Receding horizon flight control for trajectory tracking of autonomous aerial vehicles

Ionela Prodan, Sorin Olaru, Ricardo Bencatel, Joao Borges de Sousa, Cristina Stoica, Silviu-Iulian Niculescu

Receding Horizon Flight Control for Trajectory Tracking of Autonomous Aerial Vehicles[☆]

Ionela Prodan[a,b], Sorin Olaru[a], Ricardo Bencatel[c], João Borges de Sousa[c], Cristina Stoica[a], Silviu-Iulian Niculescu[b]

[a] *SUPELEC Systems Sciences (E3S) - Automatic Control Department, 3 rue Joliot Curie, 91192, Gif sur Yvette, France, {ionela.prodan,sorin.olaru,cristina.stoica}@supelec.fr*

[b] *Laboratory of Signals and Systems, SUPELEC - CNRS, 3 rue Joliot Curie, 91192, Gif sur Yvette, France {ionela.prodan,silviu.niculescu}@lss.supelec.fr*

[c] *Department of Electrical and Computer Engineering, School of Engineering, University of Porto, 4200, Porto, Portugal {ricardo.bencatel,joao.sousa}@fe.up.pt*

Abstract

This paper addresses the implementation of a predictive control strategy for Unmanned Air Vehicles in the presence of bounded disturbances. The goal is to prove the feasibility of such a real-time optimization-based control design and to demonstrate its tracking capabilities for the nonlinear dynamics with respect to a reference trajectory which is pre-specified via differential flatness. In order to benefit from the computational advantages of the linear predictive control formulations, an off-line linearization strategy of the nonlinear model of the vehicle along the flat trajectory is employed. The proposed method exhibits effective performance validated through software-in-the-loop simulations and real flight tests on different Unmanned Aerial Vehicles (UAVs).

Keywords: Unmanned Aerial Vehicles (UAVs), differential flatness, constrained Model Predictive Control (MPC)

1. Introduction

Unmanned Aerial Vehicles (UAVs) are flying devices that have received increasing attention in recent years, particularly after the tragic events of September 11, 2001, when the world has questioned the safety of the onboard aircrafts (Valavanis, 2007).

[☆]The research of Ionela Prodan is financially supported by the EADS Corporate Foundation (091-AO09-1006).

Currently, most UAVs are used for military mission planning and for combat support (Valenti et al., 2007), border surveillance (Hoffmann et al., 2011) and search for survivors (Geyer, 2008). Also, the range of possible civilian applications for UAVs is expending to traffic monitoring (Rosenbaum et al., 2009), pollution measurement (Muttin, 2011) or hazard zone inspection (Patterson et al., 2011). Many of the enabling technologies developed for military UAVs are becoming similar or identical to those required for civil UAVs (Zheng et al., 2013).

One of the main research challenges is to improve and increase their autonomy, in particular to empower the application of advanced control design methods. Their aim is to consider the limitations of the vehicle dynamics and to permit the reconfiguration of the vehicle trajectory in case of unexpected events occurring in the system. The objective of the present paper is to describe and implement a control strategy with a combined use of Model Predictive Control (MPC) and flatness concepts. This represents a challenging methodological combination allowing to handle the feedback control and the trajectory generation while cooping with the robustness issues upon set-theoretic methods. Moreover, to the best of the authors' knowledge, such a strategy has not been evaluated in flight tests on actual vehicles. In this sense, the results presented in the present paper can be considered as a validity proof for the proposed control approach for a real UAV system, all by pointing to the sensitive open problems.

There exist various techniques for the real-time control of UAVs. Usually low-complexity (and robust) control methods are preferred such as, sliding mode control (Bencatel et al., 2011), dynamic programming approach for motion control of autonomous vehicles (Silva et al., 2007) or control strategies that first estimate the speed and heading of the vehicles and then use simple feedback control laws for stabilizing different measurement error (Soares et al., 2012). In comparison to these solutions we will show that the presented MPC approach, meets the real-time computational constraints even if it presents a relatively important real-time computational load.

The present results follow a recent trend in the control literature where real-time predictive control strategies are applied to vehicles maneuvering problems. For example, in (Keviczky and Balas, 2006), the authors use a predictive guidance controller for an autonomous UAV and a fault detection filter for taking into account the disturbances. Mixed-Integer Programming (MIP) techniques combined with receding horizon strategy was useful to efficiently coordinate the interaction of multiple UAVs in scenarios with many sequential tasks and tight timing constraints (How et al., 2004; Schouwenaars et al., 2005). Furthermore, some other works investigate the capability of Nonlinear MPC for tracking control. Among these contributions, (Kim et al.,

2002) formulates a nonlinear MPC algorithm combined with the gradient-descent method for trajectory tracking, and (Fontes et al., 2009) proposes a two-layer control scheme composed by a nonlinear and a linear predictive controller for a group of nonholonomic vehicles moving in formation. The computational load highlights the importance of developing simpler real-time optimization problems embedded within predictive control formulations for plants described by nonlinear models. In this sense, the authors of (Falcone et al., 2007) consider a MPC tracking controller based on successive *on-line* linearizations of the nonlinear model of the corresponding plant. Yet, the computational complexity of the proposed MPC scheme remains significant. The approach advocated in the present paper follows a similar linearization principle from the prediction model point of view, but avoids its real-time computation by the use of a precomputed Voronoi diagram of the linearized models. This novelty is used in conjunction with an efficient trajectory generation mechanism in order to reduce the real-time computations. The propose method can be considered to be part of the larger class of parameter-varying MPC techniques. However, its particularity is that it reduces to a simple positioning mechanism in the Voronoi partition with a linear MPC for the local prediction model. This proves to be a computationally attractive alternative to the LMI-based MPC for nonlinear or time-varying dynamics (Kothare et al., 1997; Lakshmanan et al., 1999; Falugi at al., 2010) which are impractical for the real-time control of UAV systems.

From the trajectory point of view, it is important to point out that the class of nonlinear systems used in a wide range of UAV applications are differentially flat (Fliess et al., 1995; Lévine, 2009). Flatness plays an important role in the control of such systems, with practical design algorithms for motion planning, trajectory generation, and stabilization (Rouchon et al., 2003). Among the applications, (Hao and Agrawal, 2005) proposes a combination between differential flatness and a graph search method for the on-line planning and control of multiple ground mobile robots with trailers moving in groups. Also, the authors in (Van Nieuwstadt and Murray, 1998) apply a real-time trajectory generation algorithm based on flatness and receding horizon without constraints to a thrust vectored flight experiment (the Caltech ducted fan). Finally, the authors in (De Doná et al., 2009) develop a receding horizon-based trajectory generator methodology for generating a reference trajectory parameterized by splines, with the property that it satisfies performance objectives.

The paper proceeds as follows. Section 2 focuses on the description of the control system design, the aircraft model used to test the control system and the implementation of the controller intended to run in real-time in an autonomous system. In Section 3 a specified trajectory is generated for a vehicle using the *differential flatness formalism*. The proposed trajectory generation mechanism takes into account

waypoint conditions and furthermore, allows us to obtain in Section 4 *off-line* linearizations of the nonlinear vehicle model along the flat trajectory. Since the reference trajectory is available beforehand, a *real-time optimization problem* which *minimizes* the *tracking error* for the vehicle is solved in Section 5 based on a prediction of the future evolution of the system, following the model-based control principles (Goodwin et al., 2005). Furthermore, Section 6 describes the UAVs testbed hardware and software architecture. Also, the validation procedure of the proposed method tests the performances by software-in-the-loop simulations and subsequently with field tests of an offboard controller of an autonomous UAV. In this section we are also addressing all the issues concerned with the delays in communication and/or the connectivity between the Ground Station and the autonomous flight control computer. Finally, Section 7 completes the paper with concluding remarks and improvement directions.

The following notations will be used throughout the paper. A Voronoi region, $\mathcal{V}_\mathrm{i}$ associated to a collection of points p_i is defined as $\mathcal{V}_\mathrm{i} = \{x \in \mathbb{R}^n : d(x, p_\mathrm{i}) \leq d(x, p_\mathrm{r}), \forall i \neq r\}$, where $d(x, y)$ denotes the distance between the points x and y. Minkowski's addition of two sets $\mathcal{X}$ and $\mathcal{Y}$ is defined as $\mathcal{X} \oplus \mathcal{Y} = \left\{x + y : x \in \mathcal{X},\ y \in \mathcal{Y}\right\}$. Let $x(k+1|k)$ denote the value of x at time instant $k+1$, predicted upon the information available at time $k \in \mathbb{N}$. We write $R \succ (\succeq) 0$ to denote that R is a positive (semi)definite matrix. Let $s_{\max}$ denote the maximum number of steps for which an optimization problem is solved.

2. UAV model in view of control design

From the high level control point of view, the airplane model can be considered to be part of the class of nonholonomic systems. It is completely controllable, but it cannot make instantaneous turns in certain directions. This means that the vehicle state depends on the path executed until the current moment. Further, the feasible path depends on the aircraft's current state, as forces cannot be imposed in all directions concomitantly. The two main control forces of the aircraft are the lift and the thrust. Thrust magnitude is controlled directly through the motor power. Lift magnitude is controlled indirectly through the aircraft angle of attack and the aircraft airspeed.

The framework proposed in the present paper can be adapted to vehicles moving in 2D or 3D. Subsequently, the airplane can be represented by the following simplified kinematic models[1] with different Degrees-of-Freedom (DOF) (Bencatel et al., 2011):

[1] In the present paper, we adopt a kinematic model for the airplane which is not a simple double integrator. Instead, the model is written in terms of the vehicle's speed, heading and the bank.

2D model with 3-DOF

$$\begin{aligned}\dot{x}(t) &= V_{\mathrm{a}}(t)\cos\Psi(t) + W_{\mathrm{x}},\\ \dot{y}(t) &= V_{\mathrm{a}}(t)\sin\Psi(t) + W_{\mathrm{y}},\\ \dot{\Psi}(t) &= \frac{g\ \tan\Phi(t)}{V_{\mathrm{a}}(t)},\end{aligned} \tag{1}$$

3D model with 4-DOF

$$\begin{aligned}\dot{x}(t) &= V_{\mathrm{a}}(t)\cos\Psi(t) + W_{\mathrm{x}},\\ \dot{y}(t) &= V_{\mathrm{a}}(t)\sin\Psi(t) + W_{\mathrm{y}},\\ \dot{\Psi}(t) &= \frac{g\ \tan\Phi(t)}{V_{\mathrm{a}}(t)},\\ \dot{h}(t) &= \dot{h}_{\mathrm{c}}(t).\end{aligned} \tag{2}$$

In the present work we explore the case of a 2D 3-DOF model (1) of an airplane in which the autopilot forces coordinated turns (zero side-slip) at a fixed altitude. The state variables are represented by the position $(x(t), y(t))$ and the heading (yaw) angle $\Psi(t) \in [0, 2\pi]$ rad. The input signals are the airspeed velocity $V_{\mathrm{a}}(t)$ and the bank (roll) angle $\Phi(t)$, respectively. Also, the airspeed and the bank angle are regarded as the autopilot pseudo-controls. Furthermore, we assume a small angle of attack and that the autopilot provides a higher bandwidth regulator for the bank angle, making its dynamics negligible when compared to the heading dynamics. W_{x} and W_{y} are the wind velocity components on the x and y axis. Notice that the 3D 4-DOF model is the same as the 2D 3-DOF model of the airplane, except for the altitude variation $h(t)$. In this case, besides the airspeed and the bank angle, the vertical rate $\dot{h}_{\mathrm{c}}(t)$ is regarded also as the autopilot pseudo-control.

The main objective is the design of a predictive control strategy for system (1). In the view of this objective, a reference trajectory needs to be available beforehand at least for a finite prediction window. Therefore, in the following, we use flatness concepts (Fliess et al., 1995) in order to provide flat reference states and inputs for the nonlinear system (1) at the pre-design stage (trajectory generation), which can be updated in real-time in order to allow trajectory tracking and disturbance rejection.

3. Flat trajectory generation

Consider the compact notations:

$$\xi(t) = \begin{bmatrix} x^{\mathrm{T}}(t) & y^{\mathrm{T}}(t) & \Psi^{\mathrm{T}}(t)\end{bmatrix}^{\mathrm{T}}, \tag{3}$$

$$u(t) = \begin{bmatrix} V_{\mathrm{a}}^{\mathrm{T}}(t) & \Phi^{\mathrm{T}}(t)\end{bmatrix}^{\mathrm{T}}, \tag{4}$$

denoting the state vector and the input vector, respectively. Then, the general system (1) can be described as:

$$\dot{\xi}(t) = f(\xi(t), u(t)), \tag{5}$$

where $f(\cdot,\cdot) : \mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^3$ is the dynamical system.

In the following, we require the determination of a reference trajectory ($\xi^{\text{ref}}(t)$, $u^{\text{ref}}(t)$) that steers the model (1) from an initial state $\xi^{\text{ref}}(t_0)$ to a final state $\xi^{\text{ref}}(t_{\text{f}})$, over a fixed time interval $[t_0, t_{\text{f}}]$. Since the aforementioned system is controllable and allows the existence of flat outputs (see, for instance (Van Nieuwstadt and Murray, 1998; De Doná et al., 2009)), we employ flatness properties in order to construct the required reference trajectory.

The systems' state and input will be represented as functions of a finite dimensional mapping $z(t)$ and a finite number of its derivatives (in this particular case it will be shown that the second order derivative suffices):

$$\begin{aligned} \xi^{\text{ref}}(t) &= \eta_0(z(t), \dot{z}(t)), \\ u^{\text{ref}}(t) &= \eta_1(z(t), \dot{z}(t), \ddot{z}(t)). \end{aligned} \tag{6}$$

In the case of the dynamics (1), the vector $z(t) = \begin{bmatrix} z_1(t) & z_2(t) \end{bmatrix}^{\text{T}} \in \mathbb{R}^2$, called the flat output is defined as:

$$\begin{aligned} z_1(t) &= x(t), \\ z_2(t) &= y(t). \end{aligned} \tag{7}$$

It can be shown that, the corresponding reference state and input for the system (5) are obtained by replacing the reference flat output (7) into equations (6):

$$\xi^{\text{ref}}(t) = \begin{bmatrix} z_1(t) & z_2(t) & \arctan\left(\frac{\dot{z}_2(t)}{\dot{z}_1(t)}\right) \end{bmatrix}^{\text{T}}, \tag{8}$$

$$u^{\text{ref}}(t) = \begin{bmatrix} \sqrt{\dot{z}_1^2(t) + \dot{z}_2^2(t)} & \arctan\left(\frac{1}{g}\frac{\ddot{z}_2(t)\dot{z}_1(t) - \dot{z}_2(t)\ddot{z}_1(t)}{\sqrt{\dot{z}_1^2(t)+\dot{z}_2^2(t)}}\right) \end{bmatrix}^{T}, \tag{9}$$

where $t \in [t_0, t_{\text{f}}]$.

It is important to point out that the obtained trajectories (8)–(9) are consistent with the system dynamic equation (1). Practically, any trajectory for $z(t)$ in (7) that

satisfies the boundary conditions:

$$\begin{aligned}
\xi^{\mathrm{ref}}(t_0) &= \eta_0(z(t_0), \dot{z}(t_0)) = \xi(t_0), \\
u^{\mathrm{ref}}(t_0) &= \eta_1(z(t_0), \dot{z}(t_0), \ddot{z}(t_0)) = u(t_0), \\
\xi^{\mathrm{ref}}(t_{\mathrm{f}}) &= \eta_0(z(t_{\mathrm{f}}), \dot{z}(t_{\mathrm{f}})) = \xi(t_{\mathrm{f}}), \\
u^{\mathrm{ref}}(t_{\mathrm{f}}) &= \eta_1(z(t_{\mathrm{f}}), \dot{z}(t_{\mathrm{f}}), \ddot{z}(t_{\mathrm{f}})) = u(t_{\mathrm{f}}),
\end{aligned} \tag{10}$$

will enhance references (8)–(9) as feasible trajectory for system (1), while in the same time verifying the given initial and final conditions.

However, a significant shortcoming of the flatness construction is that constraints on inputs and/or state cannot be introduced easily in the trajectory generation. That is, we can impose in certain points values for state/inputs but we cannot control what happens "in-between" these points. In the forthcoming sections this will prove to be an important aspect towards the constraints validation within the predictive control setup. Therefore, it becomes important to bring the trajectory into admissible limits from its reference generation. As a solution, we propose a simplified model for the generation of the trajectory, by considering $V_{\mathrm{a}}(t)$ in (4) to be constant (chosen a priori) and let the bank angle $\phi(t)$ as the single controllable input.

This restriction leads to a simplified form of the corresponding reference signals:

$$\xi^{\mathrm{ref}}(t) = \begin{bmatrix} z_1(t) & z_2(t) & \arctan\left(\frac{\dot{z}_2(t)}{\dot{z}_1(t)}\right) \end{bmatrix}^{\mathrm{T}}, \tag{11}$$

$$u^{\mathrm{ref}}(t) = \begin{bmatrix} V_{\mathrm{a}} & \arctan\left(\frac{V_{\mathrm{a}}}{g}\frac{\ddot{z}_2(t)\dot{z}_1(t) - \dot{z}_2(t)\ddot{z}_1(t)}{\dot{z}_1^2(t) + \dot{z}_2^2(t)}\right) \end{bmatrix}^{\mathrm{T}}, \tag{12}$$

where $t \in [t_0, t_{\mathrm{f}}]$ and V_{a} denotes the constant velocity.

Remark 1. Note that this simplification (i.e., $V_{\mathrm{a}}(t)$ constant) provides yet another model which has to be dealt with. The resulting flat trajectory is still feasible for the 3-DOF model (1) since any trajectory of this restricted model is also a trajectory of the original one. Subsequently, under the hypothesis of a constant velocity, the obtained reference trajectories can be considered as being parameterized by the value of V_a. □

For a practical implementation, the flat output signal $z(t)$ is seen as a weighed sum of functions in a predefined basis. Imposing boundary constraints (or more generic waypoints) for the evolution of the differentially flat systems (see, for instance (De Doná et al., 2009)) a flat output $z(t)$ can be generated by the resolution of a *linear* system of equalities. Exploiting these principles, our approach is to further

introduce a set of waypoints through which the vehicle must pass[2] in the interval $[t_0, t_f]$:

$$\mathbb{P} \triangleq \{p^i = (\xi^i, u^i), \quad i = 0, \ldots, N_w\}, \tag{13}$$

where N_w is the number of chosen waypoints.

The list of waypoints is assumed as being provided by an operator (which can oversee the operation) and incorporates control requirements: obstacle avoidance, check points, etc. Note that producing waypoints in the reference trajectory generation is coherent with the existing software-hardware configuration which used waypoints in the communication protocol (see the description of the hardware and software architecture in Section 6.1).

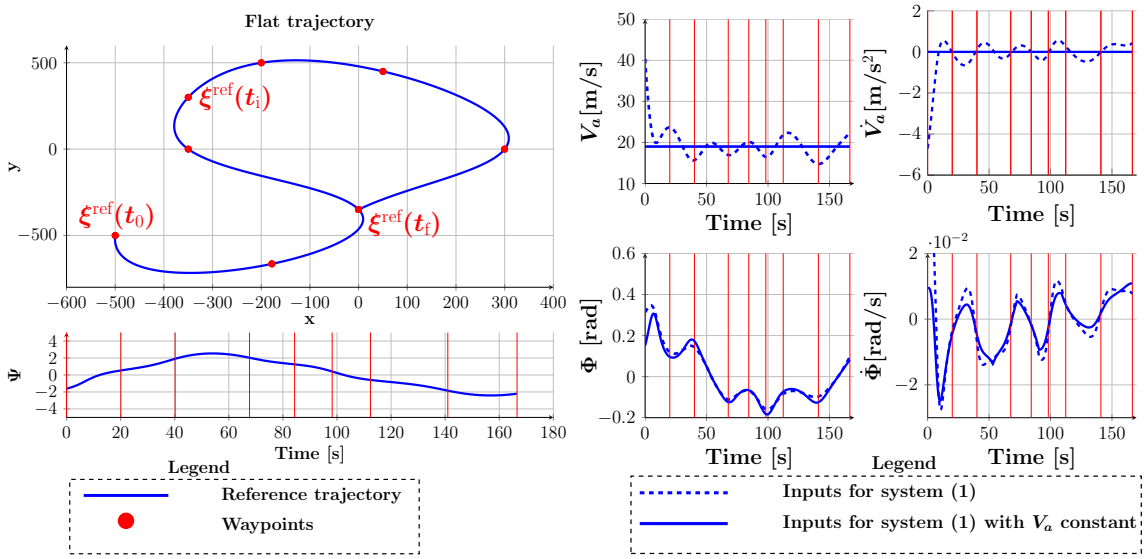
It is important to point out that polynomial basis functions are a poor choice because their dimension (degree) depends on the number of constraints imposed upon the inputs, states and their derivatives. This means that they are sensitive to the number of waypoints, which can lead to an increased numerical sensitivity when t_f grows. More precisely, in this case the trajectory needs to be computed on segments (i.e., each segment taken between two consecutive waypoints). Therefore, additional equality constraints are imposed on the flat output, leading to an increased number of polynomial basis functions beyond reasonable computation limits. For example, in our case, we need to go to higher order monomials in order to guarantee that there exists a weighted combination which satisfies all the equality constraints.

To overcome these issues, B-spline functions (De Doná et al., 2009; Suryawan et al., 2010) have been used. This is (arguably) one of the best choice in the sense that their degree does not depend on the number of waypoints. Actually, the degree depends only on the rank of derivative for which continuity need to be ensured, this being in contrast with the goal of a *simple* polynomial basis. In our particular case, the third degree order of the B-splines suffices to assure smooth bank and velocity control inputs. Practically the third order degree suffices also by the fact that the bank control is implicitly defined by the first and second derivatives (see equation (9)). Finally, this implies that the bank control is smooth only if the second and third order derivatives of the flat output are continuous.

Illustrative example for the generation of a flat trajectory

In order to illustrate the proposed flat trajectory mechanism let us consider the nonlinear model (1) and $N_w = 9$ waypoints as in (13) through which the trajectory

[2]Hereafter whenever we use the subscript we refer to time and when we use the superscript we index a point from a set of points.



(a) (b)

Figure 1: Flat trajectory which passes through a priori given waypoints.

must pass: $\mathbb{P} = \{(-500, -500, 150), (-180, -650, 150), (0, -350, 150), (-350, 0, 150), (-350, 300, 150), (-200, 500, 150), (50, 450, 150), (300, 0, 150), (0, -350, 150)\}$.

Figure 1(a) illustrates the flat trajectory in blue obtained by letting both, V_{a} to vary as well as constant (i.e., $V_{\mathrm{a}} = 19$ m/s). The difference between the two trajectories is only represented by the variation of the control inputs which, as it can be observed in Figure 1(b) in the case when V_{a} is constant (represented in solid blue) the variations of both the bank control input and its derivative are smother.

Once the reference trajectory is available, one can concentrate on the real-time control aspects. The predictive control needs to be able to handle the discretized and linearized model of the vehicle along a reference trajectory. In Section 4 the proposed linearization strategy of the nonlinear system (1) is described and further in Section 5 the MPC design is discussed.

4. Linearization of the UAV model

For computation purposes, it is convenient to use the discretized model of the nonlinear system (5):

$$\xi(k+1) = f^{\mathrm{d}}(\xi(k), u(k)). \tag{14}$$

The Euler explicit method is further used for the time discretization of system (5), where the state of the system at a later time is precalculated based on from the state of the system at the current time:

$$\xi(k+1) = \xi(k) + h \cdot f(\xi(t), u(t))|_{t=k\cdot h}, \tag{15}$$

with h is the discretization step. Even if very simple, the Euler method proved to be adequate. Of course, more complicated implementations can be adopted, in particular, it would be interesting to have a discretization which is adapted to a variable discretization step (as it is the case for the communication delays we experienced in the practical experiments).

In the sequel, we consider the linearization problem of the nonlinear discretized system (14). We take here a piece-wise affine (PWA) approach[3], that is, we consider a collection of points along the reference trajectory in which we pre-compute linear approximations of (14):

$$\mathbb{L} \triangleq \{l^{\mathrm{j}} = (\xi^{\mathrm{j}}, u^{\mathrm{j}}), \quad j = 0, \ldots, N_{\mathrm{l}}\}, \tag{16}$$

with N_{l} the number of chosen linearization points.

For a given point $l^{\mathrm{j}} \in \mathbb{L}$ we consider the following Taylor decomposition:

$$f^{\mathrm{d}}(\xi(k), u(k)) = f^{\mathrm{d}}(\xi^{\mathrm{j}}, u^{\mathrm{j}}) + A_{\mathrm{j}}(\xi(k) - \xi^{\mathrm{j}}) + B_{\mathrm{j}}(u(k) - u^{\mathrm{j}}) + \beta_{\mathrm{j}}(\xi(k), u(k)), \tag{17}$$

where the matrices $A_{\mathrm{j}} \in \mathbb{R}^{3\times 3}$ and $B_{\mathrm{j}} \in \mathbb{R}^{3\times 2}$ are defined as

$$A_{\mathrm{j}} = \frac{\partial f^{\mathrm{d}}}{\partial \xi}|_{\xi^{\mathrm{j}}, u^{\mathrm{j}}}, \qquad B_{\mathrm{j}} = \frac{\partial f^{\mathrm{d}}}{\partial u}|_{\xi^{\mathrm{j}}, u^{\mathrm{j}}} \tag{18}$$

and $\beta_{\mathrm{j}}(\xi_{\mathrm{k}}, u_{\mathrm{k}}) \in \mathbb{R}^3$ represents the terms of the Taylor decomposition of rank greater than 1 (i.e., the nonlinear residue of the linearization):

$$\beta_{\mathrm{j}}(\xi(k), u(k)) = f^{\mathrm{d}}(\xi(k), u(k)) - f^{\mathrm{d}}(\xi^{\mathrm{j}}, u^{\mathrm{j}}) - A_{\mathrm{j}}(\xi(k) - \xi^{\mathrm{j}}) - B_{\mathrm{j}}(u(k) - u^{\mathrm{j}}), \tag{19}$$

for all $j = 0, \ldots, N_{\mathrm{l}}$. Therefore, the system (14) can be linearized in $l^{\mathrm{j}} \in \mathbb{L}$ by the

[3]The PWA approach has received and extensive interest in the literature, representing a powerful technique for approximating the nonlinear systems and furthermore, proving their equivalence to other classes of hybrid systems. For a wide extent on the subject, the interested reader is referred to the work of (Sontag, 1981; Heemels et al., 2001; Ulbig et al., 2010) and the references therein.

following dynamics:

$$\xi(k+1) = f_{\mathrm{j}}^{\mathrm{d}}(\xi(k), u(k)) \triangleq A_{\mathrm{j}}\xi(k) + B_{\mathrm{j}}u(k) + r_{\mathrm{j}}, \tag{20}$$

with the affine constant terms $r_{\mathrm{j}} \in \mathbb{R}^3$ defined as:

$$r_{\mathrm{j}} = f^{\mathrm{d}}(\xi^{\mathrm{j}}, u^{\mathrm{j}}) - A_{\mathrm{j}}\xi^{\mathrm{j}} - B_{\mathrm{j}}u^{\mathrm{j}}, \tag{21}$$

for all $j = 0, \ldots, N_{\mathrm{l}}$.

In the following we consider a procedure of selecting between the predefined linearization points (16) for the current input/state values. To this end, we partition the state-space into a collection of *Voronoi cells*:

$$\mathcal{V}_{\mathrm{j}} = \left\{\xi : ||\xi - \xi^{\mathrm{j}}|| \leq ||\xi - \xi^{\mathrm{r}}||, \ \forall r \neq j\right\}, \tag{22}$$

where each cell consists of all points whose linearization error is lower with respect to linearization around the point ξ^{j} than with respect to any other point ξ^{r} from $\mathbb{L}$, with $r, j = 0, \ldots, N_{\mathrm{l}}$, $r \neq j$. This allows a practical criterion for the selection of the linearization point during runtime:

$$\text{if} \quad (\xi, u) \in V_{\mathrm{j}} \quad \text{then} \quad \xi(k+1) = f_{\mathrm{j}}^{\mathrm{d}}(\xi(k), u(k)), \quad \forall (\xi(k), u(k)) \in \mathcal{V}_{\mathrm{j}}. \tag{23}$$

It is worth mentioning that the Voronoi decomposition is unique (by its geometrical properties) and, as such, it offers a generic design tool for any disposition of the linearization points. The drawback is that this criterion is purely geometric and does not take into account the dynamical properties of the model. This disadvantage can be mitigated by two practical procedures:

- the increase of the number of linearization points;
- the computation of the maximal linearization error.

We point to (Fagiano et al., 2009) for a discussion on the accuracy of the linearization and the correspondence with a stabilizing control law.

Since $\beta_{\mathrm{j}}(\xi(k), u(k)) = f^{\mathrm{d}}(\xi, u) - f_{\mathrm{j}}^{\mathrm{d}}(\xi, u)$ it follows that the linearization error is related to the topology of its corresponding cell, $\mathcal{V}_{\mathrm{j}}$:

$$||\beta_{\mathrm{j}}(\xi(k), u(k))|| \leq \max_{(\xi,u)\in\mathcal{V}_{\mathrm{j}}} ||f^{\mathrm{d}}(\xi, u) - f_{\mathrm{j}}^{\mathrm{d}}(\xi, u)||. \tag{24}$$

Basically, a Voronoi decomposition with decreasing volume of the cells leads to an increasing quality of the PWA approximation for the function (14).

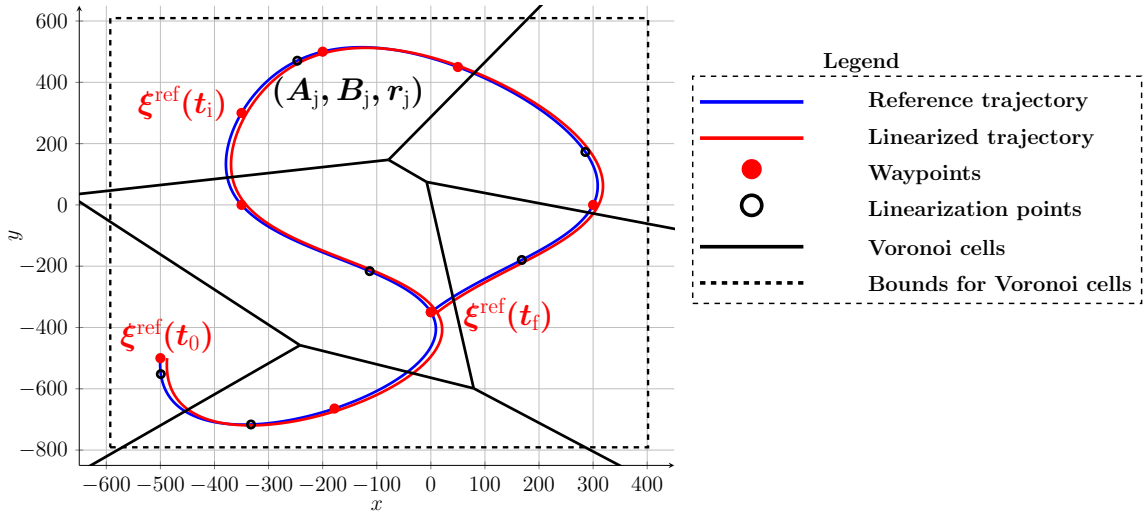


Figure 2: Real and linearized trajectories and the bounded Voronoi cells.

Remark 2. An a priori computation of the linearization (18), (19) and (21) in all feasible combinations of inputs and states is difficult to handle. As such, we prefer to select the linearization points (16) along the flat trajectory under the assumption (to be verified along the system functioning) that the real trajectory will stay in the corresponding validity domain (Voronoi cell) and thus, the chosen linearization points will remain relevant to the problem at hand. □

Illustrative example for the linearization mechanism

In order to better explain the linearization strategy, we continue with the previous example and illustrate in Figure 2 the continuous trajectory of the nonlinear system (5) (in blue) and the piecewise linearized trajectory (in red). Therefore, the linear system (20) describes dynamics of the deviations of the real nonlinear system trajectories (14) from the reference state trajectory $\xi^{\mathrm{ref}}(t)$ described by (8) at the application of an input reference signal $u^{\mathrm{ref}}(t)$ in the form (9). Several linearization points (denoted as black dots)[4] have been considered for the construction of the Voronoi cells according to relationship (22).

By linearizing the reference trajectory one can dispose of the necessary modeling

[4]Note that the waypoints (13) can also be considered between the linearization points (16). Moreover, the linearization points and the waypoints in the trajectory generation do not need to be correlated.

framework for the feedback control part of the trajectory tracking. This problem becomes the central objective for the remaining part of the paper and will be detailed in the forthcoming sections.

5. Trajectory tracking control problem

Since the reference trajectory is available beforehand (through the use of flatness procedures described in Section 3), an optimization problem, which includes the minimization[5] of the vehicle tracking error, can be formulated and included in a predictive control framework. Practically, the vehicle will be controlled in real-time to follow the reference trajectory using the available information over a finite time horizon in the presence of constraints.

For the implementation we consider the recursive construction of an optimal open-loop control sequence $\mathbf{u} = \{u(k|k), u(k+1|k), \cdots, u(k+N_\mathrm{p}-1|k)\}$ over a *finite* constrained receding horizon, which leads to a feedback control policy by the effective application of the first control action as system input:

$$u^* = \arg\min_{\mathbf{u}} \sum_{s=0}^{N_\mathrm{p}-1} (||\xi(k+s|k) - \xi^{\mathrm{ref}}(k+s|k)||_Q + ||u(k+s|k) - u^{\mathrm{ref}}(k+s|k)||_R + \\ + ||\Delta u(k+s|k)||_{R_\Delta}), \tag{25}$$

subject to the set of constraints:

$$\begin{cases} \xi(k+s+1|k) = A(k+s|k)\xi(k+s|k) + B(k+s|k)u(k+s|k) + r_\mathrm{j}, \\ \Delta u(k+s|k) = u(k+s|k) - u(k+s-1|k), \\ \xi(k+s|k) \in \mathcal{X},\ s = 1, \ldots, N_\mathrm{p} - 1, \\ u(k+s|k) \in \mathcal{U},\ s = 1, \ldots, N_\mathrm{p} - 1, \\ \Delta u(k+s|k) \in \mathcal{U}_\Delta,\ s = 1, \ldots, N_\mathrm{p} - 1, \end{cases} \tag{26}$$

with $A(k+s|k) = A_\mathrm{j}$ and $B(k+s|k) = B_\mathrm{j}$, $j = 1, \ldots, N_\mathrm{l}$ being the index selected such that $(\xi(k+s|k), u(k+s|k)) \in \mathcal{V}_\mathrm{j}$ as defined in (22). Here $Q = Q^\mathrm{T} \succeq 0$, $R \succ 0$, $R_\Delta \succ 0$ are weighting matrices and N_p denotes the length of the prediction horizon.

[5]The nominal trajectory is conceived to respect state and input constraints, but the real vehicle state may not follow exactly the reference trajectory, although it is desirable to remain as close as possible to it.

The solution of the optimization problem (25) needs to satisfy the dynamical constraints, expressed by the equality constraints in (26). In the same time, other security or performance specifications can be added to the system trajectory. These physical limitations (velocity and bank control inputs) are stated in terms of hard constraints on the internal state variables and input control action as detailed by the set constraints in (26). Practically, the sets $\mathcal{X}$ and $\mathcal{U}$ denote in a compact formulation the magnitude constraints on states and inputs, respectively. The set $\mathcal{U}_\Delta$ describes the constraints on the variations of the input control signals. In the following, all these sets are supposed to be polytopic (and by consequence bounded) and to contain the reference value. This means that $\xi^{\text{ref}}(k) \in \mathcal{X}$, $u^{\text{ref}}(k) \in \mathcal{U}$ and $u^{\text{ref}}(k) - u^{\text{ref}}(k-1) \in \mathcal{U}_\Delta$.

Note that the cost function is designed to minimize the difference between the nominal and the ideal trajectory, whereas the constraints are imposed on the real trajectory. Additionally, at each step of prediction, the current values have to be superposed over the Voronoi decomposition and the best linearization has to be selected.

The trajectory obtained by applying the optimal control u^* computed in (25)–(26) is "nominal", in the sense that it does not consider either exogenous noises (i.e., the wind) or the state-dependent linearization error (i.e., the term $\beta_{\text{j}}(\xi(k), u(k))$ from (17)). There are different approaches in the literature which deal with the reference trajectory tracking problem for dynamical systems affected by disturbances. A classical method is based on the tube MPC approach (for details, the reader is referred to (Rakovic et al., 2011; Mayne et al., 2011)) where a nominal trajectory is controlled and the real trajectory is kept into a tube around the nominal one through a suitable control action.

The "real" trajectory takes into account all the possible perturbations:

$$\xi^\circ(k+1) = A_{\text{j}}\xi^\circ(k) + B_{\text{j}}u^\circ(k) + r_{\text{j}} + \beta_{\text{j}}(\xi^\circ(k), u^\circ(k)) + w(k), \tag{27}$$

where the bounded perturbation $w(k)$ denotes the wind.

Subsequently, subtracting (20) from (27) we obtain the tracking error $z(k) = \xi^\circ(k) - \xi(k)$ measuring the difference between real $\xi^\circ(k)$ and nominal $\xi(k)$ trajectories:

$$z(k+1) = A_{\text{j}}z(k) + B_{\text{j}}u^\delta(k) + \beta_{\text{j}}(\xi^\circ(k), u^\circ(k)) + w(k),\ (\xi^\circ(k), u^\circ(k)) \in \mathcal{V}_{\text{j}}, \tag{28}$$

where $u^\delta(k) = u^\circ(k) - u(k)$ denotes the difference between "real" and "nominal" control actions.

Considering that the perturbations are bounded,[6] as long as a stabilizable control action:

$$u^{\delta}(k) = \mathcal{K}(\xi(k), \xi^{\circ}(k)), \tag{29}$$

exists, we can guarantee that the real trajectory (27) remains in a bounded neighborhood of the nominal one (20). Actually, there will exist a sequence of bounded sets which describes the tube:

$$z(k) \in S_{\mathrm{k}} \leftrightarrow \xi^{\circ}(k) \in \{\xi(k)\} \oplus S_{\mathrm{k}}, \ \forall k \geq 0. \tag{30}$$

For LTI dynamics, the choice of (29) is simply a gain matrix which makes the closed-loop dynamics stable. Here, due to the switched nature of (20) we need also to switch between the gain matrices:

$$\mathcal{K}(\xi(k), \xi^{\circ}(k)) = K_{\mathrm{j}}(\xi^{\circ}(k) - \xi(k)), \ j = 0, \ldots, N_{\mathrm{l}}, \tag{31}$$

with each gain K_{j} stabilizing the pair $(A_{\mathrm{j}}, B_{\mathrm{j}})$ in (20). Then, we obtain the switched system:

$$z(k+1) = (A_{\mathrm{j}} + B_{\mathrm{j}}K_{\mathrm{j}})z(k) + \beta_{\mathrm{j}}(\xi^{\circ}(k), u^{\circ}(k)) + w(k), \ (\xi^{\circ}(k), u^{\circ}(k)) \in \mathcal{V}_{\mathrm{j}}, \tag{32}$$

which, under the assumption of existence of a common Lyapunov function (via a piecewise Lyapunov function (Hovd et al., 2010) or alternatively by imposing constraints on the dwell time between switches (Colaneri, 2009)) is stable.

Of theoretical interest is the computation of the "tube" defined by the sets $S_{\mathrm{j(k)}}$. For LTI dynamics, the set $S_{\mathrm{j(k)}}$ is constructed to be robust positively invariant in order to minimize the on-line computations. Here the dynamics of the vehicle change whenever the linearization point changes and it may not be possible to find a common robust positively invariant set. In this case, a hybrid structure can be proposed. That is, we compute robust invariant sets for each of the linearized dynamics and change between them (or scaled versions of them, i.e., $\tilde{S}_{\mathrm{j(k)}}$ $\lambda(k)$ $S_{\mathrm{j(k)}}$) whenever the linearization point impose it. To summarize, the practical procedure is the following:

- as long as the system use the linearization point $(\xi^{\mathrm{ref}}(k), u^{\mathrm{ref}}(k))$, the set $S_{\mathrm{j(k)}}$ is defined by the same invariant shape and the scalar $\lambda(k) = 1$,
- if the index at step $k-1$ is $j(k-1)$ and $j(k-1) \neq j(k)$, then the shape of $S_{\mathrm{j(k)}}$ changes as a consequence of the fact that the linearization index changes.

[6] If the cell $\mathcal{V}_{\mathrm{j}}$ is bounded, then the nonlinear residue $\beta_{\mathrm{j}}(\xi(k), u(k))$ is also bounded.

The scalar $\lambda(k)$ is computed via the LP problem:

$$\begin{aligned} &\lambda(k) = \min \lambda \\ &\text{s.t.} \quad \lambda S_{\text{j(k)}} \supset S_{\text{j(k-1)}}. \end{aligned}$$

The danger, from the stability point of view, is to have a monotonic increase of the coefficients λ along the switches. This remark provides a simple and efficient criterion for detecting the misfunctioning of the predictive feedback loop: the violation of a pre-imposed bound on the scaling factor. Practically, as long as the change between different dynamics is slow enough, the overall stability of the tracking error, and thus of the boundedness of the tube are preserved due to local contractive properties of each LTI mode (32).

Although, the generic robust predictive control strategy would be feasible for the UAV application, we do not pursue here the invariant set manipulation in the implemented control scheme. We will test the nominal predictive control scheme and show that the inherent robustness of this control law covers in a satisfactory manner the tested maneuvers. We mention however, that whenever the control law needs to pass by certification procedures for covering important wind variations, the robust version of the design needs to be adopted. Moreover, it needs to include the tube description together with a watchdog mechanism for the invariant set scaling factor, which can commute the UAV functioning towards a "safe mode".

Algorithm 1: Trajectory tracking optimization-based control problem

Input: Consider model (1) and give the collection of waypoints $\mathbb{P}$ as in (13).

-construct the flat trajectory as in (8)–(9), passing through $p^{\text{i}} \in \mathbb{P}$;

-choose a collection of linearization points $\mathbb{L}$ as in (16);

-construct the PWA function as in (20) with $(A_{\text{j}}, B_{\text{j}}, r_{\text{j}})$ defined as in (18);

-partition the state-space into Voronoi cells as in (22);

for $s = 1 : s_{max}$ **do**

  -select the linearization point $l^{\text{j}} \in \mathbb{L}$ by testing (23);

  -select the pair $(A_{\text{j}}, B_{\text{j}}, r_{\text{j}})$ by testing (24);

  -find the optimal control action u^* by solving (25);

  -compute the next value of the state:

$$\xi(k+s+1) = A_{\text{j}}\xi(k+s) + B_{\text{j}}u(k+s) + r_{\text{j}}.$$

end

Finally, Algorithm 1 recapitualtes the mechanism implemented based on the theoretical elements previously presented.

In the sequel, we provide first the UAVs testbed hardware and software architecture description and then, software-in-the-loop simulations in comparison to the actual experimental results on real UAVs, which validate our proposed approach.

6. Practical implementation of the receding horizon flight controller on a UAV

In the current paper we present software-in-the-loop simulations and real flight tests results for the predictive control of Unmanned Aerial Vehicles (UAVs). The flight experiments took place at the Portuguese Air Force OTA airfield in May 2012 on small platforms (Figure 3, Figure 4) owned by the Air Force Academy and Underwater Systems and Technologies Laboratory (Laboratório de Sistemas e Tecnologias Subaquáticas - LSTS), University of Porto.

The control objective is to force the UAV to track some given waypoints. The control inputs are the velocity and the bank angle of the UAV. One real-world situation that matches this scenario is that of an autonomous aircraft equipped with GPS, radio communications and a camera. The mission of the aircraft is to take some snapshots at a certain time of a certain area and then, to transmit the information to the ground.

Figure 3: "Cularis-05" UAV of the LSTS laboratory from University of Porto.

In the sequel, we present first a summary of the control setup used onboard the UAV.

Figure 4: "Pilatos-03" UAV (in the right side) of the LSTS laboratory from University of Porto and "Alfa-06" UAV (in the left side) of the Portuguese Air Force Academy.

6.1. UAVs testbed hardware and software architecture description

The testbed implements a ground (off-board) control, that is, the controller runs on Matlab on a ground station computer that receives telemetry and sends flight commands through a Ground Station. The autopilot system is a Piccolo II (Vaglienti et al., 2011), which receives airspeed and bank commands from DUNE, a control software developed by the LSTS lab from University of Porto (Pinto et al., 2012).

The LSTS lab testbed software and hardware architectures enabled us to integrate and to test the proposed predictive control algorithm in a real-time environment. One fundamental control issue we deal with in this paper is the problem of tracking a given reference trajectory in the presence of constraints (see (Aguiar and Hespanha, 2007) and the references therein). This problem is even more challenging because the UAVs dynamical systems are nonlinear, underactuated and exhibit nonholonomic constraints (Li and Canny, 1993; Reyhanoglu et al., 1999).

The operation of an UAV platform is based on the existence of a hardware infrastructure and a software architecture which enable the real-time control. For the present study the Piccolo system will be used[7]. The Piccolo control system setup, shown in Figure 5, consists of four main parts (Vaglienti et al., 2011): an Avionics control system located onboard the UAV, a Ground Station, a Pilot Manual Console and the Piccolo Command Center (operator interface).

As shown in Figure 6, the controller setup has two control loops, that is, a faster

[7]The Piccolo system is manufactured by *Cloud Cap Technologies* (http://www.cloudcaptech.com/piccolo_system.shtm).

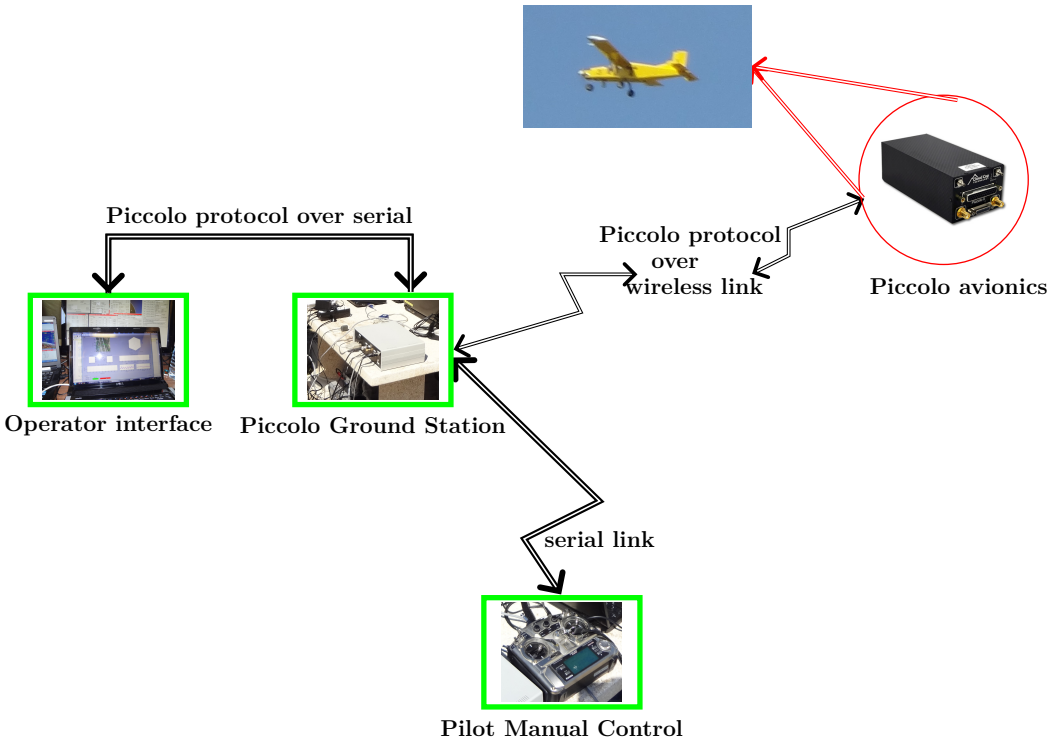


Figure 5: Operational control system setup of the UAV.

inner loop on board the UAV and a slower outer loop, which is implemented on the Ground Station control computers. The outer loop provides the path to be followed by the vehicle whereas, the inner loop controls the UAV dynamics. Furthermore, Piccolo autopilot relies on a mathematical model parameterized by the aircraft geometric data and has a built-in wind estimator. A slightly more detailed description of the main components of the Piccolo autopilot is given in the following.

Piccolo autopilot description

The **Piccolo Avionics** system is an autopilot designed to track the controlled path transmitted by the Ground Station. It relies on a group of sensors which includes: three rate gyroscopes and two axis accelerometers (represented by the Inertial Measurement Unit - IMU), a radio, a Global Positioning System (GPS) to determine its geodetic position, and a set of dynamic and static pressure sensors coupled with a thermometer to determine the airplane's true airspeed and altitude.

A Kalman filter approach is used to gather the information provided by the IMU

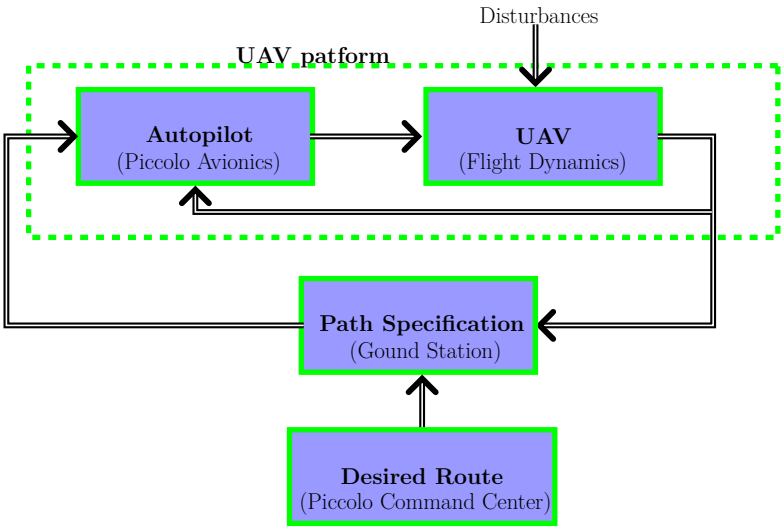


Figure 6: Piccolo closed-loop control system.

block (accelerations and angle-rates) and the GPS block (positions and velocities) in order to estimate the UAV full state. An embedded Power PC receives the state information from all sensors and runs the autopilot loops commanding the control surfaces actuators of the airplane (ailerons, elevator and rudder), the engine's thrust as well as playload ports. Telemetry data is transmitted to the Ground Station through 2.4 GHz radio modem with a telemetry rate up to 25Hz. With omni-directional antennas the signal strength is enough for a 3 km communication radius. On the other hand, with directional antennas, the communication range extends to more than 40 km.

The **Piccolo Ground Station** provides the communication link between the Piccolo Command Center, the Pilot Manual Control and the Piccolo Avionics and converts the intentions of the end user captured through the operator interface, into meaningful commands for the autopilot. The ground station can concurrently monitor up to 10 UAVs. Moreover, it performs differential GPS corrections, and updates the flight plan, which is a sequence of three dimensional waypoints connected by straight lines.

The **Piccolo Command Center** (operator interface) consists of a portable computer and a custom developed software which allows the end user to monitor the flight progress, to program a desired route for the UAV via waypoints, to send velocity, bank or altitude references and finally, to configure the desired gains of the control system in the autopilot.

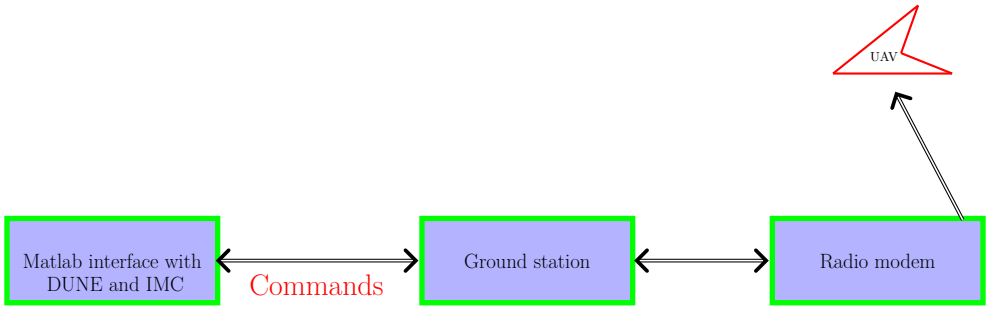


Figure 7: Software architecture.

The **Pilot Manual Control**, which is mainly used for take-off and landing, provides the end user with a way to override the commands generated by the Ground Station and allows a qualified UAV Pilot to take control of the UAV.

Software architecture

The UAV operational system block diagram, including the software architecture is depicted in Figure 7.

LSTS lab interfaced Piccolo software with MATLAB using DUNE and IMC messaging system (Pinto et al., 2012). DUNE (DUNE Unified Navigational Environment) it is a generic embedded software used to run tasks for vehicle control, navigation, communication, sensor and actuator access.

DUNE can run in simulation mode, which disables all the sensor and actuator drivers and replaces them with simulating tasks. It may also run in hardware-in-the-loop mode, which allows for some sensor or actuator drivers to be enabled, together with simulating tasks (Pinto et al., 2012).

The Inter-Module Communication (IMC) is a publish-subscribe messaging protocol designed and implemented for communication among heterogeneous vehicles (for example, aerial and marine vehicles), sensors, human operators, etc.

It is noteworthy that the core control algorithm can run in MATLAB. Both, the input to the control algorithm and the output of the control algorithm are communicated through IMC messages. The DUNE interprets the IMC messages to the corresponding commands in Piccolo software which controls the UAV (Martins et al., 2009; Dias et al., 2010).

6.2. Simulation and experimental flight tests results

For pre-flight validation of the proposed trajectory tracking method an extended model of (1) (for low-level control of an Unmanned Aerial Vehicle (UAV)) with 12-

states has been used in a 6-DOF simulation (Vaglienti et al., 2011). More precisely, the 12-states model includes the positions (x [m], y [m], z [m]), the velocities (v_{x} [m/s], v_{y} [m/s], v_{z} [m/s]), the roll, pitch and yaw angles (ϕ [rad], θ [rad], ψ [rad]), and the angular rates (p [rad/s], q [rad/s], and r [rad/s]), all measured along body X, Y, and Z axes. For the real tests the same model has been used for the different UAV platforms that have been operated, but with different parameter values.

The predictive control algorithm was implemented in MATLAB 2011a over a Windows 7x32 bit, a portable Dell machine with Intel(R) Core(TM)2 Duo CPU and 4 GB of RAM. For solving the optimization problem, which is actually a quadratic programming problem we used our own routines and also standard functions provided by the Optimization Toolbox[8]. Furthermore, for interfacing MATLAB with Piccolo we used DUNE and the IMC messaging system. Therefore, the code was running in MATLAB and then, both input to the control algorithm and the output of the control algorithm were communicated in terms of IMC messages. Furthermore, DUNE was interpreting the IMC messages and translating them to the corresponding commands in Piccolo software which was communicating with the UAV.

The UAVs were restricted to a speed range between 18 and 25 meters per second (i.e., $V_{\mathrm{a}} \in [18,\ 25]$ m/s) and to a maximum bank angle of 0.43 radians (i.e., $\phi \in [-0.43,\ 0.43]$ rad). Also, there were restrictions on the variation of the input signals:

- the variation of V_{a} is limited to $0.1 \sim 0.2$ m/s^2;
- the variation of ϕ is limited to $0.5 \sim 1.1$ rad/s.

In addition, the simulations and the flight tests took place under various wind conditions. We assumed that the intensity of the wind is bounded for some reasonable values, e.g., a maximum speed of 10m/s (for safety reasons the tests are not performed on UAVs if the wind exceeds $11 \sim 12$ m/s).

Let us summarize in Algorithm 2 the mechanism we used during the flight tests. Note that it includes the trajectory tracking optimization-based control problem presented in Algorithm 1. The workflow, as sketched in Algorithm 2 is as follows. We assume that the UAV is in flight under the control of Piccolo. Then, we take control and disable Piccolo input (step 4) and provide the control action as designed by our own control strategies (steps 5–8). Note that when we take control we need to

[8] A mitigating factor is that the most significant is the computation of the reference trajectory which is done off-line and thus, has no influence on the runtime. Also, solving the quadratic optimization problem (via the Optimization Toolbox of MATLAB) was well under the 0.02 seconds of the discretization step.

Algorithm 2: Real-time control of the UAV

Input: Give the collection of waypoints $\mathbb{P}$ as in (13)

- determine the functions for the reference state and input signals using (8)–(9) and passing through the waypoints $p^{\mathrm{i}} \in \mathbb{P}$;
- **while** *the UAV is flying and is controlled by Piccolo* **do**
  - -Piccolo data acquisition;
  - **if** *TRACK==1 (the predictive controller takes command)* **then**
    - -take the current time as the starting point for the reference trajectory ($t = 0$);
    - -compute references and linearization matrices for the interval $t : t + N_{\mathrm{p}}$;
    - -use this information to compute the optimal control action u^* by solving (25);
    - -compute the next value of the state

$$\xi(k+s+1) = A_{\mathrm{j}}\xi(k+s) + B_{\mathrm{j}}u(k+s) + r_{\mathrm{j}};$$

  - **else**
    - -the UAV is controlled by Piccolo
  - **end**
- **end**

compute the reference trajectory and that the control design is no longer done on the UAV but rather on the Ground station which has to send/receive data. This means that when communication is lost, the UAV losses its input and starts to function in "open-loop", hence the tracking errors we will see in the experimental tests results.

In the sequel, we present first software-in-the-loop simulations, that is, we implement the algorithm in DUNE which runs in simulation mode, the UAV being simulated by the previously mentioned 12-state model.

A. Software-in-the-loop simulations

We continue with the control part of the example presented along the paper (see Section 3 and Section 4). The numerical data used for the simulated vehicle trajectory tracking are showed in Table 1.

In a first stage, using the results in Section 3 we generated a flat trajectory (depicted in blue in Figure 8) starting from the current position of the vehicle and passing through the given waypoints. In a second stage, we used the linearized model

UAV platform	simulated
Waypoint list	$\mathbb{P} = \{(-500, -500, 150), (-180, -650, 150), (0, -350, 150), (-350, 0, 150), (-350, 300, 150), (-200, 500, 150), (50, 450, 150), (300, 0, 150), (0, -350, 150)\}$
Sampling time	100 ms
Tuning parameters	$Q = [10e1\ 0\ 0; 0\ 10e1\ 0; 0\ 0\ 0.1],\ P = [10e2\ 0\ 0; 0\ 10e2\ 0; 0\ 0\ 0.1],$ $R = 10e4 \cdot [10\ 0; 0\ 1],\ R_\Delta = 10e4 \cdot [10\ 0; 0\ 1],\ N_\mathrm{p} = 7$
Wind	5 m/s

Table 1: Simulation results: numerical data specifications.

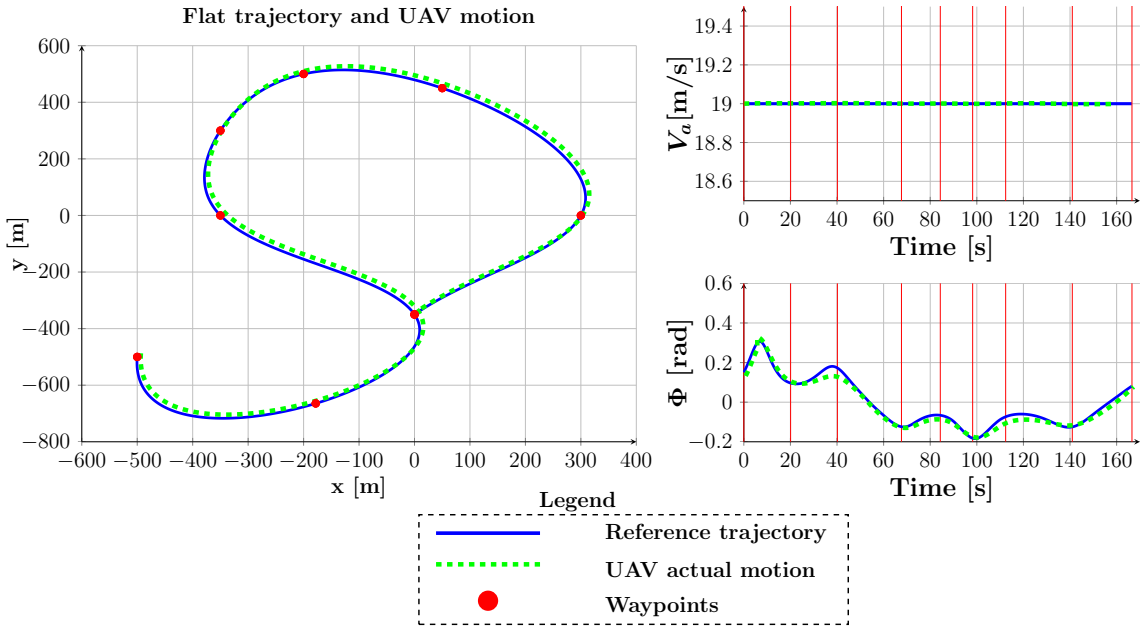


Figure 8: Reference trajectory and actual UAV motion (simulation).

(see the linearization procedure in Section 4) for the control part of the trajectory tracking problem.

Figure 8 shows simulated tracking performance in the x-y coordinate frame (i.e., north-east coordinate frame). In this case, the simulations were performed in MAT-

LAB. The vehicle tracking performances for the given reference trajectory (depicted in blue in Figure 8) are depicted in green in Figure 8. As illustrated in green in the same figure, the constraints on the velocity and bank commands are satisfied.

We have chosen to construct the reference trajectory as in (11)–(12), that is, with a constant value of V_a. Note that even for (8)–(9) where V_a is variable, the bounds for inputs (as seen in Figure 8) are respected. The problem for the latter case was the speed of variation for V_a (as seen in Figure 1). For the software-in-the-loop simulations and also in the real flight tests the UAVs model have stringent bounds for V_a variation ($0.1 \sim 0.2$ m/s^2). Consequently, we have used the former reference design, where by construction, the variation of V_a is zero (see also the illustrative simulation example in Section 3).

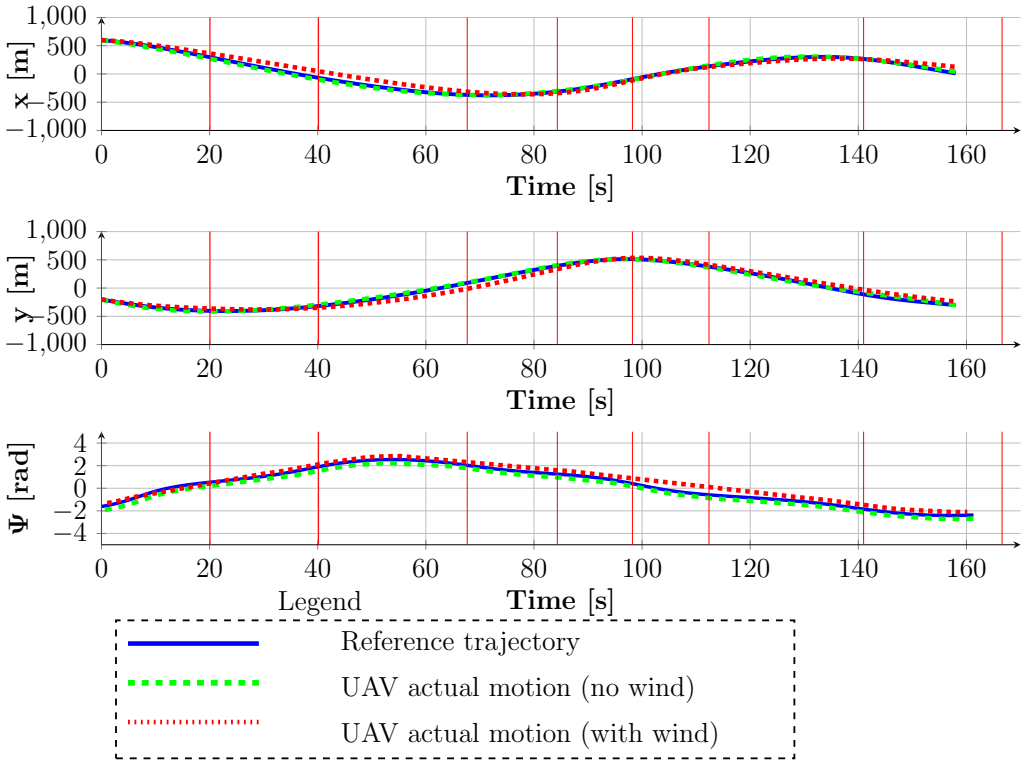


Figure 9: Comparison between actual and reference UAV motion (simulation).

The predictive tracking controller was also tested in simulations with different wind conditions and a maximum speed of 8 m/s. Figure 9 presents for exemplification the simulated reference trajectory in solid blue and the tracking performance

of the UAV with and without wind disturbances. In addition, the figure illustrates the time when the UAV passes through the waypoints. The main limitations against superior performance are the wind magnitude and the numerical issues (density of linearization points, prediction horizon length, etc.). Next, we presents the experimental results.

B. Real tests results

The numerical data used for the real-time vehicle trajectory tracking are showed in Table 2.

UAV platform	"Alfa-06", combustion motor, 2.4 m span, Piccolo II, PC104, 2h30 endurance
Waypoint list	$\mathbb{P} = \{(0, 600, 150), (300, 0, 150), (0, -600, 150), (-300, -600, 150), (-600, -300, 150), (-300, 0, 150)\}$
Sampling time	100 ms
Tuning parameters	$Q = [10e1\ 0\ 0; 0\ 10e1\ 0; 0\ 0\ 0.1]$, $P = [10e2\ 0\ 0; 0\ 10e2\ 0; 0\ 0\ 0.1]$, $R = 10e4 \cdot [10\ 0; 0\ 1]$, $R_\Delta = 10e4 \cdot [10\ 0; 0\ 1]$, $N_\mathrm{p} = 7$
Wind	$5 \sim 7$ m/s to 180 degrees (from South 6.3 m/s, from West 5.83 m/s), the airplane was flying with the wind

Table 2: "Alfa-06" platform: numerical data specifications for the experimental results.

The procedure is the same, that is, in a first stage, a basic flat trajectory generation mechanism is used (see Section 3). We underline again here the lack of constraints feasibility guarantees in-between the waypoints. More precisely, even if we impose admissible values in the waypoints, we had no guarantees about the behavior in the rest of the trajectory. This represent a problem for the real-time experiments, especially for the velocity component of the control input. We had to take into account that the rate of change of the velocity is limited to the maximum acceleration the aircraft can produce, i.e., $0.1 \sim 0.2$ m/s^2. The MPC will enforce the constraints satisfaction and thus, we can use any curve provided by the reference trajectory generation mechanism (see the illustrative simulation results in Figure 8).

In practice, it was clear that this bounds were difficult to respect by the MPC controller whenever the generated reference trajectory hit the aggressive limits on the velocity and the bank. Moreover, the system being nonlinear and the model mismatch too large, the closed loop dynamics lead to instability in these situations.

We highlight again the importance of bringing the trajectory into admissible limits from its reference generation. Hence, the equations (11)–(12) (see Section 3) were used for the trajectory generation.

This explains the fact that, in the same time, the generated trajectory is allowing a degree of freedom to the variables $V_{\mathrm{a}}(t)$ and $\phi(t)$ from the constraints activation point of view. As a consequence, these signals can be used in real time by the predictive controller to contract the tracking errors.

Figure 10 depicts in blue the reference trajectory the initial position $(-202, -7, 2.86)$ to the final position $(221, 269, 1.57)$, the waypoints (the red dots) and the control input signals. Observe that the velocity control input of the reference trajectory is constant and the bank control input varies between acceptable limits. Additionally, Figure 10 shows the derivatives of the control input components of the reference trajectory which vary between acceptable limits.

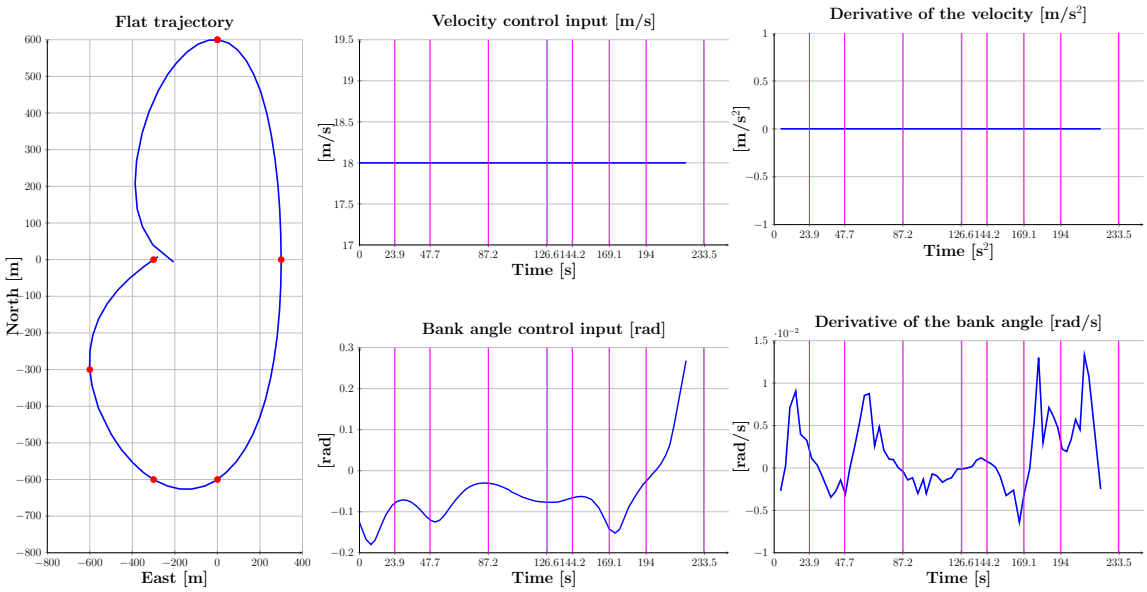


Figure 10: Reference trajectory, control input signals and their derivatives (flight experiments with Alfa06 UAV).

For the control action we have implemented the MPC mechanism using the model described in (20), where both, the bank angle and the velocity are variable. As previously mentioned, we have used this construction in DUNE where, the control action applies to a more realistic 12-states model.

Figure 11 illustrates the actual UAV motion (depicted in dashed green) in North-

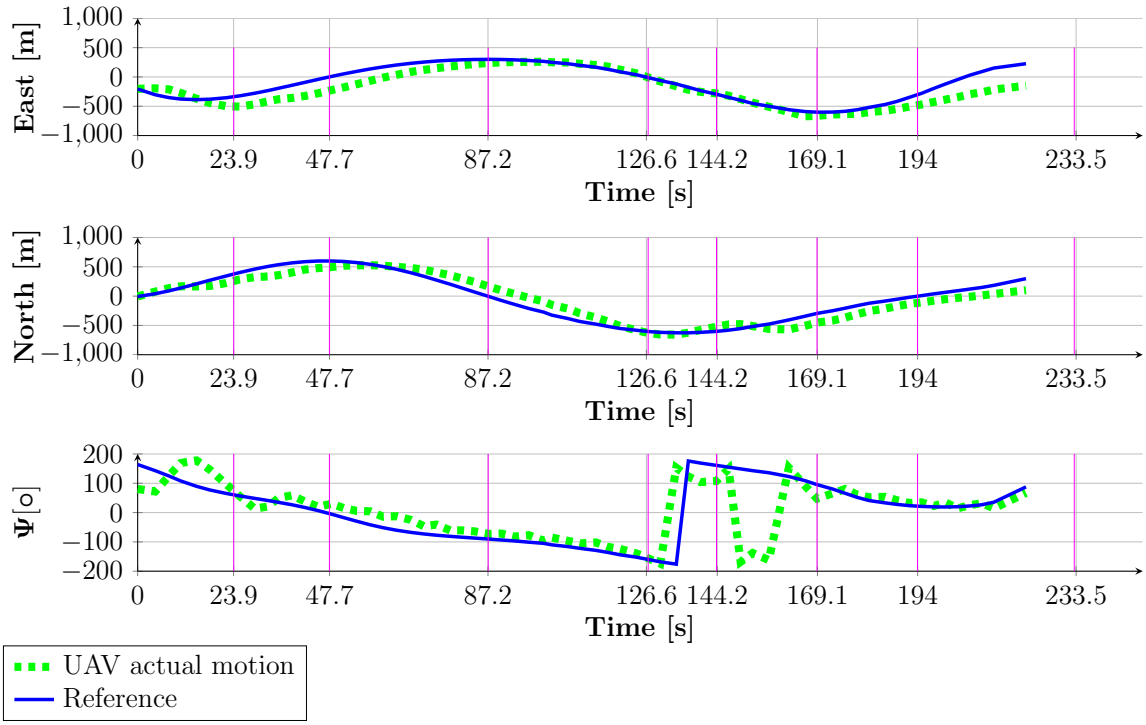


Figure 11: Reference trajectory and actual UAV motion (flight experiments with Alfa06 UAV).

East coordinate frame of the real flight test. Acceptable tracking performances for the given reference trajectory (depicted in blue in Figure 11) are obtained for the UAV. A 3D illustration of the test scenario together with the actual motion of the UAV is depicted in Figure 12, while the tracking error is depicted in Figure 13.

Remark 3. It is worth to be noted that a hierarchy of models for the UAV dynamics has been used. Firstly, we use model (1), with V_a a constant, in order to generate the flat trajectory along (11)–(12). Secondly, the MPC uses the complete model (1) to derive a control action, which is ultimately applied to a 12-states model of the UAV (or to the Piccolo control UAV dynamics during the experiments). Nonetheless, this model mismatches may represent one of the reasons for the tracking error. However, the predictive controller proves to be robust, in the sense that the error presented in real flight tests remains bounded over the time window and the real trajectory remains in a tube centered around the reference. □

In experimental settings we observe that there are losses of communication hence, the discretization step itself is not constant (see Figure 14 where the nominal and

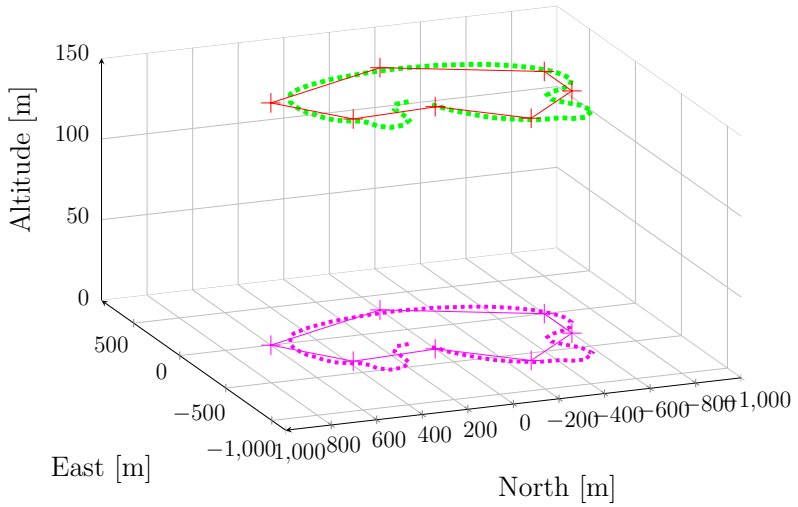


Figure 12: Actual UAV motion and its projection on the x-y space (flight experiments with Alfa06).

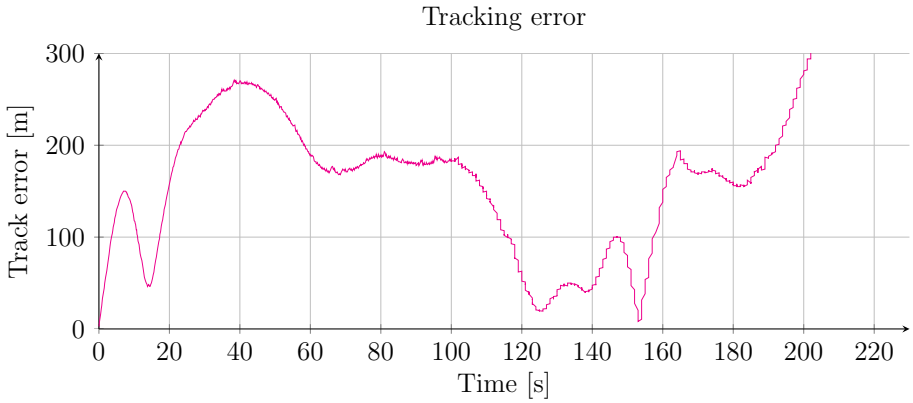


Figure 13: Tracking error (flight experiments with Alfa06).

actual UAV control motion are illustrated). Figure 15 shows the length of the discretization step, depicted in blue. Also, in the same figure, the red line denotes the average length of the discretization step, thus showing that there are values significantly above this average (i.e., there are significant losses of communication during the UAV real-time test). Observe that the values can lead up to $10 \sim 20$ steps more then the normal size (i.e., 0.1 s ~ 1 s). Some of the negative effects can be mitigated. First, the flat trajectory is generated by continuous functions subsequently it's simply

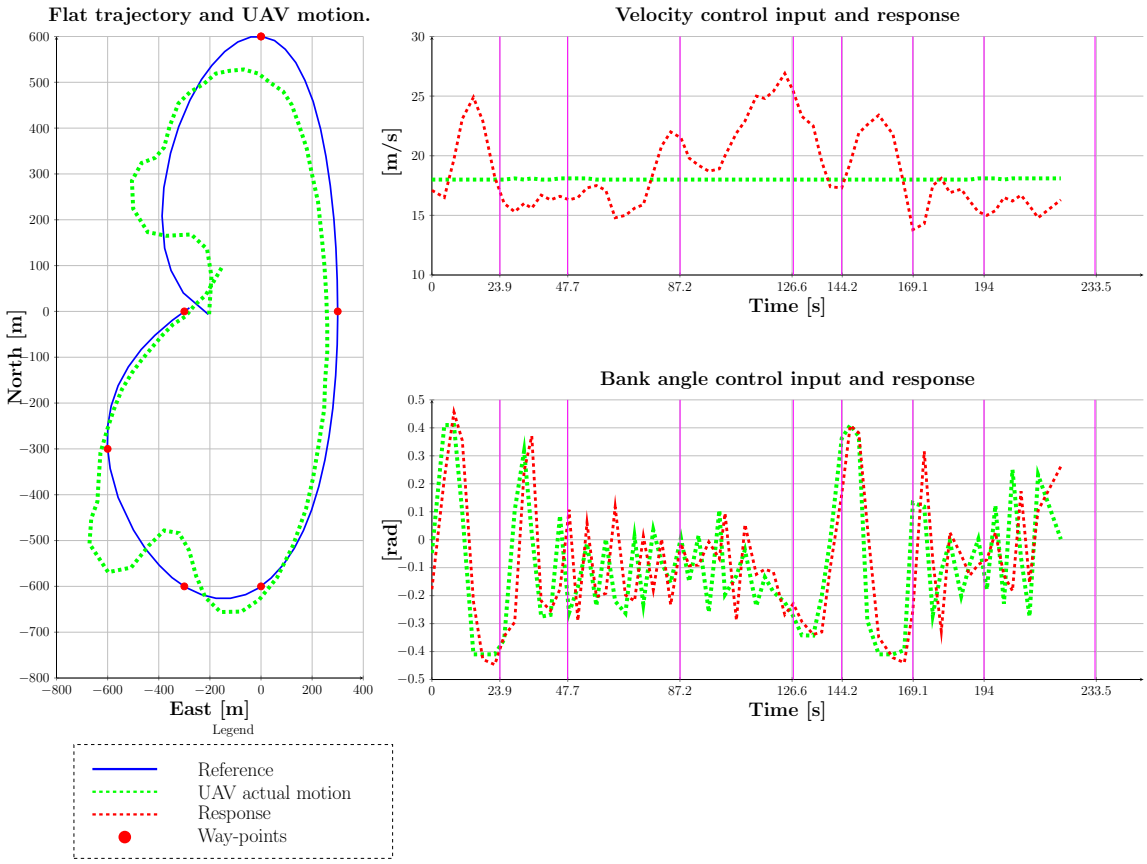


Figure 14: Nominal and UAV actual motion (flight experiments with Alfa−06).

a matter of "keeping the time" in order to provide a correct sequence of references for the MPC block. Thus, the reference signal is independent of the sampling delays. In what regards the MPC block, we consider a prediction horizon which is sampled at the nominal sampling time, but this sampling is done exclusively for the internal use in order to compute an optimal input. Therefore, the fact that we have a constant sampling time inside the MPC can coexist with the fact that the real sampling time is variable. There will be issues caused by this loss of communication[9] but, due to the fact that the UAV evolves in open-loop for a long period and not because the

[9]Note that the tracking error is mostly due to the complex dynamics involved, with significant delays in communications, and does not occur in simulation, see the illustrative figures from the simulation results.

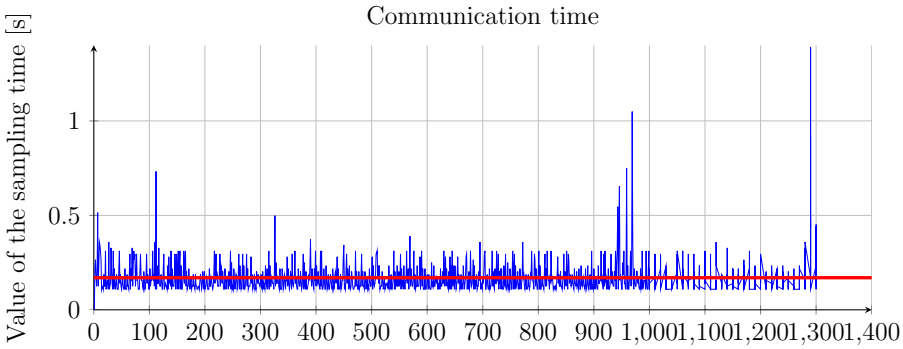


Figure 15: Value of the sampling time (flight experiments with Alfa−06).

control action is improper.

During this flight experiment, the wind amplitude was less than $5 \sim 7$ m/s. In practice, to model the influence of the wind, we considered the returned velocities and compared them with the nominal ones in order to extract the wind value. Consequently, we use this value in the cost function (25). This ad-hoc procedure is just a first step in the disturbance handling and points for the future work to an adequate use of an estimation of the wind based on a Kalman filter approach (Østergaard et al., 2007; Achour et al., 2010).

Without entering into extensive details we present next the tests results obtained on a second flight experiment. The illustrative figures show the improvements of the real-time trajectory tracking, due to the small delay in communications and better choice of the tuning parameters. The numerical data used for the real-time vehicle trajectory tracking are shown in Table 3.

Figure 16 depicts the waypoints (red dots) and the reference trajectory (blue line) that Pilatos−03 UAV needs to follow. Furthermore, in Figure 17 we illustrate, in the North-East coordinate frame, the actual motion of the UAV (in dashed green). The view of the flight scenario and the UAV motion is represented in a North-East-Altitude coordinate frame in Figure 18. Figure 20 shows the tracking error which is under 110 m, while Figure 14 illustrates the nominal and actual UAV control motion. It can be observed that the tracking performance is better than in the previous scenario and is mostly due to the small delay in communications during the flight test. This can be observed in Figure 21 which depicts in blue the length of the discretization step. In the same figure, the average length of the discretization step is represented by the red line, showing acceptable deviations above the average (i.e., the are acceptable losses of communication during the UAV testing).

UAV platform	"Pilatos-03", combustion motor, 2.82 m span, Piccolo II, PC104, 1h30 endurance
Waypoint list	$\mathbb{P} = \{(-100, -350, 150), (300, 0, 150), (0, 500, 150), (-300, 500, 150), (-500, 300, 150), (-300, 0, 150)\}$
Sampling time	100 ms
Tuning parameters	$Q = [10e1\ 0\ 0; 0\ 10e1\ 0; 0\ 0\ 1],\ P = [10e2\ 0\ 0; 0\ 10e2\ 0; 0\ 0\ 10]$, $R = 10e4 \cdot [10\ 0; 0\ 1],\ R_\Delta = 10e4 \cdot [10\ 0; 0\ 1],\ N_\mathrm{p} = 7$
Wind	$4 \sim 6$ m/s to 90 degrees (from West to East), the airplane was flying against the wind

Table 3: "Pilatos-03" platform: numerical data specifications for the experimental results.

Remark 4. Note that there are small differences between the control algorithm implementation for the practical experiments. In the first experimental result the design of the control action considers the heading, whereas in the second experiment considers the course. The difference between them is that in the second case, a more realistic value of the heading has been used, that is, the course takes implicitly into account the wind. We believe that, besides the small delay in communications, this is another reason for obtaining superior tracking performances. □

7. Concluding remarks and improvement directions

The present paper presented a Model Predictive Control (MPC) strategy for Unmanned Aerial Vehicles (UAVs). The complete design approach covering the prediction model description, the trajectory generation and the optimization-based design have been presented. The results of software-in-the-loop simulations and real flight tests confirmed the viability of the proposed approach. Moreover, the reference generation proved to be a valuable tool in the adaptation of the predictive control setup, the flat trajectory being an important element towards the constraints validation. Also, the proposed trajectory generation mechanism takes into account waypoint conditions and furthermore, allows to obtain linearizations of the nonlinear vehicle model along the flat trajectory.

As mentioned, we have encountered difficulties, both theoretical and in the practical implementation. During the presentation we gave solutions to some of them and point to possible improvements.

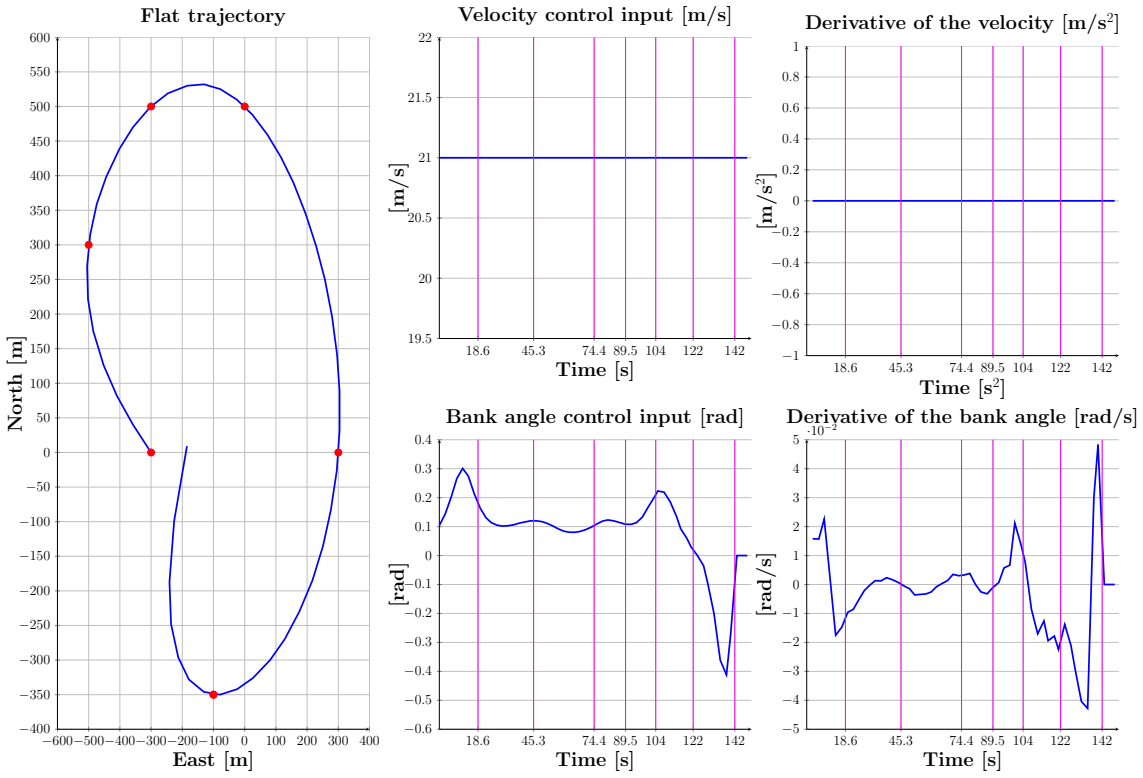


Figure 16: Reference trajectory, control input signals and their derivatives (flight experiments with Pilatos−03).

Due to the difficulties of the schema (nonlinear dynamics and the associated computational load) we pointed to the tube MPC as a theoretical framework for the robust design. In practice, the computational constraints restricted the MPC implementation to the nominal but time-varying prediction model.

Another conclusion of the present work is the need for a better mechanism for wind estimation in order to decrease the uncertainty in the prediction model (e.g., a Kalman filter implementation). Nonetheless, with all these shortcomings and with various (and usually unfavorable) wind conditions the flight tests exhibited an acceptable tracking performance. The implicit robustness of the MPC approach proved its capabilities and the results were satisfactory.

Another sensitive point revealed by practice is the discretization and the inter-sample variations. For now, we assumed a fixed length of the discretization step and computed the control action accordingly. A future improvement of the control mech-

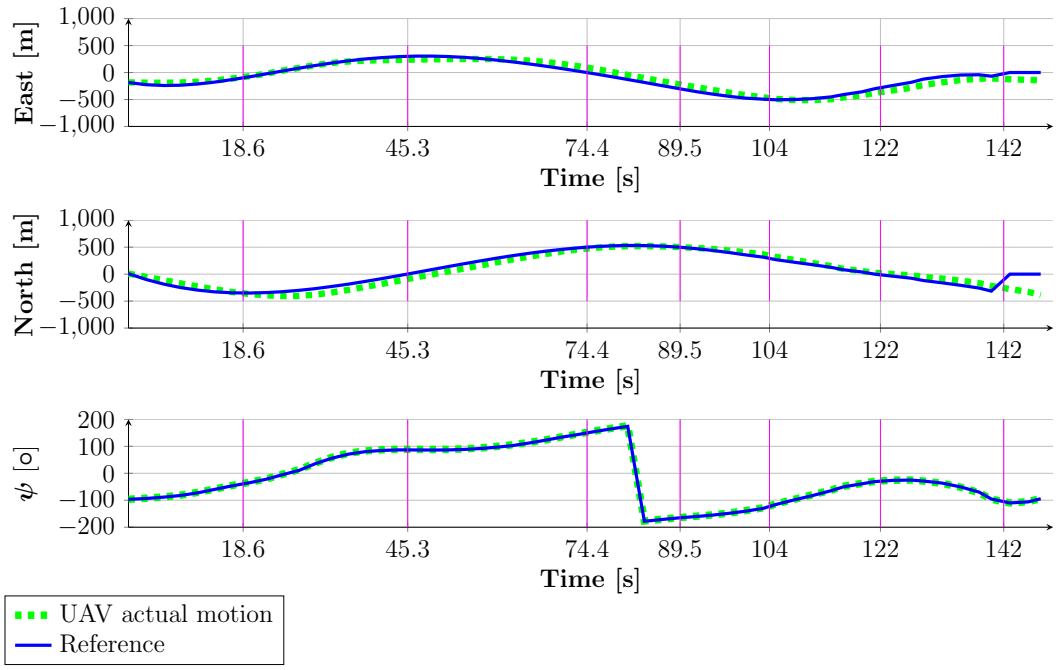


Figure 17: Reference trajectory and actual UAV motion (flight experiments with Pilatos−03).

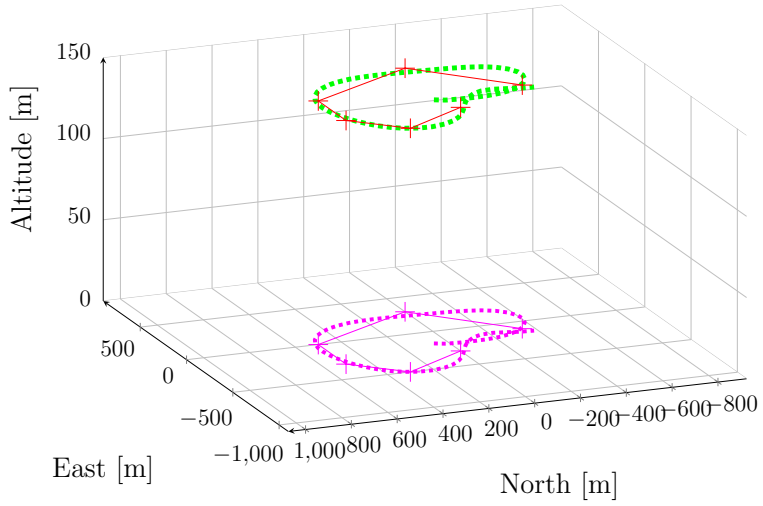


Figure 18: Actual UAV motion and its projection on the x-y space (flight experiments with Pilatos−03).

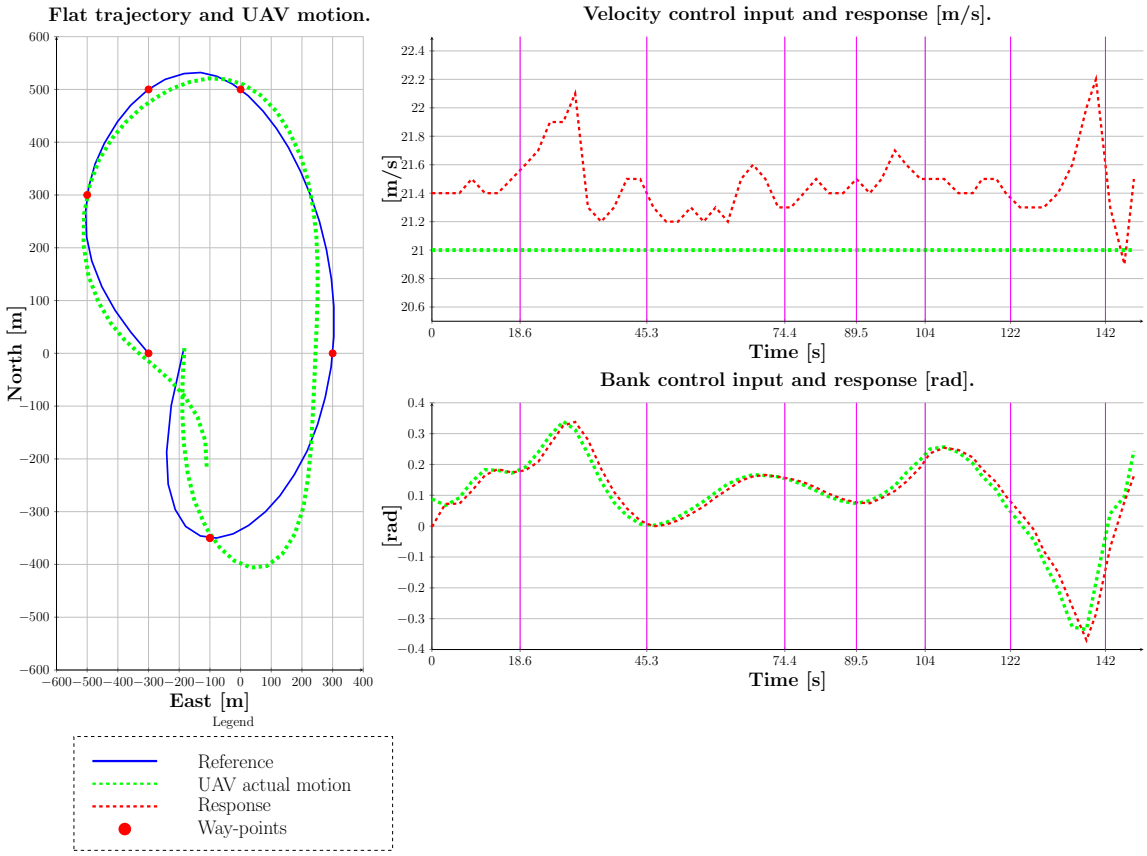


Figure 19: Nominal and actual UAV motion (flight experiments with Pilatos−03).

anism needs to include these communication delays in the design of the predictive controller.

Acknowledgments

We are grateful for the support we enjoyed from SUPELEC Systems Sciences (E3S) - Automatic Control Department, specially Patrick Boucher for the support and the interest has shown to this work.

The authors would like to thank the members of Pitvant project for for their support during this work. The authors also thank the Portuguese Air Force Academy for the wonderful support during the field tests at the OTA airfield.

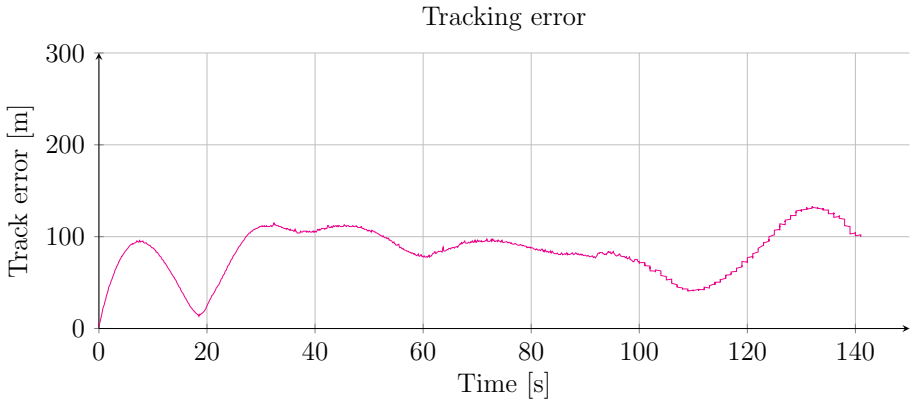


Figure 20: Tracking error (flight experiments with Pilatos−03).

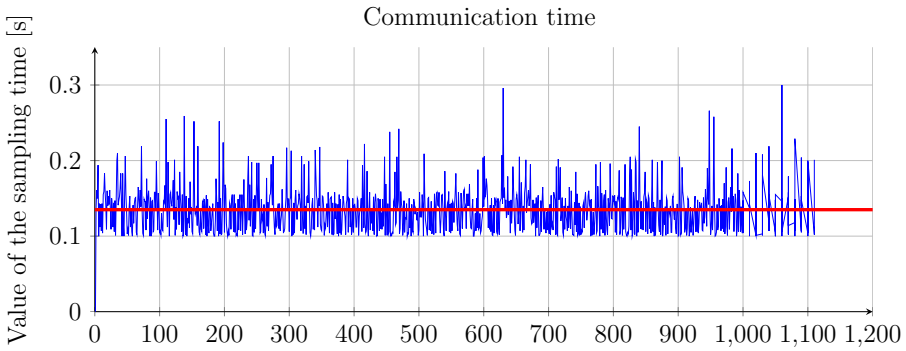


Figure 21: Value of the sampling time (flight experiments with Pilatos−03).

References

Achour W., Piet-Lahanier H., Siguerdidjane H. (2010): Wind field bounded error identification and robust guidance law design for a small-scaled helicopter *Automatic Control in Aerospace*, vol. 18(1):43–48.

Aguiar A., Hespanha J. (2007): Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, vol. 52(8):1362–1379.

Bencatel R., Faied M., Sousa J., Girard A. (2011): Formation control with collision avoidance. *In Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, Florida, USA, pp. 591–596.

Colaneri, P (2009): Dwell time analysis of deterministic and stochastic switched systems. *In Proceedings of the 10th IEEE European Control Conference*, Budapest, Hungary, pp. 15-31.

De Doná J., Suryawan F., Seron M., Lévine, J. (2009): A flatness-based iterative method for reference trajectory generation in constrained NMPC. *International Workshop on Assessment and Future Direction of Nonlinear Model Predictive Control*, Pavia, Italy, pp. 325–333.

Dias P., Pinto J., Gonçalves R., Sousa J. (2010): Enabling a dialog–a c2i infrastructure for unmanned vehicles and sensors. *In Proceedings of the IEEE International Conference on Autonomous and Intelligent Systems*, Povoa de Varzim, Portugal, pp. 1–6.

Fagiano L., Canale M., Milanese M. (2009): Set membership approximation of discontinuous nmpc laws. *In Proceedings of Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, P.R. China, pp. 8636–8641.

Falcone P., Tufo M., Borrelli F., Asgari J., Tseng H. (2007): A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. *In Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Los Angeles, USA, pp. 2980–2985.

Falugi, P. and Olaru, S. and Dumur, D. (2010): Multi-model predictive control based on LMI: from the adaptation of the state-space model to the analytic description of the control law. *International Journal of Control*, vol. 83(8):1548–1563.

Fliess M., Lévine J., Martin P., Rouchon P. (1995): Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control*, vol. 61(6):1327–1361.

Fontes F., Fontes D., Caldeira A. (2009): Model predictive control of vehicle formations. *Optimization and Cooperative Control Strategies*, Springer, pp. 371–384.

Geyer C. (2008): Active target search from uavs in urban environments. *In Proceeding of the IEEE International Conference on Robotics and Automation*, Pasadena, California, USA, pp. 2366–2371.

Goodwin G., Seron M., De Dona J. (2004). Constrained control and estimation: an optimisation approach. *Springer*.

Hao Y., Agrawal S. (2005): Formation planning and control of ugvs with trailers. *Autonomous Robots*, vol.19(3):257–270.

Heemels W., De Schutter B., Bemporad A. (2001): Equivalence of hybrid dynamical models. *Automatica*, vol. 37(7):1085–1091.

Hoffmann G., Huang H., Waslander S., Tomlin C. (2011): Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, vol. 19(9):1023–1036.

Hovd, M. and Olaru, S.(2010): Piecewise quadratic Lyapunov functions for stability verification of approximate explicit MPC. *Modeling, Identification and Control*, vol. 31(2):45–53.

How J., King E., Kuwata Y. (2004): Flight demonstrations of cooperative control for uav teams. *In AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, USA, pp. 20–23.

Keviczky T., Balas G. (2006): Software-enabled receding horizon control for autonomous unmanned aerial vehicle guidance. *Journal of Guidance Control and Dynamics*, vol. 29(3):680–694.

Kim H., Shi, D., Sastr, S. (2002): Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. *In Proceedings of the 21st IEEE American Control Conference*, Anchorage, Alaska, USA, pp. 3576–3581.

Kothare, M.V. and Mettler, B. and Morari, M. and Bendotti, P. and Falinower, C.-M. (1997): Linear parameter varying model predictive contr for steam generator level control. *Computers & Chemical Engineering*, vol. 21:S861–S866.

Lakshmanan, N.M. and Arkun, Y. (1999): Estimation and model predictive control of non-linear batch processes using linear parameter varying models. *International Journal of Control*, vol. 72(7–8):659–675.

Lévine J. (2009): Analysis and control of nonlinear systems: A flatness-based approach. *Springer*.

Li Z., Canny J. (1993): Nonholonomic motion planning. Kluwer Academic Pub. Vol. 192.

Martins R., Dias P., Marques E., Pinto J., Sousa J., Pereira F. (2009): IMC: A communication protocol for networked vehicles and sensors. *In IEEE Oceans Europe*, pp. 1–6.

Mayne D., Kerrigan E., Falugi P. (2011): Robust model predictive control: advantages and disadvantages of tube-based methods. *In Proceeding of the 18th IFAC World Congress*, Milan, Italy, pp. 148–153.

Muttin F. (2011): Umbilical deployment modeling for tethered uav detecting oil pollution from ship. *Applied Ocean Research*, vol. 33(4):332–343.

Østergaard K., Brath P., Stoustrup J. (2007): Estimation of effective wind speed. *Journal of Physics: Conference Series*, vol. 75:12–82.

Patterson T., McClean S., Parr G., Morrow P., Teacy L., Nie J. (2011): Integration of terrain image sensing with UAV safety management protocols. *Sensor Systems and Software*, pp.36–51.

Pinto J., Calado P., Braga J., Dias P., Martins R., Marques E. (2012): Implementation of a control architecture for networked vehicle systems. *In Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*, Porto, Portugal, pp. 23–28.

Prodan I., Olaru S., Stoica C., Niculescu S.-I. (2011): Predictive control for tight group formation of multi-agent systems. *In Proceeding of the 18th IFAC World Congress*, Milano, Italy, pp. 138–143.

Rakovic S., Kouvaritakis B., Cannon M., Panos C., Findeisen R. (2011): Fully parameterized tube mpc. *In Proceedings of the 18th IFAC World Congress*, Milano, Italy, pp. 197–202.

Reyhanoglu M., van der Schaft A., McClamroch N., Kolmanovsky, I. (1999): Dynamics and control of a class of underactuated mechanical systems. *IEEE Transactions on Automatic Control*, vol. 44(9):1663–1671.

Rosenbaum D., Kurz F., Thomas U., Suri S., Reinartz P. (2009): Towards automatic near real-time traffic monitoring with an airborne wide angle camera system. *European Transport Research Review*, vol. 1(1):11–21.

Rouchon P., Martin P., Murray R. (2003): Flat systems, equivalence and trajectory generation. Technical report, California Institute of Technology, Pasadena, Calif, USA.

Schouwenaars T., Valenti M., Feron E., How J. (2005): Implementation and flight test results of milp-based uav guidance. *In Proceedings of the IEEE Aerospace Conference*, USA, pp. 1–13.

Silva J., Terra B., Martins R., de Sousa J. (2007): Modeling and simulation of the lauv autonomous underwater vehicle. *In Proceedings of the 13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics*, Szczecin, Poland, pp. 51–55.

Soares J., Aguiar A., Pascoal A., Gallieri M. (2012): Triangular formation control using range measurements: An application to marine robotic vehicles. *In Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*, Porto, Portugal, pp.15–20.

Sontag E. (1981): Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, vol. 26(2):346–358.

Suryawan F., De Dona J., Seron M. (2010): Methods for trajectory generation in a magnetic-levitation system under constraints. *In Proceedings of the 18th Mediterranean Conference on Control and Automation*, Marrakech, Morocco, pp. 945–950.

Ulbig A., Olaru S., Dumur D., Boucher P. (2010): Explicit nonlinear predictive control for a magnetic levitation system. *Asian Journal of Control*, vol. 12(3):434–442.

Vaglienti B., Niculescu M., Becker J., Miley D. (2011): Piccolo system user's guide. Cloud Cap Technology (www.cloudcaptech.com).

Valavanis K., (2007): Advances in unmanned aerial vehicles: state of the art and the road to autonomy. *Springer*, vol. 33.

Valenti M., Bethke B., How J., de Farias D., Vian J. (2007): Embedding health management into mission tasking for uav teams. *In Proceedings of the 26th American Control Conference*, New York, USA, pp. 5777–5783.

Van Nieuwstadt M., Murray R. (1998): Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, vol. 8(11):995–1020.

Zheng Z., Huo, W., Wu Z. (2013): Autonomous airship path following control: Theory and experiments. *Control Engineering Practice*, vol. 21(6):769–788.