \cdots & \mathbf{p}_{i,n-1} & \mathbf{w}_i \end{pmatrix} & \text{if } \mathbf{w}_i \text{ is } \textit{lock} \\ \begin{pmatrix} \mathbf{p}_{i,n-L} & \mathbf{p}_{i,n-L+1} & \cdots & \mathbf{p}_{i,n} \end{pmatrix} & \text{if } \mathbf{w}_i \text{ is } \textit{sphere} \end{cases} \tag{4.36}$$

with the control points $\mathbf{p}_{i,j}$ in (4.36) being part of the reduced set $\widetilde{\mathbf{P}}_i$ (see (4.30)).

The computation of these derivatives can be conveniently reformulated as follows

$\forall i \in [\![1, N]\!]$

If $i = 1$

$$\mathbf{\Gamma}_i = \mathbf{A}_{1,i} \tag{4.37a}$$

Else

$$\mathbf{\Gamma}_i\Big(\boldsymbol{\Delta\tau}_{i-1}, \widetilde{\mathbf{P}}_{i-1}\Big) = \mathbf{A}_{1,i} + \mathbf{A}_{2,i}(\boldsymbol{\Delta\tau}_{i-1}) + \widetilde{\mathbf{P}}_{i-1}\,\mathbf{A}_{3,i}\,\mathbf{M}_{\text{p}\to\text{d}}(\boldsymbol{\Delta\tau}_{i-1}) \tag{4.37b}$$

with

$$\begin{aligned} \mathbf{A}_{1,i} &= \begin{cases} \begin{pmatrix} \mathbf{w}_{i-1} & \mathbf{0}_{3\times L} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{stop} \\ \mathbf{0}_{3\times(L+1)} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{lock} \text{ or } \textit{sphere} \\ \begin{pmatrix} \mathbf{w}_{i-1} & \mathbf{w}_{i-1}^{(1)} & \cdots & \mathbf{w}_{i-1}^{(L)} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{constrained} \end{cases} \\ \mathbf{A}_{2,i}(\boldsymbol{\Delta\tau}_{i-1}) &= \begin{cases} \begin{pmatrix} \mathbf{w}_{i-1} & \mathbf{0}_{3\times L} \end{pmatrix} \mathbf{M}_{\text{p}\to\text{d}}(\boldsymbol{\Delta\tau}_{i-1}) & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{lock} \\ \mathbf{0}_{3\times(L+1)} & \text{otherwise} \end{cases} \\ \mathbf{A}_{3,i} &= \begin{cases} \begin{pmatrix} \mathbf{0}_{(\widetilde{n}_{i-1}-L-1)\times(L+1)} \\ (\mathbf{I}_{L\times L} \;\; \mathbf{0}_{L\times 1}) \\ \mathbf{0}_{1\times(L+1)} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{lock} \\ \begin{pmatrix} \mathbf{0}_{(\widetilde{n}_{i-1}-L-1)\times(L+1)} \\ \mathbf{I}_{(L+1)\times(L+1)} \end{pmatrix} & \text{if } \mathbf{w}_{i-1} \text{ is } \textit{sphere} \end{cases} \end{aligned} \tag{4.38}$$

Based on $\mathbf{\Gamma}_i$, the reconstruction of the first $L+1$ control points of the piece $i \in [\![1, N]\!]$ can now be performed, as detailed in the next paragraph.

4.3.4.3 Starting derivatives to control points

The $L+1$ first control points of a piece of trajectory can be obtained from the first $L+1$ time derivatives at its beginning. Using (4.22), we can define for a given knot step vector $\boldsymbol{\Delta\tau}$ the matrix $\mathbf{Q}_l(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(L+1)\times(L+1)}$ such that, for $l \in [\![1, L]\!]$ and $\forall (i,j) \in [\![0, L]\!]^2$

$$Q_{l_{i,j}}(\boldsymbol{\Delta\tau}) = \begin{cases} 1 & \text{if } j < l \text{ and } i = j \\ \nu_j & \text{if } j \geqslant l \text{ and } i = j \\ 1 & \text{if } j \geqslant l \text{ and } i = j-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.39}$$

where

$$\nu_j = \frac{\sum_{q=0}^{l-1} \Delta\tau_q}{k+l-j} \tag{4.40}$$

For $i \in [\![1, N]\!]$ we define the *derivative to control points* matrix $\mathbf{M}_{\mathrm{d}\to\mathrm{p}}(\boldsymbol{\Delta\tau}) \in \mathbb{R}^{(L+1)\times(L+1)}$ giving the first $L+1$ control points of the i-th piece of trajectory from the first $L+1$ derivatives at the beginning of this piece of trajectory

$$\mathbf{M}_{\mathrm{d}\to\mathrm{p}}(\boldsymbol{\Delta\tau}) = \prod_{l=0}^{L-1} \mathbf{Q}_{L-l}(\boldsymbol{\Delta\tau}) \tag{4.41}$$

which gives the first $L+1$ control points

$$\mathbf{P}_{i,\mathrm{start}} = \begin{pmatrix} \mathbf{p}_{i,0} & \mathbf{p}_{i,1} & \cdots & \mathbf{p}_{i,L} \end{pmatrix} \tag{4.42}$$

from the starting derivatives. Indeed, for $i \in [\![1, N]\!]$, the following expressions hold

If $i = 1$

$$\mathbf{P}_{i,\mathrm{start}}(\boldsymbol{\Delta\tau}_i) = \boldsymbol{\Gamma}_i \, \mathbf{M}_{\mathrm{d}\to\mathrm{p}}(\boldsymbol{\Delta\tau}_i) \tag{4.43a}$$

Else

$$\mathbf{P}_{i,\mathrm{start}}\Big(\boldsymbol{\Delta\tau}_{i-1}, \boldsymbol{\Delta\tau}_i, \widetilde{\mathbf{P}}_{i-1}\Big) = \boldsymbol{\Gamma}_i\Big(\boldsymbol{\Delta\tau}_{i-1}, \widetilde{\mathbf{P}}_{i-1}\Big) \, \mathbf{M}_{\mathrm{d}\to\mathrm{p}}(\boldsymbol{\Delta\tau}_i) \tag{4.43b}$$

The reconstruction of the remaining control points is now detailed.

4.3.5 Reconstruction of the full set of control points

In some cases, the last control points are also imposed. In the case where the next waypoint is a *stop* waypoint, the last $L+1$ control points are imposed equal to $\mathbf{w}_i$. Similarly, if $\mathbf{w}_i$ is a *lock* waypoint, the last control point is constrained on the waypoint. In the case of a *sphere* waypoint, there is no constraint on the last control points.

Finally, the remaining control points (i.e. other than the first and the last ones) do not require reconstruction, as they are directly equal to the reduced control points. The overall reconstruction operation can thus be formulated as follows, for $i \in [\![1, N]\!]$

$$\mathbf{P}_i = \begin{pmatrix} \mathbf{P}_{i,\mathrm{start}} & \mathbf{p}_{i,L+2} & \mathbf{p}_{i,L+3} & \cdots & \mathbf{p}_{i,n-L-1} & \mathbf{P}_{i-1,\mathrm{end}} \end{pmatrix} \tag{4.44}$$

with $\mathbf{p}_{i,j}$ being part of the reduced control points $\widetilde{\mathbf{P}}_i$, for $j \in [\![L+2, n-L-1]\!]$ (see (4.30)). Using (4.37), (4.36) and (4.43), the full reconstruction operation can be formulated as follows

If $i = 1$

$$\mathbf{P}_i = \mathbf{T}_{\mathrm{fixed}\,i}(\boldsymbol{\Delta\tau}_i) + \widetilde{\mathbf{P}}_i \, \mathbf{T}_{\mathrm{current}\,i} \tag{4.45a}$$

Else

$$\mathbf{P}_i = \mathbf{T}_{\mathrm{fixed}\,i}(\boldsymbol{\Delta\tau}_i) + \mathbf{T}_{\mathrm{lock}\,i}(\boldsymbol{\Delta\tau}_{i-1}, \boldsymbol{\Delta\tau}_i) + \widetilde{\mathbf{P}}_{i-1} \, \mathbf{T}_{\mathrm{previous}_i}(\boldsymbol{\Delta\tau}_{i-1}, \boldsymbol{\Delta\tau}_i) + \widetilde{\mathbf{P}}_i \, \mathbf{T}_{\mathrm{current}\,i} \tag{4.45b}$$

with

$$\begin{aligned}
\mathbf{T}_{\text{fixed}\,i} &= \mathbf{A}_{1,i}\,\mathbf{M}_{\text{d}\to\text{p}}(\boldsymbol{\Delta\tau}_i)\,\mathbf{A}_4 + \mathbf{w}_i\,\mathbf{A}_{5,i} \\
\mathbf{T}_{\text{lock}\,i}(\boldsymbol{\Delta\tau}_{i-1}, \boldsymbol{\Delta\tau}_i) &= \mathbf{A}_{2,i}(\boldsymbol{\Delta\tau}_{i-1})\,\mathbf{M}_{\text{d}\to\text{p}}(\boldsymbol{\Delta\tau}_i)\,\mathbf{A}_4 \\
\mathbf{T}_{\text{previous}_i}(\boldsymbol{\Delta\tau}_{i-1}, \boldsymbol{\Delta\tau}_i) &= \mathbf{A}_{3,i}(\boldsymbol{\Delta\tau}_{i-1})\,\mathbf{M}_{\text{d}\to\text{p}}(\boldsymbol{\Delta\tau}_i)\,\mathbf{A}_4 \\
\mathbf{T}_{\text{current}\,i} &= \mathbf{0}_{\widetilde{n}_i\times(n-\widetilde{n}_i-L)}
\end{aligned} \tag{4.46}$$

and

$$\begin{aligned}
\mathbf{A}_4 &= \begin{pmatrix} \mathbf{I}_{L+1} & \mathbf{0}_{(L+1)\times(n-L)} \end{pmatrix} \\
\mathbf{A}_{5,i} &= \begin{cases} \begin{pmatrix} \mathbf{0}_{1\times(n-L)} & \mathbf{1}_{1\times(L+1)} \end{pmatrix} & \text{if } \mathbf{w}_i \text{ is } \textit{stop} \\ \begin{pmatrix} \mathbf{0}_{1\times n} & 1 \end{pmatrix} & \text{if } \mathbf{w}_i \text{ is } \textit{lock} \\ \mathbf{0}_{1\times(n+1)} & \text{otherwise} \end{cases}
\end{aligned} \tag{4.47}$$

Therefore, in the general case, the full set of control points of a given piece of trajectory

- Linearly depends on the reduced control points of this piece of trajectory
- Nonlinearly depends, on the one hand, on the knot steps of this piece of trajectory and, on the other hand, on the knot steps and the reduced control points of the previous piece of trajectory

The knot steps and the reduced control points thus constitute the degrees of freedom for adjusting the trajectory.

In this section, a novel compact way to parameterize the entire trajectory $\boldsymbol{\zeta}$ as a piecewise clamped B-spline curve, by the mean of knot step vectors and reduced sets of control points was proposed. In the next section, this convenient compact representation is used to formulate the trajectory generation process as an optimization problem.

4.4 Minimum-time trajectory

One way to obtain a smooth trajectory meeting the requirements of section 1.2 is to generate a minimum-time trajectory with constraints on the magnitude of the velocity and its derivatives. Typically, the smoothness of the trajectory can directly be obtained by bounding the magnitude of the acceleration and the jerk with low values, rather than penalizing them as in the previous chapter. One contribution of this chapter is thus to formalize the minimum-time trajectory generation as an optimization problem on the duration of the trajectory, using the formalism of B-splines. The first paragraph details one general way to formulate this problem, using the trajectory parameterization introduced in section 4.3. Then, the expression of the constraints is detailed in the following paragraphs. Notice that the formulation of the problem is specified for 3D trajectories, however it is easily adaptable to other dimensions and to other applications than quadrotor cinematography.

4.4.1 Optimization problem

The trajectory is parameterized as a piecewise clamped B-spline curve, with the compact representation presented in section 4.3. It is then parameterized by

- N knot step vectors $\boldsymbol{\Delta\tau}_i$, each vector containing $n-k$ elements

- N reduced sets of control points $\widetilde{\mathbf{P}}_i$, with the i-th set containing $\widetilde{n}_i$ control points, themselves given by 3 coordinates

This gives a total of $n_{\mathbf{x}} = N(n-k+1) + \sum_{i=1}^{N} \widetilde{n}_i$ parameters which are stored in the vector $\mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}}$, such that

$$\mathbf{x} = \left(\boldsymbol{\Delta\tau}_1 \;\; \boldsymbol{\Delta\tau}_2 \;\; \dots \;\; \boldsymbol{\Delta\tau}_N \;\; \widetilde{\mathbf{P}}_1^x \;\; \widetilde{\mathbf{P}}_1^y \;\; \widetilde{\mathbf{P}}_1^z \;\; \widetilde{\mathbf{P}}_2^x \;\; \widetilde{\mathbf{P}}_2^y \;\; \widetilde{\mathbf{P}}_2^z \;\; \dots \;\; \widetilde{\mathbf{P}}_N^x \;\; \widetilde{\mathbf{P}}_N^y \;\; \widetilde{\mathbf{P}}_N^z \right)^\top \tag{4.48}$$

containing all the information required to reconstruct the trajectory over its entire domain.

In order to complete the flight plan described in section 1.2.2, a vector $\mathbf{x}$ minimizing the duration $T \in \mathbb{R}_+^*$ of the flight plan is chosen, under similar constraints as in chapter 3

- The knot steps are strictly positive
- The trajectory does not exit the corridors
- The trajectory validates the *sphere* waypoints by passing within a neighborhood of specified radius around them
- The magnitude of the trajectory derivatives does not exceed specified bounds

The trajectory generation can then be formulated as the following nonlinear optimization problem

$$\begin{aligned} \mathbf{x}^* &= \arg \min_{\mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}}} \; T(\mathbf{x}) \\ \text{s.t.} & \begin{cases} x_i \geqslant \Delta\tau_{\min}, \; i \in [\![0, (n-k+1)N - 1]\!] \\ \mathbf{A}_{\text{corridor}}\, \mathbf{x} - \mathbf{b}_{\text{corridor}} \leqslant 0 \\ \mathbf{g}_{\text{corridor}}(\mathbf{x}) \leqslant 0 \\ \mathbf{g}_{\text{sphere}}(\mathbf{x}) \leqslant 0 \\ \mathbf{g}_{\text{derivatives}}(\mathbf{x}) \leqslant 0 \end{cases} \end{aligned} \tag{4.49}$$

with the linear cost function

$$T(\mathbf{x}) = \sum_{i=1}^{N} \sum_{j=0}^{n-k} \Delta\tau_{i,j} \tag{4.50}$$

The expressions of the constraints are detailed in section 4.4.2, section 4.4.3 and section 4.4.4.

Problem (4.49) constitutes a Nonlinear Programming (NLP) that can be solved using classical algorithms, such as a SQP for instance. This requires some care in the choice of the starting point of the algorithm. A series of rest-to-rest trajectories joining each pair of waypoints can be a candidate as initial guess for solving the problem. Though it can be quite suboptimal, there always exists a rest-to-rest trajectory joining two waypoints which satisfies the constraints of the NLP (4.49) and it is reasonably simple to compute one. An example of such an initial guess is proposed in section 4.5.

The tuning parameters of the algorithm are the polynomial degree $k \geqslant 0$ and the number of control points $(n+1) > k$ of the clamped B-splines (thus setting their number of knot steps $n-k+1$). As mentioned in section 4.2.3, a clamped B-spline with strictly positive

knot steps is $\mathcal{C}^{k-1}$ continuous. In order to obtain a trajectory that is $\mathcal{C}^L$, the polynomial degree must then verify $k > L$. The lowest value $k = L + 1$ constitutes a good candidate as it keeps the polynomial degree as low as possible. Only the parameter n then remains to be set and constitutes a tuning parameter. However, the choice of n is not entirely free. As explained in section 4.3, for a given vector of knot steps, each constraint on the time derivatives of the trajectory in (4.27) imposes the value of one control points (and this is used to reduce the number of optimized control points). For a $\mathcal{C}^L$ trajectory, there are at most $L + 1$ constraints on the derivatives at one extremity of a piece of trajectory, when validating a waypoint. Since each piece of trajectory has 2 extremities, choosing $n \geqslant 2\,(L + 1)$ guarantees that there are always enough control points to satisfy these constraints, whatever the type of waypoints.

4.4.2 Corridor constraints

In this section we propose a better formulation of the corridor constraints than in chapter 3 (section 3.5.1) by 2 means

- The convex hull property of B-spline is used rather than a gridding method. By constraining the control points of the trajectory into the flight corridors, which are convex regions, the trajectory is guaranteed to respect them everywhere.
- The lateral corridor constraints are better formulated, using polygonal sections of the flight corridors.

The flight corridors are approximated by n_{corr}-prisms ($n_{\text{corr}} \geqslant 3$), leading to $n_{\text{corr}} + 2$ constraints for each control point

- 2 longitudinal constraints ensuring that the control point lies between the two waypoints of the corresponding piece of trajectory
- n_{corr} lateral constraints ensuring that the control point respects the cross section of the prism (and thus of the cylinder), as illustrated in figure 4.7

Corridor longitudinal constraints. Given the unit vector $\mathbf{u}_i$, for $i \in [\![1, N]\!]$

$$\mathbf{u}_i = \frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2} \tag{4.51}$$

where $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2$ is always strictly positive (otherwise the flight plan would have been split, see section 3.2). Then, the longitudinal constraints are given by

$$\begin{aligned} \mathbf{u}_i^\top (\mathbf{p}_{i,j} - \mathbf{w}_{i-1}) &\geqslant -\varepsilon_{\text{corr}} \\ \mathbf{u}_i^\top (\mathbf{p}_{i,j} - \mathbf{w}_{i-1}) &\leqslant \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 + \varepsilon_{\text{corr}} \end{aligned} \tag{4.52}$$

with $\varepsilon_{\text{corr}} \geqslant 0$ a given tolerance that can be set to 0. These constraints can be rewritten as

$$\begin{aligned} -\mathbf{u}_i^\top \mathbf{p}_{i,j} + \mathbf{u}_i^\top \mathbf{w}_{i-1} - \varepsilon_{\text{corr}} &\leqslant 0 \\ \mathbf{u}_i^\top \mathbf{p}_{i,j} - \mathbf{u}_i^\top \mathbf{w}_{i-1} - \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 - \varepsilon_{\text{corr}} &\leqslant 0 \end{aligned} \tag{4.53}$$

Corridor lateral constraints. Regarding the lateral constraints, they can be formulated by considering the 2D vector which corresponds to the projection of $(\mathbf{p}_{i,j} - \mathbf{w}_{i-1})$ onto the cross section of the prism. To this aim, a 2D orthonormal basis $\{\mathbf{e}_{x_i}, \mathbf{e}_{y_i}\}$ of this cross section is constructed, in which the equation of the boundaries of the prism can be easily expressed. The vector among $\{\mathbf{u}_i \times \mathbf{x}_\mathcal{W},\ \mathbf{u}_i \times \mathbf{y}_\mathcal{W},\ \mathbf{u}_i \times \mathbf{z}_\mathcal{W}\}$, with the highest norm is normalized and chosen as the first vector of this basis, $\mathbf{e}_{x_i}$. The second vector is then $\mathbf{e}_{y_i} = \mathbf{u}_i \times \mathbf{e}_{x_i}$. On figure 4.7, the corridor between the waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ is represented in light blue and the unit vector $\mathbf{u}_i$ is represented in red. The cross section of a 5-prism is represented in dark blue, with its basis $\{\mathbf{e}_{x_i}, \mathbf{e}_{y_i}\}$ in green. The control point $\mathbf{p}_{i,j}$ (in orange) lies within the section of the prism and it is between the two waypoints, which guarantees its satisfaction of the corridor constraint.

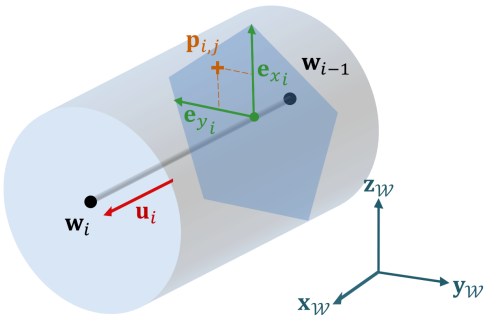

FIGURE 4.7 – Approximation of the cylindrical flight corridor by a prism

The n_{corr} lateral constraints on each control point can be expressed as follows, $\forall i \in [\![1, N]\!]$, $\forall j \in [\![0, n]\!]$, $\forall q \in [\![1, n_{\text{corr}}]\!]$

$$a_{i,q}\, \mathbf{e}_{x_i}^\top \left(\widehat{\mathbf{u}}_i \left(\mathbf{p}_{i,j} - \mathbf{w}_{i-1}\right)\right) + b_{i,q}\, \mathbf{e}_{y_i}^\top \left(\widehat{\mathbf{u}}_i \left(\mathbf{p}_{i,j} - \mathbf{w}_{i-1}\right)\right) + c_{i,q} \leqslant 0 \tag{4.54}$$

where the following coefficients are defined in the following manner

$$\begin{aligned} a_{i,q} &= \cos\left(q\delta\right) - \cos\left((q+1)\delta\right) \\ b_{i,q} &= \sin\left((q+1)\delta\right) - \sin\left(q\delta\right) \\ c_{i,q} &= -r_{\text{corr}\,i}\, \sin\left(\delta\right) \end{aligned} \tag{4.55}$$

with $\delta = \frac{2\pi}{n_{\text{corr}}}$. Regrouping the terms related to the control points and to the waypoints, the expression (4.54) can be rewritten as follows, $\forall i \in [\![1, N]\!]$, $\forall j \in [\![0, n]\!]$, $\forall q \in [\![1, n_{\text{corr}}]\!]$

$$\left(a_{i,q}\, \mathbf{e}_{x_i}^\top + b_{i,q}\, \mathbf{e}_{y_i}^\top\right) \widehat{\mathbf{u}}_i\, \mathbf{p}_{i,j} - \left(a_{i,q}\, \mathbf{e}_{x_i}^\top + b_{i,q}\, \mathbf{e}_{y_i}^\top\right) \widehat{\mathbf{u}}_i\, \mathbf{w}_{i-1} + c_{i,q} \leqslant 0 \tag{4.56}$$

Corridor inequality constraints The corridor constraints (4.53) and (4.56) for a given control point can be written as follows

$$\mathbf{\Lambda}_i\, \mathbf{p}_{i,j} - \boldsymbol{\lambda}_i \leqslant 0 \tag{4.57}$$

with $\mathbf{p}_{i,j}$ given by (4.45) and

$$\begin{aligned}
\boldsymbol{\Lambda}_i &= \begin{pmatrix}
-\mathbf{u}_i^\top \\
\mathbf{u}_i^\top \\
\left(a_{i,1}\,\mathbf{e}_{x_i}^\top + b_{i,1}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i \\
\left(a_{i,2}\,\mathbf{e}_{x_i}^\top + b_{i,2}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i \\
\vdots \\
\left(a_{i,n_{\text{corr}}}\,\mathbf{e}_{x_i}^\top + b_{i,n_{\text{corr}}}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i
\end{pmatrix} \\
\boldsymbol{\lambda}_i &= \begin{pmatrix}
-\mathbf{u}_i^\top \mathbf{w}_{i-1} + \varepsilon_{\text{corr}} \\
\mathbf{u}_i^\top \mathbf{w}_{i-1} + \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 + \varepsilon_{\text{corr}} \\
\left(a_{i,1}\,\mathbf{e}_{x_i}^\top + b_{i,1}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i\;\mathbf{w}_{i-1} - c_{i,1} \\
\left(a_{i,2}\,\mathbf{e}_{x_i}^\top + b_{i,2}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i\;\mathbf{w}_{i-1} - c_{i,2} \\
\vdots \\
\left(a_{i,n_{\text{corr}}}\,\mathbf{e}_{x_i}^\top + b_{i,n_{\text{corr}}}\,\mathbf{e}_{y_i}^\top\right)\,\widehat{\mathbf{u}}_i\;\mathbf{w}_{i-1} - c_{i,n_{\text{corr}}}
\end{pmatrix}
\end{aligned} \tag{4.58}$$

Since the reduced control points are part of the optimization vector $\mathbf{x}$ defined in (4.48), this constraint is linear relatively to $\mathbf{x}$ if $\mathbf{p}_{i,j}$ belongs to the reduced set of control points. However, the other control points of the full set must be reconstructed, using (4.45), which is a nonlinear operation on the optimization vector. As a consequence, this constraint is nonlinear for the control points needing reconstruction.

The corridor constraints which are linear in $\mathbf{x}$ can be gathered and expressed as follows

$$\mathbf{A}_{\text{corridor}}\,\mathbf{x} - \mathbf{b}_{\text{corridor}} \leqslant 0$$

while the nonlinear ones can be written as a nonlinear function of $\mathbf{x}$

$$\mathbf{g}_{\text{corridor}}(\mathbf{x}) \leqslant \mathbf{0}$$

Since they are independent of the decision variables, the matrices $\boldsymbol{\Lambda}_i$ and vectors $\boldsymbol{\lambda}_i$ are computed only once for each piece of trajectory (during the initialization phase of the optimization). Furthermore, these constraints can be removed for the control points that are imposed equal to a waypoint, i.e. on *stop* or *lock* waypoints, since the waypoints always lie inside their corridors.

This formulation of the corridors constraints, by maintaining the control points inside the flight corridors, provides a sufficient condition for guaranteeing that the trajectory lies within the corridors. As explained in [77], a certain conservatism can be expected from such a formulation. However, this conservatism can be reduced by increasing the number of control points, which allows the curve to stick closer to the control polygon. The counterpart is that, trough a finite, reasonable amount of constraints, the *entire* piece of trajectory is guaranteed to satisfy the constraints. This is a significant advantage over methods such as gridding, for instance, for which the validation of the constraints is only guaranteed on some points of the trajectory.

4.4.3 Sphere waypoints validation

When a waypoint $\mathbf{w}_i$ ($i \in [\![1, N-1]\!]$) is a *sphere* waypoint, the i-th piece of trajectory must end within a sphere of radius $r_{\mathbf{w}_i}$, centered on $\mathbf{w}_i$. This can be ensured by constraining its last control point inside this sphere, since clamped B-spline curves are used. Then, due

to the continuity of the trajectory, the next piece starts inside the sphere and validates $\mathbf{w}_i$ as well.

Each *sphere* waypoint validation can be formulated by one nonlinear constraint

$$\|\mathbf{p}_{i,n} - \mathbf{w}_i\|_2^2 - r_{\mathbf{w}_i}^2 \leqslant 0 \tag{4.59}$$

which can be rewritten

$$\mathbf{p}_{i,n}^\top \mathbf{p}_{i,n} - 2\,\mathbf{w}_i^\top \mathbf{p}_{i,n} + \mathbf{w}_i^\top \mathbf{w}_i - r_{\mathbf{w}_i}^2 \leqslant 0 \tag{4.60}$$

All theses quadratic constraints can be regrouped as a nonlinear function of $\mathbf{x}$

$$\mathbf{g}_{\text{sphere}}(\mathbf{x}) \leqslant 0 \tag{4.61}$$

These constraints can also be approximated by a set of linear constraints on $\mathbf{p}_{i,n}$, by replacing the sphere by a convex polyhedron (e.g. a platonic or archimedean solid). Since $\mathbf{p}_{i,n}$ is part of the reduced control points for a *sphere* waypoint and does not need reconstruction, these linear constraints in $\mathbf{p}_{i,n}$ are also linear in $\mathbf{x}$. In this case, the *sphere* waypoints validation constraints can be expressed as follows

$$\mathbf{A}_{\text{sphere}}\,\mathbf{x} - \mathbf{b}_{\text{sphere}} \leqslant 0 \tag{4.62}$$

4.4.4 Time derivatives bounds

For each piece of trajectory, bounds on the magnitude of the time derivatives can be imposed. The convex hull property can be used to this aim, as for the corridor constraints. For $l \in [\![1, L+1]\!]$, the l-th time derivative of the trajectory is also a clamped B-spline, of control points given by (4.18). Therefore, constraining these control points into a sphere ensures that the magnitude of this derivative will never exceed the value of the radius of this sphere

$$\forall (i,j) \in [\![1,N]\!] \times [\![0, n-l]\!] \quad \left\|\mathbf{p}_{i,j}^{(l)}\right\|_2^2 - {d_{i,l}}_{\max}^2 \leqslant 0 \tag{4.63}$$

with ${d_{i,l}}_{\max}$ the bound on the l-th time derivative for the i-th piece of trajectory and the control points $\mathbf{p}_{i,j}^{(l)}$ given by (4.18) and (4.45). Since equation (4.18) is nonlinear relatively to the knot steps, the constraint (4.63) is a nonlinear constraint in $\mathbf{x}$

$$\mathbf{g}_{\text{derivatives}}(\mathbf{x}) \leqslant \mathbf{0} \tag{4.64}$$

In order to illustrate the behavior of the overall trajectory generation strategy, it is now applied to the case of aerial cinematography with a quadrotor, on the benchmarks defined in chapter 3.

4.5 Application to aerial cinematography with quadrotors

In this section, we apply the strategy described above to perform a cinematographic flight plan with a quadrotor. To this aim, we generate a minimum-time trajectory with speed, acceleration and jerk constraints as in chapter 3. From a slow but feasible suboptimal trajectory, the knot steps of each piece of trajectory are reduced and their control points optimized until the constraints on the derivatives are active. It can be noticed that

ensuring feasibility through bounds on the derivative can introduce some conservatism in the trajectory as a quadrotor dynamics is more complex than a multiple integrator. However, in the context of video making, smoothness and video quality can be considered more important than having the most time-optimal trajectory. The minimization of the duration of the trajectory is just one way to produce a trajectory suited for cinematography. For a more generic solution, not specific to video making, this strategy could be improved by including the dynamics of the quadrotor, in a similar fashion as in [88] (where flatness properties are used jointly with uniform B-splines).

4.5.1 Tuning parameters for trajectory generation

In the previous chapter, we obtained bounds both on the acceleration and the jerk guaranteeing the feasibility of the trajectory (section 3.3). The smoothness of the trajectory was obtained by minimizing the jerk while its feasibility was given by constraints on the acceleration and jerk. However, in this minimum-time B-spline formulation, the smoothness has not been taken into account, the time is minimized so that the trajectory sticks as much as possible to the bounds on the derivatives, i.e. the speed reference and the feasibility constraints. One way to obtain a smooth trajectory is then to use stronger bounds on the acceleration and jerk than what is required for the feasibility of the trajectory. In other terms, the smoothness is now enforced through jerk and acceleration constraints. The following values are respectively chosen as bounds on the acceleration and jerk: $a_{\max} = 1\,\mathrm{m{\cdot}s^{-2}}$ and $j_{\max} = 0.3\,\mathrm{m{\cdot}s^{-3}}$, respectively.

Furthermore, a consequence of the bounded jerk is that the acceleration has to be continuous. For this reason the trajectory was chosen $\mathcal{C}^2$ continuous in chapter 3. In order to compensate the fact that the smoothness of the trajectory is now only obtained by stronger bounds on the acceleration and jerk, we decided to impose the jerk to be continuous too, i.e. we chose a $\mathcal{C}^3$ continuous trajectory. This typically leads to a continuous angular velocity of the quadrotor, as in [13] or [85].

The degree of the trajectory is then chosen as the lowest one allowing a $\mathcal{C}^3$ trajectory, i.e. since the clamped B-splines are $\mathcal{C}^{k-1}$ continuous (see section 4.2.3), $k = 4$. Finally, in order to use the initialization strategy described below, each piece of trajectory contains $n + 1 = 11$ control points and thus 7 knot steps.

For each piece of trajectory, the speed must not exceed the reference and the norm of the acceleration and of the jerk are respectively limited to $a_{\max}$ and $j_{\max}$. The snap is also limited in order to prevent the optimal trajectory to present short periods of time with an excessively high snap, leading to an almost discontinuous behavior of the jerk. In the following, we choose $s_{\max} = 0.5\,\mathrm{m{\cdot}s^{-4}}$.

4.5.2 Initial feasible trajectory

As an initial feasible guess for the optimization process, we first propose a rest-to-rest trajectory with a BOB snap profile, divided into three phases: acceleration, cruising and braking. We distinguish the following two cases: first, a case with a cruising phase (4.65a), and, second, a case where the distance is too short for this (4.65b). The initial solution is thus given as follows.

Proposition 3 *The initial piecewise clamped B-spline trajectory given by the following snap profile*

For $i \in [\![1, N]\!]$

If $4\,\widetilde{v}_i \sqrt[3]{\frac{\widetilde{v}_i}{2s}} < \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2$

$$\begin{gathered} \Delta\tau_{\text{acc}} = \sqrt[3]{\frac{\widetilde{v}_i}{2s}}, \quad \Delta\tau_{\text{cruise}} = \frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{\widetilde{v}_i} - 4\Delta\tau_{\text{acc}} \\ \boldsymbol{\Delta\tau}_{\text{init}} = (\Delta\tau_{\text{acc}} \;\; 2\Delta\tau_{\text{acc}} \;\; \Delta\tau_{\text{acc}} \;\; \Delta\tau_{\text{cruise}} \;\; \Delta\tau_{\text{acc}} \;\; 2\Delta\tau_{\text{acc}} \;\; \Delta\tau_{\text{acc}}) \\ \mathbf{P}^{(4)}_{\text{init}} = \begin{pmatrix} s\,\mathbf{u}_i & -s\,\mathbf{u}_i & s\,\mathbf{u}_i & \mathbf{0} & -s\,\mathbf{u}_i & s\,\mathbf{u}_i & -s\,\mathbf{u}_i \end{pmatrix} \end{gathered} \tag{4.65a}$$

Else

$$\begin{gathered} \Delta\tau_{\text{acc}} = \sqrt[4]{\frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{8s}} \\ \boldsymbol{\Delta\tau}_{\text{init}} = \left(\Delta\tau_{\text{acc}} \;\; 2\Delta\tau_{\text{acc}} \;\; \frac{1}{2}\Delta\tau_{\text{acc}} \;\; \frac{1}{2}\Delta\tau_{\text{acc}} \;\; \Delta\tau_{\text{acc}} \;\; 2\Delta\tau_{\text{acc}} \;\; \Delta\tau_{\text{acc}}\right) \\ \mathbf{P}^{(4)}_{\text{init}} = \begin{pmatrix} s\,\mathbf{u}_i & -s\,\mathbf{u}_i & s\,\mathbf{u}_i & s\,\mathbf{u}_i & -s\,\mathbf{u}_i & s\,\mathbf{u}_i & -s\,\mathbf{u}_i \end{pmatrix} \end{gathered} \tag{4.65b}$$

with

$$\widetilde{v}_i = \min\left(v_i, \frac{8\,a_{\max}^2}{9\,j_{\max}}\right), \quad s = \frac{3\,j_{\max}^2}{2\,a_{\max}} \tag{4.66}$$

and ν_i *the speed reference of the i-th piece of trajectory, is a feasible solution of the NLP* (4.49).

A proof of feasibility of this trajectory, relatively to the constraints of the problem (4.49), is given in the appendix C. The initial control points for this piece of trajectory can then be deduced from this snap profile using (4.22).

An example of such a trajectory is represented on figure 4.8, for which $v_i = \frac{8\,a_{\max}^2}{9\,j_{\max}}$. The blue lines represent the evaluation of the velocity, acceleration, jerk and snap of a rest-to-rest trajectory, projected along the vector $\mathbf{u}_i$ (the trajectory is a straight line of direction $\mathbf{u}_i$), while the red lines represent the bounds.

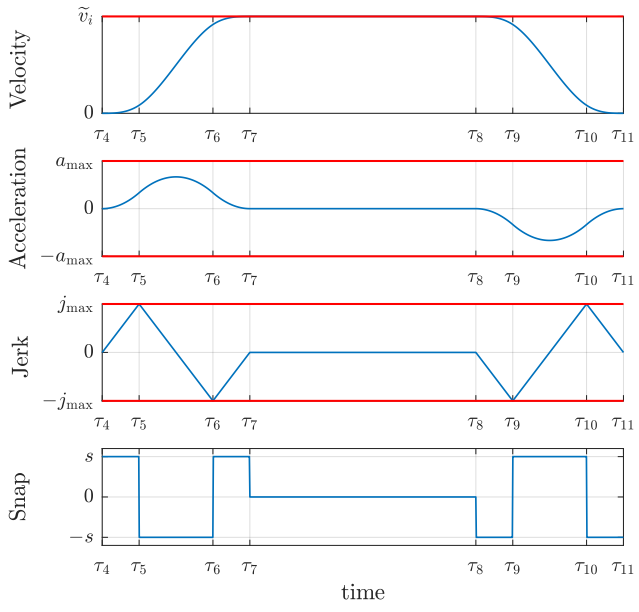

FIGURE 4.8 – Initial rest-to-rest trajectory profile

4.5.3 Simulation

The proposed trajectory generation strategy is applied to the benchmarks 1 (figure 4.9) and 2 (figure 4.10) defined in section 3.5.3.

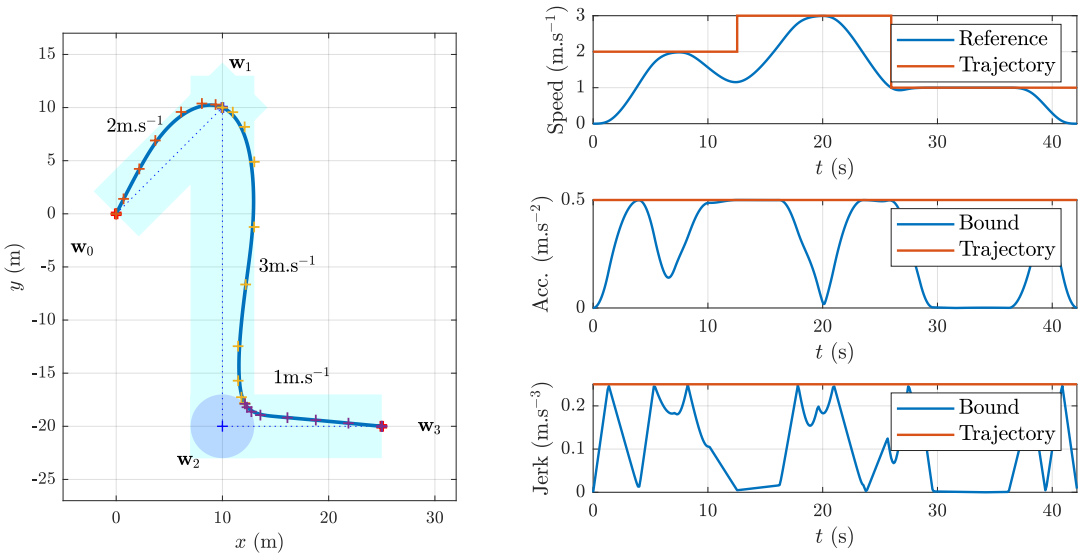

FIGURE 4.9 – Minimum-time B-spline trajectory for benchmark 1

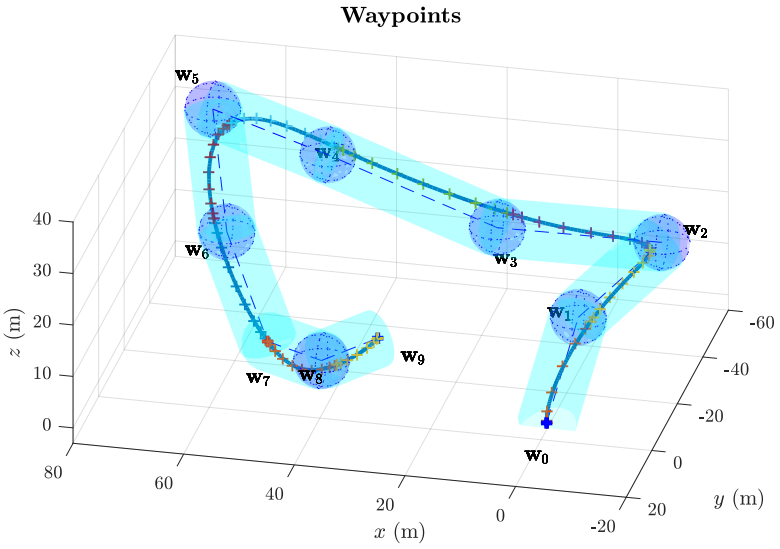

FIGURE 4.10 – Minimum-time B-spline trajectory for benchmark 2

The constraints are respected and the speed profile is much better than the speed profiles obtained with the bi-level optimization. However on benchmark 1, the position seems less smooth, since the penalty on the jerk has been removed in comparison to section 3.6.

4.5.4 Comparison with uniform clamped B-splines

Often, *uniform* clamped B-splines are used, which means that the knot steps are all equal. This is not the case in this work, where they are optimized to fully exploit the piecewise polynomial nature of B-splines.

Using uniform B-spline makes the formulation easier and the computation faster, as most of the calculations can be performed once, during the initialization phase of the resolution of the optimization problem, or even be done off-line. This is especially convenient for on-board computation. However, there are only two ways to act on the derivatives of such a spline. They can either be scaled in magnitude by scaling the knot steps (which does not change the shape of the overall curve), or be changed locally by moving the control points (4.13), which modifies the shape of the trajectory. The latter solution is limited in presence of flight corridors.

The limitations of this kind of splines are illustrated on figure 4.11. In this example, a trajectory is generated for a simple flight plan of only 2 waypoints separated by 100 m, to join with a reference speed of $1\,\mathrm{m.s^{-1}}$ (the same as on figure 3.21), using the strategy described in this chapter and with the same parameters as in section 4.5.1. In the first case, a non-uniform clamped B-spline is used (figure 4.11 top), while in a second case, an uniform B-spline is used (figure 4.11 bottom). The speed profile of each trajectory is presented on figure 4.11, with each knot marked by a vertical gray line. Notice that since the two considered waypoints are *stop*, only 3 control points are optimized, the other 8 being imposed on the waypoints.

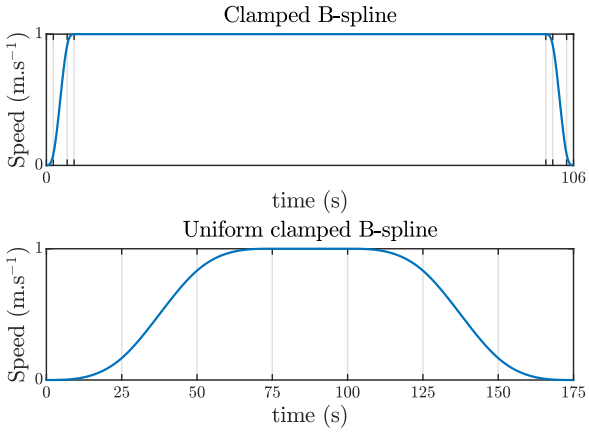

FIGURE 4.11 – Comparison of a non-uniform and an uniform minimum-time B-spline trajectories

Both solutions verify all the constraints: speed, acceleration, jerk, snap and corridor constraints, though the satisfaction of the corridor constraints is trivial as the trajectory is a straight line in this example. However, the uniform B-spline trajectory is much slower, due to long acceleration and braking phases. Indeed, for a B-spline curve of degree k, the k-th derivative is constant between 2 knots, i.e. for a duration equal to a knot step (see figure 4.8). For a uniform clamped B-spline, the knot steps being all equal, these durations are directly given by the length of the interval $[\tau_k, \tau_{n+1}[$ and by the number of knot steps, $n-k+1$, fixed by the number of control points and the polynomial degree. The constant knot steps of a uniform B-spline hence prevents the possibility to have arbitrarily long (or short) acceleration, deceleration and cruising phases.

In order to deal with this issue, one could introduce more control points, but considering

the wide variety of possible flight plans, it can be delicate to choose an adequate number of control points. In [115], control points are added or removed on-the-flight, during the optimization process, which results in a MIP that can significantly impact the computation time.

Freeing the knot steps by using non-uniform clamped B-splines, as it is proposed in this paper, allows the derivatives to be more independent relatively to the control points and the shape of the curve. This is an elegant and robust way to solve the issue, but, as for the MIP, it comes at a certain cost on the computation time.

Finally, the speed profile obtained with the non-uniform B-spline can be compared to the one obtained with the bi-level, minimum-time/minimum-jerk strategy of chapter 3, in section 3.7.4. Again, the non-uniform B-spline gives a much better speed profile. This is due to the piecewise nature of the B-spline allowing to parameterize much richer trajectories than with a single polynomial.

4.5.5 Outdoor experiment

In order to evaluate the cinematographic quality and the feasibility of the trajectories obtained with the method proposed above, a trajectory was generated *off-line* and then performed by a real quadrotor. This test consists in the performance of a cinematographic flight plan containing 6 waypoints. A minimum-time B-spline trajectory is generated to join the waypoints while respecting corridors and bounds on the derivatives. The trajectory is generated off-line, using a waypoint horizon $N_{\mathbf{w}} = 3$, which means that the trajectory is computed in 3 steps. An undersampled linear predictive controller such as described in chapter 5 is used to track the camera angles references smoothly, without introducing too much framing error thanks to the anticipative aspect of the controller and the knowledge of the overall reference trajectory. The flight plan is performed under an average wind of approximately $5.3\,\mathrm{m{\cdot}s^{-1}}$ (19 km/h), with gusts up to $9.0\,\mathrm{m{\cdot}s^{-1}}$ (32 km/h).

Though this work was initially designed for the Parrot Bebop 2, Parrot released a new product during this thesis, in 2018: the Parrot ANAFI quadrotor (see figure 4.12, [93]). Since this drone has a better camera and more capabilities, we performed the experiment with a Parrot ANAFI instead of a Bebop 2, in order to better judge the results of the minimum-time B-spline trajectory generation algorithm in the context of cinematography.

FIGURE 4.12 – Parrot ANAFI quadrotor

A video of the outdoor flight is available at https://youtu.be/A0oYx268sis. figure 4.13 presents the flight plan performed by the quadcopter and the corresponding optimal trajectory, as well as the position tracking error during the flight. The latter is computed using satellite navigation and the internal sensors of the drone.

The trajectory satisfies the specifications of section 1.2.3. The trajectory is smooth and it is difficult to guess the position of the waypoints only by looking at the trajectory or the video, which comforts the idea that the trajectory is natural. The position tracking error remains low for an outdoor flight ($< 0.4\,\mathrm{m}$ despite the wind gusts), which gives confidence

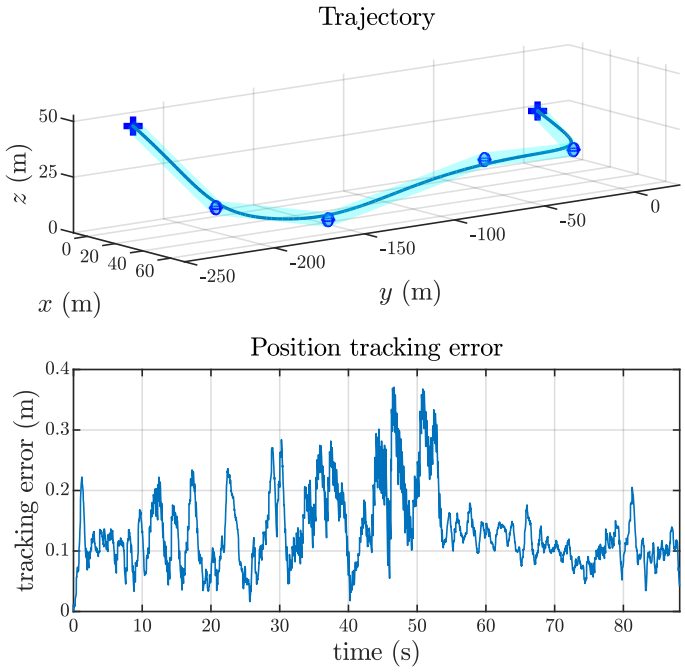

FIGURE 4.13 – Trajectory for the flight experiment (top) and position tracking error during the flight (bottom)

in the feasibility of the trajectory.

4.6 Initialization

The initialization method we developed in section 4.5.2 has the advantage to be an analytical solution, but it is largely suboptimal. Since it uses a fixed the number of knot steps, a knot insertion algorithm[1] [29] must be used if we want to use more knot steps for each piece of trajectory. However, this method prevents the use of less knot steps.

As a consequence, once the overall trajectory generation strategy was validated, we studied other ways to initialize the NLP (4.49). This section proposes 4 novel ways to generate suboptimal minimum-time clamped B-spline trajectories with corridor constraints and bounds on the derivative, that can be used to initialize the problem (4.49) for any number of control points, any polynomial degree and any dimension (and thus not restricted to the case of quadrotor cinematography).

We start by two new methods to generate generic feasible rest-to-rest uniform clamped B-spline trajectories.

4.6.1 Uniform rest-to-rest B-spline trajectory

In section 4.5.2 we proposed to initialize each piece of trajectory by a rest-to-rest trajectory, given by a *bang-off-bang* snap law. However this is not the only way to obtain a

[1]For a given B-spline $\boldsymbol{\mathcal{B}}_1$, a knot insertion algorithm returns the control points of another B-spline $\boldsymbol{\mathcal{B}}_2$, with the same knots as $\boldsymbol{\mathcal{B}}_1$ but with an additional knot $\tilde{\tau} \in [\tau_0, \tau_m]$, such that the two curves as well as their derivatives are equal.

rest-to-rest trajectory. In this section we propose another way to generate a feasible rest-to-rest B-spline trajectory using an uniform B-spline curve. For each piece of trajectory we first choose a suitable set of control points and then the knot steps.

For a trajectory of differentiability class $\mathcal{C}^L$, we suppose that the polynomial degree is at least $k \geqslant L+1$ and that there are at least $(n+1) \geqslant 2\,(L+1)$ control points for each piece of trajectory.

Control points. For the i-th piece of trajectory, between the waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$, a rest-to-rest trajectory can simply be obtained by setting the first half of the control points on $\mathbf{w}_{i-1}$ and the last half on $\mathbf{w}_i$. In case of an odd number of control points, the middle one can be set on $\mathbf{w}_{i-1}$ for instance. The initial control points are set as follows

$$\forall i \in [\![1, N]\!] \quad \forall j \in [\![0, n]\!] \quad \mathbf{p}_{i,j} = \begin{cases} \mathbf{w}_{i-1} & \text{if } j \leqslant \text{floor}(n/2) \\ \mathbf{w}_i & \text{otherwise} \end{cases} \tag{4.67}$$

Since the waypoints respect the flight corridors, so does the control points and the trajectory respect the corridor constraints, too. Finally, since each piece of trajectory is supposed to contain at least $(n+1) \geqslant 2\,(L+1)$ control points, there are at least $L+1$ control points on the waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$. As a consequence, the trajectory passes exactly on the waypoints and the L first derivatives of the position on the waypoints are null (since at least $(L+1)$ are superimposed on the waypoints, as detailed in (4.13) and section 4.3). Then, the waypoints validation criteria are also satisfied. Only the knot steps vector thus remains to be chosen.

Knot steps. In this section, in order to initialize the knot vector so that the trajectory is feasible relatively to the constraints of the problem (4.49), we propose to set all the knot steps equal. The initial trajectory is thus an uniform clamped B-spline. We now show that, having fixed the control points, the choice of the smallest knot steps respecting the constraints of (4.49) can be analytically computed.

Since we use an uniform clamped B-spline for this initial trajectory, all the knot steps are equal on a given piece of trajectory and the vector of knot steps can be expressed as follows, for the i-th piece of trajectory

$$\boldsymbol{\Delta\tau}_i = \Delta\tau_i\, \mathbf{1}_{1\times(n-k+1)} \tag{4.68}$$

This considerably simplifies the expression of the control points of the time derivatives of the B-spline curve. Indeed, the expression of the matrix $\mathbf{D}_{0\to l}$ in (4.18) then simplifies to

$$\mathbf{D}_{0\to l}(\boldsymbol{\Delta\tau}) = \frac{1}{\boldsymbol{\Delta\tau}_i^l}\, \mathbf{D}_{0\to l}\big(\mathbf{1}_{1\times(n-k+1)}\big) \tag{4.69}$$

and the expression of control points of the l-th derivative of the position becomes

$$\mathbf{P}_i^{(l)} = \frac{1}{\boldsymbol{\Delta\tau}_i^l}\, \mathbf{P}_i\, \mathbf{D}_{0\to l}\big(\mathbf{1}_{1\times(n-k+1)}\big) = \frac{1}{\boldsymbol{\Delta\tau}_i^l}\, \widehat{\mathbf{P}}_i^{(l)} \tag{4.70}$$

with $\widehat{\mathbf{P}}_i^{(l)} = \mathbf{P}_i\, \mathbf{D}_{0\to l}\big(\mathbf{1}_{1\times(n-k+1)}\big)$ fixed, representing the control points of the l-th derivative obtained for a unitary knot step. Then the bound on the magnitude of each control points of the l-th derivative of the i-th piece of trajectory is given by

$$\forall j \in [\![1, n-l]\!] \quad \Big\|\mathbf{p}^{(l)}[i,j]\Big\|_2 = \frac{1}{\boldsymbol{\Delta\tau}_i^l}\, \Big\|\hat{\mathbf{p}}^{(l)}[i,j]\Big\|_2 \leqslant d_{i,l_{\max}} \tag{4.71}$$

since the control points $\hat{\mathbf{p}}_{i,j}^{(l)}$ are fixed, this can be rewritten

$$\forall j \in [\![1, n-l]\!] \quad \boldsymbol{\Delta\tau}_i^l \geqslant \sqrt[l-1]{\frac{\left\|\hat{\mathbf{p}}_{i,j}^{(l)}\right\|_2}{d_{i,l_{\max}}}} \tag{4.72}$$

The control points respecting the corridor constraints and the waypoint validation criteria, only the bounds on the magnitude of the derivatives remain to be verified. The choice of the knot step $\boldsymbol{\Delta\tau}_i$ on each piece of trajectory can be set to the smallest value for which these bounds are met. Using the result (4.72), this can be formulated as the following optimization problem

$$\begin{aligned} \Delta\tau_i^* &= \arg \min_{\Delta\tau_i \in \mathbb{R}} \Delta\tau_i \\ \text{s.t.} &\begin{cases} \Delta\tau_i \geqslant \Delta\tau_{\min} \\ \forall j \in [\![1, n-1]\!] \quad \boldsymbol{\Delta\tau}_i \geqslant \dfrac{\left\|\hat{\mathbf{p}}_{i,j}^{(1)}\right\|_2}{d_{i,1_{\max}}} \\ \forall j \in [\![1, n-2]\!] \quad \boldsymbol{\Delta\tau}_i \geqslant \sqrt{\dfrac{\left\|\hat{\mathbf{p}}_{i,j}^{(2)}\right\|_2}{d_{i,2_{\max}}}} \\ \vdots \\ \forall j \in [\![1, n-L]\!] \quad \boldsymbol{\Delta\tau}_i \geqslant \sqrt[L-1]{\dfrac{\left\|\hat{\mathbf{p}}_{i,j}^{(L)}\right\|_2}{d_{i,L_{\max}}}} \end{cases} \end{aligned} \tag{4.73}$$

which can be rewritten in the following manner

$$\begin{aligned} \Delta\tau_i^* &= \arg \min_{\Delta\tau_i \in \mathbb{R}} \Delta\tau_i \\ \text{s.t.} \quad & \Delta\tau_i \, \mathbf{1}_{n_{\text{bounds}}} \geqslant \boldsymbol{\Delta\tau}_{\min} \end{aligned} \tag{4.74}$$

with $\boldsymbol{\Delta\tau}_{\min}$ the vector of all the right-term values of the inequality constraints in (4.73) and n_{bounds} the number of these constraints. The solution of this problem is given by the largest element of $\boldsymbol{\Delta\tau}_{\min}$, i.e.

$$\Delta\tau_i^* = \max \boldsymbol{\Delta\tau}_{\min} \tag{4.75}$$

By doing this for each piece of trajectory, we obtain a feasible rest-to-rest uniform clamped B-spline trajectory. The initial trajectory obtained by this method for the benchmark 1 is presented on figure 4.14. The trajectory parameters (number of knots, polynomial degree, derivative bounds etc.) are the same as in section 4.5.1.

Equally distributed control points. Alternatively, in order to reduce the suboptimality, some control points can be set between the waypoints rather than on them. The $L+1$ first and last control points are set on the waypoints to ensure the validation criteria of the waypoints by passing on them and cancelling the first $L+1$ derivatives of the position on the waypoints. The remaining points are then equally distributed between $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$.

$$\forall i \in [\![1, N]\!] \quad \forall j \in [\![0, n]\!] \quad \mathbf{p}_{i,j} = \begin{cases} \mathbf{w}_{i-1} & \text{if } j \leqslant L \\ \mathbf{w}_i & \text{if } j \geqslant n-L \\ \mathbf{w}_{i-1} + (j-L)\,\Delta\mathbf{w}_i & \text{otherwise} \end{cases} \tag{4.76}$$

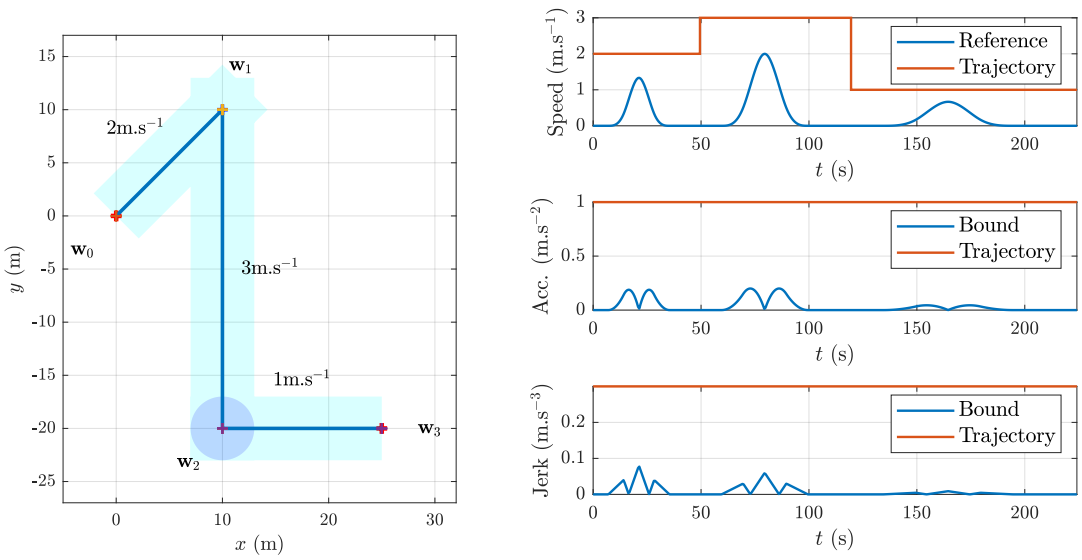

FIGURE 4.14 – Uniform rest-to-rest initialization for benchmark 1

with $\Delta \mathbf{w}_i = \frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{n-L}$. The knot steps are then obtained by the same method. The initial trajectory obtained by this method for the benchmark 1 is presented on figure 4.15. The trajectory parameters (number of knots, polynomial degree, derivative bounds etc.) are the same as in section 4.5.1.

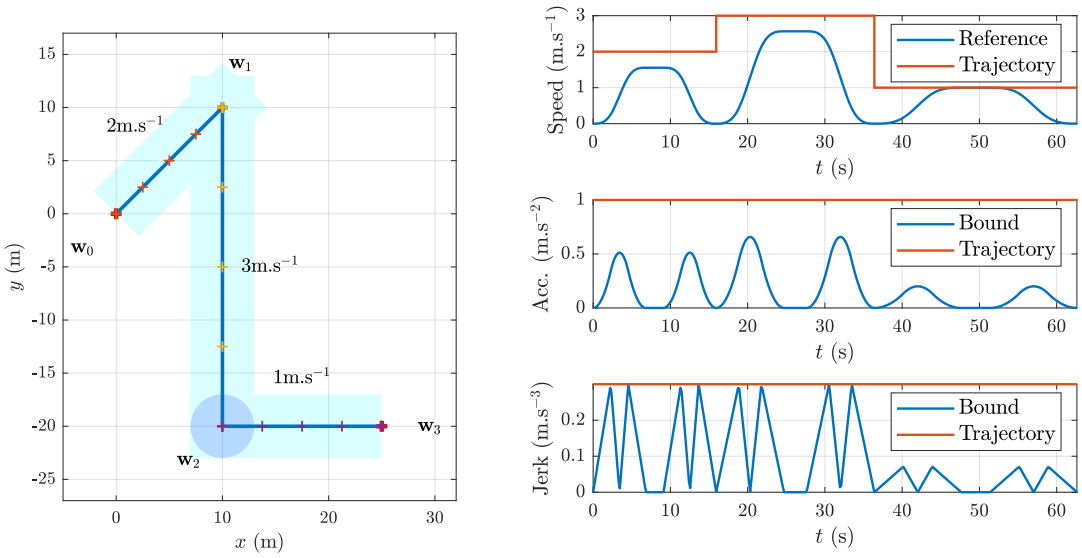

FIGURE 4.15 – Alternative uniform rest-to-rest initialization for benchmark 1

4.6.2 Uniform smooth B-spline trajectory

All the initialization methods proposed for now involve rest-to-rest trajectories. Stopping at each waypoint can induce a severe suboptimality of the trajectory. In this section,

we propose to use as initial solution for the NLP (4.49) a uniform B-spline trajectory minimizing the mean square of one of the derivatives of the position, as the jerk, for instance.

To this aim, we first set all the knot steps of each piece of trajectory to 1. Then, we optimize the control points so that the mean square of the jerk or another derivative of the overall trajectory is minimized, while verifying the waypoint validation criteria and the flight corridors. The obtained control points are fixed, and a similar method as for the rest-to-rest initialization is then used to compute the knot step vector.

The method is then similar to the one used in section 3.6 in the fact that two stages are used to generate the trajectory, one for the control points (equivalent to the computation of the polynomial coefficients in section 3.6) and one for the knot steps (equivalent to the computation of the times of flight in section 3.6). However, it differs in the fact that there is no feedback between these two levels. Here, the control points are first optimized, once, and then they are used to optimize the knot steps.

Since the knot steps are fixed, so is the duration of trajectory, and minimizing the mean square of the l-th time derivative of the position trajectory is equivalent to minimize

$$J_l = \int_{t_0}^{t_N} \left\| \frac{\mathrm{d}^l}{\mathrm{d}t^l} \boldsymbol{\zeta}(t) \right\|_2^2 \mathrm{d}t = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \left(\frac{\mathrm{d}^l}{\mathrm{d}t^l} \boldsymbol{\mathcal{B}}_i(t) \right)^\top \left(\frac{\mathrm{d}^l}{\mathrm{d}t^l} \boldsymbol{\mathcal{B}}_i(t) \right) \mathrm{d}t \tag{4.77}$$

Though the square of a B-spline can be computed analytically (it is also a B-spline, of polynomial degree $2k$) and then integrated to get the mean square of this B-spline, it is not the most convenient way to minimize this derivative. An alternative to minimize the mean square of the l-th derivative is then to minimize the norm of its control points. Indeed, since the derivative is constrained inside the convex hull of its control points, reducing their norm compresses the convex hull and thus the mean square of the derivative. As a consequence, we minimize the alternative cost function

$$J_l = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=0}^{n} \mathbf{p}_{i,j}^{(l)\top} \mathbf{p}_{i,j}^{(l)} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=0}^{n} \left(p_{i,j}^{(l),x} \right)^2 + \left(p_{i,j}^{(l),y} \right)^2 + \left(p_{i,j}^{(l),z} \right)^2 \tag{4.78}$$

We will now show that, defining $n_\mathrm{p} = \sum_{i=1}^{N} \widetilde{n}_i$ and the optimization vector $\mathbf{x_p} \in \mathbb{R}^{n_\mathrm{p}}$ such that

$$\mathbf{x_p} = \begin{pmatrix} \widetilde{\mathbf{P}}_1^x & \widetilde{\mathbf{P}}_1^y & \widetilde{\mathbf{P}}_1^z & \widetilde{\mathbf{P}}_2^x & \widetilde{\mathbf{P}}_2^y & \widetilde{\mathbf{P}}_2^z & \dots & \widetilde{\mathbf{P}}_N^x & \widetilde{\mathbf{P}}_N^y & \widetilde{\mathbf{P}}_N^z \end{pmatrix}^\top \tag{4.79}$$

With fixed the knot steps, the cost function J_l can be expressed as a quadratic form of $\mathbf{x_p}$.

Since the knot steps have been set to 1, the reconstruction of the control points of the i-th piece (4.45) can be simplified

$$\mathbf{P}_i = \begin{cases} \mathbf{B}_{\mathrm{f}i} + \widetilde{\mathbf{P}}_i \, \mathbf{B}_{\mathrm{c}i} & \text{if } i = 1 \\ \mathbf{B}_{\mathrm{f}i} + \widetilde{\mathbf{P}}_{i-1} \, \mathbf{B}_{\mathrm{p}i} + \widetilde{\mathbf{P}}_i \, \mathbf{B}_{\mathrm{c}i} & \text{otherwise} \end{cases} \tag{4.80}$$

with

$$\begin{aligned}
\mathbf{B}_{\mathrm{f}i} &= \begin{cases} \mathbf{T}_{\mathrm{fixed}i}(\mathbf{1}_{1\times n-k+1}) & \text{if } i = 1 \\ \mathbf{T}_{\mathrm{fixed}i}(\mathbf{1}_{1\times n-k+1}) + \mathbf{T}_{\mathrm{lock}i}(\mathbf{1}_{1\times n-k+1}, \mathbf{1}_{1\times n-k+1}) & \text{otherwise} \end{cases} \\
&= \begin{pmatrix} \mathbf{B}_{\mathrm{f}i}^{x} \\ \mathbf{B}_{\mathrm{f}i}^{y} \\ \mathbf{B}_{\mathrm{f}i}^{z} \end{pmatrix} \\
\mathbf{B}_{\mathrm{p}i} &= \mathbf{T}_{\mathrm{previous}_i}(\mathbf{1}_{1\times n-k+1}, \mathbf{1}_{1\times n-k+1}) \\
\mathbf{B}_{\mathrm{c}i} &= \mathbf{T}_{\mathrm{current}i}
\end{aligned} \tag{4.81}$$

In the case of fixed knot steps, the reconstruction is a linear operation on the reduced control points.

Let us focus on the x dimension of the i-th piece of trajectory, with $i > 1$. The x coordinates of the control points of the l-th time derivative of this piece of trajectory are given by

$$\mathbf{P}_i^{(l),x} = \mathbf{P}_i^l \, \mathbf{R}_l = \left(\mathbf{B}_{\mathrm{f}i} + \widetilde{\mathbf{P}}_{i-1} \, \mathbf{B}_{\mathrm{p}i} + \widetilde{\mathbf{P}}_i \, \mathbf{B}_{\mathrm{c}i} \right) \mathbf{R}_l \tag{4.82}$$

with $\mathbf{R}_l = \mathbf{D}_{0\to l}(\mathbf{1}_{1\times n-k+1})$. The sum of the squares of the x coordinates of all the control points of this piece of trajectory is given by

$$\begin{aligned}
\sum_{j=0}^{n} \left(p_{i,j}^{(l),x} \right)^2 &= \mathbf{P}_i^{(l),x} \, \mathbf{P}_i^{(l),x\top} \\
&= \widetilde{\mathbf{P}}_{i-1}^{x} \, \mathbf{B}_{\mathrm{p}i} \, \mathbf{R}_l \, \mathbf{R}_l^\top \, \mathbf{B}_{\mathrm{p}i}^\top \, \widetilde{\mathbf{P}}_{i-1}^{x\top} + \widetilde{\mathbf{P}}_i^{x} \, \mathbf{B}_{\mathrm{c}i} \, \mathbf{R}_l \, \mathbf{R}_l^\top \, \mathbf{B}_{\mathrm{c}i}^\top \, \widetilde{\mathbf{P}}_i^{x\top} + 2 \, \widetilde{\mathbf{P}}_{i-1}^{x} \, \mathbf{B}_{\mathrm{p}i} \, \mathbf{R}_l \, \mathbf{R}_l^\top \, \mathbf{B}_{\mathrm{c}i}^\top \, \widetilde{\mathbf{P}}_i^{x\top} \\
&\quad + 2 \, \mathbf{B}_{\mathrm{f}i}^{x} \, \mathbf{R}_l \, \mathbf{R}_l^\top \left(\mathbf{B}_{\mathrm{p}i}^\top \, \widetilde{\mathbf{P}}_{i-1}^{x\top} + \mathbf{B}_{\mathrm{c}i}^\top \, \widetilde{\mathbf{P}}_i^{x\top} \right) + 2 \, \mathbf{B}_{\mathrm{f}i}^{x} \, \mathbf{R}_l \, \mathbf{R}_l^\top \, \mathbf{B}_{\mathrm{f}i}^{x\top}
\end{aligned} \tag{4.83}$$

The expression for the other dimensions is obtained similarly, while the case $i = 1$ is dealt with by removing the terms relative to the previous piece of trajectory. Then, the cost function J_l can be written

$$J_l = \mathbf{x}_\mathbf{p}^\top \, \mathbf{H}_l \, \mathbf{x}_\mathbf{p} + \mathbf{f}_l^\top \mathbf{x}_\mathbf{p} + g_l \tag{4.84}$$

Since the reconstruction of the full set of control points is now a linear operation, all the corridor constraints can be expressed as linear constraints on the reduced control points rather than only the corridor constraints on the reduced control points (as in section 4.4.2). Furthermore, by using a linear formulation of the *sphere* waypoints validation constraints, the computation of the control points consists in solving a QP problem under linear constraints

$$\begin{aligned}
\mathbf{x}_\mathbf{p}^* &= \arg \min_{\mathbf{x}_\mathbf{p} \in \mathbb{R}^{n_\mathrm{p}}} \; \mathbf{x}_\mathbf{p}^\top \, \mathbf{H}_l \, \mathbf{x}_\mathbf{p} + \mathbf{f}_l^\top \mathbf{x}_\mathbf{p} \\
&\text{s.t.} \begin{cases} \mathbf{A}'_{\mathrm{corridor}} \, \mathbf{x}_\mathbf{p} - \mathbf{b}'_{\mathrm{corridor}} \leqslant 0 \\ \mathbf{A}_{\mathrm{sphere}} \, \mathbf{x}_\mathbf{p} - \mathbf{b}_{\mathrm{sphere}} \leqslant 0 \end{cases}
\end{aligned} \tag{4.85}$$

with $\mathbf{A}'_{\mathrm{corridor}}$, $\mathbf{b}'_{\mathrm{corridor}}$ the matrix and vector for expressing the linear corridor constraints and $\mathbf{A}_{\mathrm{sphere}}$ and $\mathbf{b}_{\mathrm{sphere}}$ the matrix and vector for expressing the *sphere* waypoints validation constraints.

Once the control points have been computed, the knot steps are then obtained by the same method as in the previous section and given by (4.75). The initial trajectory obtained by this method, by minimizing the jerk, for the benchmark 1, is presented on figure 4.16. The trajectory parameters (number of knots, polynomial degree, derivative bounds etc.)

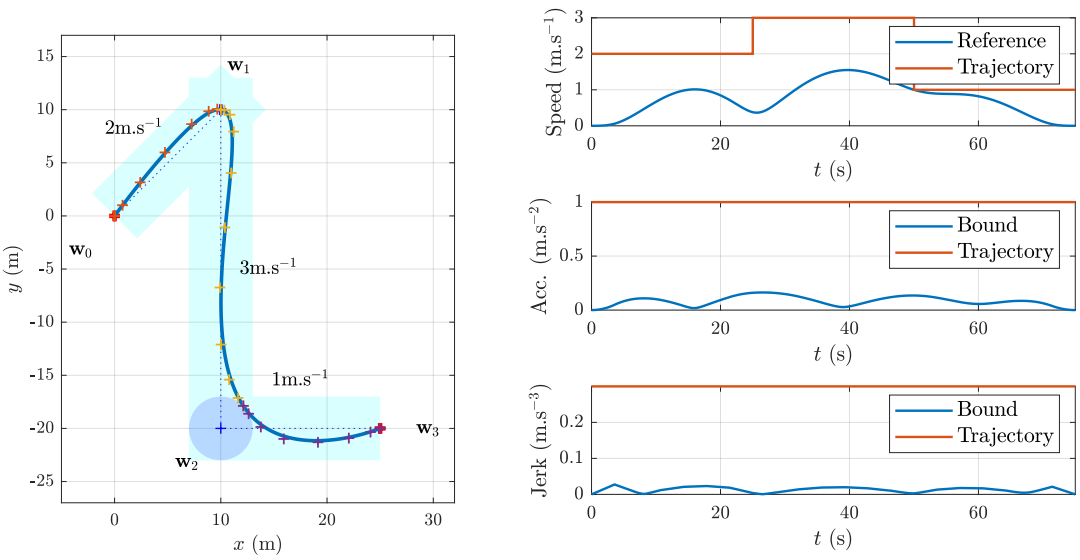

FIGURE 4.16 – Minimum-jerk uniform initialization for benchmark 1

are the same as in section 4.5.1. The obtained initial trajectory is smooth but the speed profile is still largely suboptimal. To overcome this inconvenient, we propose another to generate a feasible trajectory by using a non-uniform B-spline trajectory.

4.6.3 Non-uniform smooth B-spline trajectory

In order to improve the speed profile of the previous method in order to get closer to an optimal solution for the problem (4.49), we propose to free the knot steps of the uniform B-spline trajectory obtained in the method of section 4.6.2 and to optimize them to reduce the duration, while maintaining the feasibility. The differences with the NLP (4.49) are the following

- First, only the knot steps are optimized, the control points are fixed as the solution of the QP problem (4.85).

- Second, the control points being fixed, there is need to reconstruct the set of control points anymore. The issue is that the control points reconstruction implicitly ensures the continuity of the derivatives at the connections between the different pieces of trajectory. This continuity must then be enforced by an additional constraint on the knot steps now, as further detailed.

Defining $n_\mathrm{p} = N\,(n-k+1)$ and $\mathbf{x}_\tau \in \mathbb{R}^{n_\mathrm{p}}$ such that

$$\mathbf{x}_\tau = \begin{pmatrix} \boldsymbol{\Delta\tau}_1 & \boldsymbol{\Delta\tau}_2 & \dots & \boldsymbol{\Delta\tau}_N \end{pmatrix}^\top \tag{4.86}$$

the optimization problem to solve is then

$$\mathbf{x}_\tau^* = \arg \min_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \sum_{i=1}^{N}\sum_{j=0}^{n-k} \Delta\tau_{i,j} \quad \text{s.t.} \begin{cases} \mathbf{x}_\tau \geqslant \mathbf{x}_{\tau\min} \\ \mathbf{g}_{\text{derivatives}}(\mathbf{x}_\tau) \leqslant 0 \\ \mathbf{g}_{\text{continuity}}(\mathbf{x}_\tau) = 0 \end{cases} \tag{4.87}$$

The nonlinear continuity constraints at the connection can be formulated as follows

$$\forall i \in [\![2, N]\!] \quad \forall l \in [\![1, L]\!] \quad \mathbf{P}_{i-1}\,\mathbf{D}_{0\to l}(\boldsymbol{\Delta\tau}_{i-1}) \begin{pmatrix} \mathbf{0}_{n-l\times 1} \\ 1 \end{pmatrix} - \mathbf{P}_i\,\mathbf{D}_{0\to l}(\boldsymbol{\Delta\tau}_i) \begin{pmatrix} 1 \\ \mathbf{0}_{n-l\times 1} \end{pmatrix} = 0 \tag{4.88}$$

All these constraints can be written as a function of the decision vector $\mathbf{x}_\tau$, i.e. $\mathbf{g}_{\text{continuity}}(\mathbf{x}_\tau) = 0$. Though there are additional nonlinear equality constraints relatively to (4.49), there are also less inequality constraints and much less decision variables, as the control points are fixed. As a consequence, the resolution of this problem (4.87) is much faster than the original NLP (4.49), which allows using this method for the initialization of (4.49). The initial trajectory obtained with this method, by minimizing the jerk, for the benchmark 1, is presented on figure 4.17. The trajectory parameters (number of knots, polynomial degree, derivative bounds etc.) are the same as in section 4.5.1.

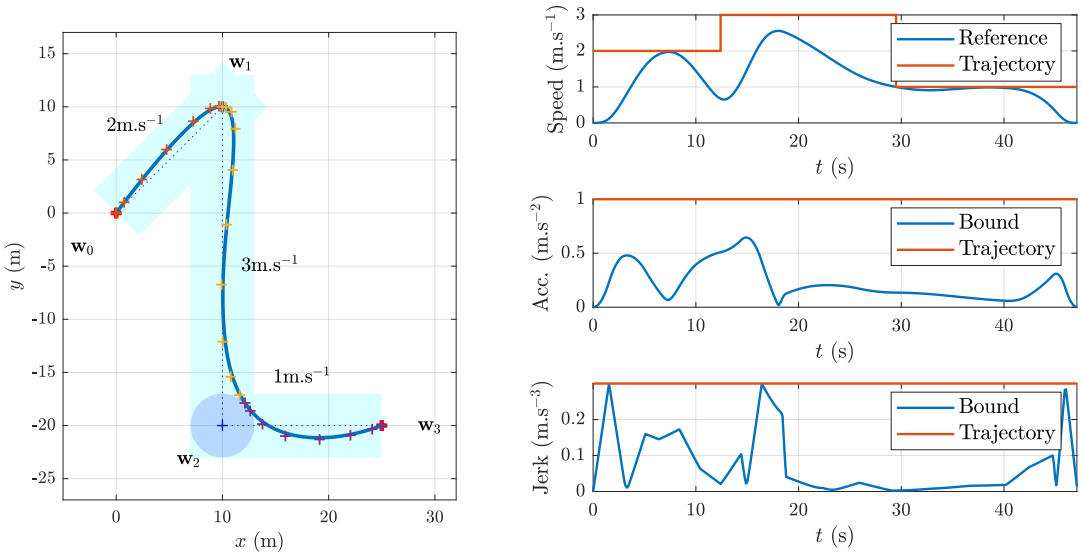

FIGURE 4.17 – Minimum-jerk non-uniform initialization for benchmark 1

Figure 4.18 illustrates an example with 15 knot steps per piece of trajectory rather than 8 as in the previous examples. The shape of the obtained initial trajectory as well as its speed profile almost meet the aesthetic requirements for cinematography. The only imperfections of this trajectory are that it is slow to decelerate at the end of the second piece of trajectory and that the acceleration and jerk profile are not really smooth nor natural.

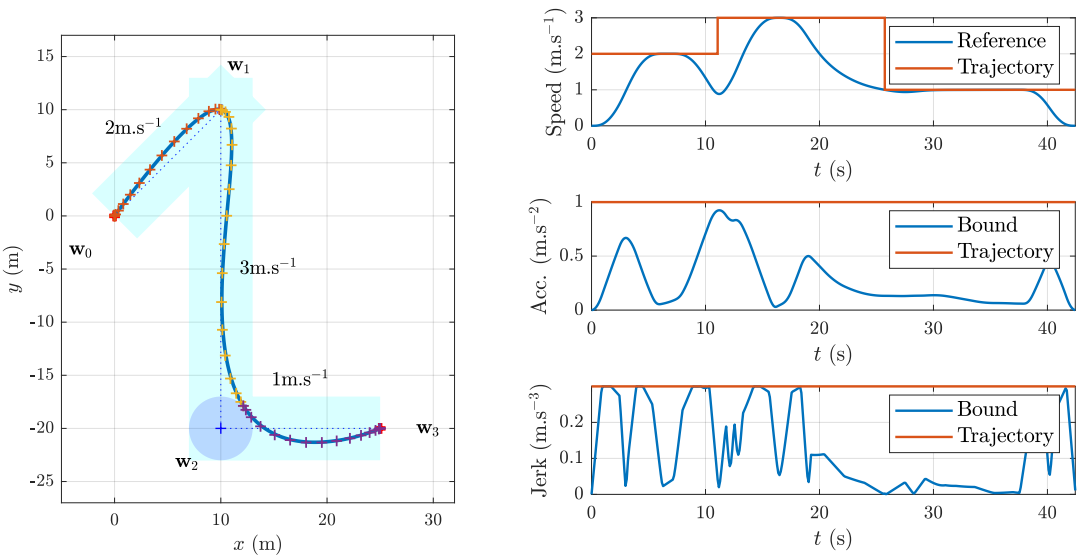

FIGURE 4.18 – Minimum-jerk non-uniform initialization for benchmark 1 with additional knots

4.6.4 Comparison of the initialization methods

In this section, we present a comparison of the different initialization methods. For the benchmark 1, we compare for each method the number of iterations required by the SQP solver of the `fmincon` function of Matlab© to solve the problem (4.49). The trajectory parameters (number of knots, polynomial degree, derivative bounds etc.) are the same as in section 4.5.1. The results are given in table 4.1, with

- The method 1 designating the *bang-off-bang* snap initialization (section 4.5.2)
- The method 2 and 3 designating, respectively, the uniform rest-to-rest initialization (section 4.6.1) with the control points on the waypoints and with the control points equally distributed
- The methods 4, 5, 6 and 7 designating the initialization proposed in section 4.6.2, minimizing respectively the speed, acceleration, jerk and snap, respectively
- The methods 8, 9, 10 and 11 designating the initialization proposed in section 4.6.3, minimizing respectively the speed, acceleration, jerk and snap, respectively

Method	1	2	3	4	5	6	7	8	9	10	11
Iterations	162	321	166	109	145	183	131	97	127	153	112

TABLE 4.1 – Comparison of the B-spline trajectory initialization methods

All the methods lead to the same solution of the NLP (4.49). For this example, it seems that the non-uniform initialization minimizing the speed performs best. However, the results on only one example are not sufficient to conclude. Future work is planned to automatically test these methods on many random flight plans.

4.7 Conclusion

This chapter presented a trajectory generation algorithm suited for a wide variety of flight plans, guaranteeing the corridors and the bounds on the derivatives. To this aim, the piecewise nature of clamped B-splines is exploited, by optimizing their knots and control points, in order to obtain a time-optimal trajectory.

To achieve this, a first contribution of this chapter is to introduce a new compact representation of a piecewise B-spline trajectory satisfying the waypoint validation criteria defined in section 1.2.2 as well as the continuity of the derivatives at the connections between the pieces.

Then, as a second contribution, we propose one technique to obtain a minimum-time B-spline trajectory, by optimizing the knot steps and the control points of the trajectory. The feasibility of the trajectory is obtained by bounding the magnitude of the time-derivatives of the position with the values obtained in section 3.3 while the validation of the flight corridors and *sphere* waypoints are obtained through linear and nonlinear constraints. To do so, we use the compact representation of a piecewise B-spline trajectory introduced in this chapter. The algorithm then consists in a Nonlinear Programming (NLP) problem.

Confronted to simulations, the method responded to the cinematographic requirements, returning smooth and feasible trajectories with better speed profiles than the bi-level formulation of chapter 3. It was successfully applied on a real drone through an outdoor experiment. The output video is available at https://youtu.be/A0oYx268sis, it is convincing and the limited tracking error comforts the feasibility of the trajectory. These results were published in [107].

Finally, as the last contribution of this chapter, we proposed 4 methods to quickly obtain an initial feasible solution for the NLP problem, some of which are satisfying enough in terms of cinematographic criteria to constitute inexpensive (though not as good) alternatives to the resolution of the optimization problem in case the computation time would be critical.

The experience acquired on B-spline curves during this work allowed the author to make an intervention on Bézier and B-splines curves [104] in the massive open online course DroMOOC [11].

In the next chapter, we tackle the last remaining task to perform the aerial shot, the generation of the magnification trajectory and the rotation motion of the on-board camera.

Chapter 5

Control strategy

5.1 Introduction

Though the previous chapter presented several techniques to generate a position trajectory suited for quadrotor cinematography, it only partially dealt with the problem to solve. In addition to waypoints to validate, speed reference and flight corridors to respect, the flight plan reference defined in section 1.2.2 also contains camera angles and magnification behaviors to adopt between the waypoints. In this chapter, given a smooth trajectory, we propose to generate smooth camera references by the mean of a linear MPC law.

First, the angle and magnification references are detailed in section 5.2. Then, we propose and detail a linear under-sampled MPC controller to smooth the camera references in section 5.3, as they can be discontinuous when changing the camera behavior on a waypoint, for instance. Finally, in section 5.4 we propose two novel approaches to merge the yaw and thrust direction references of the drone into full 3D attitude references, and compare it to a widespread method through examples on a Bebop 2 drone and a Parrot ANAFI.

We start by giving the mathematical expression of the camera references corresponding to the camera behavior defined in section 1.2.2.

5.2 Camera references

In addition to its 3D position, a flying camera also has 4 DOF, its 3D angular position (attitude) and its magnification. Its attitude is expressed by the mean of ZYX Euler angles (see section 2.3.1), denoted by ψ_c (yaw), θ_c (pitch) and φ_c (roll), while the magnification is expressed by a factor m_c (with the default magnification given by $m_\mathrm{c} = 1$). While joining a waypoint, the camera can adopt a different behavior on each one of these quantities, defined in section 1.2.2. In this section, we detail the computation of the references associated to each camera behavior.

5.2.1 References

Each camera behavior covers one piece of trajectory. For the segment $i \in [\![1, N]\!]$ between the waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$, lasting over the periode $[t_{i-1}, t_i]$, the references at a time instant $t \in [t_{i-1}, t_i]$ are computed as follows.

Constant. For a *constant* reference we have

$$\eta(t) = \eta_i \tag{5.1}$$

with η designating either ψ_c, θ_c, φ_c or m_c and η_i a constant.

Ramp. The *ramp* reference is given by 2 references: a reference η_i to join when validating the waypoint $\mathbf{w}_i$ with a constant reference speed $\dot{\eta}_i$. We then have

$$\eta(t) = \eta_{i-1} + (\eta_i - \eta_{i-1}) \frac{t - t_{i-1}}{t_i - t_{i-1}} \tag{5.2}$$

with η designating either ψ_c, θ_c, φ_c or m_c and η_{i-1} the reference when validating the previous waypoint.

Point of Interest. The *POI* reference is only available for the yaw and pitch axes. It is given by a 3D reference position $\boldsymbol{\zeta}_\mathrm{POI} = \begin{pmatrix} x_\mathrm{POI} & y_\mathrm{POI} & z_\mathrm{POI} \end{pmatrix}^\top$ in the ground fixed frame (NED, see section 2.2) and a reference framing angle η_i. The reference is then computed using the evaluation of the drone trajectory $\boldsymbol{\zeta} = \begin{pmatrix} x & y & z \end{pmatrix}^\top$. For the yaw axis, this gives

$$\psi_\mathrm{c}(t) = \arctan_2\left(y_\mathrm{POI} - y(t), x_\mathrm{POI} - x(t)\right) + \psi_{\mathrm{c}i} \tag{5.3}$$

and for the pitch axis

$$\theta_\mathrm{c}(t) = -\arctan_2\left(z_\mathrm{POI} - z(t), \sqrt{\left(x_\mathrm{POI} - x(t)\right)^2 + \left(y_\mathrm{POI} - y(t)\right)^2}\right) + \theta_{\mathrm{c}i} \tag{5.4}$$

The singular cases for which the drone is at the vertical of the POI (for the yaw) or on the POI (for the pitch) are dealt with by taking the last well defined reference (in other words, the reference is not updated for singular cases).

Tangent. The *tangent* reference is only available for the yaw and pitch axes. The reference is given by the direction of the drone velocity $\dot{\boldsymbol{\zeta}} = \begin{pmatrix} \dot{x} & \dot{y} & \dot{z} \end{pmatrix}^\top$ and a reference framing angle η_i. For the yaw axis, this gives

$$\psi_\mathrm{c}(t) = \arctan_2\left(\dot{y}(t), \dot{x}(t)\right) + \psi_{\mathrm{c}i} \tag{5.5}$$

and for the pitch axis

$$\theta_\mathrm{c}(t) = -\arctan_2\left(\dot{z}(t), \sqrt{\dot{x}^2(t) + \dot{y}^2(t)}\right) + \theta_{\mathrm{c}i} \tag{5.6}$$

As for the *POI*, the references are not updated in presence of a singular case (null velocity).

Vertigo. The *vertigo* reference is only available for the magnification and should be used along with *POI* behaviors on the pitch and yaw axes. Given a 3D reference position $\boldsymbol{\zeta}_\mathrm{POI} = \begin{pmatrix} x_\mathrm{POI} & y_\mathrm{POI} & z_\mathrm{POI} \end{pmatrix}^\top$ in the ground fixed frame, the magnification varies such that the apparent diameter of an object at the position $\boldsymbol{\zeta}_\mathrm{POI}$ stays constant (i.e. so that it occupies a constant field of view in the output video). This can be achieved by setting

$$m_\mathrm{c}(t) = \frac{m_{\mathrm{c}i}}{d_i} \left\| \boldsymbol{\zeta}_\mathrm{POI} - \boldsymbol{\zeta}(t) \right\|_2 \tag{5.7}$$

with $m_{\mathrm{c}i}$ the magnification when validating the previous waypoint, $\mathbf{w}_i$, and

$$d_i = \left\| \boldsymbol{\zeta}_\mathrm{POI} - \boldsymbol{\zeta}(t_{i-1}) \right\|_2 \tag{5.8}$$

the distance between the drone and the POI also at the validation of $\mathbf{w}_i$.

Plane. The *plane* reference is only available for the roll and should typically be used along with *tangent* behaviors on the pitch and yaw axes. It simulates the view from a fixed-wind aircraft by rolling as the rotation speed on the yaw axis increases. One way to achieve this is to set

$$\varphi_{\mathrm{c}}(t) = \varphi_{\mathrm{c}_{\max}} \tanh\left(k_{\dot{\psi}_{\mathrm{c}}}\, \dot{\psi}_{\mathrm{c}}\right) \tag{5.9}$$

with $\varphi_{\mathrm{c}_{\max}}$ the maximum admissible roll reference and $k_{\dot{\psi}_{\mathrm{c}}}$ a tuning parameter and tanh the hyperbolic tangent function, used as a smooth saturation function.

Smooth. Finally, for the *smooth* transition between 2 camera behaviors, we use a cubic polynomial interpolation between the last reference of the $(i-1)$-th piece of trajectory, at the validation of $\mathbf{w}_{i-1}$, and the first reference of the $(i+1)$-th piece of trajectory, at the validation of $\mathbf{w}_i$

$$\begin{aligned}\eta(t) = \eta_{i-1} + \Delta t_i \dot{\eta}_{i-1}\, \tau(t) + (3\eta_i - 3\eta_{i-1} - \Delta t_i \dot{\eta}_i - 2\Delta t_i \dot{\eta}_{i-1})\, \tau_i^2(t) \\ + (2\eta_{i-1} - 2\eta_i + \Delta t_i \dot{\eta}_{i-1} + \Delta t_i \dot{\eta}_i)\, \tau_i^3(t)\end{aligned} \tag{5.10}$$

with η designating either ψ_{c}, θ_{c}, φ_{c} or m_{c}, $\Delta t_i = (t_i - t_{i-1})$ and $\tau_i(t) = \frac{t-t_i}{t_i - t_{i-1}}$.

5.2.2 Speed reference limitation

Since some camera behaviors use the duration of a piece of trajectory or the position or velocity of the drone, the latter must be low enough for the resulting camera reference to be both feasible and aesthetically satisfying. One way to deal with this could have been to add constraints in the trajectory generation algorithm to take into account the camera references. However their computation can be nonlinear in regard to the position and its time derivatives (*POI*, *smooth*, etc.) which can lead to complicate optimization problems for the trajectory generation. Instead, we have chosen to approximate the angle and magnification excursions on each piece of trajectory to compute a minimum duration under which the camera references could lead to infeasible or unsatisfying footage. For instance, if a 120° yaw panorama has to be performed on a piece of trajectory i, at a reference speed of $5°{\cdot}\mathrm{s}^{-1}$, the duration of this piece of trajectory should not be lower than $\Delta t_{\min i} = 24\,\mathrm{s}$.

These minimum duration are thus converted into maximum translational speeds

$$v_{i,\mathrm{cam}} = \frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{\Delta t_{\min i}} \tag{5.11}$$

used in the preprocessing of the flight plan before generating a position trajectory (section 3.2). It should be noticed that this method has some drawbacks. Firstly, it induces some conservatism and, secondly, it does not guarantee that the camera reference variations will not be locally large. However, this is a first approach approach to facilitate the generation of the camera motion, that can be improved later.

To achieve this, since the position trajectory is not yet generated when the estimated durations are computed, it is approximated by a broken line joining the waypoints. In other words, we suppose that the waypoint validations occur on the waypoints and that the velocity on the i-th piece is collinear to the vector $\mathbf{w}_i - \mathbf{w}_{i-1}$. Then, the camera reference at the beginning and at the end of each piece of trajectory can be approximated by using the expression given in the previous paragraph and their difference gives the

excursion on each camera axis, $\Delta\eta_i$ (with η designating either ψ_c, θ_c, φ_c or m_c). Given the slope references $\dot{\eta}_i$ for the *ramp* behaviors and a general bound for the time derivative of the reference on each axis, $\dot{\eta}_{\max}$, a minimum approximated duration of the piece can be computed

$$\Delta t_{\min i} = \frac{\Delta\eta_i}{\dot{\eta}_i} \tag{5.12}$$

with $\dot{\eta}_i = \dot{\eta}_{\max}$ for camera behaviors other than the *ramp* behavior. Then (5.11) is used to obtain a speed limitation for this piece of trajectory.

In this section, we detailed how the camera references are computed for each available camera behavior. In the next section, we propose one way to smoothly track these references by the mean of an MPC law.

5.3 Camera control

As described above, different types of camera behavior can be specified over each piece of trajectory. These can lead to discontinuous camera references, for example when validating a waypoint with a *POI* yaw behavior to join another waypoint with a *tangent* yaw behavior, or when passing above a POI. In this section we propose a way to smoothly track the considered camera references while keeping the framing error low.

5.3.1 Nominal model following control 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 behavior 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 behavior, 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. This means that the disturbance rejection dynamics should be stiff, as opposed to the reference tracking which should be smooth.

To achieve this, the Nominal Model Following Control (NMFC) architecture proposed in [67] is used in this work. This architecture consists in a 2-stage controller as illustrated in figure 5.1. Its robustness regarding disturbances and its performances in decoupling reference tracking and disturbances rejection dynamics are discussed in [111].

The principle of this architecture is the following

- A first stage, called *virtual controller*, has a virtual linear drone model track the heading reference according to the requirements for a good video quality in terms of smoothness, angular speed etc. This nominal model is referred to as the *virtual model*.

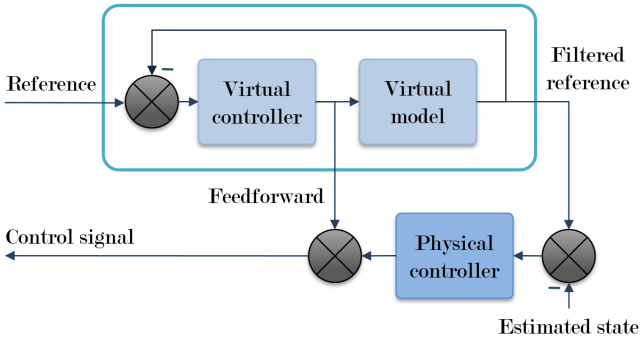

FIGURE 5.1 – NMFC architecture for the heading controller

- The output heading trajectory performed by the virtual drone is then tracked as a reference by the second stage, called the *physical controller*, acting directly on the real drone and using the real-time estimated state of the real drone.
- The control signals generated by the virtual controller are sent as a feedforward control input to the real drone. Thus, if the virtual model matched perfectly the real drone dynamics, without uncertainties and perturbations, the drone would actually be controlled in open-loop and the physical controller would not have to generate any control signal.

This architecture is a simple way to separate the reference tracking and the disturbance rejection dynamics.

Heading references can vary significantly from a piece of trajectory to another, depending on the chosen camera behavior. As a consequence, the heading reference over the entire mission can include steps, ramps or smooth parts. Due to its anticipating action, a MPC is used as virtual controller, in order to efficiently track this reference. This is illustrated on figure 5.2, a step reference is tracked both by a causal controller and a MPC. Both controller are volontarily slow in order to provide a smooth response, which lead to a large transient error (represented by the light red area) for the causal controller. On the contrary, thanks to its anticipative action, the MPC is able to reduce the tracking error while still offering a smooth and slow response [21].

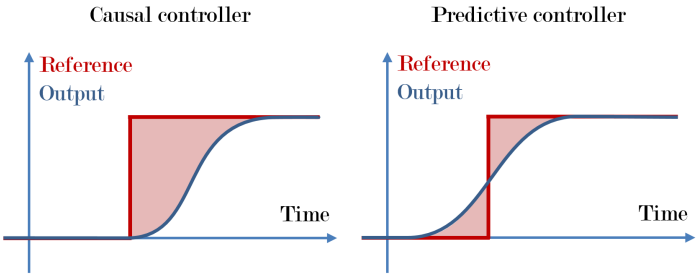

FIGURE 5.2 – Anticipative behavior of a predictive controller

Furthermore, this type of controller can explicitly take constraints into account, e.g. in order to limit the rotation speed, acceleration or jerk. The anticipative behavior of the MPC is obtained by the use of a preview on the future references and the prediction of the system response to a given series of future control inputs, obtained by propagating a model of the system. The latter is detailed in the next section.

5.3.2 Virtual model

The virtual heading dynamics is described by a continuous-time LTI system such as the one derived from the linearization of the drone dynamics near the hovering equilibrium in section 2.6. More specifically, we simplify the state space model (2.117) in order to only take into account the yaw axis of the drone

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{z}(t) &= \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \end{aligned} \tag{5.13}$$

with $\mathbf{x} \in \mathbb{R}^n$ the state vector, $\mathbf{u} \in \mathbb{R}^p$ the control vector and $\mathbf{z} \in \mathbb{R}^m$ the output vector. Here, we choose $\mathbf{u} = v_r$ (see section 2.6)

$$\mathbf{z} = \begin{pmatrix} \psi & r & \dot{r} \end{pmatrix}^\top \tag{5.14}$$

where ψ denotes the drone heading, $r \approx \dot{\psi}$ and $\dot{r}$ denote its angular velocity and acceleration around the yaw axis, respectively, and

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -c_r & 1 & \tau_r \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\varpi^2 & -2\xi\varpi \end{pmatrix} \\ \mathbf{B} &= \begin{pmatrix} 0 & 0 & 0 & \varpi^2 \end{pmatrix}^\top \\ \mathbf{C} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -c_r & 1 & \tau_r \end{pmatrix} \\ \mathbf{D} &= \mathbf{0}_{4\times 1} \end{aligned} \tag{5.15}$$

Given this model, we now detail the MPC tuning parameters and synthesis.

5.3.3 MPC controller

The virtual model (5.13) is discretized with a sampling period T_{MPC}. In the sequel, this equivalent discrete-time model is

$$\begin{aligned} \dot{\mathbf{x}}_{\text{MPC}}[k+1] &= \mathbf{A}_{\text{MPC}}\,\mathbf{x}_{\text{MPC}}[k] + \mathbf{B}_{\text{MPC}}\,\mathbf{u}_{\text{MPC}}[k] \\ \mathbf{z}_{\text{MPC}}[k] &= \mathbf{C}_{\text{MPC}}\,\mathbf{x}_{\text{MPC}}[k] + \mathbf{D}_{\text{MPC}}\,\mathbf{u}_{\text{MPC}}[k] \end{aligned} \tag{5.16}$$

This discretized model (5.16) is controlled by a MPC controller. 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 y, we denote by $\hat{y}[k+i|k]$ the prediction of the value of y at the step $k+i$, computed using its value at the step k. The MPC controller generates at each step a control signal minimizing

- The heading tracking error $\psi_{\text{ref}} - \psi$
- The angular speed r for the reasons mentioned in section 1.2.3 (aesthetic and motion blur)
- The angular acceleration $\dot{r}$ and the control input variations $\Delta\mathbf{u}_{\text{MPC}}$, strongly linked to the angular jerk, in order to obtain a smooth motion

with, at each step k, the control signal increment $\Delta\mathbf{u}_{\text{MPC}}[k] = \mathbf{u}_{\text{MPC}}[k] - \mathbf{u}_{\text{MPC}}[k-1]$. This leads to a classical quadratic cost function (as described in [21] and [73])

$$J[k] = \sum_{i=1}^{h_p} \left\| \mathbf{z}_{\text{MPC}}^{\text{ref}}[k+i] - \hat{\mathbf{z}}_{\text{MPC}}[k+i-1|k] \right\|_{\mathbf{Q}}^2 + \sum_{i=0}^{h_u-1} \|\mathbf{u}_{\text{MPC}}[k+i] - \mathbf{u}_{\text{MPC}}[k+i-1]\|_{\mathbf{R}}^2 \tag{5.17}$$

where h_p and h_u define the prediction and control horizons, respectively, and $\hat{\mathbf{z}}_{\text{MPC}}$ denotes the prediction of $\mathbf{z}_{\text{MPC}}$. The weighting terms are the diagonal matrix

$$\mathbf{Q} = \begin{pmatrix} \mu_\psi & & \\ & \mu_r & \\ & & \mu_{\dot{r}} \end{pmatrix} \tag{5.18}$$

and $\mathbf{R} = \mu_{\dot{r}}$, with μ_ψ, μ_r, $\mu_{\dot{r}}$ adjustable strictly positive weights on each controlled variable and $\mu_{\dot{r}}$ a strictly positive weight on the control signal variations.

An advantage of the MPC strategy is that it explicitly takes into account the constraints in the generation of the control signals. The control signal can thus be bounded between maximum admissible values

$$|\mathbf{u}_{\text{MPC}}| \leqslant u_{\max} \tag{5.19}$$

In order to comply with the feasibility proof of the position trajectory given in section 3.3, another constraint could be added on the angular speed, by the mean of a soft constraint as recommended in [73] for state constraints

$$\begin{aligned} &|r| \leqslant r_{\max} + \varepsilon_r \\ &\varepsilon_r \geqslant 0 \\ &\tilde{J} = J + \mu_\varepsilon \varepsilon_r \end{aligned} \tag{5.20}$$

with ε_r a slack variable and μ_ε its weight in the cost fuction. The use of a soft constraint means that the maximum angular speed can be exceeded and when it occurs, the MPC temporarily increases the priority of the reduction of the angular speed. Another, more conservative but less computation demanding way to achieve a similar goal is to bound the variations of the references

$$\widetilde{\psi}_{\text{ref}}[k] = \psi_{\text{ref}}[k-1] + \min\left\{r_{\max}, \max\left\{-r_{\max}, \psi_{\text{ref}}[k] - \psi_{\text{ref}}[k-1]\right\}\right\} \tag{5.21}$$

As for the soft constraint, this does not guarantee that the speed limitation will not be temporarily exceeded, but only that the reference to track does not exceed this limitation.

Remark 13 *Though the use of soft constraints would have constituted a more elegant solution, the very limited on-board computation power of the Bebop 2 prevented to deploy such a solution, at least in a reasonable amount of time.*

5.3.4 MPC undersampling

To be relevant, a MPC controller must be able to predict the reference and the behavior of the system over a time horizon consistent with the time response of the system and 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 the system and

reject disturbances. Typical orders of magnitude for this sampling period lie from 1 ms to 10 ms, which would result in large prediction horizons of hundreds to thousands of steps (such as in [70]). 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 produce 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 behavior, 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 $T_\mathrm{MPC} = K\,T_\mathrm{s}$, with $K \in \mathbb{N}$.

Figure 5.3 illustrates the block diagram of the proposed heading virtual MPC controller. The reference generation then works as follows

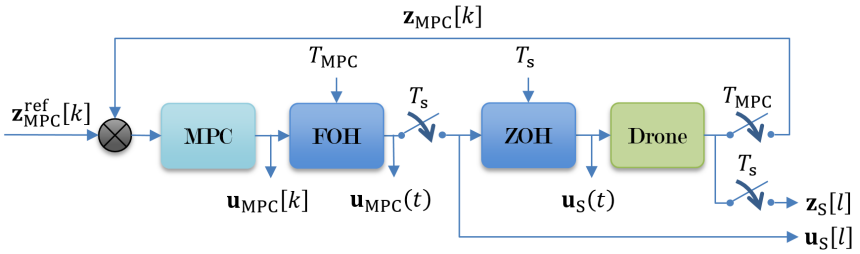

FIGURE 5.3 – Block diagram of the heading virtual controller

- The undersampled virtual model is controlled by the MPC controller, which generates an undersampled control signal $\mathbf{u}_\mathrm{MPC}[k]$
- 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 re-sampling at a higher frequency is presented on figure 5.4 with in red the

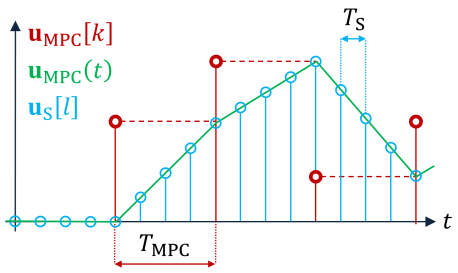

FIGURE 5.4 – Oversampling strategy for the virtual heading MPC controller

undersampled signal, in green the undersampled signal filtered by the causal FOH and in blue the oversampled signal. 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 $l\,T_{\mathrm{s}} = k\,T_{\mathrm{MPC}} + i\,T_{\mathrm{s}} = (k\,K + i)\,T_{\mathrm{s}}$, with $k \in \mathbb{N}$ and $i \in [\![0, K-1]\!]$, the following expression holds

$$\mathbf{u}_{\mathrm{s}}(t) = \mathbf{u}_{\mathrm{s}}[l] = \left(1 - \frac{i}{K}\right) \mathbf{u}_{\mathrm{MPC}}[k-1] + \frac{i}{K}\,\mathbf{u}_{\mathrm{MPC}}[k] \tag{5.22}$$

The continuous-time system is modeled by the LTI state-space representation (5.13). The ZOH discretization at the period T_{s} of this continuous-time system is given by

$$\begin{aligned} \mathbf{u}_{\mathrm{s}}[l+1] &= \mathbf{F}\,\mathbf{x}_{\mathrm{s}}[l] + \mathbf{G}\,\mathbf{u}_{\mathrm{s}}[l] \\ \mathbf{z}_{\mathrm{s}}[l] &= \mathbf{C}\,\mathbf{x}_{\mathrm{s}}[l] + \mathbf{D}\,\mathbf{u}_{\mathrm{s}}[l] \end{aligned} \tag{5.23}$$

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

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

$$\begin{aligned} \mathbf{x}_{\mathrm{s}}[(k+1)K] &= \mathbf{F}\,\mathbf{x}_{\mathrm{s}}[(k+1)K-1] + \mathbf{G}\,\mathbf{u}_{\mathrm{s}}[(k+1)K-1] \\ &= \mathbf{F}^K\,\mathbf{x}_{\mathrm{s}}[kK] + \sum_{i=0}^{K-1} \mathbf{F}^{K-1-i}\mathbf{G}\,\mathbf{u}_{\mathrm{s}}[kK+i] \end{aligned} \tag{5.24}$$

and using (5.22), it leads to

$$\begin{aligned} \mathbf{x}_{\mathrm{s}}[(k+1)K] = \mathbf{F}^K\,\mathbf{x}_{\mathrm{MPC}}[k] \\ + \left(\sum_{i=0}^{K-1} \left(1 - \frac{i}{K}\right)\mathbf{F}^{K-1-i}\mathbf{G}\right)\mathbf{u}_{\mathrm{MPC}}[k-1] \\ + \left(\sum_{i=0}^{K-1} \frac{i}{K}\,\mathbf{F}^{K-1-i}\mathbf{G}\right)\mathbf{u}_{\mathrm{MPC}}[k] \end{aligned} \tag{5.25}$$

This gives the following expression for the undersampled model

$$\begin{aligned} \mathbf{x}_{\mathrm{MPC}}[k+1] &= \mathbf{A}_{\mathrm{MPC}}\,\mathbf{x}_{\mathrm{MPC}}[k] + \mathbf{B}_{\mathrm{MPC}}\,\mathbf{u}_{\mathrm{MPC}}[k] \\ \mathbf{z}_{\mathrm{MPC}}[k] &= \mathbf{C}_{\mathrm{MPC}}\,\mathbf{x}_{\mathrm{MPC}}[k] \end{aligned} \tag{5.26}$$

with

$$\begin{aligned}
\mathbf{x}_{\text{MPC}}[k] &= \begin{pmatrix} \mathbf{x}_{\text{MPC}}[k] \\ \mathbf{u}_{\text{MPC}}[k-1] \end{pmatrix} \\
\mathbf{A}_{\text{MPC}} &= \begin{pmatrix} \mathbf{F}^K & \sum_{i=0}^{K-1} \left(1 - \frac{i}{K}\right) \mathbf{F}^{K-1-i}\mathbf{G} \\ \mathbf{0}_{p\times n} & \mathbf{0}_{p\times p} \end{pmatrix} \\
\mathbf{B}_{\text{MPC}} &= \begin{pmatrix} \sum_{i=0}^{K-1} \frac{i}{K}\, \mathbf{F}^{K-1-i}\mathbf{G} \\ \mathbf{I}_p \end{pmatrix} \\
\mathbf{C}_{\text{MPC}} &= \begin{pmatrix} \mathbf{C} & \mathbf{D} \end{pmatrix}
\end{aligned} \tag{5.27}$$

Since the inputs and outputs of these two models are the same, the expression of the cost function (5.17) is unchanged (but with the model (5.26) used for the prediction).

The weights of the cost function (5.17) remain to be adjusted, which is the topic of the next section.

5.3.5 PSO-based MPC tuning

The model (5.26) is used for the MPC controller synthesis, which requires to tune 6 strictly positive parameters: h_p, h_u μ_ψ, μ_r, $\mu_{\dot{r}}$ and $\mu_{\Delta u}$. The sampling period for the MPC controller is considered $T_{\text{MPC}} = 0.1\,\text{s}$. A time response around 1 s 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, for a given scenario, an ideal response of the oversampled model is defined, as well as a template around this ideal response. This ideal response is obtained using a linear, non causal low-pass filter. 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, overshoot or undershoot, show a non-minimum phase behavior or oscillate

$$J_{\text{tuning}} = \sum_{l=0}^{N_J} \left(\psi_{\text{ideal}}[l] - \psi[l]\right)^2 + \sum_{i=1}^{3} \varepsilon_i \tag{5.28}$$

with

$$\begin{cases}
\varepsilon_1 = \sum_{l=0}^{N_J} \mu_{\text{template}}\ \max\left\{\psi[l] - \psi_{\text{template,max}}[l], \psi_{\text{template,min}}[l] - \psi[l], 0\right\} \\
\varepsilon_2 = \sum_{l=0}^{N_J} \mu_{\text{overshoot}}\ \max\left\{\psi[l] - \psi_{\max}[l], 0\right\} \\
\varepsilon_3 = \sum_{l=0}^{N_J} \mu_{\text{undershoot}}\ \min\left\{\psi[l] - \psi_{\min}[l], 0\right\}
\end{cases} \tag{5.29}$$

with N_J the number of steps in the scenario, ψ_{ideal} the ideal response and μ_{template}, $\mu_{\text{overshoot}}$ and $\mu_{\text{undershoot}}$ strictly positive weights. The weights μ_{template}, $\mu_{\text{overshoot}}$ and

$\mu_{\text{undershoot}}$ do not require any fine tuning. The weight $\mu_{\text{undershoot}}$ must be large as a non-minimum phase behavior of the closed loop seriously impacts the quality of the video. The template penalty μ_{template} should come after, and finally the overshoot penalty $\mu_{\text{overshoot}}$, that can be set null since the template already penalizes a too large overshoot.

The task is then repeated for different values of h_p and h_u and the settings producing the lowest cost function value are chosen. This procedure is automatized for more convenience.

Simulation of the undersampled MPC tuned with the parameters obtained with this PSO tuning are shown on figure 5.5. For a ramp reference (red), the MPC generates an undersampled control signal (blue). This control signal is re-sampled 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). On the one hand, thanks to its large time horizon, the MPC is able to anticipate the reference and to confer the closed-loop system a slow but accurate response. On the other hand, the FOH strategy succeeds in smoothing the under sampled control signal of the MPC and the response of the oversampled virtual meets the smoothness requirements of section 1.2.3.

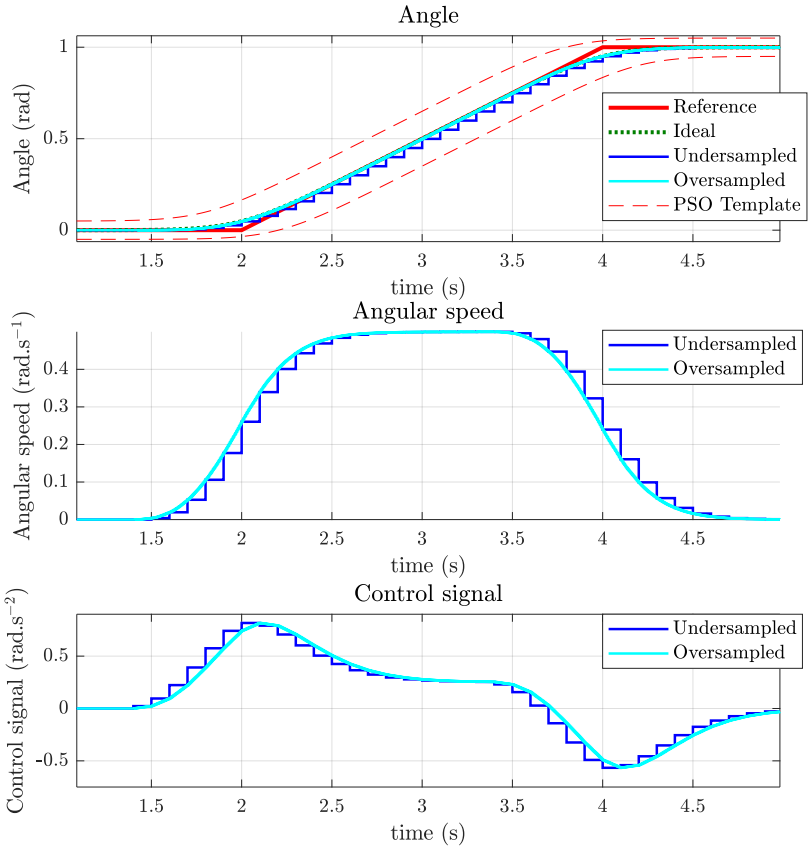

FIGURE 5.5 – Heading ramp response with the optimal MPC settings

The elevation, roll and magnification of the camera do not have any dynamics since they are given by the virtual gimbal of the Bebop 2. As a consequence, their references are filtered by a non-causal low pass filter, which does not induce any phase distortion. As with the heading MPC, the non causal filter for the other DOF of the camera are undersampled and filtered by a FOH in order to avoid hundreds of evaluations of the references at each step, while still keeping a large enough horizon for the filter to efficiently

smooth the references. The smoothed pitch, roll and magnification references obtained as outputs of these non-causal filters are sent to the digital stabilization of the Bebop 2.

As explained in section 5.3.1 the control signal generated by the heading MPC is sent to a virtual, simulated drone. The output of this virtual model is sent as a smoothed heading reference both to the virtual gimbal and to the real drone. The heading reference of the drone is thus imposed, but its reference attitude still has 2 DOF left. In the next section, we use the thrust reference of the drone for tracking the position trajectory to set these 2 DOF and reconstruct the full, 3D attitude reference to be tracked by the attitude controller of the quadrotor.

5.4 Drone full attitude reference

In addition to the heading reference, the drone also needs to track the position trajectory by adjusting its thrust and pitch/roll angles. At each time instant, a position controller is thus in charge of computing a 3D thrust reference $\mathbf{f}_{\text{ref}}$ in the ground frame NED for this task

$$\mathbf{f}_{\text{ref}} = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \tag{5.30}$$

Since, in a first approximation, the drone can only generate a thrust along its vertical vertical axis (2.48), the drone have to make its vertical axis $\mathbf{z}_{\mathcal{D}}$ coincide with the desired thrust direction $\mathbf{f}_{\text{ref}}/\|\mathbf{f}_{\text{ref}}\|_2$. As a consequence, if the heading reference sets 1 dimension of the drone attitude, the direction of the thrust $\mathbf{f}_{\text{ref}}$ requested by the position controller dictates the 2 others. This section focuses on a way to merge the thrust vector and heading references to produce 3D attitude references to be followed by the (physical) attitude controller.

5.4.1 Angle fusion for the Parrot Bebop 2

Given a yaw reference ψ (set as the output of the virtual model (5.23) in our work) we define the vector

$$\mathbf{x}_{\text{horiz}} = \begin{pmatrix} c_\psi \\ s_\psi \\ 0 \end{pmatrix}_{\mathcal{W}} \tag{5.31}$$

in the ground horizontal plan, defining the heading direction we desire to achieve. Given such a vector, it is suggested in [66] to reconstruct the reference drone axes $(\mathbf{x}_{\text{ref}}, \mathbf{y}_{\text{ref}}, \mathbf{z}_{\text{ref}})$ by the following method.

- First, the reference vertical axis $\mathbf{z}_{\text{ref}}$ of the drone is given by the reference thrust vector
$$\mathbf{z}_{\text{ref}} = -\frac{\mathbf{f}_{\text{ref}}}{\|\mathbf{f}_{\text{ref}}\|_2} \tag{5.32}$$
- Then, the y-axis $\mathbf{y}_{\text{ref}}$ can be set as a vector normal both to $\mathbf{z}_{\text{ref}}$ and the projection of $\mathbf{x}_{\text{horiz}}$ on the plan normal to $\mathbf{z}_{\text{ref}}$, i.e.
$$\mathbf{y}_{\text{ref}} = \frac{\mathbf{z}_{\text{ref}} \times \mathbf{x}_{\text{horiz}}}{\|\mathbf{z}_{\text{ref}} \times \mathbf{x}_{\text{horiz}}\|_2} \tag{5.33}$$

- Finally, the x-axis $\mathbf{x}_{\text{ref}}$ is set to complete the vector basis

$$\mathbf{x}_{\text{ref}} = \mathbf{y}_{\text{ref}} \times \mathbf{z}_{\text{ref}} \tag{5.34}$$

With this solution, the heading of the reconstructed 3D attitude (angle of the z-axis rotation of the ZYX Euler angle, see section 2.3.1) does not exactly correspond to the yaw reference in the general case. This is illustrated on figure 5.6 for which the desired heading is null ($\mathbf{x}_{\text{horiz}} = \mathbf{x}_{\mathcal{W}}$) and the desired thrust direction is $\mathbf{f}_{\text{ref}}/\|\mathbf{f}_{\text{ref}}\| = (-0.3536, 0.3536, -0.8660)_{\mathcal{W}}$.

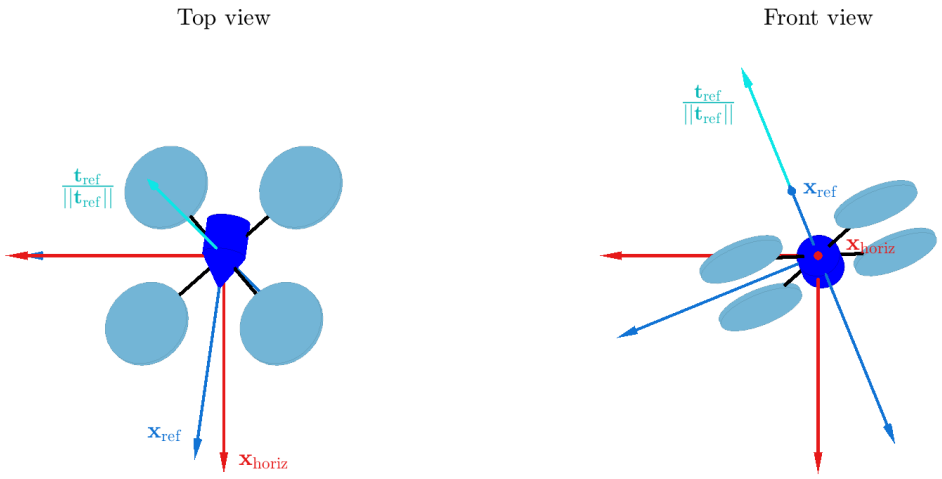

FIGURE 5.6 – Heading mismatch when using the projection method for reconstructing the 3D attitude reference

Instead, this heading actually is the one minimizing the angle between the vectors $\mathbf{x}_{\text{horiz}}$ and $\mathbf{x}_{\text{ref}}$. This is not an issue with the Bebop 2 as the FOV of the front camera allows the virtual gimbal to correct this. On the contrary, this is even an advantage as, by minimizing the angle between $\mathbf{x}_{\text{horiz}}$ and $\mathbf{x}_{\text{ref}}$, there are more chances that $\mathbf{x}_{\text{horiz}}$ lies within the FOV of the fixed camera, which is a cone of axis $\mathbf{x}_{\text{ref}}$. This method is thus suited for the Bebop 2. This last aspect will be detailed in section 5.4.3.

Nevertheless, as it has been mentioned in chapter 4, Parrot released a new quadrotor during this thesis, the Parrot ANAFI. Though this work was originally intended for the Bebop 2, we desired to extend it to the ANAFI. The latter possesses a 2-axis mechanical gimbal (roll, pitch), with a digital stabilization of the yaw axis. Contrary to the Bebop 2, the mechanical axes of the gimbal confer a much larger excursion to the y-axis of the camera for the ANAFI, but as a counter part, the digital stabilization is much more limited in range on the yaw axis, and it is critical that the heading of the drone coincides with the reference camera heading. As a consequence, the angle fusion suggested in [66] and exposed above is no longer suited. For a given yaw reference, we rather seek to have the x-axis of the drone lying in the vertical plane defined by the heading reference, $\mathbf{x}_{\text{ref}} \in (G, \mathbf{x}_{\text{horiz}}, \mathbf{z}_{\mathcal{W}})$, or in other terms, we seek to set $\mathbf{x}_{\text{ref}}$ such that its projection on the horizontal plan $(G, \mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}})$ is collinear to $\mathbf{x}_{\text{horiz}}$. In order to achieve this, we now propose a new way to merge the thrust direction and yaw references into a 3D attitude reference, suited for the Parrot ANAFI quadrotor.

5.4.2 Angle fusion for the Parrot ANAFI

As for the previous method, we set the reference z-axis of the drone as $\mathbf{z}_{\text{ref}} = -\frac{f_z}{\|\mathbf{f}_{\text{ref}}\|_2}$. This reference $\mathbf{z}_{\text{ref}}$ can be parameterized in the ground fixed frame NED by the ground angle $\alpha \in [0, \pi]$ and the angle $\beta \in [-\pi, \pi]$ such that

$$\mathbf{z}_{\text{ref}} = \begin{pmatrix} s_\beta s_\alpha \\ -c_\beta s_\alpha \\ c_\alpha \end{pmatrix}_{\mathcal{W}} \tag{5.35}$$

Conversely, these angles can be reconstructed from a given vector $\mathbf{z}_{\text{ref}}$ as follows

$$\begin{aligned} \alpha &= \arccos\left(\mathbf{z}_{\text{ref}}^\top \mathbf{z}_{\mathcal{W}}\right) \\ \beta &= \begin{cases} 0 & \text{if } \alpha = 0 \\ \arctan_2\left(\mathbf{z}_{\text{ref}}^\top \mathbf{x}_{\mathcal{W}}, \mathbf{z}_{\text{ref}}^\top \mathbf{y}_{\mathcal{W}}\right) & \text{otherwise} \end{cases} \end{aligned} \tag{5.36}$$

This is illustrated on figure 5.7, where the ground vertical axis $\mathbf{z}_{\mathcal{W}}$ is rotated of an angle α around the vector $c_\beta \mathbf{x}_{\mathcal{W}} + s_\beta \mathbf{y}_{\mathcal{W}}$, in the horizontal plan (in dashed grey on the figure).

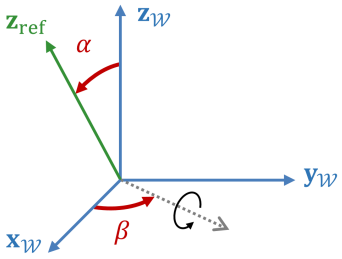

FIGURE 5.7 – Parameterization of the vertical axis of the drone with the angles α and β

Setting $\mathbf{z}_{\text{ref}} = -\frac{f_z}{\|\mathbf{f}_{\text{ref}}\|_2}$, we can compute the parameters α and β from the desired thrust direction

$$\begin{aligned} \alpha &= \pi - \arccos\left(\frac{f_z}{\|\mathbf{f}_{\text{ref}}\|_2}\right) \\ \beta &= \begin{cases} 0 & \text{if } \alpha = 0 \\ \arctan_2\left(f_x, f_y\right) + \pi & \text{otherwise} \end{cases} \end{aligned} \tag{5.37}$$

We suppose that $\alpha < \pi/2$. We can now look for the vector $\mathbf{x}_{\text{ref}}$ lying in the plan $(G, \mathbf{x}_{\text{horiz}}, \mathbf{z}_{\mathcal{W}})$ and normal to $\mathbf{z}_{\text{ref}}$.

Since it is in the plan $(G, \mathbf{x}_{\text{horiz}}, \mathbf{z}_{\mathcal{W}})$, $\mathbf{x}_{\text{ref}}$ can be expressed in the following manner

$$\mathbf{x}_{\text{ref}} = \mathbf{R}_\psi \mathbf{R}_\gamma \mathbf{x}_{\mathcal{W}} \tag{5.38}$$

where

$$\begin{aligned} \mathbf{R}_\psi &= \begin{pmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{R}_\gamma &= \begin{pmatrix} c_\gamma & 0 & s_\gamma \\ 0 & 1 & 0 \\ -s_\gamma & 0 & c_\gamma \end{pmatrix} \end{aligned} \tag{5.39}$$

with $\gamma \in [-\pi, \pi]$ represented on figure 5.8.

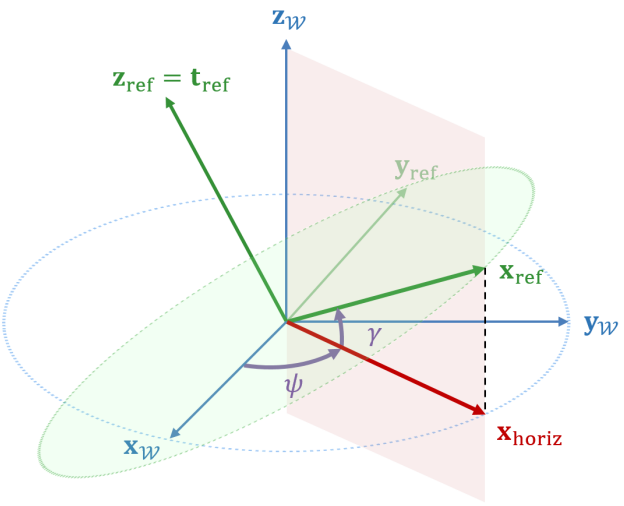

FIGURE 5.8 – Definition of the γ angle

Replacing $\mathbf{R}_\psi$ and $\mathbf{R}_\gamma$ by there expressions (5.39) in (5.38) leads to

$$\mathbf{x}_{\text{ref}} = \begin{pmatrix} c_\psi c_\gamma \\ s_\psi c_\gamma \\ -s_\gamma \end{pmatrix}_{\mathcal{W}} \tag{5.40}$$

Since $\mathbf{x}_{\text{ref}}$ is normal to $\mathbf{z}_{\text{ref}}$, we can write

$$\mathbf{x}_{\text{ref}}^\top \mathbf{z}_{\mathcal{W}} = 0 \tag{5.41}$$

By injecting (5.35) and (5.40) in this expression, we get

$$c_\gamma s_\alpha \ (s_\beta c_\psi - c_\beta s_\psi) - s_\gamma c_\alpha = 0 \tag{5.42}$$

and

$$\gamma = \arctan_2 \left(s_\alpha \left(c_\psi s_\beta - s_\psi c_\beta \right), c_\alpha \right) \tag{5.43}$$

which gives $\mathbf{x}_{\text{ref}}$ by injecting this result into (5.40). Finally, we complete the basis by setting

$$\mathbf{y}_{\text{ref}} = \mathbf{z}_{\text{ref}} \times \mathbf{x}_{\text{ref}} \tag{5.44}$$

We thus obtain the full reference attitude

$$\mathbf{R}_{\text{ref}} = \begin{pmatrix} \mathbf{x}_{\text{ref}} & \mathbf{y}_{\text{ref}} & \mathbf{z}_{\text{ref}} \end{pmatrix} \tag{5.45}$$

The results of this method applied to the same heading and thrust direction references as for the figure 5.6 are illustrated on figure 5.9.

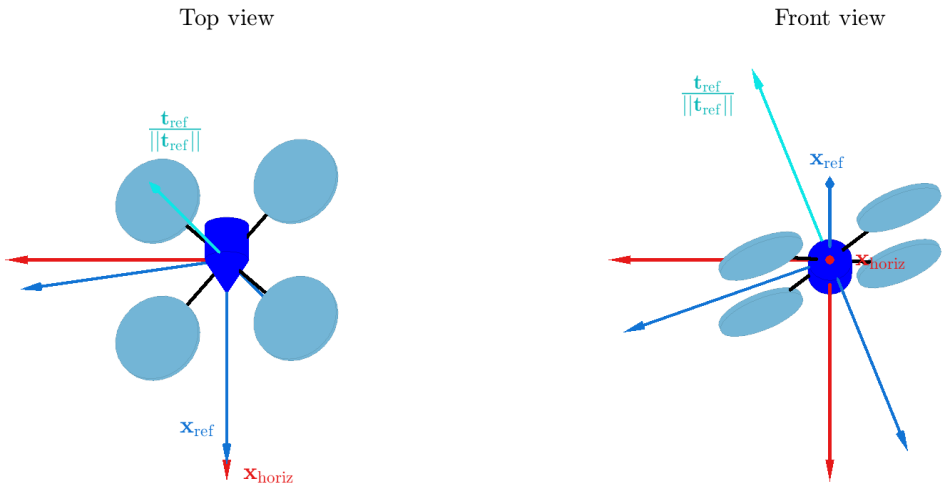

FIGURE 5.9 – Reconstructed 3D attitude reference proposed in this work

The heading extracted from the reconstructed 3D attitude reference now matches the yaw reference ψ.

In order to better understand the differences between the two methods, we compare their result on an example in the following section.

5.4.3 Comparison of the two methods

Figure 5.10 gives a simplified representation of the sets of achievable recording directions, relatively to the drone. For a camera direction $\mathbf{x}_{\text{cam}}$, if $\mathbf{x}_{\text{cam}}$ lies within this feasible set, the direction pointed by $\mathbf{x}_{\text{cam}}$ appears at the center of the video. The framing is correct. Conversely, if $\mathbf{x}_{\text{cam}}$ does not lie within the sets represented on figure 5.10, the camera has reached its maximum achievable excursion and the direction pointed by $\mathbf{x}_{\text{cam}}$ does not appear at the center of the video. The framing is then wrong. In the worst cases, the direction pointed by $\mathbf{x}_{\text{cam}}$ does not appear at all on the video and the target is out of the field of view.

FIGURE 5.10 – Simplified representations of the feasible directions of recording of the Bebop 2 and the ANAFI

These achievable sets are given by the field of view of the front fish-eye camera for the Bebop 2 which is fixed relatively to the drone. For the ANAFI, the roll axis of the mechanical gimbal first cancels the roll of the drone and then the pitch axis of this gimbal tilts the camera to the desired pitch. Consequently, at the difference of the Bebop 2, the

feasible sets of the ANAFI does not rotate with the drone when the drone rolls (as long as the roll axis of the mechanical gimbal is able to cancel the drone roll). The feasible set still pitches with the drone though. Finally, a small digital margin on the yaw axis of the camera is given by the digital stabilization.

We now illustrates on two examples how the choice of the method for reconstructing the reference attitude interacts with these achievable camera direction sets. We start with an example with the Bebop 2.

Bebop 2. The first example is illustrated on figure 5.11, with a Bebop 2 drone. The vector $\mathbf{x}_{\text{horiz}}$ is represented in red, the vector $\mathbf{x}_{\text{ref}}$ in blue and the direction of recording, $\mathbf{x}_{\text{cam}}$, in green. On the left, the method of section 5.4.1 is used for reconstructing the attitude reference, while the method of section 5.4.2 is used on the right. On the right, the heading of the drone coincides with the one given by $\mathbf{x}_{\text{horiz}}$, as explained in section 5.4.2, but this leads to a larger angle between the vectors $\mathbf{x}_{\text{ref}}$ and $\mathbf{x}_{\text{cam}}$, sending $\mathbf{x}_{\text{cam}}$ outside of the feasible set. On the contrary, on the left, using the method of section 5.4.1 leads to a smaller angle between the vectors $\mathbf{x}_{\text{ref}}$ and $\mathbf{x}_{\text{cam}}$, and $\mathbf{x}_{\text{cam}}$ is now reachable.

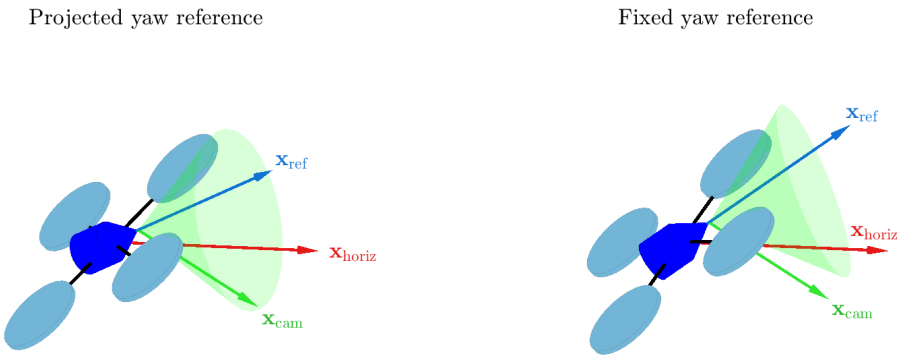

FIGURE 5.11 – Comparison of the two attitude reconstruction methods on the Bebop 2

In addition, we can propose another, new way to reconstruct the 3D attitude reference that minimizes the angle between the vectors $\mathbf{x}_{\text{ref}}$ and $\mathbf{x}_{\text{cam}}$ by replacing (5.33) by

$$\mathbf{y}_{\text{ref}} = \frac{\mathbf{z}_{\text{ref}} \times \mathbf{x}_{\text{cam}}}{\|\mathbf{z}_{\text{ref}} \times \mathbf{x}_{\text{cam}}\|_2} \tag{5.46}$$

i.e. by setting $\mathbf{x}_{\text{ref}}$ as the projection of $\mathbf{x}_{\text{cam}}$ on the plan normal to $\mathbf{z}_{\text{ref}}$, rather than the projection of $\mathbf{x}_{\text{horiz}}$. The results of this method are presented on figure 5.12, with the original method used on the left and the proposed alternative on the right. On the right, the camera direction lies within the feasible set. On the contrary, with the original method (on the left) it is inside of the admissible FOV.

Though this method does indeed reduce the angle between the vectors $\mathbf{x}_{\text{ref}}$ and $\mathbf{x}_{\text{cam}}$ compared to the method of section 5.4.1, and thus helps maintaining $\mathbf{x}_{\text{cam}}$ in the feasible set for any reference thrust direction, it leads to a significant deviation from the original yaw reference. As a consequence, the feedforward generated by the undersampled MPC of section 5.3 might not be consistent with the reconstructed heading trajectory. For this reason, we abandoned this approach for this work and decided to keep the original method proposed by [66]. However, in a different context, it could be an advantageous approach for maximizing the reachable camera directions set.

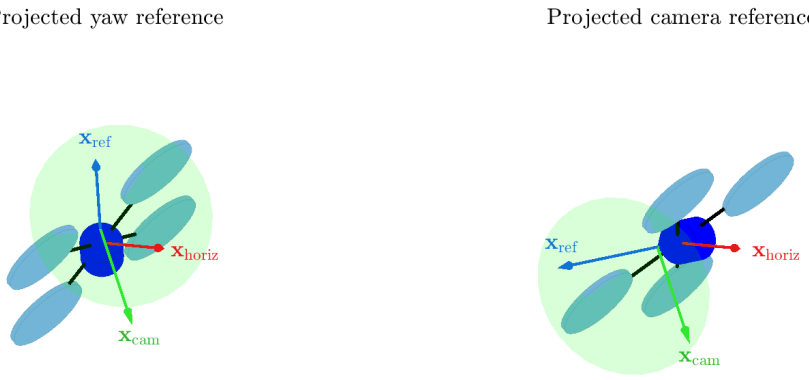

FIGURE 5.12 – Alternative attitude reconstruction methods on the Bebop 2

ANAFI. The second example is illustrated on figure 5.13, with the ANAFI. The vector $\mathbf{x}_{\text{horiz}}$ is represented in red, the vector $\mathbf{x}_{\text{ref}}$ in blue and the direction of recording, $\mathbf{x}_{\text{cam}}$, in green. On the left, the method of section 5.4.1 is used for reconstructing the attitude reference, while the method of section 5.4.2 is used on the right. On the left, the heading deviation induced by the reconstruction of the attitude makes $\mathbf{x}_{\text{cam}}$ exiting the feasible set. On the contrary, on the right, using the method of section 5.4.2 increases the pitch required by the gimbal to reach the desired camera direction but the resulting attitude reference is free of heading deviation, which brings $\mathbf{x}_{\text{cam}}$ back into the feasible set.

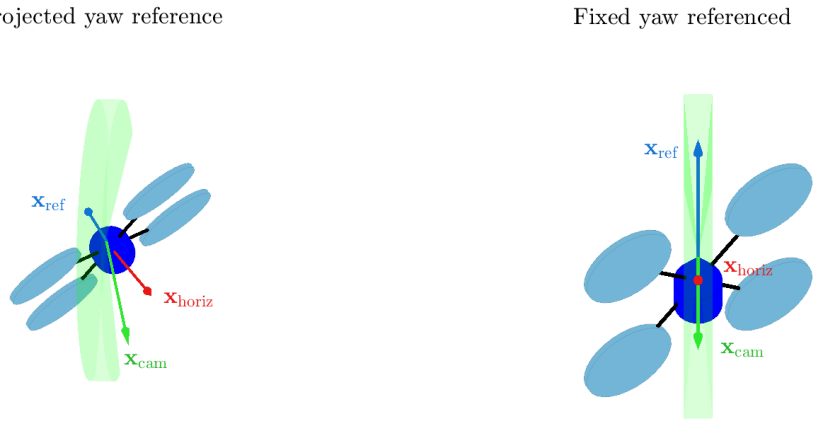

FIGURE 5.13 – Comparison of the two attitude reconstruction methods on the ANAFI

Remark 14 *In practice, the FOV of the Bebop 2 is large enough to prevent the target to completely exit it (supposing that the reconstructed attitude reference is accurately tracked). However, this issue is still significant enough in practice to prevent a correct framing, i.e. to have the desired camera direction centered in the video on the Bebop 2 if the method 2 proposed in section 5.4.2 is used for the reference attitude reconstruction. The same goes for the Parrot ANAFI, although the issue is more significant than on the Bebop 2 as the stabilization margins on the yaw axis are severe on this drone. Hence, using the method of section 5.4.1 for ANAFI can lead to significant framing errors, while they are non-existent when using the method proposed in section 5.4.2.*

5.5 Conclusion

In this chapter, we tackled the last remaining task for performing an autonomous aerial footage: the generation of the magnification trajectory and the rotation motion of the on-board camera.

To achieve this, as a first contribution, we propose to use a Nominal Model Following Control (NMFC) scheme to separate the reference tracking and the disturbance rejection dynamics. In order to have a smooth and slow reference tracking without inducing too much tracking error, we suggested to use a linear Model Predictive Control (MPC) controller as virtual controller of the NMFC. Since a large time horizon is required to obtain a smooth response, we propose as a second contribution to undersample the MPC controller and to filter the control signals it outputs by a FOH filter. The filtered control signal are then re-sampled at the higher drone frequency by a classical Zero-Order Hold (ZOH) filter. This way, the undersampling of the MPC does not induce low frequency discontinuities in the control signals, which would have degraded the smoothness of the system response. The performances of this control scheme were convincing when confronted to simulation we were able to deploy it on-board, during an outdoor flight experiment. These results were part of the publication [105].

Finally, we study the impact of the method to reconstruct the full attitude reference of the drone from the desired heading of the camera output by the NMFC controller and the thrust reference output by the position controller of the drone. Though the classical method used in the literature for this task appears suited for the Bebop 2, it is not viable for the newer Parrot drone, the ANAFI, due to its different image stabilization mechanism. To palliate this issue, we proposed, as a last contribution, a new method to merge the heading and thrust direction references into a full attitude reference which fixes the drone heading equal to the camera heading.

Chapter 6

Conclusion

6.1 Summary

Through this PhD thesis, we developed a strategy to compute a 7D cinematographic quadrotor trajectory from a generic high level reference. The proposed method is capable to deal with various types of 3D waypoints to join, along with a wide variety of camera pan, tilt, roll and magnification behaviors. The computation of the trajectory is split into, first, the generation of a position trajectory, and, second, the generation of the camera rotation motion and magnification trajectory. In both steps, aesthetic criteria are taken into account to produce an overall smooth and natural 7D trajectory suited for cinematography.

In chapter 1, after a brief introduction to the domain of aerial cinematography in section 1.1, we provided a complete specification of the goal to achieve in this work, in section 1.2.

Then, in order to propose relevant solutions for the generation of the 7D quadrotor trajectory, we conducted a deep investigation on the quadrotor model in chapter 2. Our work differs and contributes to the literature in the use of screw theory for building the model, which allowed us to perform a rigorous study of the gyroscopic and reaction torques of the propellers (appendix B). Along with the construction of the model, we provided a state-of-the-art on the modeling of quadrotors, including the different levels of modeling (section 2.1), the dynamics of the quadrotor body and the representation of its attitude (section 2.3), and the modeling of the different involved mechanical actions (section 2.4). Since the tools of rigid body mechanics do not directly allow us to build a model of the entire quadrotor, we developed the model of the drone body in order to get an equivalent nonlinear model of the entire quadrotor. In a second part (section 2.6), we linearized this model around the hovering equilibrium. This allowed us to emphasize the role of the propellers inertia when tilting the propellers, as on industrial quadrotors. In particular, we showed that this tilt resulted in the apparition of coupling terms between the 3 translation axes and 3 rotation axes of the quadrotor, as well as the apparition of a non-minimum phase behavior of the rotation dynamics of the drone. This result constitutes another contribution of this chapter and was published in [105]. Finally, we proposed a new Linear Time-Invariant (LTI) state-space model of the quadrotor near the hovering equilibrium for a generic configuration of the propellers, taking into account the impact of the inertia of the propellers as well as the dynamics of the Brushless Direct Current (BLDC) motors, controlled in closed-loop by the Electronic Speed Control (ESC).

We then tackled the generation of the position trajectory in chapter 3. As a first con-

tribution, based on the nonlinear model obtained in chapter 2, we derived a new set of constraints on the time-derivatives of the position ensuring the feasibility of the trajectory, suited for cinematography, in section 3.3. Using these constraints, we improved and extended the existing bi-level snap optimization procedure of piecewise polynomial trajectories to the case of cinematography in section 3.5 and section 3.6. More specifically, we proposed to minimize both the duration of the mission and the jerk of the trajectory in order to obtain a smooth motion respecting the speed references set by the user. As another contribution, we were able to include the drag of the quadrotor as well as the wind in the bi-level optimization method in section 3.7. We also studied in section 3.7 the impact of the initialization of the bi-level optimization procedure and the addition of soft constraints in the optimization of the polynomial coefficients for improving the speed profile of the trajectory. The strategy proposed in this chapter was successfully confronted to simulations and returned smooth and visually satisfying trajectories. To finish, we were able to confront the method to an outdoor flight experiment on a Parrot Bebop 2 quadrotor, with convincing results https://www.youtube.com/watch?v=IR9Ablo-ryI. These contributions were published in [106].

Nevertheless, after identifying several limitations of the bi-level optimization strategy, we proposed in chapter 4 a new, alternative method to generate the position trajectory, based on the use of non-uniform B-spline curves. The choice to use B-spline curves to parameterize the trajectory is motivated by their property to be easily bounded in convex regions. This allows us to conveniently guarantee the respect of the flight corridors and feasibility constraints over the entire trajectory, by constraining the control points of the position trajectory and its time-derivatives inside feasible convex regions. Starting with an overview and preliminary results on non-uniform B-spline curves in section 4.2, we introduced in section 4.3 a novel, compact way to represent a piecewise clamped B-spline that implicitly guarantees the validation criteria of the waypoints and the continuity of the first derivatives of the position. This compact representation constitutes the first contribution of this chapter and allows us to formulate the trajectory generation problem as Nonlinear Programming (NLP) problem in section 4.4. This NLP formulation represents the second contribution of this chapter, as we propose a new approach to generate a minimum-time B-spline trajectory, while other metrics than the duration are usually used in the literature for B-spline curves. The method successfully met the cinematographic requirements when confronted to simulations in section 4.5 and, as another contribution, we applied it in the context of an outdoor flight experiment on a Parrot ANAFI quadrotor, with convincing results https://youtu.be/A0oYx268sis. These original results were published in [107]. Finally, as a last contribution, we proposed 4 methods to generate a feasible B-spline trajectory for the initialization of the NLP problem in section 4.6.

Finally, chapter 5 dealt with the generation of a smooth camera rotation motion and magnification trajectory. After computing the camera references for each camera behavior defined in the specifications of section 1.2, we proposed as a first contribution to use a Nominal Model Following Control (NMFC) scheme to smoothly track the camera references in section 5.3, as it is convenient to separate the reference tracking and disturbance rejection dynamics. In order to obtain a slow but accurate tracking of the references, we use the anticipative behavior of a Model Predictive Control (MPC) controller in the NMFC scheme. Since the slow desired time response of the closed-loop system can lead to a large time horizon of the MPC, we propose to undersample the controller and to interpolate its control signals by a First-Order Hold (FOH) filter, before re-sampling them at high frequency and to send them to an oversampled, nominal model of the drone. In order to explicitly take this affine interpolation into account in the design of the MPC controller, we include the

action of the FOH filtering, the high frequency re-sampling and the following Zero-Order Hold (ZOH) filtering of the control signals generated by the MPC into the model used for the state prediction. Since the tuning of the MPC controller can be delicate, we proposed a method to choose the weights of its cost function based on a Particle Swarm Optimization (PSO) algorithm. The behavior of the undersampled MPC law was validated in simulation and tested on-board, during an outdoor flight experiment on a Parrot Bebop 2 quadrotor https://www.youtube.com/watch?v=IR9Ablo-ryI. These new results were published in [106]. Finally, in section 5.4, we studied the reconstruction of the 3D attitude reference of the drone from a thrust reference and the yaw reference of the camera. As a last contribution, we proposed 2 new techniques to reconstruct this attitude reference and we compared them to the usual method found in the literature, for 2 different image stabilization mechanisms. In particular, we showed that the existing method from the literature is not suited for a quadrotor equipped with a 2-axis mechanical gimbal.

6.2 Outlooks

Though the overall strategy developed in this PhD thesis for the autonomous performance of aerial takes with quadrotors was validated on outdoor flight experiments, it still constitutes a proof of concept. Several directions of development have been identified and could complete this work.

Concerning the impact of the inertia of the propellers identified in the section 2.6 of chapter 2 (i.e. the non-minimum phase behavior brought in the attitude dynamics by the reaction torques of the propellers), it could be interesting to deepen the study by using a more realistic, nonlinear model of the quadrotor, rather than the linearized one. This would allow to characterize the phenomenon for a larger flight envelop, and to better quantify its impact on the quadrotor flight. This would also be the opportunity to study the impact of the gyroscopic torques of the propellers and to compare it with the impact of the reaction torques for different domains of the flight envelop. Finally, the validation of these phenomena on a real system would consolidate the theoretical results. Preliminary work has been provided during this thesis to obtain such experimental results, with a specially modified Bebop 2. The prototype presented a V4 configuration with a V-angle of 45°, as illustrated on figure 6.1, in order to highlight the non-minimum phase behavior of the pitch axis. However, we did not have the opportunity to re-tune the on-board controllers of the quadrotor to suit this new configuration, as this aggressive V-angle revealed particularly destructive to the closed-loop stability, on the real system. In the future, it could nonetheless be interesting to perform more tests on this configuration.

Another potential improvement concerns the minimum-time B-spline trajectory generation algorithm proposed in chapter 4, which does not include the drag of the quadrotor, as it was done in section 3.7 for the bi-level optimization proposed in chapter 3. This could be achieved by using the fact that the product of 2 B-spline curves is also a B-spline curve. As a consequence, the left term of (3.52) can be expressed as a B-spline curve, and using the convex hull property on this curve by constraining its control points in a feasible domain could be a solution to integrate the constraint (3.52) in the NLP problem (4.49). In addition, further work is required to properly implement the minimum-time B-spline trajectory generation algorithm in order to quantify the computation load needed to generate a trajectory and to assess its feasibility in real-time. This would also constitute a valuable opportunity to compare its performances with other popular methods, in terms of aesthetic quality of the trajectory, computation time and numerical robustness.

FIGURE 6.1 – Prototype of V4 quadrotor

Furthermore, in chapter 5, the on-board computation resources prevented us from solving a complex optimization problem with the MPC controller. With the improved computation power of the most recent generations of consumer quadrotors, it could be interesting to improve this method with a more detailed optimization problem, taking into account more aesthetic requirements.

Finally, the overall method could be improved by the addition of obstacle avoidance capacities. This could be achieved by using the cardinal B-spline based, local re-planning method proposed in [122] which requires a smooth and feasible nominal global trajectory, such as the ones obtained with the procedure exposed in chapter 4.

Appendix A

Reminder on screw theory

A.1 Torsor

Vector field. Let E be a vector space of dimension n built on $\mathbb{R}$ along with a scalar product, and $\mathcal{E}$ an affine space of direction E. A function V on $\mathcal{E}$ with values in E is called a *vector field*

$$V : \begin{pmatrix} \mathcal{E} \to E \\ P \mapsto V(P) \end{pmatrix} \tag{A.1}$$

Equiprojectivity. Let V be a vector field on $\mathcal{E}$. V is *equiprojective* if it verifies

$$\forall\, (A, B) \in \mathcal{E}^2 \quad V(A) \cdot \overrightarrow{AB} = V(B) \cdot \overrightarrow{AB} \tag{A.2}$$

with $\cdot$ designating the scalar product on E.

Torsor. Let $\mathcal{E}$ be an affine space of direction $\mathbb{R}^3$. A *torsor* $\mathcal{T}$ on $\mathcal{E}$ is an *equiprojective vector field* on $\mathcal{E}$

Remark 15 *In rigid body mechanics, $\mathcal{E}$ corresponds to the 3D physical space.*

Moment. Let $\mathcal{T}$ be a torsor on $\mathcal{E}$ and $P \in \mathcal{E}$. The evaluation of $\mathcal{T}$ at P, $\mathcal{T}(P)$, is called the *moment of $\mathcal{T}$ at point P*.

Resultant. Let $\mathcal{T}$ be a torsor on $\mathcal{E}$. It has the property

$$\exists!\, \mathbf{r} \in \mathbb{R}^3, \quad \forall (A, B) \in \mathcal{E}^2 \quad \mathcal{T}(B) = \mathcal{T}(A) + \overrightarrow{BA} \times \mathbf{r} \tag{A.3}$$

The vector $\mathbf{r}$ is called the *resultant* of $\mathcal{T}$.

Remark 16 *This property is a consequence of the equiprojectivity of a torsor.*

Elements of reduction. As a consequence of (A.3), it can be deduced that the resultant of a torsor along with its moment at a given point entirely define this torsor. Such a pair defines the reduction of a torsor at a point and is sometime called a *screw*. For a torsor $\mathcal{T}$ on $\mathcal{E}$, of resultant $\mathbf{r}$, we denote its reduction at a point $A \in \mathcal{E}$ by the following notation

$$\forall A \in \mathcal{E} \quad \mathcal{T} = \begin{Bmatrix} \mathbf{r} \\ \mathcal{T}(A) \end{Bmatrix}_A \tag{A.4}$$

Remark 17 *Using* (A.3)*, the reduction of torsor on a point $B \in \mathcal{E}$ can be deduced from its reduction at point $A \in \mathcal{E}$ as follows*

$$\forall (A,B) \in \mathcal{E}^2 \quad \begin{Bmatrix} \mathbf{r} \\ \mathcal{T}(A) \end{Bmatrix}_A = \begin{Bmatrix} \mathbf{r} \\ \mathcal{T}(A) + \overrightarrow{BA} \times \mathbf{r} \end{Bmatrix}_B \tag{A.5}$$

A.2 Application in rigid body mechanics

Let $\mathfrak{S}_\mathcal{B}$ be a rigid body of mass m, center of mass G and inertia tensor $\mathbf{J}_{\mathcal{B}/G}$. Attached to this solid is the referential frame *Body*, $\mathfrak{R}_\mathcal{B}$, of origin B and vector basis $\mathfrak{B}_\mathcal{B} = (\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B})$. It is in motion relatively to the inertial reference frame *World*, $\mathfrak{R}_\mathcal{W}$, attached to the ground, of origin O and basis $\mathfrak{B}_\mathcal{W} = (\mathbf{x}_\mathcal{W}, \mathbf{y}_\mathcal{W}, \mathbf{z}_\mathcal{W})$.

Kinetic torsor. We define the *kinetic torsor* of the rigid body $\mathfrak{S}_\mathcal{B}$ relatively to the inertial reference frame $\mathfrak{R}_\mathcal{W}$

$$\mathfrak{K}_{\mathcal{B}/\mathcal{W}} = \begin{Bmatrix} m\,\mathbf{v}_{G/\mathcal{W}} \\ \mathbf{J}_{\mathcal{B}/G}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \end{Bmatrix}_G \triangleq \begin{Bmatrix} \mathbf{p}_{\mathcal{B}/\mathcal{W}} \\ \boldsymbol{\sigma}^G_{\mathcal{B}/\mathcal{W}} \end{Bmatrix}_G \tag{A.6}$$

with $\mathbf{v}_{G/\mathcal{W}}$ the velocity of G in $\mathfrak{R}_\mathcal{W}$ and $\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ the angular velocity of $\mathfrak{B}_\mathcal{B}$ relatively to $\mathfrak{B}_\mathcal{W}$. The resultant of $\mathfrak{K}_{\mathcal{B}/\mathcal{W}}$ is the *linear momentum* of $\mathfrak{S}_\mathcal{B}$ and is denoted by $\mathbf{p}_{\mathcal{B}/\mathcal{W}}$ in this document. Its moment at a point P is the *angular momentum* of $\mathfrak{S}_\mathcal{B}$ at P and is denoted by $\boldsymbol{\sigma}^P_{\mathcal{B}/\mathcal{W}}$ in this document.

Dynamic torsor. The time-derivative of the kinetic torsor is called the *dynamic torsor*

$$\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \frac{\mathrm{d}}{\mathrm{d}t}\mathfrak{K}_{\mathcal{B}/\mathcal{W}} = \begin{Bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_{\mathcal{B}/\mathcal{W}} \\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\sigma}^G_{\mathcal{B}/\mathcal{W}} \end{Bmatrix}_G \triangleq \begin{Bmatrix} \boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}} \\ \boldsymbol{\delta}^G_{\mathcal{B}/\mathcal{W}} \end{Bmatrix}_G \tag{A.7}$$

Its resultant is called the *dynamic resultant* and is denoted by $\boldsymbol{\mu}_{\mathcal{B}/\mathcal{W}}$ in this document. Its moment at a point P is called the *dynamic moment* at P and is denoted by $\boldsymbol{\delta}^P_{\mathcal{B}/\mathcal{W}}$ in this document.

Mechanical action. Any mechanical action exerted by an actor $\mathcal{A}$ on an actor $\mathcal{B}$ can be represented by a torsor, denoted at a point P by

$$\mathfrak{F}_{\mathcal{A}\to\mathcal{B}} = \begin{Bmatrix} \mathbf{f}_{\mathcal{A}\to\mathcal{B}} \\ \boldsymbol{\tau}^P_{\mathcal{A}\to\mathcal{B}} \end{Bmatrix}_P \tag{A.8}$$

where $\mathbf{f}_{\mathcal{A}\to\mathcal{B}}$ and $\boldsymbol{\tau}^P_{\mathcal{A}\to\mathcal{B}}$ are the resultant and the moment at P of this mechanical action, respectively.

Remark 18 *If the resultant is null, then this action is called a torque. If there exists a point P at which the moment is null, this action is called a force, applied at P. Any mechanical action can be represented as the sum of a torque torsor and a force torsor.*

Fundamental Principle of Dynamics. Let $\mathfrak{F}_{\text{ext}\to\mathcal{B}}$ denotes the sum of all the mechanical actions applied to the rigid body $\mathfrak{S}_{\mathcal{B}}$. The FPD applied to the solid $\mathfrak{S}_{\mathcal{B}}$ can be expressed

$$\mathfrak{D}_{\mathcal{B}/\mathcal{W}} = \mathfrak{F}_{\text{ext}\to\mathcal{B}} \tag{A.9}$$

Appendix B

Gyroscopic and reaction propellers torques

This appendix details the calculations leading to the expressions of the reaction and gyroscopic torques used in equation (2.44a). The use of screw theory to this aim constitutes a contribution of this thesis and allows to identify several assumptions required to obtain the usual expressions given, but rarely justified, in the literature.

We start by giving the general expression of the dynamic moment of a propeller at the center of mass of the quadrotor, G. Then, we propose different assumptions which allow to simplifie some of the terms of this dynamic moment. Finally, we rewrite this dynamic moment in order to define 3 fictitious mechanical actions and give their expressions in the drone vector basis.

Rotor The rotor of the i-th propeller, $\mathfrak{S}_{\mathcal{P}_i}$, is linked to its stator (part of the drone body $\mathfrak{S}_{\mathcal{B}}$) through a motorized pivot. This rotor is of mass m_i, center of mass P_i and inertia $\mathbf{J}_{\mathcal{P}_i/P_i}$. The frame $\mathfrak{R}_{\mathcal{P}_i}$ of origin P_i and vector basis $\mathfrak{B}_{\mathcal{P}_i} = (\mathbf{x}_{\mathcal{P}_i}, \mathbf{y}_{\mathcal{P}_i}, \mathbf{z}_{\mathcal{P}_i})$ is attached to the rotor. The rotor is supposed to be correctly balanced, so that, relatively to the drone, it only rotates along the axis $(P_i, \mathbf{z}_{\mathcal{P}_i})$. The *ZYZ Euler angles* ψ_i, θ_i and ϕ_i are used to parameterize the attitude of the propeller relatively to the drone body, such that

$$\mathrm{R}_{\mathcal{P}_i \to \mathcal{B}} = \begin{pmatrix} c_{\psi_i} c_{\theta_i} c_{\phi_i} - s_{\psi_i} s_{\phi_i} & -s_{\psi_i} c_{\theta_i} c_{\phi_i} - c_{\psi_i} s_{\phi_i} & s_{\theta_i} c_{\phi_i} \\ c_{\psi_i} c_{\theta_i} s_{\phi_i} + s_{\psi_i} s_{\phi_i} & -s_{\psi_i} c_{\theta_i} s_{\phi_i} + c_{\psi_i} c_{\phi_i} & s_{\theta_i} s_{\phi_i} \\ -c_{\psi_i} s_{\theta_i} & s_{\psi_i} s_{\theta_i} & c_{\theta_i} \end{pmatrix} \tag{B.1}$$

with ψ_i and θ_i fixed (defined in section 2.4.2). The angular velocity of the propeller relatively to the drone body is given by

$$\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} = \omega_i \, \mathbf{z}_{\mathcal{P}_i} \tag{B.2}$$

with $\omega_i > 0$. The angle ϕ_i gives the angular position of the rotor on its rotation axis (see figure B.1).

Dynamic torsor Supposing that the mass and inertia of the propellers are invariant, the dynamic torsor of the i-th propeller is given by

$$\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} = \left\{ \begin{matrix} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \\ \boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} \end{matrix} \right\}_{P_i} = \left\{ \begin{matrix} \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \\ \boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} + \overrightarrow{G\mathcal{P}_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}} \end{matrix} \right\}_{G} \tag{B.3}$$

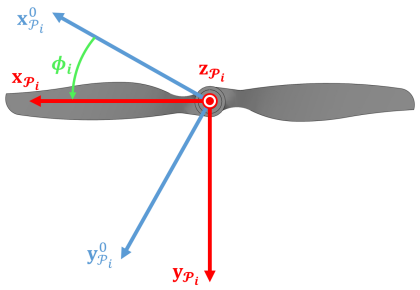

FIGURE B.1 – Propeller angular position

with $\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}$ given by equation (2.43) and $\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}}$ given by

$$\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{W}}\right) \tag{B.4}$$

Knowing that $\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{W}} = \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$, expression (B.4) can be written as follows

$$\boldsymbol{\delta}^{P_i}_{\mathcal{P}_i/\mathcal{W}} = \mathbf{J}_{\mathcal{P}_i/P_i}\left(\dot{\boldsymbol{\Omega}}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}}\right) + \left(\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}\right) \times \mathbf{J}_{\mathcal{P}_i/P_i}\left(\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}\right) \tag{B.5}$$

Simplifications The assumption that the rotor is correctly balanced leads to equation (B.2) and implies that the term $\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}}$ in equation (B.5) is null. Furthermore, for a drone such as the Parrot Bebop 2

- The inertia of the rotors are significantly smaller than the inertia of the drone body (up to 3 orders of magnitudes for the Bebop 2).
- The angular velocity and acceleration of the drone body relatively to the ground are significantly smaller than the ones of the propellers relatively to the drone body.

Then, it is reasonable to neglect the terms $\mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\boldsymbol{\Omega}}_{\mathcal{B}/\mathcal{W}}$ and $\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ in (B.5). It can be noticed that the second observation does not hold for some extreme cases, such as [84] or [131] for which the rotation speed of the drone body might be significant compared to the one of the propellers.

Fictitious mechanical actions As a consequence, the dynamic torsor of the i-th propeller can be expressed as the sum of 3 fictitious mechanical actions

$$\mathfrak{D}_{\mathcal{P}_i/\mathcal{W}} \approx -\mathfrak{F}_{\mathrm{inertia}_i\to\mathcal{B}} - \mathfrak{F}_{\mathrm{gyro}_i\to\mathcal{B}} - \mathfrak{F}_{\mathrm{lever}_i\to\mathcal{B}} \tag{B.6a}$$

with

$$\mathfrak{F}_{\mathrm{inertia}_i\to\mathcal{B}} = -\left\{\begin{array}{c}\mathbf{0}\\ \mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\boldsymbol{\Omega}}_{\mathcal{P}_i/\mathcal{B}}\end{array}\right\}_G \tag{B.6b}$$

$$\mathfrak{F}_{\mathrm{gyro}_i\to\mathcal{B}} = -\left\{\begin{array}{c}\mathbf{0}\\ \boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} + \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}}\end{array}\right\}_G \tag{B.6c}$$

$$\mathfrak{F}_{\mathrm{lever}_i\to\mathcal{B}} = -\left\{\begin{array}{c}\boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}\\ \overrightarrow{G\mathcal{P}_i} \times \boldsymbol{\mu}_{\mathcal{P}_i/\mathcal{W}}\end{array}\right\}_G \tag{B.6d}$$

respectively the reaction torque, the gyroscopic torque and the leverage action.

Reaction torque The moment of the reaction torque in the drone body basis $\mathfrak{B}_\mathcal{B}$ is given by

$$\mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\mathbf{\Omega}}_{\mathcal{P}_i/\mathcal{B}} = \left(\mathrm{R}_{\mathcal{B}\to P_i}\,\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}\begin{pmatrix}0\\0\\\dot\omega_i\end{pmatrix}\right)_\mathcal{B} \tag{B.7}$$

with $\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}$ the matrix representing $\mathbf{J}_{\mathcal{P}_i/P_i}$ in he basis $\mathfrak{B}_{\mathcal{P}_i}$.

If we suppose the basis $\mathfrak{B}_{\mathcal{P}_i}$ principal of inertia for the propeller i (which is not completely accurate in practice), $\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}$ is diagonal

$$\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i} \approx \begin{pmatrix}J_{xi}&&\\&J_{yi}&\\&&J_{zi}\end{pmatrix} \tag{B.8}$$

then expression (B.7) can be written

$$\mathbf{J}_{\mathcal{P}_i/P_i}\,\dot{\mathbf{\Omega}}_{\mathcal{P}_i/\mathcal{B}} = \begin{pmatrix}\dot\omega_i\,J_{zi}s_{\theta_i}c_{\psi_i}\\\dot\omega_i\,J_{zi}s_{\theta_i}s_{\psi_i}\\\dot\omega_i\,J_{zi}c_{\theta_i}\end{pmatrix}_\mathcal{B} \tag{B.9}$$

The expression of the reaction torque is thus

$$\mathfrak{F}_{\mathrm{inertia}_i\to\mathcal{B}} = \left\{\begin{pmatrix}\mathbf{0}\\\begin{pmatrix}-\dot\omega_i\,J_{zi}s_{\theta_i}c_{\psi_i}\\-\dot\omega_i\,J_{zi}s_{\theta_i}s_{\psi_i}\\-\dot\omega_i\,J_{zi}c_{\theta_i}\end{pmatrix}_\mathcal{B}\end{pmatrix}\right\}_G \tag{B.10}$$

Gyroscopic torque The moment of the gyroscopic torque is given by the sum of the 3 terms in (B.6c).

The expression of the term $\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}}\times\mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}}$ in the basis $\mathfrak{B}_\mathcal{B}$ is given by

$$\begin{aligned}\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}}\times\mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}} &= \left(\begin{pmatrix}p\\q\\r\end{pmatrix}\times\mathrm{R}_{\mathcal{B}\to P_i}\,\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}\begin{pmatrix}0\\0\\\omega_i\end{pmatrix}\right)_\mathcal{B}\\ &= \begin{pmatrix}\omega_i\,J_{zi}\,(qc_{\theta_i}-rs_{\theta_i}s_{\psi_i})\\-\omega_i\,J_{zi}\,(pc_{\theta_i}-rs_{\theta_i}c_{\psi_i})\\-\omega_i\,J_{zi}\,s_{\theta_i}\,(qc_{\psi_i}-ps_{\psi_i})\end{pmatrix}_\mathcal{B}\end{aligned} \tag{B.11}$$

The expressions of the last two terms are more complex and dependent on ϕ_i. However, for sufficient rotation speeds of the propellers, the angular velocities of both the propellers and the drone body can be considered constant over a complete round of the rotor. We thus propose to approximate the expressions of these two terms by their mean value over one propeller round

$$\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}}\times\mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}} = \left(\mathrm{R}_{\mathcal{B}\to P_i}\begin{pmatrix}0\\0\\\omega_i\end{pmatrix}\times\mathrm{R}_{\mathcal{B}\to P_i}\,\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}\,{\mathrm{R}_{\mathcal{B}\to P_i}}^\top\begin{pmatrix}p\\q\\r\end{pmatrix}\right)_\mathcal{B} \tag{B.12}$$

which leads to

$$\frac{1}{2\pi}\int_0^{2\pi}\mathbf{\Omega}_{\mathcal{P}_i/\mathcal{B}}\times\mathbf{J}_{\mathcal{P}_i/P_i}\,\mathbf{\Omega}_{\mathcal{B}/\mathcal{W}}\,\mathrm{d}\varphi = \begin{pmatrix}-\omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,(qc_{\theta_i}-rs_{\theta_i}s_{\psi_i})\\\omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,(pc_{\theta_i}-rs_{\theta_i}c_{\psi_i})\\\omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,s_{\theta_i}\,(qc_{\psi_i}-rs_{\psi_i})\end{pmatrix}_\mathcal{B} \tag{B.13}$$

The same can be done for the last term

$$\mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} = \left(\mathrm{R}_{\mathcal{B}\to P_i}\,\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i}\,\mathrm{R}_{\mathcal{B}\to P_i}{}^{\top} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathrm{R}_{\mathcal{B}\to P_i} \begin{pmatrix} 0 \\ 0 \\ \omega_i \end{pmatrix} \right)_{\mathcal{B}} \tag{B.14}$$

which gives

$$\frac{1}{2\pi}\int_0^{2\pi} \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}} \times \boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}}\,\mathrm{d}\varphi = \begin{pmatrix} \omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,(qc_{\theta_i} - rs_{\theta_i}s_{\psi_i}) \\ -\omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,(pc_{\theta_i} - rs_{\theta_i}c_{\psi_i}) \\ -\omega_i\,\frac{1}{2}\big(J_{xi}+J_{y_i}\big)\,s_{\theta_i}\,(qc_{\psi_i} - rs_{\psi_i}) \end{pmatrix}_{\mathcal{B}} \tag{B.15}$$

It can be noticed that the two terms (B.13) and (B.15) cancel each other and only the term $\boldsymbol{\Omega}_{\mathcal{P}_i/\mathcal{B}} \times \mathbf{J}_{\mathcal{P}_i/P_i}\,\boldsymbol{\Omega}_{\mathcal{B}/\mathcal{W}}$ remains in (B.6c). The expression of the gyroscopic torque is thus

$$\mathfrak{F}_{\mathrm{gyro}_i\to\mathcal{B}} = \left\{ \begin{pmatrix} \mathbf{0} \\ \begin{pmatrix} -\omega_i\,J_{zi}\,(qc_{\theta_i} - rs_{\theta_i}s_{\psi_i}) \\ \omega_i\,J_{zi}\,(pc_{\theta_i} - rs_{\theta_i}c_{\psi_i}) \\ \omega_i\,J_{zi}\,s_{\theta_i}\,(qc_{\psi_i} - ps_{\psi_i}) \end{pmatrix}_{\mathcal{B}} \end{pmatrix} \right\}_{G} \tag{B.16}$$

Rotor disk The more blades there are on the propellers, the more their inertia approaches the form

$$\mathrm{J}^{\mathcal{P}_i}_{\mathcal{P}_i/P_i} \approx \begin{pmatrix} J_{xi} & & \\ & J_{xi} & \\ & & J_{zi} \end{pmatrix} \tag{B.17}$$

for which the inertia on the x-axis is equal to the inertia on the y-axis. This approximation is quite valid for 3 or more blades per propeller (which is the case for the Parrot Bebop 2). If this is the case, then the propellers behave as flat cylinders of revolution axis $\mathbf{z}_{\mathcal{P}_i}$. The dependence in ϕ_i disappears in the two last terms of (B.6c) and their expression is exactly equal to the averaged expressions given above. In this case, the assumption of constant body and propeller angular velocities over one round of a propeller is then no longer required.

Appendix C

Feasible rest-to-rest B-spline trajectory

This appendix provides one way to initialize the Nonlinear Programming (NLP) (4.49) with a $\mathcal{C}^3$, feasible, clamped B-spline trajectory, for the case studied in section 4.5. This initial guess is a rest-to-rest trajectory with a Bang-Off-Bang (BOB) snap profile and it is divided into three phases: accelerating, cruising, braking. A feasibility proof of this initial solution is provided.

For a piece of trajectory i, joining two waypoints $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ such that $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 > 0$, with a speed reference $\nu_i > 0$, a maximum admissible acceleration $a_{\max} > 0$ and a maximum admissible jerk $j_{\max} > 0$, we first define the 2 scalars

$$\widetilde{v}_i = \min\left(\nu_i, \frac{8\,a_{\max}^2}{9\,j_{\max}}\right) \tag{C.1a}$$

$$s = \frac{3\,j_{\max}^2}{2\,a_{\max}} \tag{C.1b}$$

with $\widetilde{v}_i$ the maximum reachable cruising speed with the method proposed in this appendix and s a snap reference.

We now detail the case with a cruising phase.

C.1 Case with a cruising phase

In the case where

$$4\,\widetilde{v}_i \sqrt[3]{\frac{\widetilde{v}_i}{2s}} < \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 \tag{C.2}$$

we can define the following time intervals

$$\Delta\tau_{\text{acc}} = \sqrt[3]{\frac{\widetilde{v}_i}{2s}}, \quad \Delta\tau_{\text{cruise}} = \frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{\widetilde{v}_i} - 4\Delta\tau_{\text{acc}} \tag{C.3}$$

The clamped B-spline defined by the following knot steps and control points is chosen as initial solution

$$\boldsymbol{\Delta\tau}_{\text{init}} = (\Delta\tau_{\text{acc}}\ \ 2\Delta\tau_{\text{acc}}\ \ \Delta\tau_{\text{acc}}\ \ \Delta\tau_{\text{cruise}}\ \ \Delta\tau_{\text{acc}}\ \ 2\Delta\tau_{\text{acc}}\ \ \Delta\tau_{\text{acc}}) \tag{C.4a}$$

$$\mathbf{P}_{\text{init}}^{(4)} = \mathbf{u}_i \cdot (s\ \ -s\ \ s\ \ 0\ \ -s\ \ s\ \ -s) \tag{C.4b}$$

First, since $\nu_i > 0$, $a_{\max} > 0$ and $j_{\max} > 0$, we have $\Delta\tau_{\text{acc}} > 0$. Furthermore, using the assumption (C.2) and given that $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 > 0$, it follows that $\Delta\tau_{\text{cruise}} > 0$.

Using (4.22), the control points of the jerk are deduced from (C.4b), with a null jerk as initial condition

$$\mathbf{P}^{(3)}_{\text{init}} = \mathbf{u}_i \cdot (0 \;\; s\Delta\tau_{\text{acc}} \;\; -s\Delta\tau_{\text{acc}} \;\; 0 \;\; 0 \;\; -s\Delta\tau_{\text{acc}} \;\; s\Delta\tau_{\text{acc}} \;\; 0) \tag{C.5}$$

From (C.1a) we can deduce that $\widetilde{v}_i \leqslant \frac{8\,a_{\max}^2}{9\,j_{\max}}$, which, injected into (C.3), leads to

$$\Delta\tau_{\text{acc}} \leqslant \frac{2\,a_{\max}}{3\,j_{\max}} \tag{C.6}$$

Equations (C.1b) and (C.6) lead to $s\Delta\tau_{\text{acc}} \leqslant j_{\max}$. The norm of each control point of $\mathbf{P}^{(3)}_{\text{init}}$ is bounded by $s\Delta\tau_{\text{acc}}$ and thus bounded by $j_{\max}$.

Similarly, the control points of the acceleration are

$$\mathbf{P}^{(2)}_{\text{init}} = \mathbf{u}_i \cdot \left(0 \;\; 0 \;\; \frac{3}{2}s\,\Delta\tau_{\text{acc}}^2 \;\; 0 \;\; 0 \;\; 0 \;\; -\frac{3}{2}s\,\Delta\tau_{\text{acc}}^2 \;\; 0 \;\; 0\right) \tag{C.7}$$

Using (C.1b) and (C.6), we can write $\frac{3}{2}s\,\Delta\tau_{\text{acc}}^2 \leqslant a_{\max}$. The norm of each control point of $\mathbf{P}^{(2)}_{\text{init}}$ is thus bounded by $a_{\max}$.

The control points of the velocity are

$$\mathbf{P}^{(1)}_{\text{init}} = \mathbf{u}_i \cdot \big(0 \;\; 0 \;\; 0 \;\; 2s\Delta\tau_{\text{acc}}^3 \;\; 2s\Delta\tau_{\text{acc}}^3 \;\; 2s\Delta\tau_{\text{acc}}^3 \;\; 2s\Delta\tau_{\text{acc}}^3 \;\; 0 \;\; 0 \;\; 0\big) \tag{C.8}$$

Equation (C.3) directly gives $\Delta\tau_{\text{acc}}^3 = \frac{\widetilde{v}_i}{2s}$. The bound on the norm of the control points of $\mathbf{P}^{(1)}_{\text{init}}$ is then $\widetilde{v}_i \leqslant \nu_i$.

Finally, one last integration step with the initial condition $\mathbf{p}_{\text{init}_0} = \mathbf{w}_{i-1}$ gives the control points of the position

$$\mathbf{P}_{\text{init}} = \mathbf{u}_i \cdot (0 \;\; 0 \;\; 0 \;\; 0 \;\; 2\alpha \;\; 4\alpha \;\; 6\alpha \;\; 8\alpha \;\; 8\alpha \;\; 8\alpha \;\; 8\alpha) + \mathbf{w}_{i-1} \tag{C.9}$$

with $\alpha = s\Delta\tau_{\text{acc}}^3\big(\Delta\tau_{\text{acc}} + \frac{1}{4}\Delta\tau_{\text{cruise}}\big)$. Using (C.3) and $\Delta\tau_{\text{acc}}^3 = \frac{\widetilde{v}_i}{2s}$ leads to $\alpha = \frac{1}{8}\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2$. The last control point of the position is thus $\mathbf{w}_i$.

We now detail the second case with no cruising phase.

C.2 Case without a cruising phase

If (C.2) is not satisfied, i.e.

$$4\,\widetilde{v}_i \sqrt[3]{\frac{\widetilde{v}_i}{2s}} \geqslant \|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 \tag{C.10}$$

we define

$$\Delta\tau_{\text{acc}} = \sqrt[4]{\frac{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2}{8s}} \tag{C.11}$$

Injecting (C.10) into (C.11) gives the same bound on $\Delta\tau_{\text{acc}}$ as in the case with a cruising phase

$$\Delta\tau_{\text{acc}} \leqslant \sqrt[3]{\frac{\widetilde{v}_i}{2s}} \leqslant \frac{2\,a_{\max}}{3\,j_{\max}} \tag{C.12}$$

The clamped B-spline defined by the following knot steps and control points is chosen as initial guess

$$\boldsymbol{\Delta\tau}_{\text{init}} = \left(\Delta\tau_{\text{acc}}\ \ 2\Delta\tau_{\text{acc}}\ \ \frac{1}{2}\Delta\tau_{\text{acc}}\ \ \frac{1}{2}\Delta\tau_{\text{acc}}\ \ \Delta\tau_{\text{acc}}\ \ 2\Delta\tau_{\text{acc}}\ \ \Delta\tau_{\text{acc}}\right) \tag{C.13a}$$

$$\mathbf{P}^{(4)}_{\text{init}} = \mathbf{u}_i \cdot (s\ \ -s\ \ s\ \ s\ \ -s\ \ s\ \ -s) \tag{C.13b}$$

The control points of the jerk are

$$\mathbf{P}^{(3)}_{\text{init}} = \mathbf{u}_i \cdot \left(0\ s\Delta\tau_{\text{acc}}\ -s\Delta\tau_{\text{acc}}\ -\frac{1}{2}s\Delta\tau_{\text{acc}}\ 0\ -s\Delta\tau_{\text{acc}}\ s\Delta\tau_{\text{acc}}\ 0\right) \tag{C.14}$$

Therefore, given (C.1b) and (C.12), their norm is bounded by $j_{\max}$.

The control points of the acceleration are

$$\mathbf{P}^{(2)}_{\text{init}} = \mathbf{u}_i \cdot \left(0\ \ 0\ \ \frac{3}{2}s\,\Delta\tau^2_{\text{acc}}\ \ \frac{1}{4}s\,\Delta\tau^2_{\text{acc}}\ \ 0\ \ 0\ \ -\frac{3}{2}s\,\Delta\tau^2_{\text{acc}}\ \ 0\ \ 0\right) \tag{C.15}$$

Thus, given (C.1b) and (C.12), their norm is bounded by $a_{\max}$.

The control points of the velocity are

$$\mathbf{P}^{(1)}_{\text{init}} = \mathbf{u}_i \cdot \left(0\ 0\ 0\ \frac{7}{4}s\Delta\tau^3_{\text{acc}}\ 2s\Delta\tau^3_{\text{acc}}\ 2s\Delta\tau^3_{\text{acc}}\ 2s\Delta\tau^3_{\text{acc}}\ 0\ 0\ 0\right) \tag{C.16}$$

Equation (C.12) gives $\Delta\tau^3_{\text{acc}} \leqslant \frac{\widetilde{v}_i}{2s}$, hence, the norm of each control points of the velocity is bounded by $\widetilde{v}_i \leqslant \nu_i$.

Finally, the control points of the position are

$$\mathbf{P}_{\text{init}} = \mathbf{u}_i \cdot \left(0\ \ 0\ \ 0\ \ 0\ \ \frac{7}{4}\beta\ \ \frac{15}{4}\beta\ \ \frac{23}{4}\beta\ \ 8\beta\ \ 8\beta\ \ 8\beta\ \ 8\beta\right) + \mathbf{w}_{i-1} \tag{C.17}$$

with $\beta = s\Delta\tau^4_{\text{acc}}$. Given (C.11), the last control point of the position is thus $\mathbf{w}_i$.

C.3 Feasibility analysis

In both cases the position trajectory contains $n+1 = 11$ control points and $n-k+1 = 7$ knot steps. Its polynomial degree is thus $k = 4$, which implies that the position trajectory is $\mathcal{C}^3$.

The fist control point of the position is $\mathbf{w}_{i-1}$ and the last one is $\mathbf{w}_i$ (see (C.9) and (C.17)). Since clamped B-splines are used, the trajectory then starts on $\mathbf{w}_{i-1}$ and ends on $\mathbf{w}_i$. The trajectory is a straight line (see (C.9) and (C.17)), thus the lateral corridor constraints are satisfied. The position control points are set between $\mathbf{w}_{i-1}$ and $\mathbf{w}_i$ (see (C.9) and (C.17)) thus the convex hull property ensures that the longitudinal corridor constraints are satisfied too. The bounds on the control points of the time derivatives of the position are respected. Finally, the first and last control point of the velocity, the acceleration and the jerk are $\mathbf{0}$. Laying end to end several of these feasible rest-to-rest trajectories thus leads to an overall trajectory that is $\mathcal{C}^3$.

All the constraints are satisfied, the initial guess is feasible. This proves the validity of the proposition 3.

Appendix D

Résumé en français

D.1 Introduction

D.1.1 Contexte général

Grâce à leur faible coût et leur grande agilité, les véhicules sans pilote de type quadricoptère se sont largement popularisés depuis la dernière décennie. Si leur polyvalence leur a permis de trouver des applications dans divers domaines comme la surveillance, l'agriculture ou la photogrammétrie, ils se sont notamment imposés comme un moyen de prise de vue aérienne privilégié. Il est ainsi désormais courant de voir des amateurs utiliser des quadricoptères équipés de caméras stabilisées à des fins vidéographiques. Dans ce contexte, le pilotage du drone n'est pas une fin en soi, comme elle peut l'être pour un aéromodéliste, mais seulement un moyen d'acquérir des images. L'aspect pilotage de ces appareils tend par conséquent à se retrouver en arrière plan et le public se tourne maintenant vers des appareils de plus en plus intelligents, capables d'accomplir des tâches haut niveau de manière automatisée pour capturer des plans riches et complexes. Les appareils les plus récents embarquent ainsi des algorithmes de cadrage automatique de sujets, de suivi autonome de personnes ou de réalisation de plans de vol.

Le développement de tels algorithmes comme sujet de recherche au sein de la communauté scientifique est relativement récent et a réellement émergé dans la période 2015-2016, bien que de tels algorithmes étaient déjà déployés par les industriels du domaine. Ce sujet constitue aujourd'hui un sujet populaire de recherche sur lequel le nombre de publications croit chaque année. Le développement de telles capacités requiert en effet des avancées dans des domaines variés comme la vision par ordinateur (*computer vision*), l'apprentissage automatique (*machine learning*) et l'automatique. Tous les aspects de l'automatique sont notamment concernés, de la modélisation et identification à l'estimation et le contrôle. Ces derniers aspects sont généralement regroupés en trois catégories dans le contexte de la robotique

- **Guidage.** La couche de contrôle haut niveau en charge de générer des trajectoires faisables depuis des consignes ou informations haut niveau, comme des points de passages à rejoindre ou des positions d'obstacles à éviter par exemple.
- **Navigation.** L'estimation en temps réel de l'état du drone, en charge notamment de la fusion des différents capteurs embarqués.
- **Pilotage.** La couche bas niveau du contrôle en charge de générer les signaux de

commande destinés aux actionneurs, depuis les références issues du guidage et les estimations issues de la navigation.

- **Supervision.** La supervision générale du vol, en charge de générer les références haut niveau pour le guidage et de choisir les paramètres des trois autres blocs selon la mission. Ce bloc peut être un programme ou un humain équipé d'une télécommande par exemple.

Cette thèse porte sur le développement d'algorithmes de guidage et pilotage pour la réalisation autonome d'une des tâches haut niveau mentionnées plus haut, la capture d'une séquence de plans prédéfinis.

D.1.2 Spécification du problème

Le type de plan de vol à réaliser est spécifié dans le premier chapitre. Ils consistent en la réalisation d'une séquence de plans formalisée par

- **Points de passages.** Une suite de points de passage à rejoindre et valider successivement.

- **Couloirs de vol.** Des couloirs de vol à respecter entre chaque point de passage, sous forme de cylindre.

- **Vitesse.** Des consignes de vitesse pour rejoindre chaque point de passage, à ne pas dépasser.

- **Type de plan.** Un type de plan à réaliser en rejoignant chaque point de passage.

Trois types de points de passage sont proposés : *stop*, pour lequel le drone doit s'arrêter sur le point de passage (en début et en fin de plan de vol typiquement, ou pour prendre une photo), *lock*, pour lequel le drone doit passer exactement sur le point de passage (utile si le drone doit passer par une position spécifique telle qu'à travers une porte par exemple) et *sphere*, pour lequel le drone doit seulement passer dans un voisinage du point de passage. Les plans à réaliser sont quant à eux caractérisés par un comportement pour chaque degré de liberté de la caméra (autre que sa position), à savoir son cap (pan), son assiette (tilt), son inclinaison (roll) et son grossissement (zoom). Le comportement de chaque degré de liberté de la caméra peut être choisi séparément. Parmi les comportements proposés pour ces degrés de liberté figurent notamment le *travelling* (référence constante), le panorama (référence en rampe), le point d'intérêt (suivi d'une cible fixe) ou le point de vue (référence selon le vecteur vitesse du drone) ou enfin une transition entre deux comportements différents. Un exemple de plan de vol est présenté figure D.1 et tableau D.1.

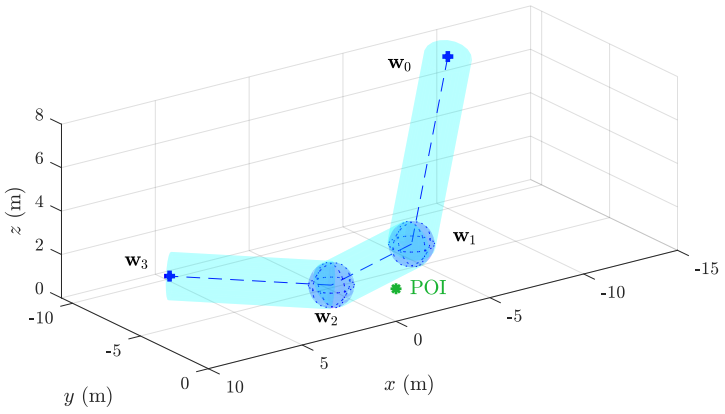

FIGURE D.1 – Exemple de plan de vol

	Type	Rayon	Couloir	Vitesse	Pan	Tilt	Roll	Zoom
$\mathbf{w}_0$	*stop*	-	-	-	*cst.* 0°	*cst.* 0°	*cst.* 0°	*cst.* 1×
$\mathbf{w}_1$	*sphere*	1 m	1 m	2 m·s^{-1}	*trs.*	*trs.*	*cst.* 0°	*cst.* 1×
$\mathbf{w}_2$	*sphere*	1 m	1 m	2 m·s^{-1}	*PI*	*PI*	*cst.* 0°	*cst.* 1×
$\mathbf{w}_3$	*stop*	-	1 m	2 m·s^{-1}	*PI*	*PI*	*cst.* 0°	*vertigo*

TABLE D.1 – Détails du plan de vol de la figure D.1

Des contraintes esthétiques et physiques sont prises en compte afin d'obtenir de ce plan de vol une vidéo lisse et naturelle. Ces contraintes sont étroitement liées au quadricoptère autour duquel s'articule les travaux de cette thèse, un Parrot Bebop 2 équipé d'une caméra fixe, à la stabilisation entièrement numérique. Ces contraintes sont les suivantes

- La position, l'orientation et le grossissement de la caméra doivent évoluer de manière douce et naturelle, sans discontinuités ou oscillations. L'accélération et le jerk quantifiant cette notion de douceur et d'à-coups, leur amplitude doit rester limitée.
- La vitesse de rotation de la caméra doit être raisonnable pour ne pas produire une vidéo chaotique et non satisfaisante d'un point de vue esthétique. La vitesse maximale de rotation acceptable a été déterminée via un entretien avec un pilote professionnel et des tests en vol.
- Le Bebop 2 utilisant une caméra à la stabilisation entièrement numérique, son angle relativement au sol doit rester en dessous d'une certaine valeur, due au fonctionnement de la stabilisation. Cela est en partie la conséquence du champ de vision limité de la caméra.
- La caméra du Bebop 2 étant fixe par rapport au drone, la vitesse de rotation de ce dernier doit restée limitée afin de ne pas produire de flou de bouger qui dégraderait la qualité de la vidéo.

Afin d'obtenir une réalisation autonome de plan de vols cinématographiques remplissant ces critères, avec un Bebop 2, ces travaux mobilisent entre autre la modélisation et la synthèse d'algorithme de guidage et pilotage du drone. Ce premier aspect est traité dans un premier chapitre, dédié à la modélisation d'un quadricoptère tel que le Parrot Bebop 2.

D.2 Modélisation

D.2.1 Construction du modèle

Le second chapitre de cette thèse consiste en une étude en profondeur de la dynamique du drone menée en vue d'établir des modèles pertinents pour le guidage et le pilotage du drone. Là où le formalisme de Lagrange est le plus souvent utilisé, le principe fondamental de la dynamique et le formalisme des torseurs sont ici utilisés pour construire un modèle de quadricoptère comme solide indéformable. Une approche à plusieurs niveaux est utilisée pour satisfaire les différentes exigences de fidélité du modèle des algorithmes de guidage et pilotage proposés dans la suite de la thèse. On préférera ainsi un modèle simplifié pour les algorithmes de guidage et un modèle plus proche des actionneurs du drone pour la synthèse des algorithmes de pilotage. Les hypothèses sous lesquelles sont construites le modèle sont explicitées et justifiées, et comptent notamment l'hypothèse de la Terre plate et de solides indéformables.

Le Bebop 2 étant constitué de plusieurs solides en interaction, son corps (représenté figure D.2, à gauche) et ses rotors (représenté figure D.2, au milieu), il n'est pas possible d'appliquer directement le principe fondamental de la dynamique directement au quadricoptère dans son ensemble. La construction du modèle est ainsi opérée en deux étapes

- La caractérisation de la dynamique et des actions subies par les différents constituants du drone, à savoir son corps et ses rotors

- La construction d'un solide virtuel équivalent au drone complet (représenté figure D.2, à droite), comprenant son corps et ses rotors

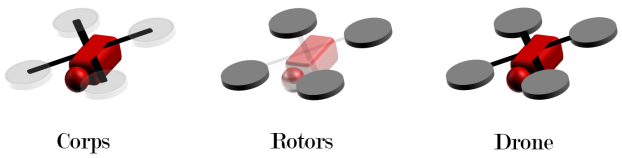

FIGURE D.2 – Schéma des différents solides considérés pour la modélisation

Les trois manières les plus répandues dans la littérature pour représenter l'attitude et leur impact dans l'expression de la dynamique du drone, les angles d'Euler, les quaternions unitaires et les matrices de rotation, sont exposées et comparées. Les différentes actions mécaniques expérimentées par le drone sont présentées, et comptent la gravité, la réaction de la masse d'air, vue comme une perturbation, et la poussée des moteurs (considérée quadratique en la vitesse de rotation des rotors) et leurs effets d'inertie. La caractérisation de la poussée des moteurs est notamment non triviale, car faisant intervenir de nombreux phénomènes physiques comme la dissymétrie de portance entre rotors, le battement des pales, la variation de densité de l'air avec l'altitude, les effets de proximité, la variation de l'angle d'attaque des rotors due aux variations de vitesse air ou encore l'impact des turbulences et de l'état de vortex entretenu. La caractérisation de la dynamique du corps du drone et de ses interactions avec ses rotors et l'environnement permet ensuite de construire

un modèle non linéaire équivalent du drone entier plutôt que du corps du drone uniquement

$$\begin{aligned}
\text{Accélération :}\ & \ddot{\boldsymbol{\zeta}} = g\,\mathbf{z}_{\mathcal{W}} + \sum_{i=4}^{4} \mathbf{f}_i + \mathbf{f}_{\text{pert}} \\
\text{Attitude :}\ & \dot{\mathbf{R}} = \mathbf{R}\,\widehat{\boldsymbol{\Omega}} \\
\text{Accélération angulaire :}\ & \mathbf{J}\,\dot{\boldsymbol{\Omega}} = \sum_{i=4}^{4} \boldsymbol{\tau}_i + \overrightarrow{GP_i} \times \mathbf{f}_i + \boldsymbol{\tau}_{\text{pert}} \\
\text{Poussées moteurs :}\ & \mathbf{f}_i = \left(\alpha_{0i} + \alpha_{1i}\,\omega_i + \alpha_{2i}\,\omega_i^2\right) \mathbf{z}_{\mathcal{P}_i} \\
\text{Couples moteurs :}\ & \boldsymbol{\tau}_i = \left(\beta_{0i} + \beta_{1i}\,\omega_i + \beta_{2i}\,\omega_i^2\right) \mathbf{z}_{\mathcal{P}_i} - \mathbf{J}_{\mathcal{P}_i}\,\dot{\omega}_i\,\mathbf{z}_{\mathcal{P}_i} - \boldsymbol{\Omega} \times \mathbf{J}_{\mathcal{P}_i}\,\omega_i\,\mathbf{z}_{\mathcal{P}_i} \\
\text{Dynamique moteur :}\ & \ddot{\omega}_i + 2\xi_i \varpi_i\,\dot{\omega}_i + \varpi_i^2\,\omega_i = \varpi_i^2\,\omega_{i\text{ref}}
\end{aligned} \tag{D.1}$$

Cette approche permet notamment de mener une étude rigoureuse des couples gyroscopiques et des couples de réaction générés par les rotors du drone. Tout au long de la construction du modèle est proposé, en parallèle, un état de l'art de la modélisation des quadricoptères. Afin de simplifier ce modèle et de mieux comprendre les effets de l'inertie des rotors dans la dynamique en rotation, ce modèle est ensuite linéarisé autour de l'équilibre correspondant au vol stationnaire.

D.2.2 Linéarisation du modèle et étude des configurations moteurs

Une linéarisation du modèle obtenu autour de l'état de vol stationnaire est étudiée et permet de mettre en évidence le découplage des dynamiques en lacet, roulis, tangage et translation verticale, ainsi que l'effet de l'inertie des rotors sur l'axe de lacet, via la matrice de mixage.

Considérant tous les groupes propulsifs identiques et décomposant leur vitesse de rotation ω_i en leur vitesse en vol stationnaire ω_{hi} et un incrément de vitesse par rapport à celle-ci $\delta\omega_i$

$$\omega_i = \omega_{hi} + \delta\omega_i \tag{D.2}$$

il est possible d'introduire des signaux de commande

$$\begin{pmatrix} \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \\ \delta\omega_4 \end{pmatrix} = \mathrm{B}_{v\to\omega} \cdot \begin{pmatrix} v_w \\ v_p \\ v_q \\ v_r \end{pmatrix} \tag{D.3}$$

tels que les expressions linéarisées de la dynamique en rotation et vitesse verticale puissent être découplées, menant aux expressions suivantes.

$$\begin{aligned}
w(s) &= \frac{k_w}{1+\tau_w\,s}\,v_w(s), \quad p(s) = \frac{k_p}{1+\tau_p\,s}\,v_p(s), \\
q(s) &= \frac{k_q}{1+\tau_q\,s}\,v_q(s), \quad\ \ r(s) = k_r\,\frac{1+\sigma_r\,s}{1+\tau_r\,s}\,v_r(s),
\end{aligned} \tag{D.4}$$

avec $\mathrm{B}_{v\to\omega}$ la matrice de mixage, fonction des paramètres moteurs, w la vitesse verticale du drone et p, q et r les composantes de la vitesse de rotation du drone sur ses axes de roulis, tangage et lacet, respectivement.

Dans la construction de la matrice de mixage pour un quadricoptère en configuration X4 (4 rotors à plat, symétriquement disposés autour du centre de gravité), nous constatons notamment le déséquilibre de l'actionnement de la rotation du drone qui présente une faiblesse sur l'axe de lacet. Cela provient du fait que dans une telle configuration, le lacet est principalement actionné par la traînée des hélices quand le roulis et le tangage sont actionnés par leur portance, qui engendre des moments bien supérieurs aux couples induits par la traînée des rotors (entre 10 et 100 fois supérieurs). Or, le contrôle de cet axe est important pour un drone comme le Bebop 2 pour lequel le cap de la caméra est fortement lié au cap du drone. Pour compenser en partie cette faiblesse d'actionnement de l'axe de lacet, une solution consiste à incliner les rotors afin de faire contribuer une partie des couples issus de la portance des hélices à l'actionnement de l'axe de lacet. Une étude est menée à l'aide du modèle linéarisé pour mettre en évidence et quantifier cet effet en fonction de l'inclinaison des moteurs. Cette étude fait également apparaître l'impact de l'inertie des hélices dans la dynamique en rotation. Si celui-ce se traduit en un zéro bénéfique dans la dynamique en lacet, il se manifeste en revanche par l'apparition de zéros à partie réelle positive dans les dynamiques de roulis et tangage lorsque les moteurs sont inclinés, menant à un comportement à non minimum de phase, comme illustré ci-dessous

$$p(s) = k_p \frac{1 - \sigma_p\, s}{1 + \tau_p\, s}\, v_p(s), \quad q(s) = k_q \frac{1 - \sigma_q\, s}{1 + \tau_q\, s}\, v_q(s) \tag{D.5}$$

Des simulations sont menées pour appuyer cet aspect et une modélisation linéaire du drone par système d'état, prenant en compte ces résultats est finalement proposée.

Les modèles obtenus sont utilisés dans la suite comme base pour synthétiser des algorithmes de pilotage ou guidage, ce qui constitue l'objet des chapitres suivants. Le développement d'algorithmes de guidage est traité en premier et constitue l'objet du chapitre trois.

D.3 Génération de trajectoire par optimisation bi-niveaux

D.3.1 Stratégie

Nous abordons dans un troisième chapitre la question de la génération de la trajectoire de la caméra volante. Celle-ci est traitée en deux parties, la génération de la trajectoire en position du quadricoptère porteur de la caméra d'un côté et de la trajectoire en rotation et grossissement de la caméra de l'autre. Deux méthodes courantes permettent d'aborder la génération de la trajectoire en position du drone, le suivi de trajectoire ou le suivi de chemin. La première consiste en la génération d'une trajectoire de référence comme une fonction du temps, qui est suivie par un correcteur. La seconde méthode s'affranchit de la dépendance au temps en s'appuyant sur une loi dont émerge le suivi d'un chemin établi. Une comparaison de ces deux méthodes dans le domaine de la cinématographie est proposée, en s'appuyant notamment sur la littérature. Si la méthode de suivi de trajectoire peut être considérée comme moins robuste que celle du suivi de chemin de par sa sensibilité aux décrochages de la trajectoire nominale de référence, elle autorise néanmoins un plus grand contrôle temporel sur l'état du drone ce qui représente un avantage pour certaines séquences de prise de vue. Une stratégie de suivi de trajectoire est ainsi choisie.

D.3.2 Contraintes de faisabilité et d'esthétique

Avant de traiter la question de la génération de la trajectoire en position, des contraintes sur sur celle-ci sont mises en évidence afin que celle-ci soit non seulement faisable, relativement à la dynamique du drone, mais aussi satisfaisante au regard de critères cinématographiques. De telles garanties sont obtenues en deux temps. Tout d'abord, le plan de vol est pré-traité afin d'assurer que les consignes de vitesse appartiennent au domaine de vol. On ajuste ainsi les consignes de vitesses excessives, pouvant conduire à un état de vortex entretenu ou pouvant nuire à la réalisation de certains plans. Si besoin, le rayon de validation des points de passage est également réduit pour empêcher la validation simultanée de plusieurs points de passages. Néanmoins, ce pré-traitement, seul, ne permet pas de garantir la faisabilité de la trajectoire ni sont esthétique. En utilisant le modèle non linéaire simplifié du drone, les contraintes de poussée minimale et maximale, de vitesse de rotation maximale et angle maximal du drone relativement au sol et de variation temporelle maximale de cap sont reformulées en termes de contraintes sur les dérivées temporelles de la position du drone. Ces contraintes se démarquent notamment de la littérature par la prise en compte de l'angle du drone ainsi que la vitesse de rotation en lacet du drone qui est amenée à varier selon les plans à réaliser. Est ainsi montré qu'une borne sur l'amplitude de l'accélération du drone permet de garantir, en l'absence de perturbation, le respect des contraintes de poussée minimale et maximale et d'angle maximal relativement au sol. Le respect des contraintes de vitesse de rotation et de variation de cap est quant à lui obtenu via une contrainte sur l'amplitude du jerk du drone

$$\Big(a \leqslant a_{\max} \triangleq \min\left\{g - f_{\min},\ f_{\max} - g,\ g\, s_{\alpha_{\max}}\right\}\Big) \Rightarrow \begin{cases} f \geqslant f_{\min} \\ f \leqslant f_{\max} \\ \alpha \leqslant \alpha_{\max} \end{cases} \tag{D.6a}$$

$$\begin{cases} \dot{\psi}_{\max} < \Omega_{\max} \\ \\ j \leqslant j_{\max} \triangleq \dfrac{\sqrt{\Omega_{\max}^2\left(1 + {s_{\alpha_{\max}}}^2\right) - \dot{\psi}_{\max}^2} - s_{\alpha_{\max}}\dot{\psi}_{\max}}{1 + {s_{\alpha_{\max}}}^2}\, f_{\min} \end{cases} \Rightarrow \left(\|\boldsymbol{\Omega}\|_2 \leqslant \Omega_{\max}\right) \tag{D.6b}$$

avec a l'accélération du drone, f la poussée du drone, $f_{\min}$ et $f_{\max}$ respectivement les poussées minimales et maximales que peut fournir le drone, α l'angle entre la verticale du drone et la verticale du sol et $\alpha_{\max}$ sa valeur maximale atteignable par le drone, j le jerk du drone, $\dot{\psi}_{\max}$ la vitesse en cap du drone et $\dot{\psi}_{\max}$ sa valeur maximale pour garantir une qualité de vidéo correcte et $\Omega_{\max}$ la vitesse de rotation maximale du drone par rapport au sol. Ces contraintes sont ensuite utilisées pour la formulation du problème de génération de trajectoire, définie dans le paragraphe suivant.

D.3.3 Optimisation bi-niveaux

Une première manière de générer une trajectoire polynomiale par morceaux via une optimisation bi-niveaux est tout d'abord étudiée. Chaque morceau polynomial de la trajectoire sert à rejoindre une paire de points de passage consécutifs. La trajectoire est alors paramétrée par deux jeux de paramètres, des temps de vols entre chaque point de passage du plan de vol d'une part et des coefficients polynomiaux d'autre part. Chacun de ces deux jeux de paramètres est optimisé par un algorithme d'optimisation différent, fonctionnant à deux niveaux différents et représentés figure D.3.

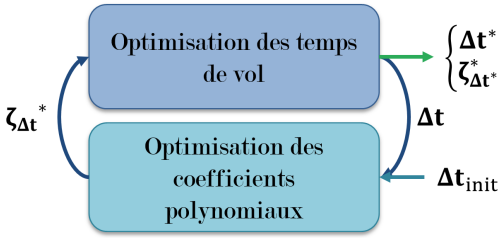

FIGURE D.3 – Principe de l'optimisation bi-niveaux

Pour un jeu de temps de vols donné, un algorithme d'optimisation de niveau bas (en bas sur la figure D.3) génère un jeu de coefficients polynomiaux paramétrant une trajectoire à jerk minimal, respectant: les temps de vol entre chaque point de passage, les critères de validation des points de passage et les couloirs de vols. Une méthode de maillage est utilisée pour cette dernière contrainte, consistant à vérifier son respect en un nombre de points fini de la trajectoire. L'algorithme assure également la continuité de la trajectoire aux connections entre chaque morceau de polynôme. La résolution de ce problème constitue un problème quadratique sous contraintes linéaires. De la minimisation du jerk résulte une trajectoire douce et naturelle, masquant les transitions entre points de passage et ainsi adaptée pour la cinématographie. Un second algorithme d'optimisation, de niveau haut (en haut sur la figure D.3), est en charge de déterminer le jeu de temps de vol minimisant la durée totale de la mission tout en s'assurant que les dérivées temporelles de la trajectoire respectent les contraintes de faisabilité et d'esthétique mises en évidence précédemment. La résolution de ce problème constitue un problème non linéaire. Cette architecture bi-niveaux permet ainsi de séparer la génération de la trajectoire en un problème quadratique "large" mais simple à résoudre, pour la génération des coefficients polynomiaux, et un problème non linéaire mais de taille réduite, pour la génération des temps de vol. Cette méthode de génération de trajectoire par optimisation bi-niveaux est confrontée à des simulations (voir figure D.4) ainsi qu'à un vol en extérieur avec un drone Parrot Bebop 2.

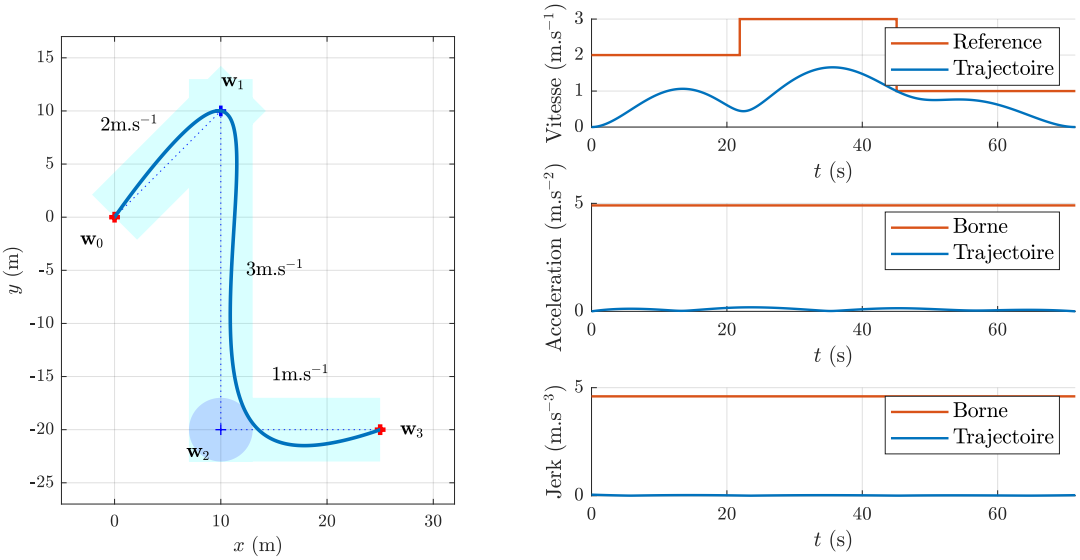

FIGURE D.4 – Trajectoire à temps minimum/jerk minimum obtenue avec l'optimisation bi-niveaux

D.3.4 Amélioration de l'algorithme

Diverses améliorations de l'algorithme sont ensuite proposées comme la prise en compte de la traînée du drone. Un modèle de traînée linéaire est ainsi inclus et la formulation de la contrainte sur l'amplitude de l'accélération est adaptée pour prendre en compte ce phénomène. Une alternative à cette formulation prenant également en compte la vitesse moyenne du vent par rapport au sol permet d'améliorer encore la faisabilité de la trajectoire. Des simulations sont réalisées pour illustrer les performances de ces formulations augmentées de la traînée, et permettent de comparer l'impact de ce phénomène sur le respect des contraintes de faisabilité. Une étude sur l'initialisation du problème de niveau haut (non linéaire) est ensuite menée et différentes méthodes d'initialisation sont comparées, dont une initialisation à vitesse supposée constante et une initialisation pour laquelle l'accélération de la trajectoire est approximée par un profil *bang-off-bang*. Leurs impacts sur le nombre d'itérations requis par l'algorithme d'optimisation pour générer les trajectoires sont présentés et comparés. Une stratégie d'horizon glissant sur les points de passage est ensuite proposée pour maîtriser la taille du problème d'optimisation à résoudre. L'impact de cette stratégie sur le temps de calcul de la trajectoire du plan de vol complet est illustré, de même que son impact sur la qualité de la trajectoire obtenue. Bien que les résultats soient satisfaisants, une limite de la formulation bi-niveaux est mise en évidence, celle-ci conduisant à des profils de vitesses parfois trop peu agressifs. Ainsi, dans le chapitre suivant, un algorithme pour générer des trajectoires plus agressives est proposé.

D.4 Génération de trajectoire B-splines non uniformes à temps minimal

D.4.1 Stratégie

Une seconde manière de générer la trajectoire est proposée dans le quatrième chapitre. La trajectoire est cette fois paramétrée par des B-splines, et l'optimisation de la trajectoire est monolithique plutôt que bi-niveaux. Les B-splines constituent des polynômes par morceaux paramétrés par des points de contrôle dans l'espace ainsi que des nœuds, correspondant aux instants pour lesquels la courbe B-spline change de représentation polynomiale. L'utilisation de B-spline présente divers avantages. Là où la première méthode utilisait un polynôme pour chaque paire de points de passages, la seconde utilise désormais une B-spline, qui constitue en soi un polynôme par morceaux. Ainsi, chaque morceau de trajectoire joignant une paire de points de passage est lui même découpée en sous-trajectoires prenant typiquement la forme de phases d'accélération, croisière et freinage. Ceci permet d'obtenir une trajectoire finale à la fois plus riche et de degré polynomial moindre. De plus la propriété d'enveloppe convexe des B-splines est utilisée pour contraindre la trajectoire dans ses couloirs de vol et limiter l'amplitude des dérivées afin de respecter les contraintes de faisabilité et d'esthétique. Ce travail se démarque notamment de la littérature par l'utilisation de B-splines non uniformes, là où des B-splines uniformes sont généralement utilisées.

D.4.2 Réduction des paramètres

Après une rapide introduction des B-splines et de leurs propriétés, une manière compacte de représenter une trajectoire B-spline non uniforme respectant les contraintes de validation

des points de passage est proposée. Ces contraintes imposent ou contraignent implicitement le positionnement des points de contrôle des B-splines autour des points de passages, qui peuvent alors être entièrement déterminés à partir des autres points de contrôle. Ces points de contrôle contraints sont ainsi retirés la représentation, et l'ensemble des points de contrôle est remplacé par un ensemble réduit de points de contrôle. Une méthode pour reconstruire l'ensemble complet des points de contrôle à partir de l'ensemble réduit et des nœuds de la B-spline est exposée, et constitue une opération linéaire en les points de contrôle réduits et non-linéaire en les nœuds. Cette représentation compacte est ensuite utilisée pour formuler la génération de la trajectoire comme une problème d'optimisation non linéaire, ce qui fait l'objet du paragraphe suivant.

D.4.3 Problème d'optimisation

Bien que d'autres métriques que la durée totale de la trajectoire soient utilisées dans la littérature pour la synthèse de trajectoires B-splines, comme sa longueur par exemple, l'utilisation de B-splines non uniformes dans ce travail permet de directement générer une trajectoire à durée minimale. La formulation du problème, de la fonction et des contraintes, est détaillée. Contrairement à la méthode bi-niveaux, les contraintes de couloirs de vol et sur l'amplitude des dérivées ne sont plus formulées par un maillage mais en utilisant la propriété d'enveloppe convexe des B-splines pour garantir la satisfaction des contraintes sur la totalité de la trajectoire. La méthode est confrontée à des simulations (voir figure D.5) ainsi qu'à un vol en extérieur. Les profils de vitesse obtenus sont comparés à ceux que l'on obtiendrait avec des B-splines uniformes et mettent en avant la pertinence de l'utilisation de B-splines non-uniformes.

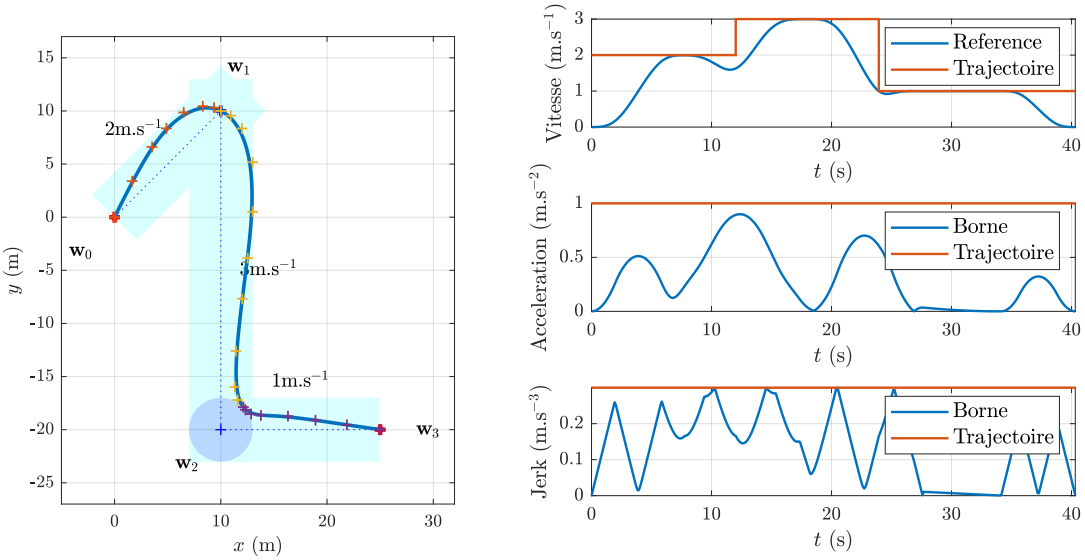

FIGURE D.5 – Trajectoire B-spline non uniforme à temps minimum

D.4.4 Étude de l'initialisation

Une étude sur l'initialisation du problème non linéaire est finalement menée. Plusieurs méthodes d'initialisation sont proposées et comparées

- Une trajectoire en ligne brisée s'arrêtant à chaque point de passage, présentant un profil *bang-off-bang* en snap
- Une B-spline dont les points de contrôle sont confondus avec les points de passage
- Une B-spline dont les points de contrôle sont uniformément distribués entre les points de passage
- Une B-spline uniforme minimisant la moyenne quadratique d'une dérivée de la position
- Une B-spline uniforme minimisant la moyenne quadratique d'une dérivée de la position

Ayant généré une trajectoire en position, il reste alors à traiter la trajectoire en rotation et grossissement de la caméra, ce qui fait l'objet du chapitre cinq.

D.5 Commande prédictive de la caméra

D.5.1 Architecture à suivi de modèle nominal

Le cinquième chapitre traite de la génération de la trajectoire de la caméra. Plusieurs modes de prises de vue sont détaillés ainsi que la manière de calculer les références en rotation et grossissement pour chacun d'entre eux, tels que la prise de vue subjective, le zoom compensé ou le point d'intérêt. Une estimation du temps minimal pour compléter certains plans, compte tenu de l'excursion angulaire qu'ils requièrent et la limitation de vitesse de rotation de la caméra, est proposée pour limiter les vitesses de références lors du pré-traitement des plans de vols (voir section D.3.2).

Si les suivis des références en tilt, roll et zoom de la caméra ne posent pas de difficulté puisqu'ils sont obtenus par la stabilisation de la caméra, le suivi des références en pan requiert plus d'attention. La pan de la caméra étant identique au cap du drone, la commande du pan de la caméra revient au commande du cap du drone. La génération d'une loi de commande sur le cap du drone, pour un suivi doux des références en pan caméra, induisant des transitions douces entre les modes de prises de vue lors de la validation des waypoints, est ainsi étudiée. Une architecture de suivi de modèle nominal (*Nominal Model Following Control*) est utilisée afin de séparer les dynamiques de suivi de référence et de rejet de perturbations (voir figure D.6).

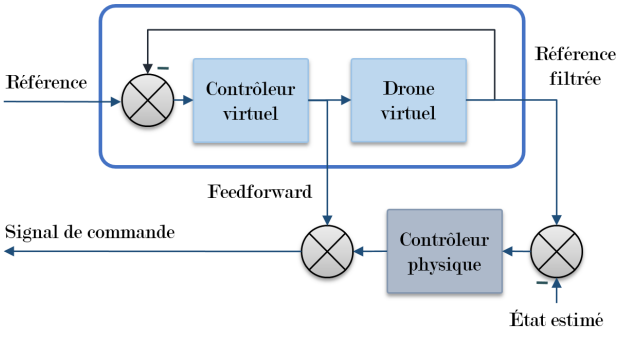

FIGURE D.6 – Architecture NMFC pour la commande en cap

Cette architecture consiste en l'asservissement d'un modèle nominal drone, simulé en temps réel, par un contrôleur virtuel. Ce contrôleur dicte la dynamique de suivi de référence, douce et avec un temps de réponse relativement élevé dans ce travail pour satisfaire les critères d'esthétique vidéo. L'état du modèle virtuel simulé asservi par le contrôleur virtuel est envoyé comme référence à un second contrôleur de rejet de perturbation à la dynamique plus raide, asservissant le drone réel cette fois. La commande générée par le contrôleur virtuel est quant à elle envoyée en *feedforward* au drone. En l'absence de perturbation et pour un modèle nominal parfait, ce *feedforward* doit ainsi induire la même trajectoire sur le drone réel que sur le modèle simulé et le contrôleur de rejet de perturbation ne travaille pas. Le drone est ainsi piloté en boucle ouverte. Dans un scénario réaliste, le contrôleur de rejet de perturbation compense les écarts à la trajectoire nominal dus aux perturbations et erreurs de modélisation. Sa dynamique est par conséquent plus raide afin de gommer ces écarts. Un contrôleur prédictif est proposé comme contrôleur virtuel et est détaillé dans le paragraphe suivant; le contrôleur physique n'est pas détaillé dans ce travail.

D.5.2 Contrôleur prédictif virtuel

Si la dynamique lente du contrôleur virtuel permet d'obtenir des mouvements caméra fluides et doux, elle risque néanmoins d'entraîner de fortes erreurs de suivi, se traduisant par des erreurs de cadrage notamment. Afin de palier ce problème, une loi de commande prédictive est utilisée pour le contrôleur virtuel, pour son aspect anticipatif. Toutefois, pour être pertinente, cette loi de commande doit disposer d'un horizon de prédiction au moins du même ordre de grandeur que le temps de réponse désiré. Étant donné la haute fréquence d'échantillonnage de la boucle de contrôle du Bebop 2, cela signifie typiquement un horizon de l'ordre de la centaine de pas. Afin de réduire ce nombre de pas, un sous échantillonnage de ce contrôleur prédictif est proposé. La commande obtenue est interpolée par un bloqueur d'ordre un avant d'être ré-échantillonnée à haute fréquence.

Afin d'être pris explicitement en compte dans la synthèse de la loi de commande, cette interpolation et le ré-échantillonnage qui la suit sont inclus dans le modèle de prédiction. Le réglage des paramètres de la loi de commande pouvant être délicat, nous proposons une méthode basée sur une optimisation par essaim particulaire pour effectuer ce réglage. Le comportement de la loi de commande est validé en simulation ainsi qu'en vol extérieur.

D.5.3 Reconstruction de l'attitude du drone

Enfin, la reconstruction de l'attitude de référence du drone depuis le référence en cap de sa caméra et la référence en vecteur poussée, ainsi que son impact sur le champ de vision de la caméra est étudié. Une méthode alternative à la reconstruction classiquement utilisée dans la littérature est proposée pour un drone plus récent de Parrot, le Parrot ANAFI, disposant d'une nacelle caméra dont deux axes sont stabilisés mécaniquement et le dernier numériquement.

D.6 Conclusion

D.6.1 Récapitulatif

Après avoir spécifié le problème de prise de vue de plan séquence aériens via des plans de vol, une analyse en profondeur de la dynamique d'un quadricoptère a été menée. Une linéarisation du modèle non linéaire obtenu autour de l'état de vol stationnaire a notamment permis de découpler les dynamiques en rotation et translation verticale du drone. L'extension de ce modèle à des quadricoptères dont les rotors sont inclinés a permis de mettre en évidence l'apparition de termes de couplages entre les dynamiques en rotation et translation ainsi que d'un comportement à non minimum de phase sur certains axes de rotations.

Dans un second temps, un première algorithme de génération de trajectoire est proposé, reposant sur l'optimisation bi-niveaux d'une trajectoire polynomiale par morceaux, visant à minimiser le jerk de la trajectoire. Cette méthode est améliorée par l'ajout d'un modèle de traînée linéaire, ainsi que d'un horizon glissant sur les points de passages. Elle a notamment été confrontée à un vol en extérieur avec un Parrot Bebop 2.

Un second algorithme de génération de trajectoire a ensuite été proposé afin d'obtenir des trajectoires plus agressives. Celles-ci sont cette fois obtenues en optimisant une trajectoire B-spline de manière à minimiser le temps de parcours du plan de vol. Une étude sur l'initialisation du problème d'optimisation a été menée afin de réduire le temps de calcul pour la génération de la trajectoire. Enfin, cet algorithme a été confronté à un vol extérieur avec un Parrot ANAFI.

Enfin, une méthode a été proposée pour asservir le cap du drone, telle que le suivi des références soit doux mais avec un impact maîtrisé sur l'erreur de suivi, et sans dégrader la dynamique de rejet de perturbation. Une étude sur la reconstruction de l'attitude de référence du drone à partir de sa poussée et son cap de référence est finalement menée.

D.6.2 Perspectives

Pour terminer, plusieurs axes de développement sont proposés. Concernant l'impact de l'inertie des hélices dans la dynamique en rotation, il pourrait être intéressant d'approfondir l'étude en utilisant un modèle plus fin, non linéaire, du quadricoptère. Cela permettrait entre autre de caractériser l'influence de ces inerties sur un domaine de vol plus large. Cela constituerait également l'opportunité d'étudier l'impact des couples gyroscopiques des rotors et de le comparer à celui de l'impact des couples de réactions. De plus la validation des résultats théoriques pour un drone à rotors inclinés sur un système réelles permettrait de les consolider et de mieux les quantifier en pratique.

Un autre axe de développement concerne l'algorithme de génération de trajectoires B-splines proposé en chapitre quatre, pour lequel n'a pas été fait le travail d'inclure le modèle de traînée linéaire du chapitre trois. Cela pourrait être accompli en utilisant le résultat que le produit de deux courbes B-spline est aussi une courbe B-spline, sur lequel s'applique aussi la propriété d'enveloppe convexe. De plus, il serait intéressant d'optimiser l'implémentation de l'algorithme afin de comparer ses performances en termes de temps de calcul à d'autres algorithmes de la littérature. Cette méthode pourrait aussi bénéficier de l'ajout de contraintes d'évitement d'obstacles.

Enfin, la puissance de calcul augmentée des drones de nouvelle génération pourrait

être mise à profit pour complexifier la loi de commande prédictive utiliser pour contrôler l'orientation de la caméra afin de prendre en compte des critères et contraintes cinématographiques plus poussés.

Bibliography

[1] Abdolhosseini, M., Zhang, Y., and Rabbath, C. A. (2013). An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 70(1-4):27–38.

[2] Adigbli, P., Mouret, J.-B., and Doncieux, S. (2007). Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques. In *Proceedings of the 3rd US-European Competition and Workshop on Micro Air Vehicle Systems*, pages 1–9, Toulouse, France.

[3] Bangura, M. (2017). *Aerodynamics and control of quadrotors*. PhD thesis, The Australian National University, Canberra, Australia.

[4] Bangura, M., Lim, H., Kim, H. J., and Mahony, R. (2014). Aerodynamic power control for multirotor aerial vehicles. In *Proceedings of the International Conference on Robotics and Automation*, pages 529–536, Hong Kong, China.

[5] Bangura, M. and Mahony, R. (2012). Nonlinear dynamic modeling for high performance control of a quadrotor. In *Proceedings of the Australasian Conference on Robotics and Automation*, pages 1–10, Wellington, New Zealand.

[6] Bangura, M. and Mahony, R. (2014). Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780.

[7] Bangura, M. and Mahony, R. (2017). Thrust control for multirotor aerial vehicles. *Transactions on Robotics*, 33(2):390–405.

[8] Bangura, M., Melega, M., Naldi, R., and Mahony, R. (2016). Aerodynamics of rotor blades for quadrotors. *arXiv preprint arXiv:1601.00733*.

[9] Bellocchio, E., Ciarfuglia, T. A., Crocetti, F., Ficola, A., and Valigi, P. (2016). Modelling and simulation of a quadrotor in V-tail configuration. *International Journal of Modelling, Identification and Control*, 26(2):158–170.

[10] Bernard, D. D. C., Riccardi, F., Giurato, M., and Lovera, M. (2017). A dynamic analysis of ground effect for a quadrotor platform. *IFAC-PapersOnLine*, 50(1):10311–10316.

[11] Bertrand, S., Marzat, J., Stoica Maniu, C., Makarov, M., Filliat, D., and Manzanera, A. (2018). DroMOOC: a massive open online course on drones and aerial multi robot systems. In *Proceedings of the 12th UKACC International Conference on Control*, pages 434–434, Sheffield, England.

[12] Bhat, S. P. and Bernstein, D. S. (1998). A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon. In *Proceedings of the American Control Conference*, volume 5, pages 2785–2789, Philadelphia, Pennsylvania, USA.

[13] Boeuf, A., Cortés, J., Alami, R., and Siméon, T. (2014). Planning agile motions for quadrotors in constrained environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 218–223, Chicago, Illinois, USA.

[14] Bonatti, R., Ho, C., Wang, W., Choudhury, S., and Scherer, S. (2019). Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments. *arXiv preprint arXiv:1904.02319*.

[15] Bouabdallah, S., Noth, A., and Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 3, pages 2451–2456, Sendai, Japan.

[16] Bouabdallah, S. and Siegwart, R. (2007). Full control of a quadrotor. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 153–158, San Diego, California, USA.

[17] Bouffard, P. (2012). On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments. Technical report, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

[18] Brescianini, D., Hehn, M., and D'Andrea, R. (2013). Nonlinear quadrocopter attitude control. Technical report, ETH Zurich.

[19] Bristeau, P.-J., Callou, F., Vissiere, D., and Petit, N. (2011). The navigation and control technology inside the AR.Drone micro uav. *IFAC Proceedings Volumes*, 44(1):1477–1484.

[20] Bristeau, P.-J., Martin, P., Salaün, E., and Petit, N. (2009). The role of propeller aerodynamics in the model of a quadrotor UAV. In *Proceedings of the European Control Conference*, pages 683–688, Budapest, Hungary.

[21] Camacho, E. F. and Bordons, C. (2008). *Model predictive control, Second Edition*. Springer.

[22] Casau, P., Sanfelice, R. G., Cunha, R., Cabecinhas, D., and Silvestre, C. (2015). Robust global trajectory tracking for a class of underactuated vehicles. *Automatica*, 58:90–98.

[23] Castillo, P., Dzul, A., and Lozano, R. (2004). Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on control systems technology*, 12(4):510–516.

[24] Chamseddine, A., Zhang, Y., Rabbath, C. A., Join, C., and Theilliol, D. (2012). Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):2832–2848.

[25] Chaturvedi, N. A., Sanyal, A. K., and McClamroch, N. H. (2011). Rigid-body attitude control. *Control Systems Magazine*, 31(3):30–51.

[26] Chen, Y.-A., Wu, T.-Y., Chang, T., Liu, J. Y., Hsieh, Y.-C., Hsu, L. Y., Hsu, M.-W., Taele, P., Yu, N.-H., and Chen, M. Y. (2018). ARPilot: designing and investigating AR shooting interfaces on mobile devices for drone videography. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–42, Barcelona, Spain.

[27] Crouch, P. E. and Grossman, R. (1993). Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science*, 3(1):1–33.

[28] Das, J., Cross, G., Qu, C., Makineni, A., Tokekar, P., Mulgaonkar, Y., and Kumar, V. (2015). Devices, systems, and methods for automated monitoring enabling precision agriculture. In *Proceedings of the International Conference on Automation Science and Engineering*, pages 462–469, Gothenburg, Sweden.

[29] De Boor, C. (1978). *A practical guide to splines*, volume 27 of *Mathematics of Computation*.

[30] de Marina, H. G. and Smeur, J. J. (2019). Flexible collaborative transportation by a team of rotorcraft. *arXiv preprint arXiv:1902.00279*.

[31] Diaz, T. J. (2015). Lights, drone... action. *IEEE Spectrum*, 52(7):36–41.

[32] Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35.

[33] Dupont, Q. F., Chua, D. K., Tashrif, A., and Abbott, E. L. (2017). Potential applications of UAV along the construction's value chain. *Procedia Engineering*, 182:165–173.

[34] Eberly, D. (2002). Rotation representations and performance issues. Technical report, Geometric Tools.

[35] Engelhardt, T., Konrad, T., Schäfer, B., and Abel, D. (2016). Flatness-based control for a quadrotor camera helicopter using model predictive control trajectory generation. In *Proceedings of the 24th Mediterranean Conference on Control and Automation*, pages 852–859, Athens, Greece.

[36] Fang, Z., Wang, X.-y., and Sun, J. (2010). Design and nonlinear control of an indoor quadrotor flying robot. In *Proceedings of the 8th World Congress on Intelligent Control and Automation*, pages 429–434, Jinan, China.

[37] Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

[38] Fleureau, J., Galvane, Q., Tariolle, F.-L., and Guillotel, P. (2016). Generic drone control platform for autonomous capture of cinema scenes. In *Proceedings of the 2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, pages 35–40, Singapore, Singapore.

[39] Formentin, S. and Lovera, M. (2011). Flatness-based control of a quadrotor helicopter via feedforward linearization. In *Proceedings of the Conference on Decision and Control and European Control Conference*, pages 6171–6176, Orlando, Florida, USA.

[40] Galvane, Q., Christie, M., Ronfard, R., Lim, C.-K., and Cani, M.-P. (2013). Steering behaviors for autonomous cameras. In *Proceedings of ACM SIGGRAPH Conference on Motion in Games*, pages 93–102, Dublin, Ireland.

[41] Galvane, Q., Fleureau, J., Tariolle, F.-L., and Guillotel, P. (2017). Automated cinematography with unmanned aerial vehicles. In *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing*, Lisbonne, Portugal.

[42] Galvane, Q., Lino, C., Christie, M., Fleureau, J., Servant, F., Tariolle, F.-L., and Guillotel, P. (2018). Directing cinematographic drones. *ACM Transactions on Graphics*, 37(3):34.

[43] Gebhardt, C., Hepp, B., Nägeli, T., Stevšić, S., and Hilliges, O. (2016). Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 2508–2519, San Jose, California, USA.

[44] Gebhardt, C., Stevšić, S., and Hilliges, O. (2018). Optimizing for aesthetically pleasing quadrotor camera motion. *ACM Transactions on Graphics*, 37(4):90.

[45] Goldman Sachs (2016). Drones: Reporting for work. Technology Driving Innovation.

[46] Goodarzi, F., Lee, D., and Lee, T. (2013). Geometric nonlinear PID control of a quadrotor UAV on se(3). In *Proceedings of the European Control Conference*, pages 3845–3850, Zürich, Switzerland.

[47] Gschwindt, M., Camci, E., Bonatti, R., Wang, W., Kayacan, E., and Scherer, S. (2019). Can a robot become a movie director? Learning artistic principles for aerial cinematography. *arXiv preprint arXiv:1904.02579*.

[48] Guenard, N., Hamel, T., and Moreau, V. (2005). Dynamic modeling and intuitive control strategy for an X4-flyer. In *Proceedings of the International Conference on Control and Automation*, volume 1, pages 141–146, Budapest, Hungary.

[49] Ham, Y., Han, K. K., Lin, J. J., and Golparvar-Fard, M. (2016). Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (UAVs): a review of related works. *Visualization in Engineering*, 4(1):1–8.

[50] Hamel, T., Mahony, R., Lozano, R., and Ostrowski, J. (2002). Dynamic modelling and configuration stabilization for an X4-flyer. *IFAC Proceedings Volumes*, 35(1):217–222.

[51] Hilario, L., Montés, N., Mora, M. C., and Falcó, A. (2011). Real-time Bézier trajectory deformation for potential fields planning methods. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1567–1572, San Fransisco, California, USA.

[52] Hoffmann, G. M., Huang, H., Waslander, S. L., and Tomlin, C. J. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the Guidance, Navigation and Control Conference and Exhibit*, pages 1–20, Hilton Head, South Carolina, USA.

[53] Hönig, W., Preiss, J. A., Kumar, T. S., Sukhatme, G. S., and Ayanian, N. (2018). Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869.

[54] Hossain, M. S., Kabir, A. M., Mazumder, P., Aziz, A., Hassan, M., Islam, M. A., and Saha, P. K. (2012). Design and development of an Y4 copter control system. In *Proceedings of the 14th International Conference on Computer Modelling and Simulation*, pages 251–256, Cambridge, United Kingdom.

[55] Huang, C., Yang, Z., Kong, Y., Chen, P., Yang, X., and Cheng, K.-T. T. (2018). Through-the-lens drone filming. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 4692–4699, Madrid, Spain.

[56] Huang, H., Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J. (2009). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Proceedings of the International Conference on Robotics and Automation*, pages 3277–3282, Kobe, Japan.

[57] INACHUS (2015). System to quickly search for survivors after disasters. European Project.

[58] Joubert, N. (2017). Tools to facilitate autonomous quadrotor cinematography. Technical report, Stanford University.

[59] Joubert, N., Jane, L. E., Goldman, D. B., Berthouzoz, F., Roberts, M., Landay, J. A., and Hanrahan, P. (2016). Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles. *arXiv preprint arXiv:1610.01691*.

[60] Joubert, N., Roberts, M., Truong, A., Berthouzoz, F., and Hanrahan, P. (2015). An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics*, 34(6):238–238.

[61] Kamel, M., Alexis, K., Achtelik, M., and Siegwart, R. (2015). Fast nonlinear model predictive control for multicopter attitude tracking on SO(3). In *Proceedings of the Conference on Control Applications*, pages 1160–1166, Sydney, Australia.

[62] Karakostas, I., Mademlis, I., Nikolaidis, N., and Pitas, I. (2018). UAV cinematography constraints imposed by visual target tracking. In *Proceedings of the International Conference on Image Processing*, pages 76–80, Athens, Greece.

[63] Khan, M. A., Ectors, W., Bellemans, T., Janssens, D., and Wets, G. (2017). UAV-based traffic analysis: A universal guiding framework based on literature survey. *Transportation Research Procedia*, 22:541–550.

[64] Lee, T. (2013). Robust adaptive attitude tracking on SO(3) with an application to a quadrotor UAV. *Transactions on Control Systems Technology*, 21(5):1924–1930.

[65] Lee, T., Leok, M., and McClamroch, N. H. (2008). Optimal attitude control of a rigid body using geometrically exact computations on SO(3). *Journal of Dynamical and Control Systems*, 14(4):465–487.

[66] Lee, T., Leok, M., and McClamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on SE(3). In *Proceedings of the 49th Conference on Decision and Control*, pages 5420–5425, Atlanta, Georgia, USA.

[67] Li, G., Tsang, K. M., and Ho, S. L. (1998). A novel model-following scheme with simple structure for electrical position servo systems. *International Journal of Systems Science*, 29(9):959–969.

[68] Li, J. and Li, Y. (2011). Dynamic analysis and PID control for a quadrotor. In *Proceedings of the International Conference on Mechatronics and Automation*, pages 573–578, Beijing, China.

[69] Liew, C. F., DeLatte, D., Takeishi, N., and Yairi, T. (2017). Recent developments in aerial robotics: A survey and prototypes overview. *arXiv preprint arXiv:1711.10085*.

[70] Ligthart, J. A. J., Poksawat, P., and Wang, L. (2017). Experimentally validated model predictive controller for a hexacopter. *IFAC-PapersOnLine*, 50(1):4076–4081.

[71] Liu, H., Liu, D., and Zuo, Z. (2016). Robust attitude control for uncertain quadrotors with input time delays. In *Proceedings of the European Control Conference*, pages 2312–2315, Ålborg, Denmark.

[72] Liu, S., Mohta, K., Atanasov, N., and Kumar, V. (2018). Search-based motion planning for aggressive flight in SE(3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446.

[73] Maciejowski, J. M. (2002). *Predictive control with constraints*. Prentice Hall.

[74] Marthinsen, A. and Owren, B. (2001). A note on the construction of crouch-grossman methods. *BIT Numerical Mathematics*, 41(1):207–214.

[75] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2520–2525, Shanghai, China.

[76] Mellinger, D., Michael, N., and Kumar, V. (2012). Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674.

[77] Mercy, T., Van Parys, R., and Pipeleers, G. (2017). Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 26(6):1–8.

[78] Milam, M. B. (2003). *Real-time optimal trajectory generation for constrained dynamical systems*. PhD thesis, California Institute of Technology.

[79] Milam, M. B., Mushambi, K., and Murray, R. M. (2000). A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proceedings of the 39th Conference on Decision and Control*, pages 845–851, Sydney, Australia.

[80] Milam, M. B., Petit, N., and Murray, R. M. (2001). Constrained trajectory generation for micro-satellite formation flying. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pages 328–333, Montreal, Canada.

[81] Mokhtari, A. and Benallegue, A. (2004). Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2359–2366, New Orleans, Louisiana, USA.

[82] Moutinho, A., Mateos, E., and Cunha, F. (2015). The tilt-quadrotor: concept, modeling and identification. In *Proceedings of the International Conference on Autonomous Robot Systems and Competitions*, pages 156–161, Vila Real, Portugal.

[83] Mueller, M. W. and D'Andrea, R. (2013). A model predictive controller for quadrocopter state interception. In *Proceedings of the European Control Conference*, pages 1383–1389, Zürich, Switzerland.

[84] Mueller, M. W. and D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *Proceedings of the International Conference on Robotics and Automation*, pages 45–52, Hong Kong, China.

[85] Mueller, M. W., Hehn, M., and D'Andrea, R. (2015). A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, 31(6):1294–1310.

[86] Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D., and Hilliges, O. (2017a). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, 2(3):1696–1703.

[87] Nägeli, T., Meier, L., Domahidi, A., Alonso-Mora, J., and Hilliges, O. (2017b). Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics*, 36(4):1–10.

[88] Nguyen, N. T., Prodan, I., and Lefèvre, L. (2018). Flat trajectory design and tracking with saturation guarantees: a nano-drone application. *International Journal of Control*, pages 1–14.

[89] Omari, S., Hua, M.-D., Ducard, G., and Hamel, T. (2013). Nonlinear control of VTOL UAVs incorporating flapping dynamics. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2419–2425, Tokyo, Japan.

[90] Orsag, M. and Bogdan, S. (2012). Influence of forward and descent flight on quadrotor dynamics. In *Recent Advances in Aircraft Technology*.

[91] Parrot Drones (2015a). Parrot Bebop 2. https://www.parrot.com/fr/drones/parrot-bebop-2.

[92] Parrot Drones (2015b). Parrot Bluegrass, Drone pour l'analyse des cultures. https://www.parrot.com/solutions-business/parrot-professional/parrot-bluegrass.

[93] Parrot Drones (2018). Parrot ANAFI. https://www.parrot.com/fr/drones/anafi.

[94] Petit, N., Milam, M. B., and Murray, R. M. (2001). Inversion based constrained trajectory optimization. *IFAC Proceedings Volumes*, 34(6):1211–1216.

[95] Piegl, L. and Tiller, W. (2012). *The NURBS book*. Springer Science & Business Media.

[96] Pounds, P., Mahony, R., and Corke, P. (2010). Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699.

[97] Powers, C., Mellinger, D., Kushleyev, A., Kothmann, B., and Kumar, V. (2013). Influence of aerodynamics and proximity effects in quadrotor flight. In *Proceedings of the 13th International Symposium on Experimental Robotics*, pages 289–302, Québec City, Canada.

[98] Qingji, G., Fengfa, Y., and Dandan, H. (2014). Research of precision flight control for quadrotor UAV. In *Proceedings of the Chinese Guidance, Navigation and Control Conference*, pages 2369–2374, Yantai, China.

[99] Remondino, F., Barazzetti, L., Nex, F., Scaioni, M., and Sarazzi, D. (2011). UAV photogrammetry for mapping and 3D modeling–current status and future perspectives. *International archives of the photogrammetry, remote sensing and spatial information sciences*, 38(1):25–31.

[100] Reyes-Valeria, E., Enriquez-Caldera, R., Camacho-Lara, S., and Guichard, J. (2013). LQR control for a quadrotor using unit quaternions: Modeling and simulation. In *Proceedings of the 23rd International Conference on Electronics, Communications and Computing*, pages 172–178, Cholula, Puebla, Mexico.

[101] Richter, C., Bry, A., and Roy, N. (2013). Polynomial trajectory planning for quadrotor flight. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–8, Karlsruhe, Germany.

[102] Richter, C., Bry, A., and Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research: The 16th International Symposium of Robotics Research*, pages 649–666.

[103] Roberts, M. and Hanrahan, P. (2016). Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics*, 35(4):1–11.

[104] Rousseau, G. (2019). Introduction to Bézier and B-spline curves. Intervention in the massive open online course DroMOOC - A Massive Open Online Course on Drones and Aerial Multi Robot Systems - created by ONERA, CentraleSupélec and ENSTA.

[105] Rousseau, G., Stoica Maniu, C., Tebbani, S., and Babel, M. (2017). Impact of propellers inertia and asymmetries on a V-shaped quadrotor. In *Preprints of the 20th IFAC World Congress*, pages 10686–10690, Toulouse, France.

[106] Rousseau, G., Stoica Maniu, C., Tebbani, S., Babel, M., and Martin, N. (2018). Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control. In *Proceedings of the European Control Conference*, pages 2897–2903, Limassol, Cyprus.

[107] Rousseau, G., Stoica Maniu, C., Tebbani, S., Babel, M., and Martin, N. (2019). Minimum-time B-spline trajectories with corridor constraints. Application to cinematographic quadrotor flight plans. *Control Engineering Practice*, 89:190–203.

[108] Seddon, J. and Newman, S. (2001). *Basic helicopter aerodynamics*. American Institute of Aeronautics and Astronautics.

[109] Shetty, O. and Selig, M. (2011). Small-scale propellers operating in the vortex ring state. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pages 1254–1269, Orlando, Florida, USA.

[110] Simplício, P., Pavel, M. D., van Kampen, E.-J., and Chu, Q. (2013). An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. *Control Engineering Practice*, 21(8):1065–1077.

[111] Skoczowski, S. (2000). The robust control system with use of nominal model of controlled plant. *IFAC Symposium on Advanced Control of Chemical Processes*, 33(10):911–916.

[112] Skylogic Research (2018). 2018 drone market repport. Sector Report.

[113] Smeur, E. J. J., de Croon, G. C. H. E., and Chu, Q. (2016). Gust disturbance alleviation with incremental nonlinear dynamic inversion. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 5626–5631, Daejeon, South Korea.

[114] Steinbach, M. C. (1997). Optimal motion design using inverse dynamics.

[115] Stoican, F., Prodan, I., Popescu, D., and Ichim, L. (2017). Constrained trajectory generation for UAV systems using a B-spline parametrization. In *Proceedings of the 25th Mediterranean Conference on Control and Automation*, pages 613–618, Valletta, Malta.

[116] Svacha, J., Mohta, K., and Kumar, V. (2017). Improving quadrotor trajectory tracking by compensating for aerodynamic effects. In *Proceedings of the International Conference on Unmanned Aircraft Systems*, pages 860–866, Miami, Florida, USA.

[117] Szádeczky-Kardoss, E. and Kiss, B. (2006). Tracking error based on-line trajectory time scaling. In *Proceedings of the International Conference on Intelligent Engineering Systems*, pages 80–85, London, UK.

[118] Tadokoro, Y., Ibuki, T., and Sampei, M. (2017). Maneuverability analysis of a fully-actuated hexrotor UAV considering tilt angles and arrangement of rotors. *IFAC-PapersOnLine*, 50(1):8981–8986.

[119] Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a VTOL quadrotor aircraft. *Transactions on Control Systems Technology*, 14(3):562–571.

[120] Tomić, T., Lutz, P., Schmid, K., Mathers, A., and Haddadin, S. (2018). Simultaneous contact and aerodynamic force estimation (s-CAFE) for aerial robots. *arXiv preprint arXiv:1810.12908*.

[121] Torres-González, A., Capitán, J., Cunha, R., Ollero, A., and Mademlis, I. (2017). A multidrone approach for autonomous cinematography planning. In *Iberian Robotics Conference*, pages 337–349, Seville, Spain.

[122] Usenko, V., von Stumberg, L., Pangercic, A., and Cremers, D. (2017). Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 215–222, Vancouver, Canada.

[123] Van Loock, W., Pipeleers, G., and Swevers, J. (2015). B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction. *Mechanical Sciences*, 6(2):163–171.

[124] Van Parys, R. and Pipeleers, G. (2017). Spline-based motion planning in an obstructed 3D environment. *IFAC-PapersOnLine*, 50(1):8668–8673.

[125] Von Stryk, O. and Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373.

[126] Wen, J.-Y. and Kreutz-Delgado, K. (1991). The attitude control problem. *Transactions on Automatic control*, 36(10):1148–1162.

[127] Xie, K., Yang, H., Huang, S., Lischinski, D., Christie, M., Xu, K., Gong, M., Cohen-Or, D., and Huang, H. (2018). Creating and chaining camera moves for qadrotor videography. *ACM Transactions on Graphics*, 37(4):88.

[128] Yali, Y., Changhong, J., and Haiwei, W. (2010). Backstepping control of each channel for a quadrotor aerial robot. In *Proceedings of the International Conference on Computer, Mechatronics, Control and Electronic Engineering*, volume 3, pages 403–407, Changchin, China.

[129] Yuan, C.and Ghamry, K. A., Liu, Z., and Zhang, Y. (2016). Unmanned aerial vehicle based forest fire monitoring and detection using image processing technique. In *Proccedings of the Chinese Guidance, Navigation and Control Conference*, pages 1870–1875, Nanjing, China.

[130] Zachariadis, O., Mygdalis, V., Mademlis, I., Nikolaidis, N., and Pitas, I. (2017). 2D visual tracking for sports UAV cinematography applications. In *Global Conference on Signal and Information Processing*, pages 36–40, Montreal, Quebec, Canada.

[131] Zhang, W., Mueller, M. W., and D'Andrea, R. (2016). A controllable flying vehicle with a single moving part. In *Proceedings of the International Conference on Robotics and Automation*, pages 3275–3281, Stockholm, Sweden.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Trajectoires optimales et commande prédictive d'un quadricoptère pour la réalisation de plans de vol cinématographiques

Mots clés : Quadricoptère, Génération de trajectoire, Cinématographie, B-spline, UAV

Résumé : Cette thèse s'intéresse à la réalisation autonome de plans de vol cinématographiques par un quadrotor équipé d'une caméra. Ces plans de vol consistent en une série de points de passage à rejoindre successivement, en adoptant diverses méthodes de prise de vue et en respectant des références de vitesse ainsi que des couloirs de vols. Une étude approfondie de la dynamique du quadrotor est tout d'abord proposée, et utilisée pour construire un modèle linéarisé du drone autour de l'équilibre de vol stationnaire. L'analyse de ce modèle linéaire permet de mettre en évidence l'impact de l'inertie des rotors du drone dans sa dynamique, notamment l'apparition d'un comportement à non minimum de phase en roulis ou tangage, lorsque les moteurs sont inclinés. Dans un second temps, deux algorithmes de génération de trajectoires lisses, faisables et adaptées à la cinématographie sont proposés. La faisabilité de la trajectoire est garantie par le respect de contraintes sur ses dérivées temporelles, adaptées pour la cinématographie et obtenues grâce à l'étude du modèle non linéaire du drone. Le premier repose sur une optimisation bi-niveaux d'une trajectoire polynomiale par morceaux, dans le but de trouver la plus rapide des trajectoires à minimum de jerk permettant d'accomplir la mission. Le second algorithme consiste en la génération de trajectoires B-spline non-uniformes à durée minimale. Pour les deux solutions, une étude de l'initialisation du problème d'optimisation est présentée, de même qu'une analyse de leurs avantages et limitations. Pour ce faire, elles sont notamment confrontées à des simulations et vols extérieurs. Enfin, une loi de commande prédictive est proposée pour asservir les mouvements de la caméra embarquée de manière douce mais précise.

Title : Optimal trajectory planning and predictive control for cinematographic flight plans with quadrotors

Keywords : Quadrotor, trajectory generation, cinematography, B-spline, UAV

Abstract : This thesis deals with the autonomous performance of cinematographic flight plans with camera equipped quadrotors. These flight plans consists in a series of waypoint to join while adopting various camera behaviors, along with speed references and flight corridors. First, an in depth study of the nonlinear dynamics of the drone is proposed, which is then used to derive a linear model of the system near the hovering equilibrium. An analysis of this linear model allows us to emphasize the impact of the inertia of the propellers when the latter are tilted, such as the apparition of a nonminimum phase behavior of the pitch or dynamics. Then, two algorithms are proposed to generate smooth and feasible trajectories suited for cinematography. The feasibility of the trajectory is ensured by constraints on its time derivatives, suited for cinematography and obtained with the use of the nonlinear model of the drone. The first algorithm proposed in this work is based on a bi-level optimization of a piecewise polynomial trajectory and try to find the fastest feasible minimum jerk trajectory to perform the flight plan. The second algorithm consists in the generation of feasible, minimum time, nonuniform B-spline. For both solutions, a study of the initialization of the optimization problem is proposed, as well as a discussion about their advantages and limitations. To this aim, they are notably confronted to simulations and outdoor flight experiments. Finally, a predictive control law is propose to smoothly but accurately control the on-board camera.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France