Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
BIOMEDICAL, ELECTRICAL & SYSTEM ENGINEERING

Ciclo XXXV

Settore Concorsuale
09/G1 AUTOMATICA

Settore Scientifico Disciplinare
ING-INF/04 AUTOMATICA

ON THE MOTION PLANNING & CONTROL OF NONLINEAR ROBOTIC SYSTEMS

by LORENZO GENTILINI

Coordinatore Dottorato
PROF. MICHELE MONACI

Supervisore
PROF. LORENZO MARCONI

Esame Finale - Anno 2023

FACULTY:
Ph.D. in Biomedical, Electrical & System Engineering
Alma Mater Studiorum - Università di Bologna
Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" (DEI)

SUPERVISOR:
Prof. Lorenzo Marconi

LOCATION:
Bologna, Italy

ABSTRACT

In the last decades, we saw a soaring interest in autonomous robots boosted not only by academia and industry, but also by the ever increasing demand from civil users. As a matter of fact, autonomous robots are fast spreading in all aspects of human life, we can see them clean houses, navigate through city traffic, or harvest fruits and vegetables. Almost all commercial drones already exhibit unprecedented and sophisticated skills which makes them suitable for these applications, such as obstacle avoidance, simultaneous localisation and mapping, path planning, visual-interial odometry, and object tracking. The major limitations of such robotic platforms lie in the limited payload that can carry, in their costs, and in the limited autonomy due to finite battery capability. For this reason researchers start to develop new algorithms able to run even on resource constrained platforms both in terms of computation capabilities and limited types of endowed sensors, focusing especially on very cheap sensors and hardware. The possibility to use a limited number of sensors allowed to scale a lot the UAVs size, while the implementation of new efficient algorithms, performing the same task in lower time, allows for lower autonomy. However, the developed robots are not mature enough to completely operate autonomously without human supervision due to still too big dimensions (especially for aerial vehicles), which make these platforms unsafe for humans, and the high probability of numerical, and decision, errors that robots may make. In this perspective, this thesis aims to review and improve the current state-of-the-art solutions for autonomous navigation from a purely practical point of view. In particular, we deeply focused on the problems of robot control, trajectory planning, environments exploration, and obstacle avoidance. The proposed methodology enbraces a control theoretic view, where control and planning algorithms are designed to approach the system theory field. Under this point of view, this thesis aspires to create a bridge between control system approaches and robotics problems to let both fields borrow useful tools to improve domain-specific solutions.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

[1] I. A. Azzollini, N. Mimmo, L. Gentilini, and L. Marconi. "UAV-Based Search and Rescue in Avalanches using ARVA: An Extremum Seeking Approach." In: *arXiv preprint arXiv:2106.14514* (2021).

[2] L. Gentilini, M. Bin, and L. Marconi. "Adaptive Nonlinear Regulation via Gaussian Process." In: *arXiv preprint arXiv:2206.12225* (2022).

[3] L. Gentilini, M. Bin, and L. Marconi. "Data-driven Output Regulation via Gaussian Processes and Luenberger Internal Models." In: *arXiv preprint arXiv:2210.15938* (2022).

[4] L. Gentilini, D. Mengoli, and L. Marconi. "Direct Bézier-Based Trajectory Planner for Improved Local Exploration of Unknown Environments." In: *arXiv preprint arXiv:2203.00968* (2022).

[5] L. Gentilini, S. Rossi, D. Mengoli, A. Eusebi, and L. Marconi. "Trajectory Planning ROS Service for an Autonomous Agricultural Robot." In: *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE. 2021, pp. 384–389.

[6] B. Trimarchi, L. Gentilini, F. Schiano, and L. Marconi. "Data-Driven Analytic Differentiation via High Gain Observers and Gaussian Process Priors." In: *arXiv preprint arXiv:2210.15528* (2022).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CBF	Control Barrier Function
ECBF	Exponential Control Barrier Function
ESDF	Euclidean Signed Distance Field
EKF	Extended Kalman Filter
FLANN	Fast Library for Approximate Nearest Neighbor
FOV	Field-Of-View
GP	Gaussian Process
GPS	Global Positioning System
GTO	Gradient-based Trajectory Optimisation
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IPP	Informative Path Planning
LTI	Linear Time-Invariant
MPC	Model Predictive Control
NBV	Next Best View
NURBS	Non Uniform Rational B-Spline
OCP	Optimal Control Problem
PGO	Path-Guided Optimisation
PMF	Probability Mass Function
QP	Quadratic Programming
RRT	Rapidly-Exploring Random Tree
RKHS	Reproducing Kernel Hilbert Space
SE	Special Euclidean
SLAM	Simultaneous Localisation And Mapping
SO	Special Orthogonal
SOR	Subset Of Regressors

SVM Support Vector Machine

UAV Unmanned Aerial Vehicle

UVD Uniform Visibility Deformation

VTOL Vertical-TakeOff-and-Landing

INTRODUCTION TO THE THESIS

This thesis is born from the combination in different shapes of three main research branches. On one side, we have a completely robotic view with the development of a fully autonomous aerial vehicle, while on the other side we have a study about nonlinear control techniques especially focused in the field of output regulation. The link between these two topics is represented by an unsupervised learning technique known as Gaussian process regression. The ambitious objective of this thesis is to join the two fields of robotics and control theory in order to let both of them borrow tools from its counterpart to improve domain-specific solutions. Being this goal very ambitious, the thesis aims to start building this link from the ground, using the learning tool as the initial bridge. In this scenario, we started our discussion by reviewing the main problems encountered when approaching the autonomous navigation problem, from the modeling and control, to the trajectory planning, and ending with autonomous exploration. For each proposed chapter, namely each faced problem, we proposed a literature review, followed by the implementation and testing of one of the most promising state-of-the-art algorithms, then we try to question the major limitations of such approach, proposing, when possible, novel solutions focused on improving the detected cons. The reader will find some pure robotics solutions as well as control theory oriented ones, which represent, under the aforementioned perspective, the main proposed vision of this thesis. Approaching the end, we discuss advanced control techniques involving output regulation tools to design robust control laws able to face the noisy and varying nature of real robot applications. The difficulty in establishing actuator saturations makes these approaches not mature enough to be applied in real contexts yet. All the discussion is seasoned with a strong use of Gaussian process regression that turns out to be a fundamental tool to deal with the high uncertainties affecting physical models and with the time-varying external disturbances.

The thesis unfolds as follows, Chapter 1 briefly analyses the motivations behind this work, with an eye to the proposed project where the Ph.D. has been developed, and describes the benchmark platform used to test the developed algorithms, while Chapter 2 recaps the "golden standard" in quadrotor modeling and control. Chapters 3, 4, 5, and 6 are devoted to discuss, analyse, and implement solutions to the four basic problems in autonomous navigation, i.e. localisation and mapping, environment exploration, trajectory planning, and obstacle avoidance. Finally Chapter 7 proposes a new approach to the output regulation problem, with an eye to the applica-

tion in the robotic control framework. Chapter 7.6 closes the thesis with some final considerations.

Part I

QUADROTOR MODELING AND CONTROL

1 INTRODUCTION

This thesis found a place inside the big and ambitious topic of motion planning, decision making, and control of highly nonlinear robotic systems, with particular attention to the case of quadrotor flight platforms. In this context, we aim to discuss and review the main problems which arise when approaching this field and try to contribute by proposing novel solutions enabling the possibility to safely use these kinds of robots in humans' everyday life. In the next chapters, we follow a tight golden line that allows touching each aspect of this field starting from the used hardware, to the problem of mapping and localisation, ending with the motion planning, obstacle avoidance, and advanced control techniques that make the robot acts safely in any conditions. This chapter unfolds as follows, in Section 1.1 and Section 1.2 we briefly analyse the motivations behind this work, with an eye to the proposed project where the Ph.D. has been developed. In Section 1.3 we describe the used UAV hardware, pointing out its sensing capabilities and the main challenges that emerge from its usage in cluttered environments, finally in Section 1.4 we recap the main contributions of this dissertation.

1.1 MOTIVATIONS

In the last decades, we saw a soaring interest in autonomous robots boosted not only by academia and industry, but also by the ever increasing demand from civil users. As a matter of fact, autonomous robots are fast spreading in all aspects of human life, we can see them clean houses, navigate through city traffic, or harvest fruits and vegetables. This trend is motivated by the fact that autonomous robots can assist humans in a plethora of daily activities ranging from transportation and surveillance, to handling heavy loads and inspection. In particular, UAVs can perform aerial inspection of industrial facilities or hazardous areas [110], surveillance and monitoring of conurbations [43], crowded public places and warfare zones. The automatisation of such activities can improve their quality, efficiency, and effectiveness, especially when performed by robots designed to use the most advanced technologies in analyzing and understanding the surrounding environment, such as LiDAR laser scanners [95], RGB-D and thermographic cameras [85], event-based cameras [149], and others more. Offloading error-prone, and potentially dangerous activities to robots that can automatically cope with them also increases the quality of life for the human operators. In this respect, UAVs can

contribute to preventing critical situations and optimizing the management of urban environments [151, 152], performing search and rescue missions in hazardous scenarios [1], performing film shooting in dangerous areas [70], and monitoring cultivated fields [144]. In order to perform the aforementioned tasks, the UAV platform must be endowed with a high level of onboard intelligence that process the information gathered by the carried sensors and take real-time decisions. Taking decisions is not the only workload that the onboard intelligence must sustain, as the UAV is also required to fly smoothly without colliding and often mapping and memorising the explored environment. Almost all commercial drones already exhibit unprecedented and sophisticated skills such as obstacle avoidance [68], simultaneous localisation and mapping [41], path planning [111], visual-interial odometry [49], and object tracking [104]. The major limitations of such robotic platforms lie in the limited payload that can carry, in their costs, and in the limited autonomy due to finite battery capability. Nowadays, industries try to overtake these problems by designing very oversized structures able to carry a large variety of sensors and very big batteries that can benefit from a very high autonomy. This solution is not resolutive at all since it limits the UAVs field of application to very large environments, without containing their cost and making their use unsafe for human operators. For this reason researchers start to develop new algorithms able to run even on resource constrained platforms both in terms of computation capabilities and limited types of endowed sensors, focusing especially on very cheap sensors and hardware. The possibility to use a limited number of sensors allowed to scale a lot the UAVs size, while the implementation of new efficient algorithms, performing the same task in lower time, allowed for lower autonomy. In this respect, a new wave of innovation in aerial robotics is rapidly soaring: the miniaturization of vehicles [47, 61, 159]. Insect-scale autonomous UAVs can extend the applicability of flying robotic-helpers, making them even more pervasive in everyday life. Thanks to their small form-factor, they can reach places otherwise inaccessible and increase the safety of operations in human-populated and indoor environments. Although the field of nano- and pico-size UAVs is nowadays very widespread, the available algorithms are not mature enough to cope with the limited resources and sensor data available onboard of these small platforms. Besides that, a second big problem is related to the robustness of the adopted solutions as often require fine-tuning procedures to be really effectiveness. In this respect, the focus of this thesis lies on reviewing and discussing current state-of-the-art solutions to the problem of motion planning and control specially tailored for autonomous aerial vehicles, then we aim to propose a bunch of new robust approaches focused on minimising the required computational load with an eye

Figure 1: Snapshot of the synthetic environment used during the Leonardo drone contest.

to their flexibility and applicability in a wide range of different environments.

1.2 THE LEONARDO DRONE CONTEST

The project behind the study which lead to the development of this dissertation consists of a drone competition proposed by Leonardo S.p.A. to encourage the research in the field of autonomous flight robots. The proposed contest completed five different Italian universities that developed five fully autonomous drones able to navigate and explore a completely unknown environment and perform landing or inspection operations inside that area. Leonardo builds a synthetic scenario mimicking an urban canyon of $20 \times 10 \times 3$ meters (see Figure 1) to test the proposed solutions in three different application scenarios of increasing complexity. In the following, we briefly report a description of the three proposed challenges, one for each Ph.D. study year.

The first year challenge.
In the first proposed challenge, the autonomous UAV was required to navigate and explore as fast as possible a completely unknown environment with the twice objective to generate a complete and precise map, and find out a set of ten ArUco markers [57] scattered inside the environment. The UAV then had to perform a precise sequence of landing and takeoff actions on the detected markers.

The second year challenge.
In the second challenge, the autonomous UAV was required to localise itself inside a previously mapped area, and to explore the environment to search for an autonomous ground agent moving randomly through the obstacles. Then the UAV had to track the found agent for a fixed amount of time without losing its sight, reading and recognizing an alphanumerical sequence printed on the chased robot. The read string contained the sequence of landing that the UAV was required to perform. The UAV had to collaborate with an external surveillance camera for better performing both the tasks of localisation and intrude finding.

The third year challenge.
In the third proposed challenge, the autonomous UAV was required to localise itself inside a previously mapped area, and to explore the environment to search for an autonomous ground agent moving randomly through the obstacles. During such an exploration procedure, the UAV had to avoid unmapped static obstacles that suddenly appeared inside the area. Then the UAV had to track the found ground robot for a fixed amount of time without losing its sight. Once done that, a human operator committed a sequence of landing or buildings inspection actions that the drone was required to perform.

As the reader can conceive, the proposed challenges cover almost all the aspects of the big field of autonomous navigation. As a matter of fact, the developed platform must be able to localise itself, map the surrounding environment, plan safe paths or trajectories through already mapped obstacles, and react to the unknown by replanning previously established safe paths. Moreover, the UAV must be able to reliably chase moving objects, explore unknown environments, performing precise landings, and plan highly informative trajectories to perform building inspection. The list of required capabilities is not short, and for each of them a deeper study of the current state-of-the-art had been carried out. In this thesis, the reader will find a brief review of most of the aforementioned topics along with a description of how each problem has been solved in the *Leonardo drone contest*.

1.3 THE BENCHMARK PLATFORM

The quadcopter deployed during the Leonardo drone contest (see Figure 2) is a commercial drone *Holybro X500* customized for our particular application, of dimension $750 \times 750 \times 370$ millimeters, with a weight of 1530 grams, and is powered with a 6200 *mAh* battery. The quadcopter is endowed with a *Pixhawk 4* [101] computational unit running the *PX4 Autopilot* [100]. Besides this unit, the UAV car-

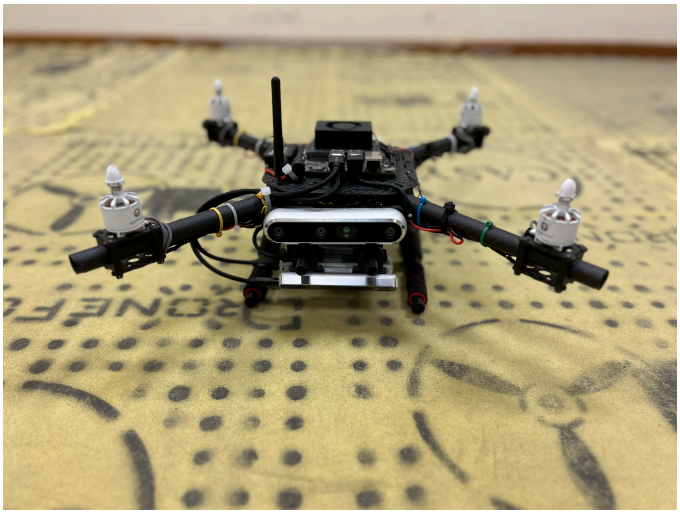


Figure 2: Developed drone used during the Leonardo drone contest.

ries a *Jetson Xavier NX* companion computer responsible for running all control, planning, localisation, and perception algorithms. All the developed algorithms are tightly integrated inside a ROS network and exchange messages with the Pixhawk computational unit via *MAVLink* interface [83]. The chosen sensor suite consists of a 9-axis Inertial Measurement Unit (IMU), inclusive of accelerometers, gyroscopes, and magnetometers, and two stereo cameras pointing backward and forward, respectively. The sensor suite is intentionally poor and GPS free, to let the robot navigate in GPS-denied environments such as indoor areas, and to keep the overall platform cost low. The overall quadcopter comprising of all sensors and computational units weights 1960 grams and has a payload of 500 grams at 60% of throttle.

1.4 CONTRIBUTIONS

The contribution of this thesis is twofold, on one side it is meant to present and describe a practical software solution to the problem of autonomous navigation in unknown, or partially known, environments. Such a solution has been extensively tested and validated in real scenario experiments and presented as a final UAV architecture during the Leonardo drone contest (see Section 1.2). The proposed solution is often built upon existing stat-of-the-art algorithms properly modified and robustified to cope with the strong real-time and reliability requirements imposed by the contest. A reader only interested in the aforementioned architecture hardly finds a smooth discussion of all software modules. The thesis presentation is in fact intentionally left unstructured, although the chapters sequence remark the localisation-planning-control pipeline, each chapter decomposes the problem at hand and in addition to providing a practical solution, it aspires to present innovative ideas and contributions. This is the

second contribution of this thesis that, for each aspect of autonomous navigation, tries to improve the current state-of-the-art by leveraging on solutions which reduce, or eliminate, the major limitations of the most popular algorithms.

2

QUADROTOR MODELING AND CONTROL

In this chapter we briefly collect and review the "golden standards" in quadcopter modeling and control, borrowing the formalism and the results from [42, 44, 75, 102, 128, 137, 138]. The chapter unfolds as follows, in Section 2.1 we develop the quadrotor dynamical model, in Section 2.2 we describe a very useful property of the developed model known as *differential flatness*. Then in Section 2.3 we briefly describe two possible quadcopter control approaches. This chapter is intentionally poor in terms of scientific contribution as it is meant to introduce the reader to the complex world of quadcopter motion planning and control.

2.1 QUADROTOR MODELING

Let $\mathcal{I} = \{e_1^{\mathcal{I}}, e_2^{\mathcal{I}}, e_3^{\mathcal{I}}\}$ denotes a right-hand inertial frame stationary with respect to the earth and such that $e_3^{\mathcal{I}}$ denotes the vertical direction downwards into the earth. Let the vector $\boldsymbol{\xi} = (x, y, z)^\top \in \mathbb{R}^3$ denotes the position of the centre of mass of the object in the frame $\mathcal{I}$ relative to a fixed origin $O_{\mathcal{I}} \in \mathbb{R}^3$. Let $\mathcal{B} = \{e_1^{\mathcal{B}}, e_2^{\mathcal{B}}, e_3^{\mathcal{B}}\}$ be a body-fixed reference frame whose center coincides with the center of mass of the vehicle and such that $e_3^{\mathcal{B}}$ is in the opposite direction of thrust generation. The attitude of the body-fixed frame is represented by a rotation matrix $R \in SO(3) : \mathcal{B} \mapsto \mathcal{I}$, with $SO(3)$ the Special Orthogonal group of dimension 3. Applying the Newton-Euler equations to the system, it is possible to retrieve the translational and rotational kinematics

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v}, \\ \dot{R} &= RS(\boldsymbol{\omega}), \end{aligned} \tag{1}$$

with $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ denoting the vector of coordinates of the angular velocity and the linear vehicle velocity with respect to $\mathcal{I}$, while $S(\boldsymbol{\omega})$ denotes the skew-symmetric matrix associated with the vector $\boldsymbol{\omega}$, and the translational and rotational dynamics

$$\begin{aligned} \dot{\boldsymbol{v}} &= m^{-1} f R e_3^{\mathcal{I}} - g e_3^{\mathcal{I}}, \\ \dot{\boldsymbol{\omega}} &= -S(\boldsymbol{\omega})\boldsymbol{\omega} + J^{-1}\boldsymbol{\tau}, \end{aligned} \tag{2}$$

with $m \in \mathbb{R}$ and $J \in \mathbb{R}^{3\times 3}$ being the quadrotor mass and inertia matrix with respect to the frame $\mathcal{B}$, $g \in \mathbb{R}$ the gravity acceleration, $f \in \mathbb{R}$ the collective thrust generated by the four rotors, and $\boldsymbol{\tau} \in \mathbb{R}^3$ the torque vector expressed in the frame $\mathcal{B}$. The system inputs

are the collective thrust f and the torque vector $\boldsymbol{\tau}$ which are directly generated by the rotation of the four rotors. In particular, each rotor has an angular speed ω_i and produces a force, F_i, and moment, M_i, according to

$$F_i = k_F \omega_i^2, \quad M_i = k_M \omega_i^2,$$

therefore the control input to the system can be expressed as

$$\begin{pmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{pmatrix} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix},$$

with $L \in \mathbb{R}$ the distance from the axis of rotation of the rotors to the center of the quadrotor. For further details about the modeling of the constants $k_F \in \mathbb{R}$ and $k_M \in \mathbb{R}$ the reader is referred to [75]. Equations (1) and (2) express the system dynamics with the aid of the rotational matrix R, the same equation can be equivalently expressed using the quaternion dynamics as

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v}, \\ \dot{\boldsymbol{q}} &= 0.5\Lambda(\boldsymbol{\omega})\boldsymbol{q}, \\ \dot{\boldsymbol{v}} &= m^{-1} f R(\boldsymbol{q}) e_3^{\mathcal{I}} - g e_3^{\mathcal{I}}, \\ \dot{\boldsymbol{\omega}} &= -S(\boldsymbol{\omega})\boldsymbol{\omega} + J^{-1}\boldsymbol{\tau}, \end{aligned} \tag{3}$$

with

$$R(\boldsymbol{q}) = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & 1 - 2(q_x^2 + q_y^2) \end{pmatrix},$$

and the matrix $\Lambda \in \mathbb{R}^{4\times 4}$ defined as

$$\Lambda(\boldsymbol{\omega}) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}.$$

In the aforementioned relation $\boldsymbol{q} \in \mathbb{H}$ is the normed quaternion attitude vector. Overall the state of the system is given by the position and velocity of the center of mass and the orientation (locally parameterized by Euler angles) and the angular velocity

$$\boldsymbol{x} = \left(x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z\right)^\top.$$

2.2 DIFFERENTIAL FLATNESS

In this section we recall the results of [102] showing that the quadrotor dynamics (1)-(2) is differentially flat [148], i.e. the states and the inputs can be written as algebraic functions of four carefully selected flat outputs and their derivatives. This is a fundamental result as it can ease a lot the process of trajectory generation and optimisation, since any smooth trajectory (with reasonably bounded derivatives) in the space of flat outputs can be followed by the underactuated quadrotor. Let $\sigma = (x, y, z, \psi)^\top$ the selected set of flat outputs, with $\boldsymbol{\xi} = (x, y, z)^\top$ coordinates of the center of mass in the world coordinate system and with ψ the yaw angle, and define $\sigma(t)$ as a smooth curve in the space of flat outputs

$$\sigma(t) : [0, T_F] \mapsto \mathbb{R}^3 \times SO(2),$$

then the objective is to show that the state of the system and its control inputs can be rewritten in terms of $\sigma(t)$ and its derivatives. First of all, the position and velocity of the center of mass are simply the first three terms of σ and $\dot{\sigma}$. In order to reconstruct R as a function of the selected flat outputs, let us define it as $R = {}^{\mathcal{I}}R_{\mathcal{C}}\, {}^{\mathcal{C}}R_{\mathcal{B}}$ where ${}^{\mathcal{I}}R_{\mathcal{C}}$ represents the yaw rotation to the intermediate frame $\mathcal{C}$ and ${}^{\mathcal{C}}R_{\mathcal{B}}$ represents the effect of roll and pitch. In this setting, considering Equation (2), we can write

$$e_3^{\mathcal{B}} = \frac{m}{f}\left(\ddot{\sigma}_x, \ddot{\sigma}_y, \ddot{\sigma}_z + g\right), \tag{4}$$

which defines the body frame z-axis of the quadrotor. From the yaw angle, $\sigma_4 = \psi$, we can derive the unit vector

$$e_1^{\mathcal{C}} = \left(\sin\left(\sigma_4\right), \cos\left(\sigma_4\right), 0\right)^\top,$$

while $e_1^{\mathcal{B}}$ and $e_2^{\mathcal{B}}$ can be determined as

$$e_2^{\mathcal{B}} = \frac{e_3^{\mathcal{B}} \times e_1^{\mathcal{C}}}{\left\| e_3^{\mathcal{B}} \times e_1^{\mathcal{C}} \right\|}, \qquad e_1^{\mathcal{B}} = e_2^{\mathcal{B}} \times e_3^{\mathcal{B}},$$

provided that $e_3^{\mathcal{B}} \times e_1^{\mathcal{C}} \neq 0$. Finally, the rotational matrix R is uniquely determined as

$$R = \left(e_1^{\mathcal{B}}, e_2^{\mathcal{B}}, e_3^{\mathcal{B}}\right).$$

Now, take the first derivative of Equation (2)

$$\ddot{\boldsymbol{v}} = m^{-1}\left(\dot{f} e_3^{\mathcal{B}} + \boldsymbol{\omega} \times f e_3^{\mathcal{B}}\right),$$

projecting this equation along $e_3^{\mathcal{B}}$ and using the fact that $\dot{f} = e_3^{\mathcal{B}} \cdot m\ddot{\boldsymbol{v}}$, we can define the vector $\boldsymbol{h} \in \mathbb{R}^3$ as

$$\boldsymbol{h} = \boldsymbol{\omega} \times e_3^{\mathcal{B}} = \frac{m}{f}\left(\ddot{\boldsymbol{v}} - \left(e_3^{\mathcal{B}} \cdot \ddot{\boldsymbol{v}}\right) e_3^{\mathcal{B}}\right).$$

In this settings, $\boldsymbol{h}$ is the projection of $\frac{m}{f}\ddot{\boldsymbol{v}}$ along the $e_1^{\mathcal{B}} - e_2^{\mathcal{B}}$ plane. Thus, decoposing $\boldsymbol{\omega}$ into its components $\boldsymbol{\omega} = pe_1^{\mathcal{B}} + qe_2^{\mathcal{B}} + re_3^{\mathcal{B}}$, we can write

$$
\begin{aligned}
p &= -\boldsymbol{h} \cdot e_2^{\mathcal{B}}, \\
q &= \boldsymbol{h} \cdot e_1^{\mathcal{B}}, \\
r &= \dot{\sigma}_4 e_3^{\mathcal{I}} \cdot e_3^{\mathcal{B}}.
\end{aligned}
$$

Finally, the components of the angular acceleration $\boldsymbol{\alpha}$ are found by computing the second of derivative of Equation (2) and following the same procedure as above. Having σ, $\boldsymbol{\omega}$, and $\boldsymbol{\alpha}$ at hand we can exploit Equation (2) and Equation (4) to directly compute the inputs f and $\boldsymbol{\tau}$.

2.3 QUADROTOR CONTROL

The literature is cluttered with works about the stabilisation and control of aerial vehicles such as quadcopters. In the first stage, given the unstable nature of the quadrotors, the initial works were focused on achieving stable hovering and near-hover flights. Thanks to the small-angle assumptions in these conditions, linear control methods such as PID and LQR demonstrate sufficiently good performance [37, 79]. However, as increased the necessity to push these platforms toward the boundaries of their dynamical capabilities, these assumptions were no longer valid. In particular, the nonlinearities coming from the attitude dynamics were no longer negligible, for this reason researchers started developing controllers based on feedback linearization [153], backstepping [92], and geometric properties [87]. Once the differential flatness property has been revealed [102], the differential flatness-based controller becomes the most used regulator for trajectory tracking, as it showed the best tracking performance at relatively high speeds [42, 138]. Besides that, recently, some studies started using Model Predictive Controls (MPCs), jointly with the full quadrotor dynamics, to compute optimal control inputs able to both stabilise the quadcopter flight and track a given reference trajectory [12, 48, 142]. These methods either directly use the optimized single rotor thrust commands [12] or send intermediate states from the solution (such as the angular rates) to a low-level controller [48, 142]. A recent study [48] demonstrates the ability of the full-model MPC with a PID low-level controller in tracking a pre-planned race trajectory at speed up $20m/s$ which surpasses the top speed of $12.9m/s$ reported in [138] using a differential flatness-based controller, in spite of a much larger tracking error.

In this chapter, we briefly review these two main used control techniques based on the flatness property explained in Section 2.2 and on the MPC tool. In the particular case of the Leonardo drone contest, we employed a non-linear model predictive controller encoding the full quadrotor dynamics (3).

2.3.1 *Differential Flatness-Based Controller*

Let define the errors on position and velocity as

$$\boldsymbol{e}_{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}_{\text{ref}}, \quad \boldsymbol{e}_{v} = \boldsymbol{v} - \boldsymbol{v}_{\text{ref}},$$

then the desired force vector is

$$F_{\text{des}} = -K_{\xi}\boldsymbol{e}_{\xi} - K_{v}\boldsymbol{e}_{v} + mge_3^{\mathcal{I}} + m\dot{\boldsymbol{v}}_{\text{ref}},$$

where K_{ξ} and K_v are positive definite gain matrices. In order to compute the desired force for the quadrotor, that correspond to the first input f, we simply project the desired force vector onto the actual body frame z-axis

$$f = F_{\text{des}} \cdot e_3^{\mathcal{B}}.$$

To determine the other three inputs, we must consider the rotation errors. First, observe that the desired $e_3^{\mathcal{B}}$ direction is along the desired thrust vector

$$e_{3_{\text{des}}}^{\mathcal{B}} = \frac{F_{\text{des}}}{\|F_{\text{des}}\|},$$

thus the desired rotation is given by

$$R_{\text{des}}e_3 = e_{3_{\text{des}}}^{\mathcal{B}},$$

with $e_3 = (0, 0, 1)^{\top}$. Knowing the specified yaw angle along the trajectory, $\psi(t)$, we can compute $e_{1_{\text{des}}}^{\mathcal{B}}$ and $e_{2_{\text{des}}}^{\mathcal{B}}$ as

$$e_{1_{\text{des}}}^{\mathcal{C}} = (\cos(\psi), \sin(\psi), 0)^{\top},$$

and

$$e_{2_{\text{des}}}^{\mathcal{B}} = \frac{e_{3_{\text{des}}}^{\mathcal{B}} \times e_{1_{\text{des}}}^{\mathcal{C}}}{\left\| e_{3_{\text{des}}}^{\mathcal{B}} \times e_{1_{\text{des}}}^{\mathcal{C}} \right\|}, \quad e_{1_{\text{des}}}^{\mathcal{B}} = e_{2_{\text{des}}}^{\mathcal{B}} \times e_{3_{\text{des}}}^{\mathcal{B}}.$$

provided that $e_{3_{\text{des}}}^{\mathcal{B}} \times e_{1_{\text{des}}}^{\mathcal{C}} \neq 0$. Next, define the error in orientation

$$\boldsymbol{e}_R = 0.5\left(R_{\text{des}}^{\top}R - R^{\top}R_{\text{des}}\right)^{\wedge},$$

where $R_{\text{des}} = \left(e_{1_{\text{des}}}^{\mathcal{B}}, e_{2_{\text{des}}}^{\mathcal{B}}, e_{3_{\text{des}}}^{\mathcal{B}}\right)$ and $^{\wedge}$ represents the *vee map* which takes elements of $so(3)$ to $\mathbb{R}^3$. The angular velocity error is simply the difference between the actual and desired angular velocity in body frame

$$\boldsymbol{e}_{\omega} = \boldsymbol{\omega} - \boldsymbol{\omega}_{\text{ref}},$$

then the desired moments can be computed as

$$\boldsymbol{\tau} = -K_R\boldsymbol{e}_R - K_{\omega}\boldsymbol{e}_{\omega},$$

where K_R and K_{ω} are diagonal gain matrices.

2.3.2 *Nonlinear Model Predictive Controller*

Model predictive control generates control commands by solving a finite-time Optimal Control Problem (OCP) in a receding horizon fashion. Given a reference trajectory, the cost function is the error between the predicted states and the reference states inside the time horizon, meaning that multiple reference points in the time horizon are used. In order to perform numerical optimizations, we discretize the states and inputs into N equal intervals over the time horizon $\rho \in [t, t+h]$ of size $\Delta_t = h/N$ with h denoting the horizon length, yielding a constrained nonlinear optimization problem

$$\begin{aligned} \boldsymbol{u} = \arg\min_{\boldsymbol{u}} \sum_{k=0}^{N-1} \left(\|\boldsymbol{x}_k - \boldsymbol{x}_{k,\text{ref}}\|_Q + \|\boldsymbol{u}_k - \boldsymbol{u}_{k,\text{ref}}\|_R \right) + \|\boldsymbol{x}_N - \boldsymbol{x}_{N,\text{ref}}\|_{Q_N} \\ sub.to.\ \ \boldsymbol{x}_{k+1} = f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \\ \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \\ \boldsymbol{u} \in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right], \\ \boldsymbol{\omega} \in \left[\boldsymbol{\omega}_{\min}, \boldsymbol{\omega}_{\max}\right], \end{aligned}$$

where the state vector is defined as $\boldsymbol{x} = \left(\boldsymbol{\xi}^\top, \boldsymbol{v}^\top, \boldsymbol{q}^\top, \boldsymbol{\omega}^\top\right)^\top$, the input vector as $\boldsymbol{u} = \left(f, \boldsymbol{\tau}^\top\right)^\top$, and Q, R, and Q_N are positive definite matrices that shape as

$$Q = \text{diag}\left(Q_\xi, Q_v, Q_q, Q_\omega\right), \quad R = \text{diag}\left(Q_f, Q_\tau\right), \quad Q_N = Q.$$

The reference state vector $\boldsymbol{x}_{\text{ref}}$ and input $\boldsymbol{u}_{\text{ref}}$ can be obtained from a trajectory generator procedure as described in the next chaptes, while the function $f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)$ is the discretized version of the full nonlinear quadrotor model (3).

Remark 2.3.1. *In the above optimization problem, the following abuse of notation is used when calculating quaternion error*

$$\boldsymbol{q} - \boldsymbol{q}_{ref} = \boldsymbol{q} \otimes \boldsymbol{q}_{ref}^{-1}$$

The above MPC solves the full nonlinear model of a quadrotor, instead of resorting to a cascaded structure, or linear assumptions. The solution presented during the Leonardo drone contest resorts on this control thecnique, where the quadratic nonlinear optimization problem is solved by a Sequential Quadratic Programming (SQP) algorithm executed in real-time. The algorithm has been implemented using ACADO [o] toolkit with qpOASES [o] as solver.

Part II

AUTONOMOUS MOTION PLANNING

3 ENVIRONMENT MAPPING AND LOCALISATION

3.1 THE FRAMEWORK

The problem of Simultaneous Localisation And Mapping (SLAM) is related to the issue of concurrently estimating the robot position, in a *local* or *global* reference system, and mapping the obstacles, or visual clues, present in the environment under exploration. SLAM is the process by which a robot builds a map of the environment and, at the same time, uses this map to compute its location, as the reader can conceive such a problem undergoes the so-called *chicken-egg* dilemma as a map is needed to perform localisation, but a pose estimation is necessary to build the map. To enable planning and navigation a reliable and resilient localisation is of fundamental importance as it can provide the reference systems necessary to ground the onboard drone intelligence. Such a problem has been widely studied and a large number of solutions have been presented along the literature, for a large number of different sensors, starting from frame monocular [49, 107, 125] and stereo cameras [20, 58, 108], to event cameras [150], and LiDAR sensors [19, 80, 162, 163, 168]. In the specific case of the Leonardo drone contest (see Section 1.2), the autonomous robot was required to perform SLAM with some apriori information at hand, as the environment under exploration was already mapped by means of user-friendly tools, and a three-dimensional geometrical reconstruction was made available for the robot. In these settings, the only way to localise inside the precomputed environment reconstruction was to exploit the endowed stereo camera to build a local pointcloud, via stereo matching, and match it with the provided 3D map, in a purely LiDAR SLAM paradigm [19, 80, 162, 163, 168]. At the same time, visual SLAM approaches, as well as the build pointcloud, could have been used to enrich the initial map with new details useful to improve the localisation stability. Following the latter intuition, we choose to deploy two different SLAM solutions working in parallel on two different sets of data. On one side, we employ state-of-the-art visual-inertial-based SLAM approaches to estimate the robot motion with respect to an *odometry* frame [107, 125], by means of hardware-calibrated stereo images, while, on the other hand, we borrow from the LiDAR SLAM field the idea of pointcloud matching to precisely align the sensed pointcloud with the apriori provided map.

In this chapter we review and adapt a novel pointcloud matching algorithm, designed for LiDAR-based localisation, to the case of quadrotor localisation in cluttered environments using limited FoV

sensors, such as stereo cameras. For further details the reader is referred to the original work [19].

3.1.1 *Related Works*

The literature is cluttered with works trying to solve both the problems of LiDAR odometry and SLAM, although very few of them deeply discuss the problem of map-based localisation, considering it a sub-class of problem strictly related to the SLAM one. A recent comprehensive review of LiDAR-based localisation solutions for ground vehicles can be found in [39]. The basic idea, on which most of the solutions are grounded, is to register (aka match) the current LiDAR reading into a local submap, being it composed by the last LiDAR measure or a collection of data from previously registered points. As the reader can conceive, matching every point of the LiDAR data with every point of the current map is a process that quickly grows and reaches infeasibility due to the limited computation capabilities available, especially onboard drones. To overtake this limitation, most state-of-the-art solutions make use of geometric features such as corners or planes directly extracted from the sensors pointcloud [26, 88, 168]. The key idea is to select points or areas that are easy to identify and match when reviewed again from a different perspective, point of view, or distance. LOAM [168] exploits exactly this idea, as it extracts point features on sharp edges and planar surfaces and matches them between consecutive scans. On the other hand [38] uses a similar approach, pointcloud segments and descriptors are used to perform localisation; while MULLS [115] tries to increase the localisation accuracy by combining a feature-based front-end SLAM with a multi-cetric ICP for map alignment. Although feature-based LiDAR-to-map registration approaches show very good results with low computational burden, they may fail in poor texture scenes, or when multiple occlusions happen. Iterative Closest Points (ICP)-based approaches [24] showed better performance and robustness with respect to the aforementioned issues, as their use the raw pointcloud for registration, and no features or interest points are extracted. This accuracy and stability come at the price of a high computational burden, especially when dealing with large pointcloud maps. Moreover, the ICP convergence strongly depends on the quality of the provided initial pose estimation. In the literature can be found a bunch of works aiming to approximate the neighbors search procedure [40, 105], which turns out to be the major computational bottleneck. While many ICP variants have been proposed to improve the global robustness against noise or bad initialization by means of new criteria as Branch-and-Bound in GO-ICP [167], or reformulating the ICP problem as Expectation-Maximization in EM-ICP [60], or as a Truncated Lest Squares in TEASER [166]. The Normal Distribution Transform

(NDT) was the first approach to propose nearest-neighbor-free solution for pointcloud registration [66, 93] allowing for very high accuracy while avoiding the computational burden induced by ICP approaches. The NDT is used to encode both scan and map using a probabilistic representation. Registration is then performed between these representations and not the original scans. Such a representation allows for numerical optimization methods, and does not require expensive nearest-neighbor search procedures. Recently, [19] proposed a direct approach for registration, where the raw pointcloud is registered inside the map in a nearest-neighbor-free fashion. The registration is reformulated as a non-linear least square optimisation problem which aims to minimise the scan points distance to the current environment map continuously maintained as a Euclidean Signed Distance Field (ESDF).

3.2 AN OPTIMIZATION-BASED LOCALIZATION APPROACH

3.2.1 *Problem Formulation*

Let $\mathcal{M} = \{e_1^{\mathcal{M}}, e_2^{\mathcal{M}}, e_3^{\mathcal{M}}\}$ denotes a right-hand inertial frame associated with the provided three-dimensional reconstruction, and let $\mathcal{S} = \{e_1^{\mathcal{S}}, e_2^{\mathcal{S}}, e_3^{\mathcal{S}}\}$ be a sensor-fixed reference frame, rigidly attached to the body-fixed one $\mathcal{B}$ by means of a static transformation ${}^{\mathcal{B}}T_{\mathcal{S}} = \left[{}^{\mathcal{B}}R_{\mathcal{S}}, \{\boldsymbol{t}_{\mathcal{S}}\}_{\mathcal{B}}\right] \in SE(3)$, with $SE(3)$ the Special Euclidean group of dimension 3. Assume that the endowed sensor, being a stereo-frame camera in the Leonardo drone contest case, streams as output a pointcloud $\{\mathcal{PC}^k\}_{\mathcal{S}} = \{\{\boldsymbol{p}_0^k\}_{\mathcal{S}}, \{\boldsymbol{p}_1^k\}_{\mathcal{S}}, \ldots, \{\boldsymbol{p}_{N_p^k}^k\}_{\mathcal{S}}\}$ at each timestamp $k \in \mathbb{R}_{\geq 0}$, consisting of N_p^k points $\{\boldsymbol{p}_i^k\}_{\mathcal{S}} \in \mathbb{R}^3$ expressing the obstacles position in the sensor reference frame $\mathcal{S}$. Moreover, assume that a map of the environment is also available as a pointcloud $\{\mathcal{PC}^k\}_{\mathcal{M}} = \{\{\boldsymbol{p}_0^k\}_{\mathcal{M}}, \{\boldsymbol{p}_1^k\}_{\mathcal{M}}, \ldots, \{\boldsymbol{p}_{N_m^k}^k\}_{\mathcal{M}}\}$, where each point $\{\boldsymbol{p}_i^k\}_{\mathcal{M}}$ is static inside the interval $[k, k+1]$. In this setting, the localisation goal is to find the transformation ${}^{\mathcal{M}}T_{\mathcal{S}} = \left[{}^{\mathcal{M}}R_{\mathcal{S}}, \{\boldsymbol{t}_{\mathcal{S}}\}_{\mathcal{M}}\right] \in SE(3)$ that better aligns $\{\mathcal{PC}^k\}_{\mathcal{S}}$ with $\{\mathcal{PC}^k\}_{\mathcal{M}}$. In other terms, the goal is to find out the minimiser of the optimisation problem

$$\underset{{}^{\mathcal{M}}T_{\mathcal{S}}}{\arg\min} \sum_{i=0}^{N_p^k} \left\| {}^{\mathcal{M}}T_{\mathcal{S}}\{\boldsymbol{p}_i^k\}_{\mathcal{S}} - \underset{\{\boldsymbol{p}^k\}_{\mathcal{M}}}{\arg\min} \left\| {}^{\mathcal{M}}T_{\mathcal{S}}\{\boldsymbol{p}_i^k\}_{\mathcal{S}} - \{\boldsymbol{p}_j^k\}_{\mathcal{M}} \right\|^2 \right\|^2, \quad (5)$$

where

$$\underset{\{\boldsymbol{p}^k\}_{\mathcal{M}}}{\arg\min} \left\| {}^{\mathcal{M}}T_{\mathcal{S}}\{\boldsymbol{p}_i^k\}_{\mathcal{S}} - \{\boldsymbol{p}_j^k\}_{\mathcal{M}} \right\|^2$$

represents the *map* point closest to ${}^{\mathcal{M}}T_{\mathcal{S}}\{\boldsymbol{p}_i^k\}_{\mathcal{S}}$. Solving the aforementioned problem needs to deal with two major challenges, first how to determine the closest map point, and then how to solve a massively

overdetermined non-linear optimization problem. ICP solutions work by iteratively searching for pairs of nearby points in the two pointclouds and minimising the sum of all point-to-point distances, while NDT approaches model the pointclouds as combinations of normal distributions instead of individual points, describing the probability of finding a point at a certain position. The latter piecewise smooth representation allows for standard numerical optimisation methods for registration. Since the major bottleneck in registration is represeted by the nearest neighbor search, NDT solutions generally work better, with high performance. For this reason, [19] follows the same NDT idea, and converts the registration process in a non-linear optimization problem where pointclouds are modeled as Euclidean distance fields. Let $d^k(\boldsymbol{p}) : \mathbb{R}^3 \mapsto \mathbb{R}$ be the ESDF build up with the map points $\{\mathcal{PC}^k\}_{\mathcal{M}}$ at time k, then (5) boils down to

$$\underset{{}^{\mathcal{M}}T_{\mathcal{S}}}{\arg\min} \sum_{i=0}^{N_p^k} d^k\left({}^{\mathcal{M}}T_{\mathcal{S}}\{\boldsymbol{p}_i^k\}_{\mathcal{S}}\right)^2. \tag{6}$$

The map $d^k(\boldsymbol{p})$ is everywhere continuous and smooth, except to the object boundaries where the gradient is discontinuous [73], so the aforementioned optimisation problem can be solved using off-the-shelf non-linear least squares solvers (eg. Ceres [8]).

Remark 3.2.1. *Due to the high nonlinearities encoded in (6), its convergence to the global optimum is ensured only in front of good enough initial conditions. To overtake this problem, a state-of-the-art visual-inertial-based SLAM approach has been employed to estimate the robot motion between two different time instants k and $k+1$, in order to supply (6) with a very good initial condition.*

Remark 3.2.2. *When dealing with (6), it is of fundamental importance to keep care about possible outliers. In particular, the sensed pointcloud may contain objects not mapped yet, such as dynamic obstacles that can cross the environment, or previously unmapped static obstacles, or may contain points that fall outside the map bounds. At the same time, the sensed pointcloud may be affected by noise and can contain many points that do not belong to any object in the scene. To overtake these problems three actions have been taken to improve the solver convergence*

1. *The sensed pointcloud is pre-elaborated with a bunch of downsampling and outliers removal filters.*
2. *A robust Cauchy kernel is applied to each loss factor in (6), in order to penalize large costs induced by outliers and not matched points (in the nearest neighbor sense).*
3. *If a point lies outside the map, the ESDF returns a zero value, producing both error and gradient equal to zero to avoid affecting the optimization process.*

3.2.2 *Signed Distance Field Computation*

As emerges from the previous section, [19] encodes the ESDF inside the optimisation problem (5) in place of the nearest neighbor search, to speed-up its solution. This choice has the advantage to avoid heavy nearest neighbor searches, at the expense of ESDF computation, which can turn out to be very computationally expensive for large environments. As a matter of fact, [19] proposes to build-up the Euclidean field offline, and assume the environment to be static, so the precomputed map is always valid. Such an assumption is clearly not satisfied in the Leonardo drone contest settings (Section 1.2), thus we require to build and adapt the signed distance online to account for environments modifications. Although along the literature there exist several different approaches to compute ESDFs, both in discrete [62, 113], and continuous settings [122, 136, 160], none of these is able to cope with prior map information, so we propose a novel structured approach able to account for previous maps and to locally modify the computed ESDF to cope with environment modifications. Unlike state-of-the-art approaches, our method assumes a bounded map and is able to add and remove newly sensed obstacles, but is unable to remove previously already mapped objects.

The proposed ESDF computation follows two steps. In the first offline stage it builds up the distance field of the apriori mapped environment, to do so the overall environment is discretised with a fixed resolution $\Delta = \left[\Delta_x, \Delta_y, \Delta_z\right]$, then each cell grid $\boldsymbol{\xi}_i \in \mathbb{R}^3$ is filled with the squared distance of the closest map point $\{\boldsymbol{p}^0\}_{\mathcal{M}}$

$$\min_{\{\boldsymbol{p}^0\}_{\mathcal{M}}} \left\| \boldsymbol{\xi}_i - \{\boldsymbol{p}_j^0\}_{\mathcal{M}} \right\|^2 .$$

The latter nearest neighbor search is performed via FLANN KDTree search [106].

Remark 3.2.3. *The built FLANN KDTree is kept maintained in memory and used later for ESDF online adaptation.*

The second step is performed online, at each new sensor update, the incoming cloud $\{\mathcal{PC}^k\}_{\mathcal{S}}$ is downsampled and filtered against possible outliers, then it is converted in map coordinate via the estimated transformation ${}^{\mathcal{M}}T_{\mathcal{S}}$ and inserted inside an octree probability occupancy map [67], build-up with the same discretisation resolution Δ. The latter insertion is fundamental to robustify the approach both in terms of sensor noise and possible localisation errors. All previously free cell grids branded as *occupied*, and all previously occupied cells branded as *free* are grouped into two vectors and used for the distance field update.

From free to occupied.
A new (very small) FLANN KDTree is build out of the provided

point vector, then each ESDF cell grid is updated with the minimum between the current cell value and the distance computed over the last built KDTree.

From occupied to free.
Two new (very small) FLANN KDTrees are build out of the provided point vector, and the current occupied voxels from the occupancy map. Then each ESDF cell grid is updated with the minimum between the original map distance and the one evaluated on the occupancy map KDTree, only if the current cell value matches with the distance evaluated on the point vector KDTree.

The first step explains the insertion procedure, which turns out to be very easy, as each cell grid is replaced with the new minimum distance, while the second step analyzes the removal procedure. In the latter step, the point vector KDTree is used to identify each map cell affected by the removal, then it is updated with the minimum between the original map distance and the distance evaluated on the new sensed obstacles.

Remark 3.2.4. *Due to the discrete nature of the adopted ESDF, the map $d^k(\boldsymbol{p})$ presents discontinuities at the cell borders, losing the required smoothness to let (6) converge properly. To overtake this limitation we approximate the real value of $d^k(\boldsymbol{p})$ via trilinear interpolation.*

Remark 3.2.5. *The overall approach can be speeded-up by restricting the occupancy points insertion to only those that do not belong to previously mapped obstacles (i.e. belong to new scene objects).*

Remark 3.2.6. *The proposed approach is not able to deal with the removal of previously mapped obstacles that have been moved, leaving the previously occupied space free. This represents a strong limitation in the case of high dynamic scene, where obstacles may continuously change their location, but on the other hand, this assumption leads to a conservative solution suitable for our case of study.*

3.3 CONTRIBUTIONS

This chapter is devoted to discuss the problem of localisation and mapping in dynamic scenarios where some apriori information, in the format of three-dimensional geometrical reconstruction, of the navigating environment was provided. The discussion unfolds by first reviewing the current state-of-the-art solutions both in terms of SLAM, occupancy mapping, and ESDF reconstruction, then we propose the adaptation of the most promising solution, to the quadcopter navigation case, where stereo-frame cameras are used to retrieve a pointcloud reconstruction of the surroundings. The selected solution

is then extended to dynamic environment scenarios via a novel structured approach to continuously update the computed ESDF. The proposed solution has the limitation of obstacle removal, as it is not able to remove objects scene already mapped inside the initial geometrical reconstruction. The proposed localisation approach has been successfully developed and deployed inside the Leonardo drone contest.

4

ENVIRONMENT EXPLORATION

4.1 THE FRAMEWORK

The ability to autonomously plan and execute informative trajectories in previously unknown, or partially known, environments is a fundamental requirement for mobile robots. As a matter of fact, they started to be employed in a huge number of different applications which require the ability to efficiently collect new information about the surroundings, such as surface inspection, object search, weed recognition, search and rescue missions, and others. The problem of Informative Path Planning (IPP), jointly with the problem of environment exploration, has been extensively studied in literature and a wide variety of approaches have been proposed so far. The majority of recent works had focused on novel hybrid approaches leveraging the interplay between the concepts of *local* and *global* exploration [130, 132]. In particular, the major issue behind such works is related to how efficiently combine the two local and global exploration steps, and how to plan highly informative global paths out of the current environment information. A limited number of papers focused on improving the local exploration step [132]. On the other hand, the necessity to push the autonomous platform always toward unknown frontiers limited the research of solutions where the environment is a priori known, or partially known, and the agent should perform patrolling-like operations [31]. In this chapter, we present two novel approaches aiming, in the first stage, to push the current state-of-the-art toward more smooth and resilient solutions for local exploration in cluttered and possibly varying environments, and, in the second stage, to adapt global exploration tools to the problem of fast environment patrolling and object search.

4.1.1 *Related Works*

Although the number of solutions presented in the literature is quite variegated, the majority of them can be classified as *frontier-based* or *sampling-based* methods. The former class was pioneered in [165] and later more comprehensively developed in [74]. The key idea is to guide the agent toward the borders between free and unmapped space (aka frontiers), since these points may represent those with higher potential information gain. Exploration is then carried out by extracting the map frontiers and by navigating through them sequentially. Several works propose to extend it by adding constraints to en-

sure low localization errors [135]. The basic frontier-based approach has been also extended to high-speed flight for fast exploration in [27]. In this case, the authors propose to extract frontiers only inside the current Field-Of-View (FoV) and select the one leading to the minimal change in velocity. In recent years, other works focused on rapid exploration [173], by planning global coverage paths and optimising them with respect to the robot dynamics, and on the reformulation of the frontier information gain as a differentiable function [34], allowing paths to be optimised with gradient information. On the other hand, *sampling-based* methods typically sample random viewpoints to explore the space in a Next-Best-View (NBV) fashion [28, 96]. Much of the work in this domain can be traced back to [59], where the NBV problem has been moved for the first time from the computer graphic field to the robotic domain, with the introduction of the notion of reconstruction gain. The concept of NBV exploration has been afterward extended by [16], where the building of a Rapidly-Exploring Random Tree (RRT) allows one to weigh both the amount of information gained at the viewpoints and during the agent motion to reach each viewpoint. Unlike frontier-based methods, which are difficult to adapt to other tasks, the sampling-based ones have the advantage to allow any kind of gain formulation. Thanks to that, the original NBV algorithm was extended to consider the uncertainty of localisation [116, 143] and the visual importance of different objects [32]. On the other hand, sampling-based methods suffer from stacking in local minima, leading to a premature ending of the exploration procedure in unlucky scenarios. For this reason, the recent trend is to merge the two frontiers and sampling-based approaches in a local-global exploration fashion. The pioneer of this idea was [22], which makes use of a frontier method to detect global goals and supplements these with motion primitives for local exploration. More recent approaches, instead, leverage the capabilities of sampling-based methods and employ additional planning stages to escape from local minima [30, 132]. Other approaches focus on memorising previously visited places and sampled information under the format of roadmaps [158, 164]. Similarly, the work presented in [130] continuously maintains and expands a single RRT of candidate paths. Although the literature has seen some impressive works in the field of NBV, there are very few works concentrating on fast exploration. Even if previous solutions are able to quickly plan global coverage paths [82, 130, 164], the problem of efficient trajectory planning is rarely addressed. Both under the perspective of local exploration and the patrolling, or object finding, problem.

4.1.2 *Problem Definition*

The problem addressed in this chapter consists of exploring, as fast as possible, a bounded 3D volume $\mathcal{V} \subset \mathbb{R}^3$. Each point $x = (x, y, z) \in \mathcal{V}$ of this space can take only two values, i.e. *free* or *occupied*, and the problem of exploration ideally consists of clustering the volume $\mathcal{V}$ in the free $\mathcal{V}_{\text{free}} = \{x \in \mathcal{V} \mid \Gamma(x) = \text{free}\}$ or occupied $\mathcal{V}_{\text{occ}} = \{x \in \mathcal{V} \mid \Gamma(x) = \text{occ}\}$ volumes, with a mapping function $\Gamma(x) : \mathcal{V} \mapsto \{\text{free}, \text{occ}\}$. Since this operation is subjected to the agent constraints and its sensing capabilities, it may happen that some map point cannot be observed by construction. These points will remain unmapped and, for this reason, we introduce a further set $\mathcal{V}_{\text{res}}$ collecting the residual points that cannot be observed. Thus, the exploration will be considered complete when $\mathcal{V}_{\text{free}} \cup \mathcal{V}_{\text{occ}} = \mathcal{V} \setminus \mathcal{V}_{\text{res}}$. Due to the nature of the problem, it has to be solved in real-time by planning suitable safe trajectories through the free space which is not known a priori.

4.2 LARGE-SCALE ENVIRONMENTS EXPLORATION

In this section, we propose a novel solution to the aforementioned exploration problem (Section 4.1.2) in a setting where timing is the major issue during the task of completely unknown environment exploration. The basic idea behind the proposed solution is that efficient local procedures allow for a very high speed up in the overall task. Although the global-local paradigm is fundamental to ensure consistency and completeness of the exploration procedure, it is not the key to compute efficient exploration paths, due to the fact that the switching between local and global may lead to a waste of time, requiring the agent to change direction very frequently. On the other hand, optimising local trajectories with respect to the agent capabilities and the locally gathered information of the environment under exploration may lead to faster motions, low energy consumption, and lower waste of time. This is especially true when dealing with Unmanned Aerial Vehicles (UAVs), whose high maneuverability motivates solutions able to stress the quadrotor to fully exploit both its computational and dynamical capabilities. Similarly to [132] and [16], the core of the proposed solution consists of an RRT-inspired [84] sampling-based exploration algorithm that aims to directly plan highly informative feasible trajectories in known space, leading to an optimal local exploration procedure. The obtained tree is executed one node at a time, in a receding-horizon fashion. Unlike previous works, our algorithm employs a Bézier curve parameterisation to grow and maintain a tree of possible trajectory segments. The proposed approach weights both potential information gain and trajectory cost during the selection of the next goal. Moreover, the planned trajectory does not constrain the end velocity to zero, thus the exploration can

be performed quickly by avoiding "stop-and-go" like behaviors. Motivated by the promising results obtained using hybrid approaches [132], we allow adaptability of the proposed solution by letting the tree be easily extended with global exploration routines. In particular, we implemented efficient rewiring procedures in order to keep in memory and continuously refine the same tree, following the ideas of roadmaps memorisation [164] and continuous tree expansion [130]. We show that the combination of the planning of both path and the associated timing law leads to a solution outperforming the state-of-the-art approaches in this field, which usually plans point-to-point trajectories requiring stopping the robot at each exploration step.

4.2.1 *Bézier Trajectory Parameterisation*

In this framework, instead of using traditional polynomial functions, we adopt the Bernstein polynomial basis and define trajectories as piecewise Bézier curves. A Bézier curve is completely defined by its degree p and a set of $m = p + 1$ control points $\mathcal{CP} = \begin{bmatrix} \boldsymbol{q}_0 \cdots \boldsymbol{q}_p \end{bmatrix}$, with $\boldsymbol{q}_i \in \mathbb{R}$. The curve can be evaluated, for any $u \in [0, 1]$, as

$$\boldsymbol{s}(u) = \sum_{i=0}^{p} B_i^p(u)\, \boldsymbol{q}_i, \tag{7}$$

where the basis functions $B_i^p(u)$ are pth degree *Bernstein basis polynomials* [11, 46] of the form

$$B_i^p(u) = \frac{p!}{i!(p-i)!} u^i (1-u)^{p-i}.$$

A gentle introduction to Bézier curves can be found in Appendix A, to seek of clarence we report here the major properties required to properly understand the subsequent analysis. The aforementioned polynomials enjoy a partition-of-unity property (i.e. $\sum_{i=0}^{p} B_i^p(u) = 1$ for all u), by which the curve defined by Equation (7) is constrained inside the convex hull generated by its control points $\mathcal{CP}$. Moreover, a p-degree Bézier curve is always p times differentiable and its derivatives preserve a Bézier structure of lower degree. In particular, $s'(u) := ds/du$ is a Bézier curve of order $p - 1$ whose control points $\mathcal{CP}'$ can be evaluated as $\boldsymbol{q}_i' = p\left(\boldsymbol{q}_{i+1} - \boldsymbol{q}_i\right)\ \forall i = 0, \ldots, p-1$. The overall quadrotor reference trajectory can be expressed through the evolution of its *flat outputs* [102], as explained in Section 2.2, $\sigma = [\mathbf{r}, \phi]^\top$, where $\mathbf{r} = [x, y, z]^\top \in \mathbb{R}^3$ represents the coordinates of the center of mass in the world coordinate system, while $\phi \in \mathbb{R}$ is the yaw angle.

Both the quantities $\mathbf{r}$ and ϕ are expressed as l-segments piecewise Bézier curves of order $p_{\mathbf{r}}$ and p_ϕ, respectively

$$\mathbf{r}(t) = \begin{cases} \sum_{i=0}^{p_{\mathbf{r}}} B_i^{p_{\mathbf{r}}}(\zeta_1)\mathbf{r}_i^1 & t \in [T_0, T_1], \\ \sum_{i=0}^{p_{\mathbf{r}}} B_i^{p_{\mathbf{r}}}(\zeta_2)\mathbf{r}_i^2 & t \in [T_1, T_2], \\ \vdots & \vdots \\ \sum_{i=0}^{p_{\mathbf{r}}} B_i^{p_{\mathbf{r}}}(\zeta_l)\mathbf{r}_i^l & t \in [T_{l-1}, T_l], \end{cases}$$

$$\phi(t) = \begin{cases} \sum_{i=0}^{p_\phi} B_i^{p_\phi}(\zeta_1)\phi_i^1 & t \in [T_0, T_1], \\ \sum_{i=0}^{p_\phi} B_i^{p_\phi}(\zeta_2)\phi_i^2 & t \in [T_1, T_2], \\ \vdots & \vdots \\ \sum_{i=0}^{p_\phi} B_i^{p_\phi}(\zeta_l)\phi_i^l & t \in [T_{l-1}, T_l], \end{cases}$$

with $\zeta_i = \frac{t - T_{i-1}}{T_i - T_{i-1}}$. The quantities $\mathbf{r}_i^j \in \mathbb{R}^3$ and $\phi_i^j \in \mathbb{R}$ describe the i^{th} control point of the j^{th} trajectory segment of $\mathbf{r}(t)$ and $\phi(t)$ respectively, while T_{j-1} and T_j are the start and end time of the j^{th} trajectory segment. Note that the introduced time scaling does not affect the spatial path described by the Bézier curve, but strongly affects its derivatives as

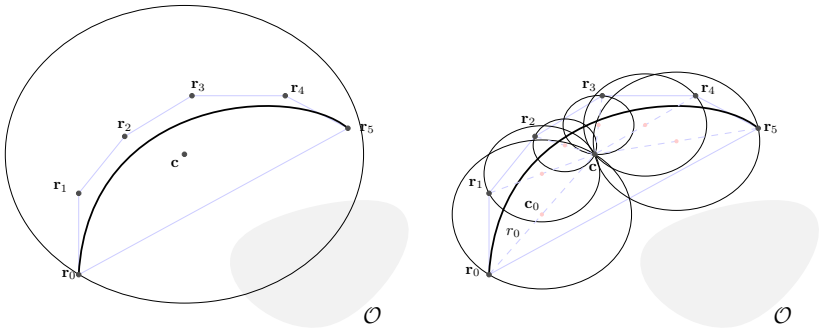
$$\mathbf{r}'^j_i = \frac{p_{\mathbf{r}}(\mathbf{r}^j_{i+1} - \mathbf{r}^j_i)}{T_{j-1} - T_j} \quad \forall i = 0, \dots, p_{\mathbf{r}} - 1,$$

$$\phi'^j_i = \frac{p_\phi(\phi^j_{i+1} - \phi^j_i)}{T_{j-1} - T_j} \quad \forall i = 0, \dots, p_\phi - 1.$$

The *convex hull containment* property is a powerful tool to verify both the trajectory feasibility in terms of dynamic constraints, such as velocity or acceleration bounds, and to check for collisions. Figure 3a reports the classical condition used for collision checking with Bézier curves [139], where the overall curve is constrained inside a *safe* sphere. The aforementioned approach often results in being too conservative, as a matter of fact the considered sphere is far to be tight over the convex hull, and thus over the curve itself. For this reason we formulate a new proposition that represents a less conservative tool to verify collision (see Figure 3b).

Proposition 4.2.1. *Let $\mathbf{r}(u)$ be a Bézier curve of order p, with control points $\mathcal{CP} = [\mathbf{r}_0, \dots, \mathbf{r}_p]$. Moreover, let $r_i \in \mathbb{R}$ and $\mathbf{c}_i \in \mathbb{R}^3$ with $i = 0, \dots, p$ be respectively the radii and centre of p spheres ($\mathcal{C}_0 \dots \mathcal{C}_p$), defined as:*

$$\mathbf{c}_i = (\mathbf{r}_i + \mathbf{c})/2,$$
$$r_i = \|\mathbf{r}_i - \mathbf{c}_i\|,$$



(a) (b)

Figure 3: Representation of a fifth-degree Bézier curve with (a) the classical sphere used for collision checking [139] (b) the proposed multiple spheres envelope. The $\mathcal{O}$ shaded gray area represents a generic obstacle.

with $\mathbf{c}$ *be the centre of the convex hull generated by* $\mathcal{CP}$*, i.e.* $\mathbf{c} = \sum_{i=0}^{p} \mathbf{r}_i / p$*. Then the curve* $\mathbf{r}(u)$ *is entirely contained inside the spheres envelope, namely*

$$\mathbf{r}(u) \in \bigcup_{i=1}^{p} \mathcal{C}_i \quad \forall u \in [0,1].$$

Proof. The proof follows from the fact that $\mathbf{c}$ belongs to the convex hull and that the spheres envelope composed by $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ always contains the convex hull edge $\overline{\mathbf{r}_i \mathbf{r}_{i+1}}$. The first statement is true by construction, since $\mathbf{c}$ is a linear combination of $\mathbf{r}_i$, while the second one follows from the triangle inequality

$$\|\mathbf{r}_i - \mathbf{r}_{i+1}\| \leq \|\mathbf{r}_i - \mathbf{c}\| + \|\mathbf{r}_{i+1} - \mathbf{c}\| . \qquad \square$$

Proposition 4.2.1 states that the convex hull containment property can be reformulated taking into account a set of p spheres. Since this set of spheres results to be tighter around the curve with respect to a single big safe ball, the use of this proposition in formulating a new collision condition results in a less conservative approach. The following proposition states the sufficient condition for non-collision as a corollary of Proposition 4.2.1.

Proposition 4.2.2. *Let* $\mathbf{r}(u)$ *be a Bézier curve of order* p*, with control points* $\mathcal{CP} = [\mathbf{r}_0, \ldots, \mathbf{r}_p]$*. Moreover, let* $\mathbf{c}_i$ *and* r_i *with* $i = 0, \ldots, p$ *be the centre and radii of* p *spheres defined as in Proposition 4.2.1. The curve* $\mathbf{r}(u)$ *is said to be collision-free, with a safety bound of* $d^{safe} \in \mathbb{R}_+$*, if the condition*

$$r_i - d^{obs}_{\mathbf{c}_i} - d^{safe} > 0 \quad \forall i = 0, \ldots, p$$

holds, where $d^{obs}_{\mathbf{c}_i}$ *represents the Euclidean distance of* $\mathbf{c}_i$ *from the closest obstacle.*

From now on, we use fifth-order Bézier curves to represent the quadrotor position ($p_{\mathbf{r}} = 5$), while the yaw trajectory is parameterised using third-order Bézier curves ($p_\phi = 3$).

4.2.2 *Tree Structure*

The proposed algorithm works by growing and maintaining, at each iteration, a tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ of possible trajectories. Such tree consists of a set of nodes $\mathcal{N} = \{N_1, \dots, N_{n_N}\}$ and a set of edges $\mathcal{E} = \{E_i, \dots, E_{n_E}\}$. Each node N_i is completely defined by the following five quantities

$$N_i = \left\{ g_i, c_i, \delta_i, \mathcal{CP}_i^{\mathbf{r}}, \mathcal{CP}_i^{\phi} \right\}$$

where $g_i = g(N_i)$ represents the amount of information gained if that node is executed, and $c_i = c(N_i)$ is the cost associated to the node execution. $\mathcal{CP}_i^{\mathbf{r}}$ and $\mathcal{CP}_i^{\phi}$ are the two sets of control points defining the trajectories $\mathbf{r}_i(t)$ and $\phi_i(t)$, while δ_i is the execution time. Two nodes N_{i-1} and N_i are connected by an edge E_{i-1} only if the first $(p_{\mathbf{r}/\phi} + 1)/2$ control points of the latter node satisfy some continuity criterion with the last $(p_{\mathbf{r}/\phi} + 1)/2$ control points of the former one. This constraint is required to ensure continuity among all trajectory segments of the tree. In particular, since $p_{\mathbf{r}} = 5$ and $p_\phi = 3$, we enforce continuity up to the third derivative along $\mathbf{r}(t)$ and continuity up to the second derivative along $\phi(t)$, namely

$$\mathbf{r}_0^i = \mathbf{r}_5^{i-1}, \tag{8}$$

$$\frac{1}{\delta_i}(\mathbf{r}_1^i - \mathbf{r}_0^i) = \frac{1}{\delta_{i-1}}(\mathbf{r}_5^{i-1} - \mathbf{r}_4^{i-1}), \tag{9}$$

$$\frac{1}{\delta_i^2}(\mathbf{r}_2^i - 2\mathbf{r}_1^i + \mathbf{r}_0^i) = \frac{1}{\delta_{i-1}^2}(\mathbf{r}_5^{i-1} - 2\mathbf{r}_4^{i-1} + \mathbf{r}_3^{i-1}), \tag{10}$$

$$\phi_0^i = \phi_3^{i-1}, \tag{11}$$

$$\frac{1}{\delta_i}(\phi_1^i - \phi_0^i) = \frac{1}{\delta_{i-1}}(\phi_3^{i-1} - \phi_2^{i-1}). \tag{12}$$

The aim is to plan sub-optimal trajectories by maximising a user-specified utility function $\mathcal{J}(\mathcal{R}(N_i))$, with $\mathcal{R}(N_i)$ be the sequence of nodes connecting N_i to the tree root, which properly combines gains and costs of all nodes in $\mathcal{R}(N_i)$. In this context, the tree root is defined as the tree node which is about to be executed by the flying agent. It results that the agent behavior strongly depends on the choice of functions $g(N_i)$, $c(N_i)$ and $\mathcal{J}(\mathcal{R}(N_i))$. The proposed algorithm is agnostic with respect to these functions. Therefore, the user can specify any formulation of them by ensuring that the following criteria are satisfied [130]:

1. $g(N_i)$ should be a function that depends on the trajectory end position only ($g\left(\mathbf{r}_5^i, \phi_3^i\right)$),

2. all node gains should be mutually independent,
3. $c(N_i)$ is required to be an intrinsic property of the trajectory ($c\left(\mathcal{CP}_i^{\mathbf{r}}, \mathcal{CP}_i^{\phi}, \delta_i\right)$).

4.2.3 *Tree Update*

The tree, initially composed by just one root node, is iteratively expanded by randomly sampling viewpoints inside a sphere centred on the current *best node* (N_{best}), namely the one among all tree nodes that maximise the utility function $\mathcal{J}(\cdot)$. In particular, the sphere is centred exactly on the last control point of $\mathcal{CP}_{\text{best}}^{\mathbf{r}}$, i.e. $\mathbf{r}_5^{\text{best}}$, while its radius ($r_{\text{sp}}$) is a user chosen value defined as a parameter for the algorithm. The sampled viewpoint is retained only if it belongs to a known and free part of the environment under exploration and, at the same time, it is far enough from the mapped obstacles. Such viewpoint is considered as the last control point of the next trajectory segment ($\mathbf{r}_5^i$). Moreover, due to Condition (8), also the control point $\mathbf{r}_0^i$ is already defined to ensure position continuity. As regards the heading trajectory, the first control point (ϕ_0^i) is established through Condition (11), while the last one (ϕ_3^i) is chosen as the value that maximise the potential information gain $g(\mathbf{r}_5^i, \phi)$, namely

$$\phi_i^3 = \arg\max_{\phi} \; g(\mathbf{r}_5^i, \phi),$$

in a similar way as done in [132]. The choice of the remaining points ($\mathcal{CP}_i^{\mathbf{r}}[1:4]$, $\mathcal{CP}_i^{\phi}[1:2]$) and the trajectory duration (δ_i) is performed concurrently. In particular, the interval of admissible trajectory duration $\Delta = [\delta_{\min}, \delta_{\max}]$ is uniformly discretised as

$$\Delta_{\text{d}} = \{\delta_{\min},\ \delta_{\min} + \Delta_\delta,\ \delta_{\min} + 2\Delta_\delta,\ \ldots,\ \delta_{\max}\},$$

with $\Delta_\delta = \frac{\delta_{\max} - \delta_{\min}}{r}$, leading to $r+1$ possible time intervals. For any $\delta \in \Delta_{\text{d}}$, the control points $\mathbf{r}_1^i$ and $\mathbf{r}_2^i$ are computed exploiting Condition (9) and Condition (10). If the obtained points do not satisfy Proposition 4.2.2 the current δ is discarded, otherwise also the points $\mathbf{r}_3^i$ and $\mathbf{r}_4^i$, as well as ϕ_1^i (Condition (12)) and ϕ_2^i are computed. Note that the quantities $\mathbf{r}_3^i$, $\mathbf{r}_4^i$ and ϕ_2^i are not constrained by any conditions (8)–(12), thus these points are computed by optimising the induced cost $c(\mathcal{CP}_i^{\mathbf{r}}, \mathcal{CP}_i^{\phi}, \delta_i)$. In the same way as before, if the computed points violate Proposition 4.2.2, or the induced velocity or acceleration exceed the dynamic bounds, the current δ is discarded. Once all $\delta \in \Delta_{\text{d}}$ have been considered, the one leading to the optimal value of c_i is selected with the corresponding computed control points and the node is added to the tree. The tree growth continues until it became impossible to find a new node with higher information gain $g(\cdot)$ and the number of sampled nodes goes beyond a given threshold ($n_{\max}$). Once the tree expansion is terminated, the branch leading

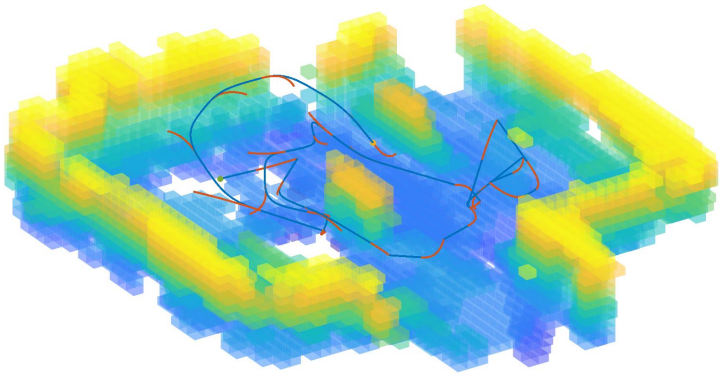


Figure 4: Qualitative evaluation of the proposed method in a real-world experiment. The Bézier-based exploration succeeded in fast planning motion inside the unknown area without forcing zero end velocities and successfully avoiding the two obstacles placed at the center. In the figure, blue lines represent the reference trajectory, while in red are depicted the planned *safe* maneuvers.

to the best node is extracted and only the first node of such branch (N_{opt}) is executed.

It may happen that the tree growth procedure takes too much time, or it may result impossible to find a valid candidate as next trajectory segment. In order to handle these issues, at each iteration two trajectory segments are computed. The first one corresponds to the execution of the best branch, while the second one is a *safe* trajectory, linked via Constraints (8)–(12) to the first committed segment. The *safe* trajectory constrains the final velocity to be zero, as well as the final acceleration, and it is executed every time the algorithm fails in planning a new node.

4.2.4 *Reconstruction Gain, Trajectory Cost & Total Utility*

The algorithm presented in Section 4.2.3 is used to plan spatial trajectories by maximising the total utility function $\mathcal{J}(\cdot)$. As a consequence, since this function combines both node gain and cost, the choice of functions $g(\cdot)$ and $c(\cdot)$, as well as $\mathcal{J}(\cdot)$ itself, is crucial for the success of the exploration procedure and of its performance. In this context, the reconstruction gain is defined as the amount of space that can be discovered if the agent is located in the considered position ($\mathbf{r}$) and oriented with a given heading angle (ϕ). The function $g(\cdot)$ can be computed by casting rays outward from the sensor and summing up all the unmapped volume elements that the ray crosses. Although there exist very efficient procedures useful to compute $g(\cdot)$, such as sparse ray-casting [132], its explicit evaluation is still the bottleneck for most of the exploration algorithms proposed within the

literature. The work [132], motivated by the continuous nature of the reconstruction gain over its domain, tries to overtake this problem by modelling it as a realisation of a Gaussian process [127]. The idea is to infer, when possible, the gain value using previously sampled data, avoiding its explicit computation at each exploration iteration. This approach has the limitation that when the process returns a *poor* (in terms of resulting variance) estimation, the gain must be explicitly re-computed, leading to a higher overhead due to the double computation. In this work we show that it is possible to completely avoid the gain explicit computation in real-time, during the planning procedure, and it can be left as a background thread. Unlike previous approaches, we propose to evaluate $g(\cdot)$ exclusively through Gaussian Process inference. The motivating assumption is that in previously unexplored areas the reconstruction gain evaluates as the sensor FoV volume. Therefore we impose a GP prior $g(\mathbf{r}) \sim \mathcal{GP}(V_{\text{fov}}, k(\mathbf{r}, \mathbf{r}', \tau))$ consisting of a constant mean function, equivalent to the sensor FoV volume, and a *squared-exponential* kernel

$$\kappa(\mathbf{r}, \mathbf{r}', \tau) = \exp\left(-\frac{\|\mathbf{r} - \mathbf{r}'\|_2^2}{2\tau^2}\right),$$

where τ is a hyper-parameter known as *characteristic length-scale*, iteratively estimated by minimising the associated log-likelihood function [127]. The proposed approach alternates between gain prediction, using the currently sampled data and the current estimation of the hyper-parameter, and correction, where τ is estimated by minimising the log-likelihood over the data.

In autonomous exploration applications, where classical RRT algorithms are employed, the trajectory cost is usually associated with node distance [16, 132], or execution time [130]. The former penalises long trajectories, while the latter pushes the agent near its dynamical limits in order to execute the task as fast as possible. Recent studies about frontier exploration [27] have shown great results in terms of execution time and traveled distance. In these works, viewpoints are selected considering minimal variations in velocity. Encouraged by the success of these algorithms and keeping in mind the necessity to end the exploration as fast as possible, we propose a trajectory cost that weight execution time and total control effort. The overall trajectory cost is formalised as

$$c\left(\mathcal{CP}_i^{\mathbf{r}}, \mathcal{CP}_i^{\phi}, \delta_i\right) = \mu_1 \delta_i + \mu_2 c_{\mathbf{r}}\left(\mathcal{CP}_i^{\mathbf{r}}, \delta_i\right) + \mu_3 c_{\phi}\left(\mathcal{CP}_i^{\phi}, \delta_i\right),$$

where $\mu_{1:3}$ are tuning parameters, while $c_{\mathbf{r}}(\cdot)$ and $c_{\phi}(\cdot)$ take the following form

$$c_{\mathbf{r}}\left(\mathcal{CP}_i^{\mathbf{r}}, \delta_i\right) = \int_0^{\delta_i} \left\| \frac{d^k \mathbf{r}^i\left(\frac{\tau}{\delta_i}\right)}{d\tau^k} \right\|^2 d\tau, \tag{13}$$

$$c_{\phi}\left(\mathcal{CP}_i^{\phi}, \delta_i\right) = \int_0^{\delta_i} \left\| \frac{d^p \phi^i\left(\frac{\tau}{\delta_i}\right)}{d\tau^p} \right\|^2 d\tau. \tag{14}$$

In this particular case we selected $k = 2$ and $p = 1$, leading to trajectories with minimal accelerations and angular velocities. Minimising the angular velocity has several benefits in terms of mapping reconstruction accuracy, due to the fact that the captured data present low blur effect, especially when working with RGBD cameras. Note that the three components of $\mathbf{r}(\cdot) = [r_x(\cdot), r_y(\cdot), r_z(\cdot)]$ are decoupled inside the cost function, thus Equation (13) can be rewritten as

$$c_{\mathbf{r}}\left(\mathcal{CP}_i^{\mathbf{r}}, \delta_i\right) = \sum_j^{x,y,z} \int_0^{\delta_i} \frac{d^k r_j^i\left(\frac{\tau}{\delta_i}\right)^2}{d\tau^k} d\tau. \tag{15}$$

The Bernstein basis parameterisation is closed with respect to operations of derivative, power elevation and integral [46], thus Equation (14) and Equation (15) can be evaluated in closed form just acting on the trajectory control points in the following way

$$c_{\mathbf{r}}\left(\mathcal{CP}_i^{\mathbf{r}}, \delta_i\right) = \begin{bmatrix} \mathbf{r}_0^i & \cdots & \mathbf{r}_{n_{\mathbf{r}}}^i \end{bmatrix} B_{\mathbf{r}}\left(\delta_i\right) \begin{bmatrix} \mathbf{r}_0^i & \cdots & \mathbf{r}_{n_{\mathbf{r}}}^i \end{bmatrix}^\top,$$

$$c_{\phi}\left(\mathcal{CP}_i^{\phi}, \delta_i\right) = \begin{bmatrix} \phi_0^i & \cdots & \phi_{n_{\phi}}^i \end{bmatrix} B_{\phi}\left(\delta_i\right) \begin{bmatrix} \phi_0^i & \cdots & \phi_{n_{\phi}}^i \end{bmatrix}^\top,$$

where $B_{\mathbf{r}}(\delta_i)$ and $B_{\phi}(\delta_i)$ are the matrix form of the Bézier curves $c_{\mathbf{r}}(\cdot)$ and $c_{\phi}(\cdot)$ [124]. We take advantage of this property during the planning stage, when selecting the remaining free points $\mathbf{r}_3^i$, $\mathbf{r}_4^i$ and ϕ_2^i. These are computed solving the following optimisation problem

$$\min_{\mathbf{r}_3^i, \mathbf{r}_4^i, \phi_2^i} \begin{bmatrix} \mathbf{r}_0^i \\ \vdots \\ \mathbf{r}_{n_{\mathbf{r}}}^i \\ \phi_0^i \\ \vdots \\ \phi_{n_{\phi}}^i \end{bmatrix}^\top \begin{bmatrix} \mu_2 B_{\mathbf{r}}(\delta_i) \\ \mu_3 B_{\phi}(\delta_i) \end{bmatrix} \begin{bmatrix} \mathbf{r}_0^i \\ \vdots \\ \mathbf{r}_{n_{\mathbf{r}}}^i \\ \phi_0^i \\ \vdots \\ \phi_{n_{\phi}}^i \end{bmatrix},$$

which results in an unconstrained QP problem, solvable by equalising the gradient to zero in a similar way as done in [128]. Finally, the total utility function, responsible to merge gains and costs of the tree nodes in only one utility value, has been chosen borrowing the idea of [130] that proposes a total utility function based on the notion of efficiency

$$\mathcal{J}\left(\mathcal{R}(N_i)\right) = \frac{\sum_{N_l \in \mathcal{R}(N_i)} g_l}{\sum_{N_l \in \mathcal{R}(N_i)} c_l}.$$

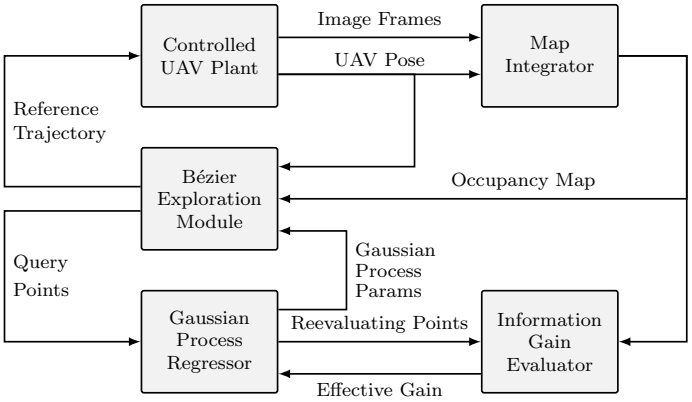


Figure 5: The overall scheme of the proposed exploration system.

4.2.5 *Implementation Details*

The overall structure of the proposed framework is shown in Figure 5. The planning framework is built on top of a reliable UAV control scheme and an occupancy map integrator whose construction is out of the scope of this chapter. The solution relies on three different threads running in parallel.

1. *Exploration Module*.
   This thread acts as the exploration supervisor and is in charge of computing the reference trajectories to be executed from the UAV platform. The exploration module continuously grows and maintains the trajectories tree via random sampling and by reevaluating each sampled non-executed node at each new iteration. Since the tree is executed in a receding-horizon fashion, every time a new trajectory is commissioned, a bunch of previously planned trajectories may become infeasible due to continuity issues. To handle this problem an activation flag is added as a node property. Note that at each sampling, only the active branches are taken into consideration. Furthermore, the explorer node also keeps care about possible *deadline violations*. In these cases the executed safe trajectory is added to the tree and rewired to all nearby active nodes. This prevent us to lose previous sampled possible promising trajectories.

2. *Gaussian Regressor*.
   This thread receives all sampled points from the exploration module and implements a policy to allow the cache of only the most informative ones. In particular, a new point is retained only if it belongs to a new and not explored area. The implementation of a R-tree structure allows for fast point insertion and retrieval, moreover it eases the insertion condition check. In order to be consistent with the exploration task, and the evolution of the known map, all cached points are reevaluated pe-

riodically via explicit gain computation. Such a module is in charge to train the Gaussian process parameters used by the explorer.

3. *Information Gain Evaluator*.
   This thread receives the evaluating point and computes explicitly the information gain via sparse ray-casting as in [132].

The architectural subdivision of the implemented algorithm in three different threads allows fast computing high informative trajectories without the explicit gain computation bottleneck. The whole algorithm has been implemented as a ROS network and paired with the PX4 autopilot both for the software-in-the-loop simulations and the real-world tests. As a map representation we use OctoMap [67].

The implemted code is open-source and completely available at the GitHub repository github.com/casy-lab/BezierFastExploration.

4.2.6 *Experimental Evaluation*

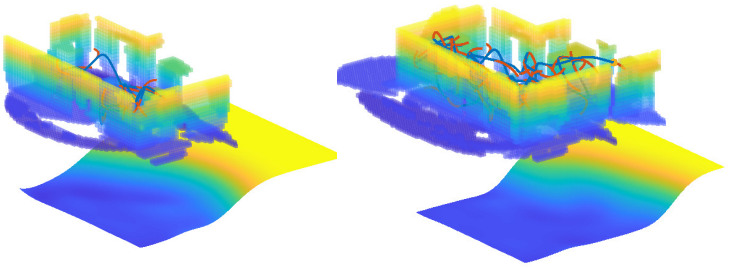
The proposed approach has been evaluated via Gazebo-based simulations, exploiting the environment RotorS [50] along with the provided 3DR Iris quadrotor model, endowed of a depth sensor. The algorithm performance have been also qualitatively evaluated in real-

Max Vel.	$1.5 m/s$	Max Acc.	$1.5 m/s^2$
Sampled Nodes	40	Max Length	$3m$
Min Range	$0.3m$	Max Range	$5.0m$
Camera FoV	115×60 deg	Map Res.	$0.2m$
μ_1 μ_2 μ_3	0.5 0.1 0.1	Time Res.	$0.5s$
Min Time	$1s$	Max Time	$5s$

Table 1: Parameters used in simulations.

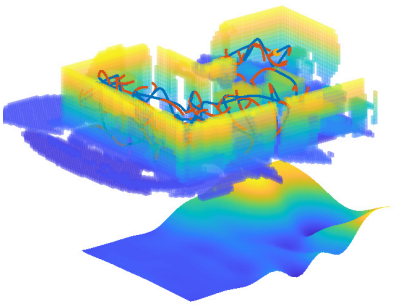
Max Vel.	$0.5 m/s$	Max Acc.	$0.5 m/s^2$
Sampled Nodes	20	Max Length	$3m$
Min Range	$0.3m$	Max Range	$3.0m$
Camera FoV	87×58 deg	Map Res.	$0.2m$
μ_1 μ_2 μ_3	0.5 0.1 0.1	Time Res.	$0.5s$
Min Time	$1s$	Max Time	$5s$

Table 2: Parameters used in the real-world experiments.



(a)

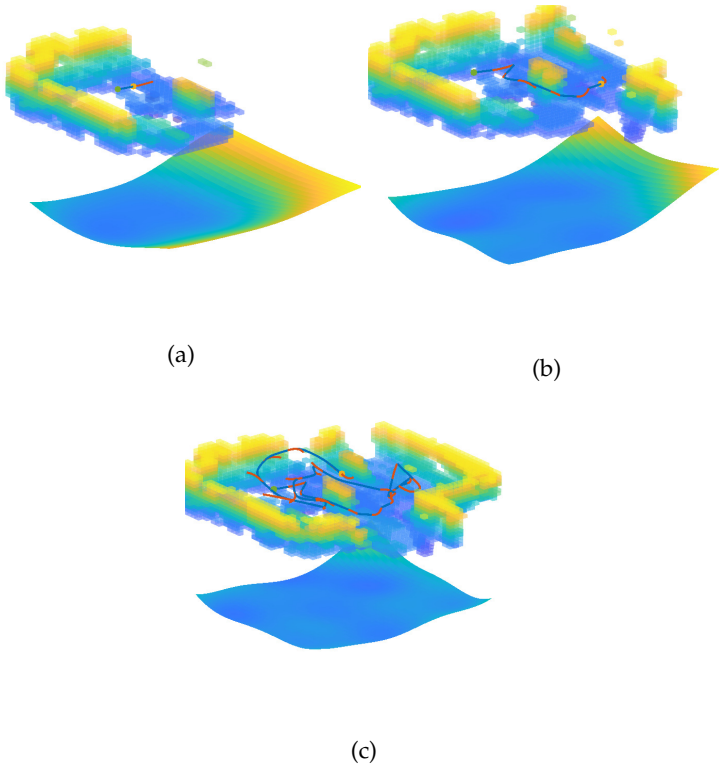
(b)



(c)

Figure 6: Results of the simulation tests. The exploration algorithm runs over a map of $20 \times 10 \times 3$ meters and was able to complete the exploration after only 400 seconds. In the image, in order, (a) exploration state at 100 seconds, (b) exploration state at 200 seconds, and (c) exploration state at 300 seconds. At the bottom of each time snapshot, a visual representation of the Gaussian inferred information gain is reported.

world scenario tests. In all tests the agent starts in the origin with zero yaw angle. The agent performs an initial rotation of 360 degrees around the initial hovering point in order to be sure to start the exploration with some initial information at hand. The parameters used during the simulation tests are reported in Table 1. Figure 6 shows the obtained simulation results when agent is required to map an $20 \times 10 \times 3$ urban canyon. In particular, in Figure 6, the blue lines represent the reference trajectory, the red ones are the planned safe motions (both executed and non-executed), while the bottom surface represents the current Gaussian process state. It can be noticed that, the Gaussian process is constantly kept updated with the current map



(a) (b)

(c)

Figure 7: Results of the real-world exploration test. The exploration algorithm was run using only integrated onboard sensors and computational capabilities. In the image, in order, (a) exploration state at 20 seconds, (b) exploration state at 57 seconds, and (c) exploration state at 167 seconds. At the bottom of each time snapshot, a visual representation of the Gaussian inferred information gain is reported.

information and it results to be consistent, at each time instant, with the exploration task. In order to evaluate the performances against the state-of-the-art solutions, the proposed algorithm has been compared with the *Autonomous Exploration Planner* (AEP) described in [132]. Figure 8 compares the amount of explored volume over time by both the approaches. The blue line represents the average of explored area obtained deploying our approach over 10 experiments, with the associated standard deviation represented in shaded blue. Conversely, the line and shade red reports the results obtained via AEP, with the global exploration module disabled, on the same number of experiments. It can be noticed that both algorithms achieve comparable results at the beginning of the exploration, where most of the volume needs to be explored, then our solution tends to get higher exploration rate, thanks to the ability to fast plan the next trajectory. Moreover, it is worth noting that our solution provides more consistency between different tests, as the variance is narrower with respect to the AEP, thus guaranteeing better repeatability of the experiment and a

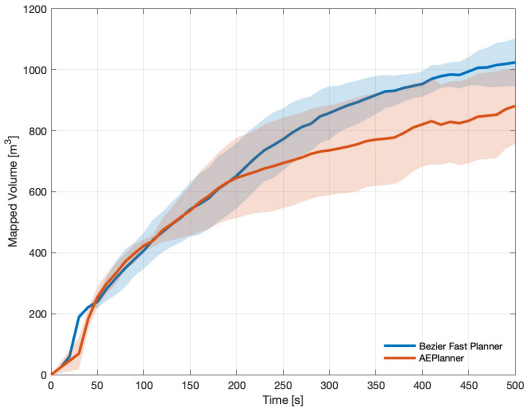


Figure 8: Exploration progress for the urban $20 \times 10 \times 3$ canyon. Mean and standard deviation over 10 experiments are shown. Notice that due to the employing of pierced nets as maps borders makes the overall explored volume higher than the real volume.

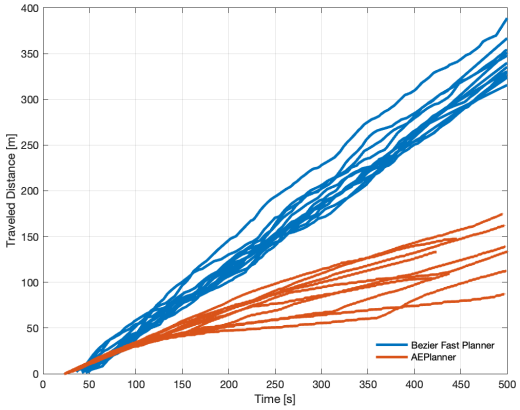


Figure 9: Overall traveled distance in the urban canyon. The traveled distances over 10 experiments are shown.

mitigation of the worst case scenarios. Figure 9 depicts the overall travelled distance on same experiments. Since our solution plans trajectories by never stopping the UAV motion, this leads to an overall travelled distance 2 times higher than the AEP solution. The solution has been tested using a real UAV inside an indoor scenario using our office spaces. The obtained results are depicted in Figure 7. The used parameters are reported in Table 2. The available area was $9 \times 6 \times 2.5$ meters and it was successfully mapped in 170 seconds, the maximum camera range was saturated at 3 meters in order to stress navigation trajectories around obstacles. The used UAV was powered by the PX4 autopilot and endowed with a depth Intel RealSense D455 camera, mounted frontally. Visual odometry, used to localize the UAV in the indoor scenario, was provided by an Intel Realsense T265 tracking camera.

4.3 PATROLLING: A DIFFERENT EXPLORATION PERSPECTIVE

In this section, we review a novel and different solution to the exploration problem stated in Section 4.1.2, in the setting where the environment is completely known and the only goal is to find out interesting points, or objects, inside the explored area. In this framework, the general exploration problem can be specialised by considering a third attribute, jointly with *free* and *occupied*, namely *interest*. A subset of the overall volume $\mathcal{V} \in \mathbb{R}^3$ will be categorised as $\mathcal{V}_{\text{int}} = \{x \in \mathcal{V} \mid \Gamma(x) = \text{int}\}$, thus the exploration task will be considered complete when $\mathcal{V}_{\text{free}} \cup \mathcal{V}_{\text{occ}} \cup \mathcal{V}_{\text{int}} = \mathcal{V} \setminus \mathcal{V}_{\text{res}}$, with $\mathcal{V}_{\text{free}}$, $\mathcal{V}_{\text{occ}}$, and $\mathcal{V}_{\text{res}}$ defined as in Section 4.1.2. Let us suppose to completely apriori known the set $\mathcal{V}_{\text{free}}$, then the exploration goal reduces to correctly classify $\mathcal{V} \setminus \mathcal{V}_{\text{res}} \setminus \mathcal{V}_{\text{free}}$ between $\mathcal{V}_{\text{occ}}$ and $\mathcal{V}_{\text{int}}$. The proposed approach is based on the idea that the employed software pipeline used to recognise the objects of interest may fail in doing that, and the probability of failure is jointly linked with the capability of the endowed sensor. Moreover, the autonomous platform may not be able to perform the commissioned motion due to environment modifications, or unexpected and unmapped obstacles. The stochastic nature of the problem at hand, along with the stochasticity of the adopted pipeline, embraces a purely probabilistic approach, so the locations of the objects are expressed in a likelihood form, that in the following takes the name of *probabilistic map*.

Similar to the algorithm described in Section 4.2, the core of the proposed solution consists of an RRT of possible and feasible motions. Unlike the previous approach, here we do not constrain the tree to directly represents pieces of possible trajectories. In this case, the tree is built to describe possible collision-free paths inside the environment, while the computation of the associated timing laws is left as next step, during the *trajectory optimisation* stage. In this view, the timing law is selected to ensure trajectory feasibility in terms of induced maximum velocity and acceleration. The tree expansion is supplied with the aforementioned probabilistic map, which is copied and kept updated on each new sampled node. The extension of the node attributes set with a copy of the current likelihood map allows for time consistency. The map in each node is in fact constantly kept updated, allowing for cost increase or decrease if the exploration time grows, or if a particular node has been visited, respectively. The remainder of this section unfolds as follows Section 4.3.1 and Section 4.3.2 deeply analyze the aforementioned mapping strategy, bridging the gap between discrete and continuous maps, and proposing a novel mapping framework. In Section 4.3.3 the tree structure is briefly described, along with the adopted growth policy, Section 4.3.4 introduces the used time allocation procedure and how the final trajectory is iteratively refined before its commissioning. Finally, Section 4.3.5 reports

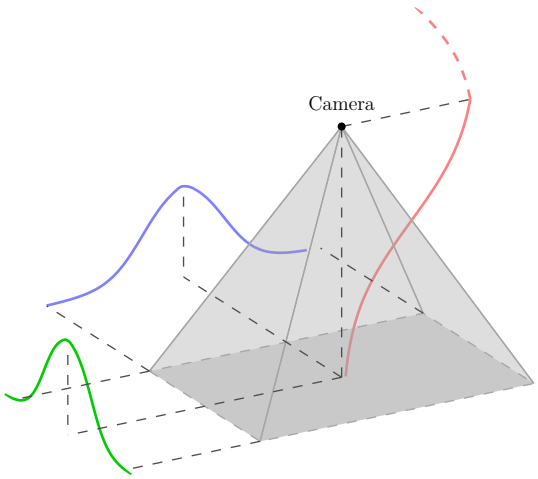


Figure 10: Probabilistic adopted camera model.

the obtained results when the proposed approach is exploited to perform target search in a completely known environment.

4.3.1 *Mapping Strategy*

In this solution we employ a discrete occupancy-based mapping approach [67] to store information about occupied and free spaces, while interesting environment features, or objects, are stored by following a continuous probability mapping paradigm. In particular, Gaussian processes are used to infer the belief of features existence. This approach allows to increase the map reliability by introducing spatial correlations between data while maintaining the memory consumption low, and to formulate an information gain that trades-off between target re-observation and exploration for new ones. In a probabilistic framework, the target positions are usually represented through a set of discrete random variables defined over a discretization of the search space ($\mathcal{E} \subset \mathbb{R}^3$). This approach leads to a discrete occupancy map ($\mathcal{M}$) where each grid cell is associated with an independent Bernoulli random variable (ζ_i) whose Probability Mass Function (PMF) represents the probability of target occupancy [121]

$$P(\zeta_i = k) = \begin{cases} p_i & \text{if } k = 1, \\ 1 - p_i & \text{if } k = 0. \end{cases}$$

The value of p_i for each $\zeta_i \in \mathcal{M}$ inside the sensor FoV is updated, at each new sensor measurement, via the Bayes formula

$$P\left(\zeta_i \mid z^{1:t}, s^{1:t}\right) = \frac{P\left(z^t \mid \zeta_i, s^t\right) P\left(\zeta_i \mid z^{1:t-1}, s^{1:t-1}\right)}{P\left(z^t\right)},$$

where z^t is the current observation at sensor position s^t. The aforementioned relation is usually rewritten in log-likelihood notation, where products are replaced by additions, allowing for faster updates [67]

$$L\left(\zeta_i \mid z^{1:t}, s^{1:t}\right) = L\left(\zeta_i \mid z^{1:t-1}, s^{1:t-1}\right) + L\left(z^t \mid \zeta_i, s^t\right), \tag{16}$$

with

$$L\left(\zeta_i\right) = \log\left(\frac{P\left(\zeta_i = 1\right)}{1 - P\left(\zeta_i = 1\right)}\right).$$

In Equation (16) the quantities $L\left(\zeta_i \mid z^{1:t-1}\right)$ and $L\left(\zeta_i \mid z^{1:t}\right)$ represent the prior and posterior probability, respectively. While, $L\left(z^t \mid \zeta_i\right)$ is the observation likelihood that strongly depends on the adopted probabilistic sensor model. In our specific case, we employ a downward-facing monocular camera to detect targets on the ground, Figure 10 shows the probabilistic model for such type of sensor. The probability to retrieve correct measures decreases approaching the FoV boundaries, as well as the maximum distance allowed. This kind of behavior is well described by a modified logistic function of the kind

$$P\left(z^t = 1 \mid \zeta_i, s^t\right) = 0.5 + \frac{p_{\max}}{1 + \exp\left(\left\|x_i - s^t\right\|_W - c\right)},$$

where $x_i \in \mathcal{M}$ represents the grid cell associated with ζ_i, while $s^t \in \mathbb{R}^3$ is the current sensor position. The weight matrix $W \in \mathbb{R}^{3\times3}$, and the parameters $p_{\max}$ and c are responsible to shape the function over the camera field of view.

Despite the power of discrete maps, in our solution we choose to adopt a continuous mapping strategy. Continuous maps allow to incorporate spatial correlations of input data and, potentially, require lower memory to maintain the overall map. These advantages are not for free since continuous maps present a higher computational complexity. Gaussian processes are well suited for this purpose since can encode spatial correlations in a probabilistic way and, with a careful choice of training points, take low computational and storage memory. With this idea in mind, the log-likelihood of occupancy is treated, in this case, as a continuous function over the search space $L : \mathcal{E} \subset \mathbb{R}^3 \mapsto \mathbb{R}$. The key idea is to model such function as a realization of a Gaussian process. A GP model is completely defined by a mean function $m\left(x\right) : \mathcal{E} \mapsto \mathbb{R}$ and a covariance function $\kappa\left(x, x'\right) : \mathcal{E} \times \mathcal{E} \mapsto \mathbb{R}$

$$L\left(x\right) \sim \mathcal{GP}\left(m\left(x\right), \kappa\left(x, x'\right)\right).$$

Although there are many possible choices for mean and covariance functions, usually the structure of m and κ are assumed to be known up to certain hyper-parameters ($m(\cdot, \vartheta_m)$, $\kappa(\cdot, \vartheta_\kappa)$). In the particular case of target search, the mean function should be chosen on the basis of some a prior notions about the target existence, thus it is a priori fixed and does not depend on any hyper-parameter. In those cases in which no prior information is available, the mean function is simply $m(x) = 0 \; \forall x \in \mathcal{E}$, that corresponds to an occupancy probability of 0.5. The covariance function, aka *kernel*, influences the GP behavior when fed with training data points. The results presented afterward are obtained using a *Matérn* 3/2 *kernel*, that takes the form

$$\kappa(x, x', \vartheta) = \left(1 + \frac{\sqrt{3}d}{\vartheta}\right) \exp\left(-\frac{\sqrt{3}d}{\vartheta}\right),$$

where $d = \|x - x'\|_2^2$, while ϑ is a hyper-parameter known as *characteristic length-scale* [127]. The Matérn 3/2 kernel is very common in geostatistical analysis, due to its capability to capture discrete targets [97], thus it results very suitable also for this particular application. Given N noisy observations $y^{1:N} \in \mathbb{R}^N$ and the respective sampling locations $x^{1:N} \in \mathbb{R}^{N\times 3}$, the posterior distribution of L is a Gaussian distribution, whose mean and variance can be evaluated as

$$\mu(x) = m(x) + \boldsymbol{\kappa}(x)^\top \left[\boldsymbol{\mathcal{K}} + \sigma_n^2 I\right]^{-1} \left(y^{1:N} - m\left(x^{1:N}\right)\right), \tag{17}$$

$$\sigma^2(x) = \kappa(x, x) - \boldsymbol{\kappa}(x)^\top \left[\boldsymbol{\mathcal{K}} + \sigma_n^2 I\right]^{-1} \boldsymbol{\kappa}(x). \tag{18}$$

To deal with the unknown hyper-parameter, we follow an empirical Bayes approach [10], where the prediction steps are alternated with parameters estimation steps via maximum likelihood

$$\vartheta = \arg\max_{\vartheta} p\left(y^{1:N} \mid x^{1:N}; \vartheta\right).$$

Under the Gaussian process prior, the quantity $p\left(y^{1:N} \mid x^{1:N}; \vartheta\right)$ can be evaluated in closed form as log-likelihood (dropping the apex $^{1:N}$)

$$\begin{aligned}\log\left(p\left(y \mid x; \vartheta\right)\right) = \\ -\frac{1}{2}\left(m(x) - y\right)^\top \left[\boldsymbol{\mathcal{K}} + \sigma_n^2 I\right]^{-1} \left(m(x) - y\right) \\ -\frac{1}{2}\log\left(\left|\boldsymbol{\mathcal{K}} + \sigma_n^2 I\right|\right) - \frac{N}{2}\log(2\pi).\end{aligned}$$

4.3.2 *Complexity Reduction & Spatial Partitioning*

Exact GP inference is expensive due to matrix inversion, whose computational complexity scales with the number of training points as $\mathcal{O}\left(N^3\right)$. Moreover, after training, the computation of posterior mean (17) and variance (18) costs $\mathcal{O}(N)$ and $\mathcal{O}\left(N^2\right)$ respectively, per testing

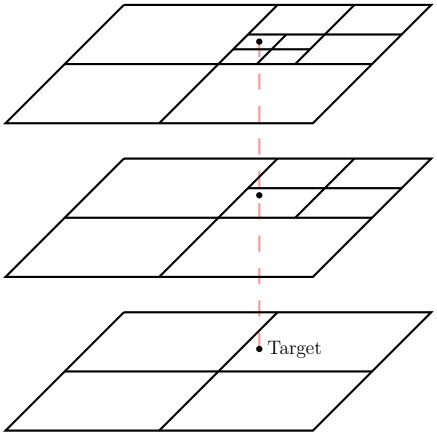


Figure 11: 2-Dimensional example of the adopted environment discretization. In this particular case, the presence of a target makes the likelihood level increases in that area, leading to a finer adopted discretization.

point. This makes the application of GP regression in real-time challenging, in particular for large environments. The common approach to deal with this issue was to divide the environment into regions and train several experts, one for each region [81]. This approach was effective, but the complexity problem still persists as long as the number of observations per region increases. Recently, approaches that leverage on *approximate regression* have shown great success thanks to their good scaling capabilities [56, 126, 157]. To reduce the GP complexity, the adopted approach leverages on the Subset of Regressors (SoR) method [126] to compute an approximation of the gram matrix $\mathcal{K}$ and the kernel vector $\boldsymbol{\kappa}$, jointly with a spatial partitioning and a targeted choice of the training points. In particular, a set of N_p *inducing variables* ($u^{1:N_p} \in \mathbb{R}^{N_p \times 3}$) are arbitrarily selected out of the training points set ($x^{1:N}$), then the covariance quantities can be approximated as

$$\begin{aligned} \mathcal{K} &\sim \mathcal{K}_{xu} \mathcal{K}_{uu}^{-1} \mathcal{K}_{ux}, \\ \boldsymbol{\kappa}(x) &\sim \mathcal{K}_{xu} \mathcal{K}_{uu}^{-1} \boldsymbol{\kappa}_u(x). \end{aligned} \tag{19}$$

Where $\mathcal{K}_u \in \mathbb{R}^{N_p \times N_p}$ and $\boldsymbol{\kappa}_u \in \mathbb{R}^{N_p}$ are the covariance matrix and vector computed along the inducing points, while $\mathcal{K}_{xu} \in \mathbb{R}^{N \times N_p}$ is the cross-covariance between $x^{1:N}$ and $u^{1:N_p}$, with entries

$$\mathcal{K}_{xu}^{i,j} = \kappa\left(x^i, u^j\right) \text{ with } i \in [1, N] \text{ and } j \in \left[1, N_p\right].$$

Plugging Equation (19) in Equation (17) and Equation (18) leads to

$$\mu(x) = m(x) + \boldsymbol{\kappa}_u(x)^\top \left[\boldsymbol{\mathcal{K}}_{ux}\boldsymbol{\mathcal{K}}_{xu} + \sigma_n^2\boldsymbol{\mathcal{K}}_{uu}\right]^{-1}\boldsymbol{\mathcal{K}}_{ux} \cdot \cdot \left(y^{1:N} - m\left(x^{1:N}\right)\right),$$

$$\sigma^2(x) = \boldsymbol{\kappa}_u(x)^\top \left[\boldsymbol{\mathcal{K}}_{ux}\boldsymbol{\mathcal{K}}_{xu} + \sigma_n^2\boldsymbol{\mathcal{K}}_{uu}\right]^{-1}\boldsymbol{\kappa}_u(x).$$

The SoR approximation method allows to reduce the computation complexity of the training procedure up to $\mathcal{O}\left(NN_p^2\right)$, while the evaluation complexity of predictive mean and variance is kept constant ($\mathcal{O}(N)$ and $\mathcal{O}\left(N^2\right)$, respectively). The hyper-parameter estimation is carried out by maximizing the log-likelihood function evaluated just on the inducing points ($\log\left(p\left(y^{1:N_p} \mid u^{1:N_p}; \vartheta\right)\right)$). In order to further decrease the map computation complexity, we propose a novel methodology to proper select the expert training points. A good approach should trade-off between process complexity and map precision. To achieve that, the 3-dimensional environment is discretized with a very coarse resolution, then we let this discretization be flexible in areas where higher precision is required. The discretization level is then dependent on the local likelihood value: higher is the probability of target existence and finer is the adopted discretization. In particular, the probability range $[0, 1]$ is divided into q intervals, each of which corresponds to a given map resolution. Note that those intervals are such that a discretization level and the consecutive one differ of a factor equal to 2. This leads to a quad-tree space discretization (see Figure 11). Training points are store in memory only when a measure of them is retrieved, each new point inherits the current local GP value, then is updated at each new sensor measurement through Equation (16). The adopted hierarchical structure allows to select the inducing variables as those training points obtained from the coarsest discretization, associated with null probability. Moreover, when a training point gets a promotion to the upper discretization level, that point is also promoted to inducing variable, while a set of new training points are added to still be compliant with the associated discretization level.

4.3.3 *Tree Structure & Update*

The proposed algorithm works by iteratively growing an RRT, $\mathcal{T} = (\mathcal{N}, \mathcal{E})$, of possible feasible motions, whose each node $N_i \in \mathcal{N}$ is described by means of six quantities

$$N_i = \left\{g_i, c_i, \mathbf{r}_i, y_i^{1:N}, x_i^{1:N}, u_i^{1:N_p}\right\},$$

where $g_i = g(N_{i-1}, N_i)$ and $c_i = c(N_{i-1}, N_i)$ are the information gain and the cost, respectively, associated with the execution of node N_i, i.e. navigate from N_{i-1} to N_i. The quantity $\mathbf{r}_i \in \mathbb{R}^3$ describes the node

position, while $y_i^{1:N}$, $x_i^{1:N}$, and $u_i^{1:N_p}$ represent the parameterisation of the likelihood map, at the time of node creation. As already mentioned, unlike the approach described in Section 4.2, here the adopted tree does not directly represent a set of possible trajectories, but instead describes a set of feasible paths through the environment. As a matter of fact, the aforementioned node description does not take into account neither a trajectory parameterisation, nor a possible execution time. In this sense, two nodes N_{i-1} and N_i are connected by an edge E_{i-1} if the straight path connecting $\mathbf{r}_{i-1}$ and $\mathbf{r}_i$ is collision-free, no other continuity constraints are required. Although the nodes present a slight different structure, the aim of this tree is the same of the one build in Section 4.2.2. Basically, the goal is to compute sub-optimal paths that maximise a given utility function $\mathcal{J}(\mathcal{R}(N_i))$, with the latter that combines gains and costs of all nodes in $\mathcal{R}(N_i)$. It results that the final agent behavior strongly depends on the chosen functions $g(N_{i-1}, N_i)$, $c(N_{i-1}, N_i)$ and $\mathcal{J}(\mathcal{R}(N_i))$. Unlike general planners, who exploit RRT structures to plan flyable trajectories, in this case the tree expansion logic must be designed around the aforementioned loss functions, with particular attention to the gain $g(N_{i-1}, N_i)$. Tailoring the algorithm behavior on g allows for keeping the exploration task consistent during the motion, and avoids cases in which the agent gets stuck in little areas, without exploring the overall environment. This is particularly true when the chosen utility function presents many local minima, which are difficult to escape. In this respect, we formulate a new information gain index that takes into account the amount of improvement, in the current likelihood map, brought by the execution of the considered node. In particular, consider two nodes N_{i-1} and N_i, with their respective map parameterisation $\mu_{i-1}\left(y_{i-1}^{1:N}, x_{i-1}^{1:N}, u_{i-1}^{1:N_p}, x\right)$ and $\mu_i\left(y_i^{1:N}, x_i^{1:N}, u_i^{1:N_p}, x\right)$, the gain associated with the node N_i can be evaluated as

$$g_i(N_i, N_{i-1}) = \int_{\mathcal{E}} \frac{\exp(\mu_{i-i}(\cdot, x))}{1+\exp(\mu_{i-i}(\cdot, x))} - \frac{\exp(\mu_i(\cdot, x))}{1+\exp(\mu_i(\cdot, x))} dx,$$

where the quantities $y_i^{1:N}$, $x_i^{1:N}$, and $u_i^{1:N_p}$ are computed out of the corresponding values of the N_{i-1} node via Equation (16), along the path connecting $\mathbf{r}_{i-1}$ to $\mathbf{r}_i$. Note that, unlike previous works in this field [121, 130], here the information gain is not a function of the node end-position only, thus allows us to consider also frustrum overlap between consecutive poses and the amount of information gained while moving from N_{i-1} to N_i. On the other hand, the node execution cost is fixed as the intra-node Euclidean distance $c_i(N_i, N_{i-1}) = \|\mathbf{r}_{i-1} - \mathbf{r}_i\|$, while the total utility function as been designed following the same idea of efficiency stated in Section 4.2.4

$$\mathcal{J}(\mathcal{R}(N_i)) = \frac{\sum_{N_l \in \mathcal{R}(N_i)} g_l}{\sum_{N_l \in \mathcal{R}(N_i)} c_l}.$$

With these notion at hand, we can now proceed to the design of the core of the algorithm. The chosen tree updating procedure is as simple as it is effective, and it is designed to cope with the multi-modal nature of the chosen utility function, as well as the possibility to inject prior notions about the obstacles occupancy. The tree, initially composed of just one root node, is iteratively expanded via random sampling of the next node position $\mathbf{r}_i$. In particular, we constrain the sample points to lie inside a circular crown of radii R_{min} and R_{max}, centered on the current best node (N_{best}), namely the one among all tree nodes that maximise the utility function $\mathcal{J}(\cdot)$. In order to escape from local minima and let the tree grow in all directions, the best node is reevaluated each n_{sample} samples, ensuring that every expanded node presents exactly n_{sample} childs. This choice, jointly with the circular crown constraint, allows the algorithm to expand the tree without getting stuck in little areas, with many local minima. Each sampled viewpoint is retained only if it belongs to a known and free part of the environment and, at the same time, the line connecting it with the respective father is far enough from the mapped obstacles. The tree expansion continues until the number of sampled nodes goes beyond a given threshold n_{max}, then the path connecting the root node to the best one is extracted and completely executed. It may happen that, the autonomous robot is not able to complete the commissioned path due to unexpected obstructions, in this case the algorithm keeps track of the executed nodes and reinitialize the tree with the final executed node as root.

4.3.4 *B-Spline Trajectory Optimisation*

As already pointed out in previous sections, the built tree does not directly represent flyable trajectories, but just feasible paths through the environment's obstacles. For the purpose of control, a timing law is essential to ensure the dynamic feasibility of the resulting reference. In the specific case of a quadcopter, such a reference can be computed just in terms of heading angle ϕ and space position $\mathbf{r} = [x, y, z] \in \mathbb{R}^3$, following the concept of differential flatness [102]. The latter property can be used only if the heading trajectory is guarante to be at least a class $\mathcal{C}^1$ function, while $\mathbf{r}$ must be at least class $\mathcal{C}^3$. Contrary to the solution presented in Section 4.2, here we enforce the use of B-splines as time parameterisation tool for the planned collision-free path. B-splines are a generalization of Bézier curves which allows for a more compact representation, especially when dealing with long trajectory segments. Unlike Bézier curves, B-splines are defined by a set of control points $\mathcal{CP} = [\boldsymbol{q}_0, \dots, \boldsymbol{q}_m]$, and a set of time knots $\boldsymbol{u} = \left[u_0, \dots, u_{m+p+1}\right]$, as

$$\boldsymbol{s}(u) = \sum_{i=0}^{m} B_i^p(u)\, \boldsymbol{q}_i,$$

with p order of the spline and B_i^p basis functions, recursively computed via the De-Boor formula [33]. B-splines and Bézier curves share several properties, among all, the most important ones at the time being, are the (a) convex hull containment and (b) closure with respect to differentiation. These two properties allow to easily formulate a constrained quadratic programming problem and solve it for the final trajectory control points. A deeper analysis of B-spline properties and their evaluation is reported in Section A. In the specific case of patrolling, since we are using a downward-facing camera, the heading trajectory loses importance, thus we consider it as a constant reference in time, and focus only on solving the optimisation problem for the x, y, and z components. Suppose to discretize the node sequence $\mathcal{R}(N_{\text{best}})$ with a given resolution $\Delta_{\mathbf{r}}$, leading to a sequence of waypoints, $\mathbf{r}_0, \cdots, \mathbf{r}_m$, moreover let $p = 7$ and fix the knot vector as

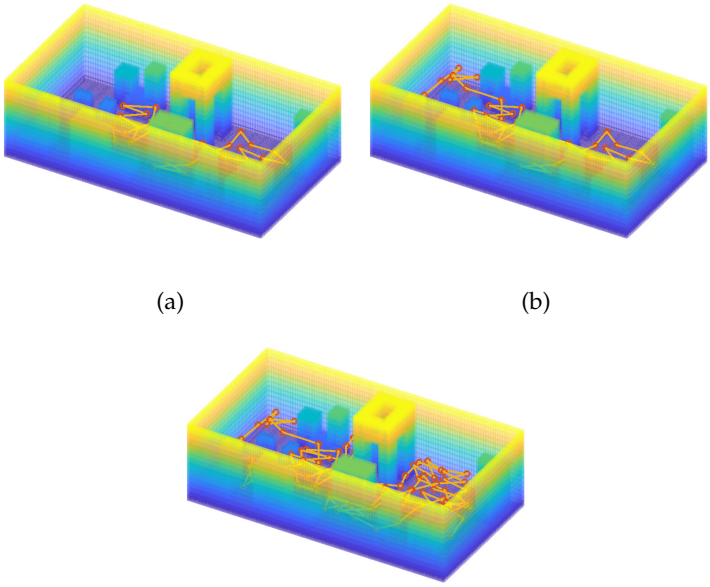
$$\boldsymbol{u} = \left[0_{0:7}, \quad \Delta_u, \quad 2\Delta_u, \quad \ldots, \quad (m-p)\Delta_u, \quad T_{0:7}\right],$$

with $T = \sum_{i=0}^{m-1} \|\mathbf{r}_i - \mathbf{r}_{i+1}\| / v_{\max}$ and $\Delta_u = T/(m-p+1)$. Then, the problem of feasible trajectory generation can be solved via the constrained quadratic programming

$$\begin{aligned} \min_{\boldsymbol{q}_0,\ldots,\boldsymbol{q}_m} \; & \lambda_1 \int_0^T \left\| \frac{d^3 \boldsymbol{s}(u)}{du^3} \right\|^2 du + \lambda_2 \sum_{i=0}^{m} \|\boldsymbol{q}_i - \mathbf{r}_i\|^2, \\ \text{sub.to.} \; & \boldsymbol{q}_i' \leq v_{\max} \quad \forall i = 0, \ldots, m-1, \\ & \boldsymbol{q}_i'' \leq a_{\max} \quad \forall i = 0, \ldots, m-2, \\ & \boldsymbol{q}_0 = \mathbf{r}_0, \quad \boldsymbol{q}_m = \mathbf{r}_m, \\ & \boldsymbol{q}_0' = v_{\text{init}}, \quad \boldsymbol{q}_0'' = a_{\text{init}}, \end{aligned} \tag{20}$$

with $\lambda_{1,2} \in \mathbb{R}$ tuning parameters and $v_{\max}$, $a_{\max}$, v_{init}, and a_{init} provided as inputs. The aforementioned optimisation problem, framed in the context of quadcopter control, tries to compute minimal effort trajectories that propagate in time close to the sampled waypoints. The B-spline properties are used here to constrain the dynamical quantities to strictly remain inside the imposed structural limits and to shape the curve along the selected viewpoints. Note that no final velocity nor acceleration constraints have been imposed, letting the algorithm select the optimal ones on its own. This choice is motivated by the discussion carried out in Section 4.2, where emerges that constraining the robot to stop at each exploration step may degrade the algorithm performance. As in the case of Bézier curves (see Section 4.2.4), the loss function in problem (20) can be fully decomposed along the tree axis, yielding

$$\begin{aligned} \min_{\boldsymbol{q}_0,\ldots,\boldsymbol{q}_m} \; & \sum_{j}^{x,y,z} \left[\lambda_1 \int_0^T \left(\frac{d^3 \boldsymbol{s}_j(u)}{du^3} \right)^2 du + \lambda_2 \sum_{i=0}^{m} \left(\boldsymbol{q}_{j,i} - \mathbf{r}_{j,i} \right)^2 \right], \\ \text{sub.to.} \; & \text{Same of problem (20).} \end{aligned}$$



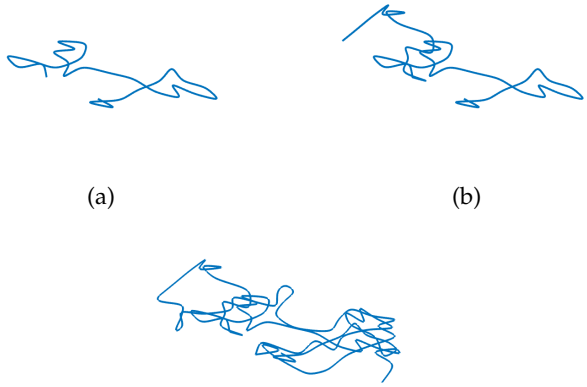
(a) (b)

(c)

Figure 12: Results of the proposed search algorithm. The figure depicts the simulated environment along with the built tree of possible viewpoints. Red circles represent the tree nodes, while the yellow connecting lines its edges. In the image, in order, (a) exploration state after only one step, (b) after two steps, and (c) after five steps. Note how the tree grows toward previously unexplored areas.

4.3.5 *Experimetal Results & Future Directions*

The proposed approach has been tested on a synthetic urban canyon scenario, similar to the one proposed during the Leonardo drone contest during the third year (see Section 1.2). In this scenario, the $10 \times 20 \times 3$ meters map was already completely known in advance and the flying agent was required to find as fast as possible a single moving unmanned ground robot. Thanks to these experiment settings the explorer was initialized with a prior guess of probability distribution that equalizes to zero near and over the known obstacles. The obtained results are reported in Figure 12, 13, and 14. In particular, Figure 12 depicts the built tree, where red circles represent nodes and the yellow connecting lines the tree edges, along with a discrete representation of the environment map. Overlapping Figure 12 with Figure 13 the blue continuous line represents the final trajectory followed by the drone during the exploration task. Finally, Figure 14 reports the used probability map. In all the figures, in order, has been depicted the exploration state after one, two, and five steps. Note how the agent tends to explore quickly previously unseen areas, while the probability increases over time in not visited areas and decreases a



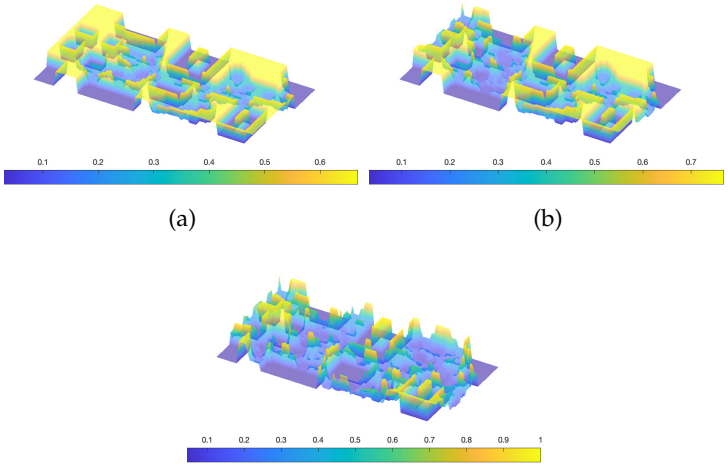
(a) (b)

(c)

Figure 13: Results of the proposed search algorithm. The figure depicts the planned reference trajectory after (a) one exploration step, (b) two exploration steps, and (c) five exploration steps. The reported trajectory is built on top of the viewpoints tree reported in Figure 12.

lot after one agent visit. The proposed solution has been successfully employed during the Leonardo challenge, where the drone was able to find out the ground agent in almost a couple of minutes, with a maximum velocity of $0.5m/s$ and starting always from random initial positions. The algorithm was able to run completely real-time with only onboard computation and sensing capabilities. For a complete list of the used parameters, please refer to Table 3.

In this section, we proposed a novel solution to the exploration problem in the setting where target search is the only final goal. This approach is in contrast with the one described in Section 4.2 under different aspects, first it plans paths instead of directly flyable trajectories, this approach may lead to degraded performances in all those cases in which the environment is not a priori known, constraining the agent to perform several stops before starting a new exploration step. On the other hand, the latter solution allows for continuous mapping of the target probability, which is constantly kept updated during the tree expansion. Future research directions in this field may aim to fill the gap between these two solutions and try to solve both the problem of environment exploration and target finding concurrently. At the same time, we wish to import the local-global paradigm even in this class of problems, which currently is not considered yet.



(a) (b)

(c)

Figure 14: Results of the proposed search algorithm. The figure depicts the costmap state during the searching task after (a) one exploration step, (b) two exploration steps, and (c) five exploration steps. The reported costmap is used to build the tree of possible viewpoints reported in Figure 12. Note how the cost decreases a lot after each agent visit, how it increases in time in not visited areas, and how the cost is almost zero on obstacle cells. The latter property can be easily integrated into the algorithm thanks to its flexibility to apriori knowledge about probability distributions.

4.4 CONTRIBUTIONS

This chapter is devoted to reviewing two original approaches to the problem of rapid exploration in two different contexts. In the first stage, the focus is on the robot's capabilities and the employed algorithm is meant to carry out the exploration task as fast as possible pushing the drone on the boundaries of its dynamical limits. While

$n_{\max}$	1500	n_{sample}	60
$R_{\min}$	$0.5m$	$R_{\max}$	$4.0m$
$v_{\max}$	$0.5m/s$	$a_{\max}$	$0.5m/s^2$
p^{inc}	1.01	$p^{\text{dec}}_{\max}$	0.5
W	$[5,0;0,5]$	c	0.0
λ_1	1.0	λ_2	10.0

Table 3: Parameters used to test the object search algorithm.

in the second stage, the focus is on the exploration for target search. In this second case, the proposed solution mixes a continuous mapping paradigm with the random nature of RRTs. It results in two approaches able to sequentially build high informative trajectories, or paths, in unknown, partially known or completely known environments. In both cases, Gaussian processes have been used to fast infer the node's information gain, and both approaches have been validated with real scenario tests. As previously mentioned, we propose, as future work to bridge the gap between these two solutions and try to solve both the problems of environment exploration and target searching concurrently, by planning directly sub-optimal and feasible trajectories. We believe that the introduction of these solutions opens a new research frontier able to cope with the gap between the learning theory, from supervised to unsupervised techniques, and the robotic world, where the limited computational power does not allows for online network training or big data processing.

5 TRAJECTORY PLANNING

5.1 THE FRAMEWORK

In the previous chapter, we tracked the problem of autonomous environment *exploration* from a very practical point of view, providing a couple of possible solutions to efficiently compute high informative trajectories. Here, we moved to a slightly different problem as the environment to navigate is completely known and the main objective is to move the autonomous agent from an initial position, to a final one, without getting in collision with the mapped obstacles and without the necessity to gather information about the surroundings. In these settings, the planned trajectory is required only to be *safe* in the sense that does not collide with any obstacles and develops inside known and free environment areas, and to fully satisfy the agent dynamical capabilities. As the trajectory is not meant to gather as much data as possible, it can be designed to optimise whatever user-defined loss function, from the overall travel time, to control effort, or high derivatives. In the Leonardo drone contest case, a trajectory planning module was required to navigate the environment and reach a goal position smoothly and quickly, so the quadrotor is steered in a way that the localisation and the collision avoidance (see later in Chapter 6) parts can work as well as possible.

In this chapter, after a brief literature review, we analyze and adapt a state-of-the-art solution to the trajectory planning problem. The proposed solution has been successfully implemented and deployed during the Leonardo drone contest. For further details the reader is referred to the original work [171].

5.1.1 *Related Works*

Although the literature is cluttered with possible solutions to the problem of trajectory planning, most of them can be classified solely into two categories: *hard-constrained* and *soft-constrained* methods. Hard-constrained methods reformulate the planning problem as a convex optimisation one, where collisions are avoided by constraining the planned path to be inside a set of convex flight corridors. This idea was pioneered by [102] who proposes to generate minimum-snap trajectories via Quadratic Programming (QP) approach, and by [128] who, in turn, presented a closed-form approach to the minimum-snap trajectory generation problem. Both the latter approaches parametrise the overall trajectory as n-segments piecewise polynomial curve, and

ensure the trajectory safety by iteratively adding intermediate waypoints. Similar approaches generate safe trajectories via convex optimisation [23, 35, 36, 53, 55, 89], but, unlike the pioneer solutions, here the problem is solved in a two-steps pipeline. First, a sequence of cubes [55], spheres [54], or polyhedrons [89], are fitted inside the free and navigable space, then the reconstructed flight corridors are used as convex constraints inside the optimisation problem. The works of [35, 36] proposed a B-Spline-based search able to generate trajectories used as an initial guess for the subsequent optimisation step. Thanks to the convex reformulation, hard-constrained methods guarantee global optimality at the expense of all possible nonconvex costs and constraints, such as distance from obstacles and conservative kinodynamic constraints, leading often to unsafe and conservative paths. On the other hand, soft-constrained methods reformulate trajectory planning as a non-linear optimisation problem able to keep smoothness, feasibility, and safety into account. In [174] the authors generate discrete-time trajectories minimizing both smoothness and risk of collision via gradient descent methods, while in [76] the optimisation is solved in a gradient-free fashion. The work of [112] extended the latter approaches to continuous-time polynomial trajectories, thanks to the continuous-time formulation it avoids numeric differentiation errors leading to a more accurate representation. However, the approach proposed by [112] suffers from low success rate, being the proposed optimisation problem solved without a good initial guess. As a matter of fact, the success rate considerably improved when [52] proposes to solve the same problem with a collision-free path used as an initial condition. The latter path is obtained via sampling-based path searching methods. In [147] the trajectory is parameterized as an uniform B-Spline, whose intrinsic continuity allows to reduce the overall number of constraints. Soft-constrained solutions make use of gradient information to push the final trajectory far from obstacles, but may suffer from local minima and do not have any guarantee of success rate and kinodynamic feasibility. The work of [171] proposes a novel soft-constrained method where the B-Spline parameterisation is used to both speed-up the cost computation and simplify the collision checking thanks to its convex hull containment property (see Appendix A).

5.2 QUADROTOR TRAJECTORY GENERATION

In this section we review a trajectory planning approach first proposed by [171]. The considered approach aims to efficiently compute minimum effort safe trajectories, balancing both the induced control effort and the overall trajectory time. This soft-constrained approach develops in two steps, first a collision-free suboptimal trajectory is computed via *hybrid-state* A* search, then the obtained solution is it-

eratively refined via non-linear optimisation. In this second step the B-Spline formulation is adopted to improve the algorithm convergency rate.

5.2.1 *Hybrid-State A⋆*

The first step consists of raw trajectory planning via A⋆ graph search, unlike the standard A⋆ formulation, here the proposed solution expands and optimises a graph of possible kinodynamic feasible trajectories (from this property the *hybrid-state* attribute). In this settings, the A⋆ optimisation is applied to a tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ consisting of a set of nodes $\mathcal{N} = \{N_1, \dots, N_{n_N}\}$ and a set of edges $\mathcal{E} = \{E_1, \dots, E_{n_E}\}$, where each node is completely defined by the following six quantities

$$\mathcal{N}_i = \left\{ \boldsymbol{x}_i, \boldsymbol{u}_i, \delta_i^t, \chi_i, g_i, h_i \right\},$$

where $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ represent the current quadrotor state and the applied input used to reach that state, δ_i^t is the amount of time for which $\boldsymbol{u}_i$ is applied, χ_i is an unique integer indexing $\boldsymbol{x}_i$ over a voxel grid, and (g_i, h_i) are two costs associated with the current node useful for the graph optimisation, in particular g_i is the *true* trajectory cost, while h_i represents a heuristic estimating the cost-to-go to reach the goal state. The graph edges are generated by motion primitives respecting the quadrotor dynamics, instead of standard straight lines. As in Section 4.2, here we enforce the differential flatness property commonly used in quadrotor planning, and express the final state trajectory through the evolution of only the three axis-wise positions so $\boldsymbol{x} = (p_x, p_y, p_z) \in \mathbb{R}^3$, where each component can be designed as a time-parameterized polynomial function

$$p_j(t) = \sum_{k=0}^{n} a_k t^k. \tag{21}$$

The aformentioned formulation correspond to a Linear Time-Invariant (LTI) system, letting $\boldsymbol{z} = \left(\boldsymbol{x}^\top, \boldsymbol{x}^{(1)^\top}, \dots, \boldsymbol{x}^{(n-1)^\top}\right)^\top \in \mathbb{R}^{3n}$ the state vector, and $\boldsymbol{u} = \boldsymbol{x}^{(n)} \in \mathcal{U} = [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}] \subset \mathbb{R}^3$ the control input, then the state-space model can be defined as

$$\dot{\boldsymbol{z}} = A\boldsymbol{z} + B\boldsymbol{u},$$

with $A \in \mathbb{R}^{3n \times 3n}$ and $B \in \mathbb{R}^{3n \times 3}$ matrices in prime form. The complete solution for the state equation, giving the quadrotor trajectory from the initial condition $\boldsymbol{z}_0$, is expressed as

$$\boldsymbol{z}(t) = \exp(At)\,\boldsymbol{z}_0 + \int_0^t \exp(A(t-\tau))\,B\boldsymbol{u}(\tau)\,d\tau. \tag{22}$$

In these settings, a set of motion primitive, leading to a set of possible graph edges for each node, can be efficiently computed by propagating Equation (22) for a fixed time step δ^t, applying a set of discretised control inputs $\mathcal{U}_{\mathrm{D}} \subset \mathcal{U}$. From a practical point of view, the

set $[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]$ can be uniformly discretised in r steps along each axis, yielding to r^3 possible motion primitives. The graph search then unfolds as a classical A* search, for each node a set of edges and children nodes are generated via motion primitive propagation, then each child is checked against dynamical feasibility and possible collision with map obstacles, and only those nodes considered as feasible are retained for future expansions. To speed-up the search a voxel grid discretisation is imposed over the trajectory space so all children nodes whose associated index χ matches in the list of already expanded cells are culled out from the open set.

In order to let the adopted search effective, we need an admissible and informative loss formulation for both the true trajectory cost and the estimated cost-to-go. Under this perspective, since the main trajectory planning objective is to generate motions balancing both control effort and overall traveling time, the natural loss formulation is

$$\mathcal{J} = T + \rho \int_0^T \|\boldsymbol{u}(\tau)\|^2 \, d\tau, \tag{23}$$

where T is the total trajectory time, $\boldsymbol{u}$ is the control action, and $\rho \in \mathbb{R}_{>0}$ is a weight coefficient balancing between minimum time and minimum effort trajectories. Under the loss function (23), the true cost g_i, representing the trajectory cost form the start position, can be computed summing up the cost of all primitives applied so far

$$g_i = g_{i-1} + \left(1 + \rho \|\boldsymbol{u}_i\|^2\right) \delta_i^t.$$

As regards the heuristic cost-to-go h_i, we need an estimation of $\mathcal{J}$ from the current state $\boldsymbol{x}_i$ to the goal one. To retrieve this estimation, we compute a closed-form trajectory, from $\boldsymbol{x}_i$ to the goal state, that minimizes the cost $\mathcal{J}$ by applying the Pontryagins minimum principle [103]. In particular, letting $n = 3$ in Equation (21), then the optimal trajectory $p(t)$ minimising $\mathcal{J}$ can be computed as

$$p_\mu^\star = \frac{1}{6}\alpha_\mu t^3 + \frac{1}{2}\beta_\mu t^2 + v_{\mu_0} t + p_{\mu_0} \quad \text{with } \mu = x, y, z, \tag{24}$$

where the parameters α_μ and β_μ follow the relation

$$\begin{pmatrix} \alpha_{mu} \\ \beta_\mu \end{pmatrix} = \frac{1}{T^{\star 3}} \begin{pmatrix} -12 & 6T^\star \\ 6T^\star & -12T^{\star 2} \end{pmatrix} \begin{pmatrix} p_{\mu_{\text{goal}}} - p_{\mu_0} - v_{\mu_{\text{goal}}} T^\star \\ v_{\mu_{\text{goal}}} - v_{\mu_0} \end{pmatrix},$$

and p_{μ_0}, $p_{\mu_{\text{goal}}}$, v_{μ_0}, and $v_{\mu_{\text{goal}}}$ represent the initial and goal position and velocity, respectively. The optimal time $T^\star$ can be compute by deriving Equation (24) and plugging it inside Equation (23)

$$T^\star = \arg\min_{T>0} \sum_{\mu \in \{x,y,z\}} \frac{1}{3}\alpha_\mu^2 T^3 + \alpha_\mu \beta_\mu T^2 + \beta_\mu^2 T.$$

The optimal cost $\mathcal{J}^\star = \mathcal{J}(p^\star, T^\star)$ is then used in place of h_i as cost-to-go estimation.

Remark 5.2.1. *The choice $n = 3$ leads to* minimum jerk *trajectories as the control action $\boldsymbol{u} = \boldsymbol{x}^{(3)}$.*

Remark 5.2.2. *The computed optimal trajectory $p^\star$ contributes to the planning procedure more than only providing an admissible heuristic. As a matter of fact, $p^\star$ is useful twice as it can be used to stop the search in advance, provided that the computed trajectory satisfies all dynamical and collision constraints. Moreover, such an analytic expansion is of fundamental importance as, due to the control discretisation, it is difficult to have a primitive end exactly in the goal state.*

5.2.2 *B-Spline Trajectory Optimisation*

The second step of the proposed trajectory planning method consists of a trajectory optimisation procedure. This step is fundamental in smoothing the initial trajectory produced by the path searcher, which is suboptimal due to the control discretisation, and to push the final trajectory away from the obstacles, since the distance information in the free space is ignored at search time. Here, as in Section 4.3.4, to speed-up the solver and improve the planner success rate, we enforce the use of B-Spline curves (see Section A) as time parameterisation for the quadrotor trajectory. As already mentioned in Section 4.3.4, B-splines are defined by a set of control points $\mathcal{CP} = [\boldsymbol{q}_0, \dots, \boldsymbol{q}_m]$, and a set of time knots $\boldsymbol{u} = \left[u_0, \dots, u_{m+p+1}\right]$, as

$$\boldsymbol{s}(u) = \sum_{i=0}^{m} B_i^p(u)\, \boldsymbol{q}_i,$$

with p order of the spline and B_i^p basis functions, recursively computed via the De-Boor formula [33]. The use of B-Spline in trajectory optimisation is attractive thanks to its convex hull containment property, which allows approximating the collision condition of the overall trajectory by directly conditioning its control points $\mathcal{CP}$ lying to a free non-necessary convex environment area. Moreover, by evaluating the control points distance from the obstacles we can directly assess the trajectory safeness. Another useful property is represented by the closure with respect to integration and derivation, so the derivatives $\boldsymbol{s}^{(d)}$ of a B-Spline still maintain a B-Spline structure with control points

$$\boldsymbol{q}_i^{(d)} = p\frac{\boldsymbol{q}_{i+1}^{(d-1)} - \boldsymbol{q}_i^{(d-1)}}{u_{i+p+1}^{(d-1)} - u_{i+1}^{(d-1)}}.$$

These properties, jointly with the differential flatness one, can be used to ease the trajectory optimisation procedure as both the collision and dynamics constraints can be refurmulated linearly in terms of the trajectory control points, which are then used as decision variables.

For this purpose, we fix an uniformly distributed knot vector of the form

$$\boldsymbol{u} = \left[0_{0:7},\quad \Delta_u,\quad 2\Delta_u,\quad \ldots,\quad (m-p)\Delta_u,\quad T_{0:7}\right],$$

with $\Delta_u = T/(m-p+1)$, and T the total trajectory time computed in the search phase. Then we formulate the trajectory optimisation step as an uncostrained optimisation problem of the form

$$\min_{\boldsymbol{q}_0,\boldsymbol{q}_1,\cdots,\boldsymbol{q}_m} \lambda_\mathrm{s}\mathcal{J}_\mathrm{s} + \lambda_\mathrm{c}\mathcal{J}_\mathrm{c} + \lambda_\mathrm{f}\left(\mathcal{J}_\mathrm{v} + \mathcal{J}_\mathrm{a}\right), \tag{25}$$

where $\lambda_\mathrm{s} \in \mathbb{R}_{<0}$, $\lambda_\mathrm{c} \in \mathbb{R}_{<0}$, and $\lambda_\mathrm{f} \in \mathbb{R}_{<0}$ are weight coefficients balacing between trajectory smoothness $\mathcal{J}_\mathrm{s}$, distance from obstacles $\mathcal{J}_\mathrm{c}$, velocity feasibility $\mathcal{J}_\mathrm{v}$, and acceleration feasibility $\mathcal{J}_\mathrm{a}$. Each cost is formulated taking into account the B-Spline parameterisation and properties. The smoothness cost $\mathcal{J}_\mathrm{s}$ tries to pursue the same goal carried out by the searcher to minimise the overall control effort, thus with $n = 3$, $\mathcal{J}_\mathrm{s}$ reads to

$$\mathcal{J}_\mathrm{s} = \int_0^T \left\| \frac{d^n \boldsymbol{s}(u)}{du^n} \right\|^2 du,$$

which can be computed in closed form for B-Spline parameterisation (see Section A). The collision cost $\mathcal{J}_\mathrm{c}$ is formulated as the repulsive force of the obstacles acting on each control point

$$\mathcal{J}_\mathrm{c} = \sum_{i=0}^{m} \begin{cases} \left(d(\boldsymbol{q}_i) - d_\mathrm{thr}\right)^2 & d(\boldsymbol{q}_i) \leq d_\mathrm{thr}, \\ 0 & d(\boldsymbol{q}_i) > d_\mathrm{thr}, \end{cases}$$

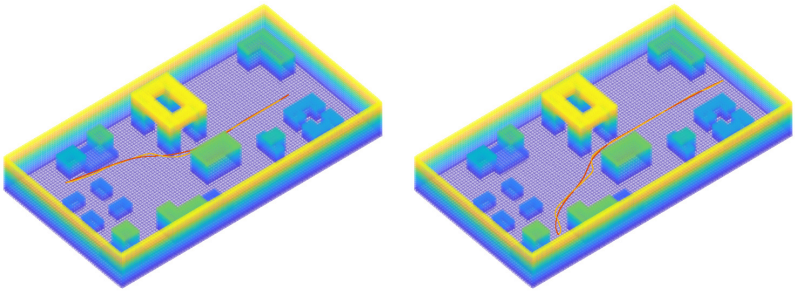
where $d(\boldsymbol{q}_i)$ is the distance of $\boldsymbol{q}_i$ from the closer mapped obstacle, and d_thr is a minimum distance considered as *safe*. Finally the velocity and acceleration feasibility cost is formulated as

$$\mathcal{J}_\mathrm{v} = \sum_{\mu=\{x,y,z\}} \sum_{i=0}^{m-1} \begin{cases} \left(\boldsymbol{q}_i^{(1)^2} - v_\mathrm{max}^2\right)^2 & \boldsymbol{q}_i^{(1)^2} \leq v_\mathrm{max}^2, \\ 0 & \boldsymbol{q}_i^{(1)^2} < v_\mathrm{max}^2, \end{cases}$$

$$\mathcal{J}_\mathrm{a} = \sum_{\mu=\{x,y,z\}} \sum_{i=0}^{m-2} \begin{cases} \left(\boldsymbol{q}_i^{(2)^2} - a_\mathrm{max}^2\right)^2 & \boldsymbol{q}_i^{(2)^2} \leq a_\mathrm{max}^2, \\ 0 & \boldsymbol{q}_i^{(2)^2} < a_\mathrm{max}^2, \end{cases}$$

where v_max and a_max are the axis-wise maximum velocity and acceleration, respectively.

Remark 5.2.3. *The problem (25) does not ensure neither safety, nor dynamical feasibility as the constraints has been softly injected inside the cost. Nevertheless, there exists a local optimal close to the initial trajectory planned by the searcher, so the problem (25) quickly converges to the local optimal solution if initialised with a good and feasible initial condition.*



(a) (b)

Figure 15: Results of the proposed trajectory planning algorithm. The figure depicts the simulated environment along with the planned trajectory by the searcher (yellow line), and the output of the trajectory optimisation step (red line). Notice how the optimised trajectory smoothly follows the yellow path without colliding with the environment obstacles.

(a) (b)

Figure 16: Results of the proposed trajectory planning algorithm. The picture reports the same results displayed in Figure 15 without the environment map for better visualisation.

5.2.3 *Experimental Results*

The results obtained applying the proposed approach are reported in Figures 16a and 16b, where the yellow lines represent the trajectories computed by the path searcher, while the red lines are the outputs of the trajectory optimisation step. As the reader can observe, both yellow and red lines lie to the free part of the environment, ensuring safeness of the overall trajectory, moreover the yellow path presents sharp turns which are not present in the red one. This latter property is the smoothing effect introduced by the optimiser, that, unlike the search part, makes use of a continuous set of possible inputs $\boldsymbol{u}$.

5.3 CONTRIBUTIONS

The chapter is devoted to discuss the problem of trajectory planning in completely known and mapped environments. The discussion unfolds by first reviewing the current state-of-the-art solutions then we review and implement one of the most promising approach to the problem of quadrotor trajectory planning in cluttered environments. The proposed solution has been deeply analyzed, implemented, and deployed during the Leonardo drone contest with great results. Moreover, the implemented solution has been fully integrated inside a complete navigation pipeline which makes use, and thus may be affected by noise and errors, of a localisation, mapping, planning, replanning, and control pipeline. The implemented solution has been tested in a bunch of different scenarios where the surrounding map was continuously kept updated and the planning procedure may, sometimes, not have a feasible solution.

6

TRAJECTORY REPLANNING FOR OBSTACLE AVOIDANCE

6.1 THE FRAMEWORK

React to the unknown, avoid previously unseen obstacles, and replan trajectories in real-time are three basics capabilities that take part inside the set of elementary abilities that an autonomous robot must master before being able to start moving in real-world environments, cluttered of objects, people, animals, cars, and hopefully other robots, which may behave unpredictably. Due to the importance of the problem at hand, a huge number of solutions have been presented in the literature, first in a pure path planning approach [98], where the timing law is not considered during the planning stage and its computation is left as a posteriori task, later in a direct trajectory planning framework [36], where the path and timing law are concurrently computed, letting the planner to exploit the full robot capabilities and to shape the path in order to ease the time allocation for a final feasible trajectory. Although we saw very impressive works in this field, very few of them try to approach the planning, or replanning, problem from a numeric point of view, and even fewer try to use consolidated control system theory to guarantee the convergence of the planning algorithm. It results that, although the proposed solutions work well in simulations, or in not-too-complex environments, their failure rate in real environments is very high. Besides that, careful parameters tuning is often required, yielding to a very poor generalization. With the aforementioned problems in mind, we try to contribute by firstly analyzing, and implementing on our flight platform, a carefully selected state-of-the-art replanning algorithm, to test its performance in real-world scenarios, then we try to perturb the current trend of purely algorithmic solutions by proposing two numerical and control-oriented approaches. In this chapter, we aim to review three different avoidance techniques starting from a purely algorithmic solution, borrowed from the state of the art, to more numerical and control-oriented ones, which instead represent original unpublished works. The core concept behind the proposed solutions is represented by the idea that treating the problem of obstacle avoidance numerically allows assessing its performances via stability analysis tools, borrowed from the control theory field, which ensures convergence in all cases where some conditions are fulfilled, unlike purely algorithmic approaches where the convergence to a correct solution is not guaranteed at all. Although the reader hardly finds in this the-

sis the aforementioned stability analysis, the algorithm presentation, as well as the research carried out in this field, is oriented in this direction.

6.1.1 *Related Works*

The trajectory replanning framework is a large field of study that may embrace several different solutions, often quite variegated with different inputs and outputs, and suitable for different environmental conditions where only static obstacles are present, where moving objects may suddenly appear, or where other robotic agents move, with the possibility to networking and share motion information. Due to this variety, the referring literature is as large as to be unmanageable, even focusing on a specific problem, so we choose to mention here the most important works which motivated this thesis more. In particular, we first focused on the problem of static obstacles avoidance, then moved to the case with moving objects, overlooking the multi-agent case, being that a specialization of the more general one where objects move with a priori unknown pattern. In this field, the first sharp split in the research direction was caused by the introduction of the Bézier curves as a new trajectory parameterisation approach, pioneer of this idea was [25], where this parameterisation was used to plan fixed-wing aircraft trajectories. The good properties of such curves allowed for easy arc length computation and collision checking, thanks to which the approaches developed with this tool outperform classical solutions based on polynomial trajectories [112, 128]. The work [25] introduced also a novel concept of time allocation as curve composition, allowing to use the same curve properties not only for spatial paths but also to shape the associated timing law. This innovative idea dropped into obsolescence due to the complexity of evaluating high-order derivatives. As a matter of fact, further works in this field, which exploit the Bézier curves properties, were devoted only to modifying the current agent path, ensuring dynamic continuity, but without keeping into account the possibility of locally changing the allocated timing law [98, 99]. Bézier curves were subsequently used in an uncountable number of works, together with their strict brother, the B-splines. The work of [54] proposes to use Bézier curves to plan trajectories inside the pointclouds retrieved from LiDAR sensor updates, while [172] makes use of B-splines to ensure trajectory feasibility during optimisation. The works of [117] and [118] exploit the convex hull containments property to avoid collisions, constraining the planned curve inside a set of convex safe corridors iteratively updated in time, while [141] used B-splines to represent both robot and obstacle dynamics, a carefully selected and optimized set of separation planes was used to keep the planned path safe. A new recent trend was finally introduced by [36], who first proposed a novel

sampling-based approach meant for directly planning B-spline trajectories. The basic idea was to translate the classic robot pose sampling, into a higher dimensional space where each sample configuration represents directly a piece of possible trajectory, allowing for dynamical considerations. Although some state-of-the-art works may consider time allocation as an active part of the problem at hand [36, 118, 141] there is no explicit optimization procedure that allows its computation yet, losing a set of degrees of freedom especially useful in highly cluttered environments. Besides the approaches just analyzed, which consider a time-parameterised trajectory as algorithm output, there exist other solutions that try to elevate the problem abstractness to a higher level, considering velocity, or acceleration, as algorithm output, allowing for easier control algorithms at the plant side. In this field, it is worth mentioning the works of [27] and [45] which select the robot velocity on the basis of a reconstructed map of the environment in the former case, and on the basis of event camera updates in the latter one. Furthermore, learning-based techniques obtained impressive results when trained to compute safe velocity commands [90, 91]. Finally, other approaches take into consideration the full robot dynamics via model predictive control [120], or control barrier functions [78, 133, 155] solutions. The latters, although more complicated, allow the use of control system tools to verify stability, ensuring convergence to a feasible solution under a set of well-specified assumptions.

6.1.2 *Problem Definition*

The problem of trajectory replanning is strictly linked to the problem of planning, with the only difference that, in the former case, is supposed a prior notion of a dynamically feasible trajectory, which may become unfeasible due to possible collisions when new sensor readings become available. Let $(x(t), u(t))$ be an initial feasible state trajectory, solution of

$$\dot{x} = f(x, u),$$

and satisfying $x \in \mathcal{X}$ and $u \in \mathcal{U}$. Where $f(\cdot)$ represents the robot dynamics, while $\mathcal{X} \in \mathbb{R}^{n_x}$ and $\mathcal{U} \in \mathbb{R}^{n_u}$ are the allowed sets for x and u, respectively. Then the problem of replanning can be formalized as the computation of the tuple $(x^\star(t), u^\star(t))$ as a solution of the optimisation problem

$$\begin{aligned} &\min_{x', u'} \left\| x - x' \right\| + \left\| u - u' \right\|, \\ \text{subj. to} \quad & x' \in \mathcal{X}', \quad u' \in \mathcal{U}', \\ & \dot{x}' = f\left(x', u'\right), \end{aligned} \tag{26}$$

with $\mathcal{X}'$ and $\mathcal{U}'$ the new allowed sets computed after the integration of the new sensor readings. Note that (26) addresses the problem of

replanning by looking for a new feasible trajectory that lies as close as possible to the original one. This choice is motivated by the idea that if an initial trajectory was provided, it was optimal for the task at hand, and computed by minimising a precise index. Although problem (26) embraces a large number of use cases addressed in the literature, it misses the case where the safe set $\mathcal{X}$ is not fixed in time, thus in the environment are present moving objects, or other agents. In this view, let us suppose to have the knowledge of how these objects, or agents, move, having at hand an estimation of their current trajectory in terms of $\mathbf{r}_o = (x_o, y_o, z_o)$ position. Moreover, let $\Gamma(x)$ the map able to extract, from the current state trajectory, only the x, y, z components expressing the agent position in space. Thus problem (26) can be rewritten as

$$\begin{aligned} &\min_{x',u'} \left\|x - x'\right\| + \left\|u - u'\right\|, \\ \text{subj. to} \quad & x' \in \mathcal{X}', \quad u' \in \mathcal{U}', \\ & \dot{x}' = f\left(x', u'\right), \\ & \left\|\Gamma\left(x\left(t\right)\right) - \mathbf{r}_o\left(t\right)\right\| > d_{\text{safe}} \quad \forall t \in \mathbb{R}_+, \end{aligned}$$

with $d_{\text{safe}} \in \mathbb{R}_{\geq 0}$ expressing the minimum distance intra-agent, or between the considered agent and the moving obstacle. In the particular case of a quadcopter, its differential flatness property allows to ease the optimisation procedure by directly optimise on the space of its flat outputs $\sigma = (x, y, z, \phi) = (\mathbf{r}, \phi)$. For more details about this property please refer to Section 2.2.

6.2 ON FLIGHT TRAJECTORY REPLANNING

In this section we review and adapt a purely algorithm solution to the problem stated in Section 6.1.2, in particular we focused on the solution proposed by [172] being the state-of-the-art work that shows the most promising results. We borrowed from [172] the core idea, while the algorithm implementation, as well as its integration inside our operative framework has been completely developed internally. The reviewed solution has been successfully coupled with a fast and reliable perception stack, that allows to detect possible collisions and triggers the replanning system, as well as the control layer. Moreover, we proposed a novel solution, exploiting the B-spline properties, to fast link the replanned trajectory with the previously provided one, ensuring continuity up to the adopted spline order. Experimental results show how the proposed stack is effective in fast replanning colliding trajectories, and how continuity is always guaranteed.

The proposed replanning system takes the outputs of a global planning procedure, along with the perception, or mapping, output, and the current robot position, and deforms the global reference trajectory locally to avoid previously unknown obstacles. The replanning

works in two steps. Firstly, a set of local guiding paths are generated through the free space, although may there exists an infinite number of possible paths we restrict the final choice only to those dinstictive paths considered as *topologically different*, by pruning the ones bringing more detour from the initial planned trajectory. Secondly, a B-spline-based Path-Guided Optimisation (GTO) step is in charge to build a set of locally optimal trajectories out from the found paths. The "best" trajectory is then extracted and executed.

6.2.1 *Collision Perception & Replanning Trigger*

In order to check the trajectory safeness, we employ two different data structures (a) an Euclidean Signed Distance Field (ESDF), which is in charge to represent the known environment obstacles, and is continuously updated with new sensor readings, and (b) a ktree structure built on the last pointcloud computed via stereo image matching. The necessity to employ two different data structures may seem as an useless redundancy, since them are actually representing the same information, but is of fundamental importance to maintain the time consistency of the data. As a matter of fact, the integration of new measurements inside any map structure is costly and cannot be made completely real-time, moreover checking the current trajectory in the sensor frame may improve the solution robustness against position estimation errors and sensor noise. The same structures are then later used to generate the topological paths, and to steer the path-guided optimisation away from the detected obstacles. The perception layer works by continuously checking, at each new sensor reading, the currently running trajectory for possible collisions inside both the current pointcloud, via ktree nearest search, and inside the current reconstructed ESDF. In particular, the running trajectory is checked for collisions in a sliding time window of fixed-dimension $T_{\max}$, at a fixed resolution of Δ_T. The initial time of the window is chosen in order to correspond exactly to the current robot position, while $T_{\max}$ results from a tradeoff between algorithm performances, sensor range, and the robot capabilities. If a collision is detected very close to the current position, let's say there exists a minimum allowed time $t_{\min}$, then an emergency stop procedure is triggered, and the robot tries to apply all efforts in slowing down and stop as fast as possible. On the other hand, if a collision is detected on a feasible time, the replanning procedure is triggered. In order to let the solution be resilient versus sensor noise and false detections, we inject a lower bound $N_{\min}$ of consecutive detections before considering the current trajectory unsafe.

6.2.2 *Topological Path Searching*

The core of the proposed solution is a Gradient-based Trajectory Optimisation (GTO) which allows to formulate locally optimal and safe trajectories in real-time. GTO methods, which typically formulate trajectory generation as non-linear optimization problems, trading off smoothness, safety and dynamic feasibility, are shown to be particularly effective in local replanning [52, 112, 147, 170], but previous works [131] showed that GTO methods are very sensitive to unfavorable initialization, which may even lead to unfeasible solutions. Typical GTO methods incorporate the gradients of a ESDF in a collision cost to push the trajectory out of obstacles. Yet there are some "valleys" or "ridges" in the ESDF, around which the gradients differ greatly. Consequently, if a trajectory is in collision and crosses such regions, the gradients of ESDF will change abruptly at some points. This can make gradients of the collision cost push different parts of the trajectory in opposing directions and fail the optimisation. Normally, such points, which correspond to the space that has an identical distance to the surfaces of nearby obstacles, are difficult to avoid, especially in complex environments. Therefore, optimization depending solely on the ESDF fails inevitably at times. To solve the problem, it is essential to introduce extra information that can produce an objective function whose gradients consistently deform the trajectory to the free space. For this reason, we adopt a sampling-based topological path searching to find a collection of distinctive paths, later used inside the proposed GTO method reformulated as a GPO.

Whereas there exists an infinite number of possible paths, we restrict the GPO procedure to apply to a subset of distinctive paths, that are considered topologically different. In this sense, we employ the notion of *Uniform Visibility Deformation* (UVD) firstly introduced in [172], which provides a constructive method to assess the sampled paths equivalency.

Definition 6.2.1. *Two trajectories* $s_1(u)$, $s_2(u)$ *parameterized by* $u \in [0,1]$ *and satisfying* $s_1(0) = s_2(0)$, $s_1(1) = s_2(1)$, *belong to the same uniform visibility deformation class, if for all* u, *the line* $\overline{s_1(u)\, s_2(u)}$ *is collision-free.*

The proposed solution works by building a roadmap capturing an abundant set of paths from different UVD classes. Unlike standard roadmap planning algorithms, which create maps containing many redundant loops, the adopted method generates a more compact roadmap where each UVD class contains just one or a few paths. The final roadmap is iteratively created as a graph connecting a series of nodes, randomly sampled in the x, y, z configuration space. Each sampled node can be recognised as a *guard* or a *connector*. Guards are responsible for exploring different parts of the free space, while connectors connect two guards to create a feasible piece of path. Any two guards $g_1 \in \mathbb{R}^3$ and $g_2 \in \mathbb{R}^3$ of the graph cannot be visible to

each other, i.e. the line $\overline{g_1 g_2}$ is in collision, thus every time a sampled point is invisible to all other guards, a new guard is created. On the other hand, if a sampled point is visible at least from two guards, a new connector is created and all possible paths, connecting the latter node to all visible guards, are generated. Finally, if a sampled node is visible from only one guard is discarded. The newly generated paths are added to the global roadmap only if belong to a different UVD class, or the counterpart of the same class represents a longer path. In the beginning, the roadmap graph is initialized with exactly two guards, representing the starting and goal points. The start position is computed as the point corresponding to half of the detected collision time t_{coll} (i.e. $t_{\text{start}} = 0.5 t_{\text{coll}}$), while the goal is selected as the point far at least d_{obs} from the detected collision point, at time t_{goal}. In this setting, halving the collision time turns out to be effective to instantiate enough time for the replanning procedure, while the robot is moving, and d_{obs} is provided as an input parameter. We stress the fact that d_{obs} is representative of a priori notion about the surrounding environment, being it ideally the maximum obstacle size that the robot should avoid. A wrong tuning of d_{obs} may lead to further unnecessary replanning procedure, which may even fail if the guessed value is too far from the real one. The graph growth continues until a limit of time $\Delta_{\max}$ of a limit of sampling nodes $N_{\max}$ is reached. With the roadmap at hand, a depth-first search algorithm, augmented by a visited node list, is applied to search for all the possible paths between the start and goal node, in a similar way as done in [129].

6.2.3 *Path Shortening and Optimisation*

The extracted paths may present two distinctive pathologies, on the first stage they may present very high detours from the original trajectories, and in the second stage may be redundant in the UVD sense, even if redundant connections between two guards are avoided. In order to correct these unwanted pathologies, we firstly search for topologically equivalent shortcut path for each one, then we check the equivalence between any two paths and only preserve topologically distinct ones. In particular, to perform shortening, each found path is discretised with a fixed resolution d_{res} and a new path is generated by adding the discretised points, one to each other, only if the considered point is not visible from the last added point. The aforementioned procedure generates shorter, but infeasible, paths, due to not visibility condition. To solve this problem, all infeasible points of the new paths are pushed away from obstacles in the direction of the ESDF gradient, of a fixed distance d_{safe}. Then the generated paths are checked for equivalence to preserve only topologically distinct ones. It is worth to remark that the number of distinctive paths grows exponentially with the number of obstacles, and in case of complex

environments, it is computationally intractable to use all paths. For this reason, we only select the first N_{paths} shortest paths.

Once a set of topologically different short paths, connecting the start and goal node, have been collected, a PGO procedure is employed to optimise all found paths and allocate a feasible timing law. The optimisation procedure is efficiently performed using B-spline parameterisation and by parallelizing the computation load on all available cores. The proposed PGO method follows a two steps approach, the first phase is devoted to generate a *warmup trajectory* by deforming the current one toward the selected path, that lies on the free and flyable space. While, in the second phase the obtained solution is iteratively refined via nonlinear optimisation to push it away from obstacles and to guarantee its dynamical feasibility. Going down in the details, the trajectory segment in collision is reparameterized as a p-degree B-spline $s(u)$ with control points $\mathcal{CP} = [\boldsymbol{q}_0, \ldots, \boldsymbol{q}_m]$ and knot vector

$$\boldsymbol{u} = \left[0_{0:p}, \quad \Delta_u, \quad 2\Delta_u, \quad \ldots, \quad (m-p)\Delta_u, \quad T_{0:p}\right],$$

where m is given by the chosen discretisation resolution d_{res}, while T correspond with the time elapsed from the selected initial to the goal point. In this setting, the first phase correspond to the solution of the following optimisation problem

$$\min_{\boldsymbol{q}_0,\ldots,\boldsymbol{q}_m} \lambda_1 \int_0^T \left\| \frac{d^3 \boldsymbol{s}(u)}{du^3} \right\|^2 du + \lambda_2 \sum_{i=0}^{m} \|\boldsymbol{q}_i - \mathbf{r}_i\|^2, \tag{27}$$

with $\lambda_1, \lambda_2 \in \mathbb{R}_+$, and $\mathbf{r}_i$ with $i = 0, \ldots, m$ represents the discretised points obtained from the collected topological paths. In problem (27), the first loss term aims to improve the final trajectory smoothness, while the second one penalizes its distance from the guiding path. Both the terms can be simplified in their formulation using the B-spline properties (Section A), yielding to an unconstrained quadratic programming problem, that can be easily solved in closed form. The first phase outputs a smooth trajectory in the vicinity of the guiding path. Since the path is already collision-free, usually the warmup trajectory is also so. Even though it is not completely collision-free, its major part will be attracted to the free space. At this stage, the gradients of ESDF along the trajectory vary smoothly, and the gradients of the objective function push the trajectory to the free space in consistent directions. In the second phase, we adopt a nonlinear

optimisation framework to further refine the warmup trajectory into a smooth, safe, and dynamically feasible one.

$$\begin{aligned} \min_{\boldsymbol{q}_0,\dots,\boldsymbol{q}_m} \ & \lambda_1 \int_0^T \left\| \frac{d^3 \boldsymbol{s}(u)}{du^3} \right\|^2 du + \lambda_2 \sum_{i=0}^{m} \mathcal{F}(\boldsymbol{q}, d_{\text{safe}}), \\ \text{sub.to.} \ & \boldsymbol{q}_i' \leq v_{\max} \quad \forall i = 0,\dots,m-1, \\ & \boldsymbol{q}_i'' \leq a_{\max} \quad \forall i = 0,\dots,m-2, \\ & \boldsymbol{q}_0 = \mathbf{r}_0, \quad \boldsymbol{q}_m = \mathbf{r}_m, \\ & \boldsymbol{q}_0' = v_{\text{init}}, \quad \boldsymbol{q}_0'' = a_{\text{init}}, \end{aligned} \tag{28}$$

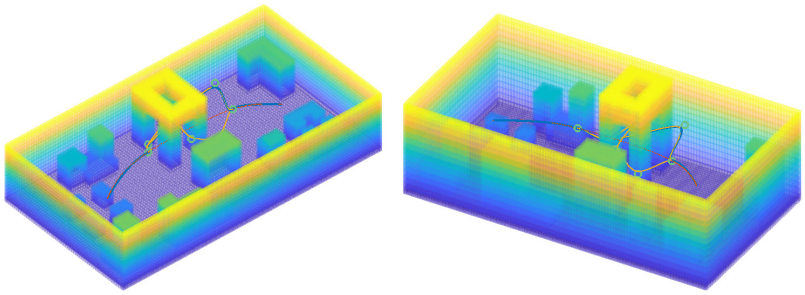
where $\mathcal{F}(\boldsymbol{q}, d_{\text{safe}})$ shapes as

$$\mathcal{F}(\boldsymbol{q}, d_{\text{safe}}) = \begin{cases} 0 & \text{if } d(\boldsymbol{q}) \geq d_{\text{safe}}, \\ (d(\boldsymbol{q}) - d_{\text{safe}})^2 & \text{if } d(\boldsymbol{q}) < d_{\text{safe}}. \end{cases}$$

In the aforementioned equation, $d(\boldsymbol{q})$ is the distance of $\boldsymbol{q}$ from the closest obstacle evaluated using both the ESDF map and the ktree built out of the current sensor reading. Once all topological paths as been time parameterised and optimised, the trajectory leading to the minimum cost is extracted and executed. Although the proposed PGO has one more step of optimization compared with previous methods, it can generate better trajectories within shorter time. The first-phase takes only negligible time, but generate a warmup trajectory that is easier to be further refined, which improve the overall efficiency.

6.2.4 *B-Spline Trajectory Injection*

The optimisation problem (28) yields to locally optimal trajectories guaranteed to be continuous up to the second derivative, with the initial colliding trajectory. This is a fundamental feature since the tree segments, namely the first initial trajectory, the replanned piece, and the final one, must be executed one after each other, consecutively. Some issues may arise when the replanning procedure is called several times, during the execution of a previously replanned segment. Indeed, the replanner may commission an even large number of trajectory pieces, that quickly become intractable for the reference generator. Moreover, if the application at hand requires a higher level of continuity, this new requirement must be encoded inside (28) which at the end may take too much time to solve. In this section we propose a novel method to join the new replanned segment with a previously computed B-spline trajectory. The proposed method is as simple as effective, it ensures continuity up to the spline order and outputs only one trajectory, allowing for any replanning procedure as required by the surrounding environment. The key idea comes from the B-spline property to be shaped, at each time instant $u \in [0, T]$, by only $p + 1$



(a) (b)

Figure 17: Results of the reviewed approach in the synthetic environment adopted in the Leonardo drone contest. The solution was able to replan feasible and safe trajectories in real-time, without forcing the quadcopter to an emergency stop. In the figure, the red path represents the initial colliding trajectory, the yellow points are the optimised hypothesis for replanned trajectory, and the blue path represents the final choice. Images (a) and (b) depict the same simulation, captured from two different points of view.

control points. It follows that, splitting the control points vector exactly in the correspondence of the final $(p+1)$th control point that spans the curve at time t_{start} and the first control point that spans the curve at t_{goal}, allows for inserting the new set of points, identifing the replanned curve, without loosing any spline continuity feature. Finding the splitting points can be easily done by checking the knot span for the first $u_i \geq t_{\text{start}}$ and $u_j \geq t_{\text{goal}}$, then converting the found value in terms of the corresponding control point indeces i_{cp} and j_{cp} as

$$\begin{aligned} i_{\text{cp}} &= i - \lceil p/2 \rceil - 2, \\ j_{\text{cp}} &= j - \lceil p/2 \rceil - 1. \end{aligned}$$

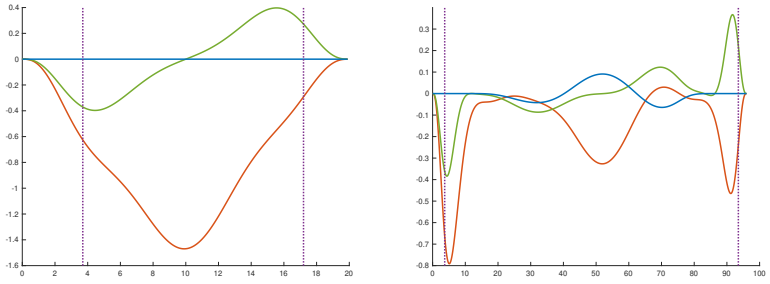
Giving two control points sequences $\mathcal{CP}^1 = \left[\boldsymbol{q}^1_0, \ldots, \boldsymbol{q}^1_{m^1}\right]$ and $\mathcal{CP}^2 = \left[\boldsymbol{q}^2_0, \ldots, \boldsymbol{q}^2_{m^2}\right]$, with the corresponding knot vectors $\boldsymbol{u}^1 = \left[0_{0:p}, \Delta^1_u, \ldots, (m^1 - p)\Delta^1_u, T^1_{0:p}\right]$ and $\boldsymbol{u}^2 = \left[0_{0:p}, \Delta^2_u, \ldots, (m^2 - p)\Delta^2_u, T^2_{0:p}\right]$, representing the initial and replanned trajectory respectively, then the composed curve with control points

$$\mathcal{CP} = \left[\boldsymbol{q}^1_0, \ldots, \boldsymbol{q}^1_{i_{\text{cp}}}, \boldsymbol{q}^2_0, \ldots, \boldsymbol{q}^2_{m^2}, \boldsymbol{q}^1_{j_{\text{cp}}}, \ldots, \boldsymbol{q}^1_{m^1}\right],$$

and knot vector

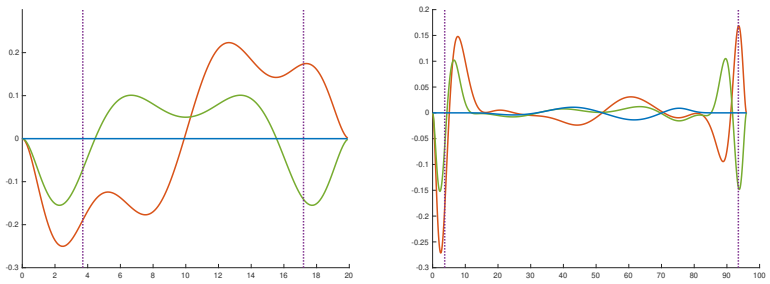
$$\boldsymbol{u} = \left[0_{0:p}, \quad \Delta^1_u, \quad 2\Delta^1_u, \quad \ldots, \quad ((m^2 + m^1 - j_{\text{cp}} + i_{\text{cp}}) - p)\Delta^1_u, \quad T_{0:p}\right],$$

preserves the path described by the consecution of the aforementioned three segments, ensuring continuity up to the chosen spline order.



(a) (b)

Figure 18: Initial velocity trajectory (a) and replanned one (b). In the images, the red line represents the velocity along the x-axis, the green one is the velocity along the y-axis, and the blue represents the velocity along the z-axis. The two vertical purple lines mark the initial and final cutting points, where the initial trajectory is broken and reconnected with the replanned one. Note that the velocity continuity is completely preserved. In both cases the velocity is kept below the safe level of $1.5m/s$.

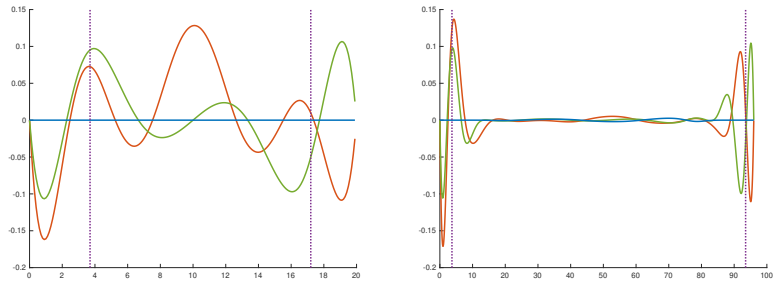


(a) (b)

Figure 19: Initial acceleration trajectory (a) and replanned one (b). In the images, the red line represents the acceleration along the x-axis, the green one is the acceleration along the y-axis, and the blue represents the acceleration along the z-axis. The two vertical purple lines mark the initial and final cutting points, where the initial trajectory is broken and reconnected with the replanned one. Note that the acceleration continuity is completely preserved. In both cases the acceleration is kept below the safe level of $0.5m/s$.

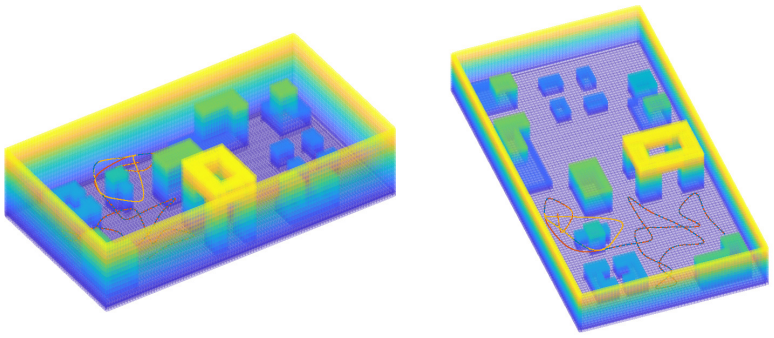
6.2.5 *Experimental Results*

The proposed approach has been successfully applied to the synthetic environment proposed during the Leonardo drone challenge (see Section 1.2), in two meaningful contexts. In the first stage, we supposed that the robot was performing a point-to-point trajectory to reach a particular area, the committed initial trajectory was computed without the obstacles knowledge, leading to an unsafe motion.



(a) (b)

Figure 20: Initial jerk trajectory (a) and replanned one (b). In the images, the red line represents the jerk along the x-axis, the green one is the jerk along the y-axis, and the blue represents the jerk along the z-axis. The two vertical purple lines mark the initial and final cutting points, where the initial trajectory is broken and reconnected with the replanned one. Note that the jerk continuity is completely preserved. No jerk limits have been fixed in this simulation.



(a) (b)

Figure 21: The reviewed approach applied to the specific case of exploration. In the image the trajectory has been replanned without a real collision in order to test its performance against previously unseen static obstacles, which may appear during the exploration task. As emerges from the figure, the replanning stack was able to successfully replan the exploring trajectory.

Next, we recreated the same exploration settings of Section 4.1.2, where the robot was performing an exploration task driven by the algorithm proposed in Section 4.3. In this second case, the initial trajectory was computed using the environment knowledge, thus the performed trajectory was not in collision with obstacles, in order to trigger the replanning procedure we inject a false sensor update containing an obstacle exactly on the motion direction. The obtained results are reported in Figure 17 for the first case, and in Figure 21 for the exploration case. In both the images is reported the environment

occupancy map, the initial trajectories, the optimised ones, and the sampled nodes. In particular, the red continuous line represents the initial colliding motion, which requires replanning, in Figure 17 the collision is particularly clear, while in Figure 21 the red path is not colliding with any obstacles due to the false measurement injected. The green circles represent the topologically different sampled and then shortened paths, the yellow lines are the trajectories obtained after path-guided optimisation, and the blue line is the final selected trajectory. As the reader can notice, the blu line is always overlapped to the red one in the beginning end at the end, while is overlapped to the yellow one inside the replanned segment. Figures 18, 19, and 20 show the behavior of velocity, acceleration, and jerk before and after replanning. In the images, the red lines are the quantities along the x-axis, the green ones are the same quantities along the y-axis, and the z-axis is shown in blue. The purple vertical lines remark the start and goal points, where the initial colliding trajectory has been broken. Note how the continuity is preserved, among the different trajectory segments, up to the chosen trajectory order p. A full list of parameters used to carry out the aforementioned simulations is reported in Table 4.

6.3 SPATIO-TEMPORAL CURVES SEPARATION

Despite the effectiveness of the replanning algorithm described in Section 6.2, it may fail in environments where are present moving objects, as these are all treated as static obstacles. This simplifying assumption can be fatal to the replanning procedure as the final trajectory may still be unsafe during the next time instant. In this view, we proposed a novel approach to the replanning problem described in Section 6.1.2, in the setting where the safe set is not fixed in time, thus moving obstacles may cross the robot motion. The proposed solution

$v_{\max}$	$1.5m/s$	$a_{\max}$	$0.5m/s^2$
$T_{\max}$	$2.0s$	Δ_T	$0.1s^2$
$T_{\min}$	$0.5s$	$N_{\min}$	3
d_{obs}	$4m$	$\Delta_{\max}$	$0.1s$
$N_{\max}$	3000	d_{res}	$0.2m$
d_{safe}	$0.5m$	p	7
λ_1	1.0	λ_2	10.0

Table 4: Parameters used to test the replanning algorithm.

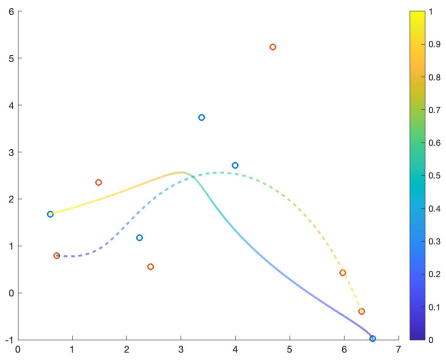


Figure 22: Example of two colliding Bézier curves. In the figure, the blue circles represent the randomly sampled control points of the continuous curve, while the red ones are the randomly sampled control points of the dotted one. The two curves are obtained as the composition of two Bézier curves of order 5 for the position and 3 for the timing law. The color shadows represent the time behavior of the two curves, normalized inside the interval $[0, 1]$.

is grounded on the assumption to have a priori knowledge of the surrounding environment, as well as an estimation of the obstacle to which the agent may collides. We stress the fact that, even if we especially focus on the particular case of one single moving obstacle, the proposed approach can easily extended to the multi-objects or multi-agent ones, where some of them are static.

Motivated by the success of GTO approaches [52, 112, 147, 170], and inspired by the flexibility of Bézier curves in trajectory planning [54, 98, 118], we propose a novel optimisation-based replanning paradigm where the Bézier parameterisation is employed twice in expressing the path and the associated timing law. The final trajectory recalls the same structure chosen in Section 4.2.1, where the piecewise structure allows for splitting the planning problem in several segments, and considering one environment area, or obstacle collision, at a time. In the next sections we firstly remark how the double parameterisation is the key to fast plan optimal avoiding trajectories and state the final trajectory equations (Section 6.3.1), then we formulate the proposed solution as a single step optimisation problem (Section 6.3.2).

6.3.1 *Spatio-Temporal Parameterisation & Composition*

The key idea behind the proposed solution is to parametrise both path and timing law by means of two Bézier curves. In order to understand how this choice may helps during the planning stage, let

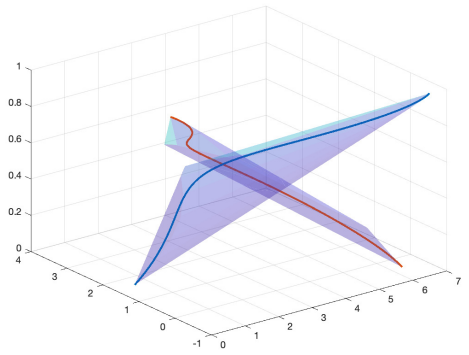


Figure 23: Convex hull representation of Figure 22. The third dimension is represented by the time, normalized inside the interval $[0, 1]$.

us consider the example reported in Figure 22. In this 2-dimensional case, two 5-order Bézier curves

$$\mathbf{r} = \sum_{i=0}^{p} \boldsymbol{q}_i B_i^p(u),$$
$$\mathbf{r}^o = \sum_{i=0}^{p} \boldsymbol{q}_i^o B_i^p(u^o),$$

with $p = 5$ and $\boldsymbol{q}_i$, $\boldsymbol{q}_i^o$ randomly sampled in $\mathbb{R}^2$. The timing laws fixing u and u^o are chosen as other two Bézier curves ensuring the contidition $u \in [0, 1]$ for any $t \in \mathbb{R}_{\geq 0}$

$$u = \sum_{i=0}^{p_u} u_i B_i^{p_u}(t),$$
$$u^o = \sum_{i=0}^{p_u} u_i^o B_i^{p_u}(t).$$

In the aforementioned relation, $p_u = 3$, while $u_0 = u_0^o = 0$ and $u_{p_u} = u_{p_u}^o = 1$ to constraining $u, u^o \in [0, 1]$. The remaining free points have been selected randomly in $\mathbb{R}$. In this settings, $\mathbf{r}$ is thought of as the robot trajectory, while $\mathbf{r}^o$ as the obstacle one. Figure 22 depicts the curves behavior on the $\mathbb{R}^2$ plane and along time thanks the reported colormap, in particular $\mathbf{r}$ is depicted as a continuous line, with the blue circles as associated control points, while $\mathbf{r}^o$ is represented as a dotted line, with the orange circles as control points. The reader can immediately recognize a possible collision around 0.5/0.6 seconds, where the two lines cross each other. The replanning problem may be solved just by moving the path control points $\boldsymbol{q}_i$, leading to a very huge detour from the initial planned trajectory, or by slowing down the agent until its motion is safe from possible collisions. In this second case, instead of moving the position control points we require to change the timing law, making the agent trajectory slower and with

a higher execution time. Although very appealing, the latter solution is not feasible as long as the relation between the timing law and the induced agent-obstacle distance is not clear. In this framework, the Bézier curve properties help us in formalizing such a relation exploiting first the composition rule, then the difference rule, and finally the product one. Although the reader can find a complete analysis in Section A, we recall here these properties applied to the considered specific case. Let us suppose we want formalize the agent-obstacle squared distance $d(t)$ as a Bézier curve, whose control points d_i must be function of the initial quantities $\boldsymbol{q}_i$, $\boldsymbol{q}_i^o$, u_i, and u_i^o. Applying the composition rule to $\mathbf{r}$ and u, and $\mathbf{r}^o$ and u^o leads to

$$\begin{aligned} \boldsymbol{q}_{u_i} &= \lambda_{0,j}^p \quad j = 0, \dots, pp_u, \\ \boldsymbol{q}_{u_i}^o &= \rho_{0,j}^p \quad j = 0, \dots, pp_u, \end{aligned} \tag{29}$$

where

$$\lambda_{i,j}^k = \binom{kp_u}{j}^{-1} \sum_{l=\max(0,j-p_u)}^{\min(j,kp_u-p_u)} \binom{kp_u - m_u}{l} \binom{p_u}{j-l} \left[(1-u_{j-l})\, \lambda_{i,l}^{k-1} + u_{j-l} \lambda_{i+1,l}^{k-1} \right], \tag{30}$$

$$\rho_{i,j}^k = \binom{kp_u}{j}^{-1} \sum_{l=\max(0,j-p_u)}^{\min(j,kp_u-p_u)} \binom{kp_u - m_u}{l} \binom{p_u}{j-l} \left[\left(1-u_{j-l}^o\right) \rho_{i,l}^{k-1} + u_{j-l}^o \rho_{i+1,l}^{k-1} \right], \tag{31}$$

for $k = 1, \dots, p$, $i = 0, \dots, p-k$, and $j = 0, \dots, kp_u$ and setting $\lambda_{i,0}^0 = \boldsymbol{q}_i$. The obtained control points $\boldsymbol{q}_{u_i}$ and $\boldsymbol{q}_{u_i}^o$ represent two new curves of order pp_u, encoding both path and timing law, then the agent-obstacle distance, along the two axis, can be evaluated as

$$\boldsymbol{q}_{\Delta_i} = \boldsymbol{q}_{u_i} - \boldsymbol{q}_{u_i}^o \quad \forall i = 0, \dots, pp_u. \tag{32}$$

The final step consists in converting the axis-wise distance, to a squared Euclidean one. To do this, $\boldsymbol{q}_{\Delta_i} = \left[\boldsymbol{q}_{\Delta_i}^x, \boldsymbol{q}_{\Delta_i}^y \right]$ must be first decomposed alog the two axis, then each component must be squared up as

$$\left(\boldsymbol{q}_{\Delta_i}^k \right)^2 = \sum_{j=\max(0,i-pp_u 2)}^{\min(pp_u,i)} \frac{\binom{pp_u}{j}\binom{pp_u}{i-j}}{\binom{2pp_u}{i}} \boldsymbol{q}_{\Delta_j}^k \boldsymbol{q}_{\Delta_{i-j}}^k, \tag{33}$$

with $k = x, y$ and $i = 0, \dots, 2pp_u$. Finally, the squared Euclidean distance can be evaluated as a Bézier curve $d(t)$ of order $2pp_u$ with control points

$$d_i = \left(\boldsymbol{q}_{\Delta_i}^x \right)^2 + \left(\boldsymbol{q}_{\Delta_i}^y \right)^2. \tag{34}$$

Remark 6.3.1. *Equations* (29)-(34) *look complicated, but at the end, the relation between* d_i *and* $\boldsymbol{q}_i$, u_i, $\boldsymbol{q}_i^o$, *and* u_i^o *results to be a simple algebraic relation. Let define the map* $\Gamma : \mathbb{R}^{2(p+p_u)} \mapsto \mathbb{R}^{2pp_u}$ *as the function mapping the initial path and time control points to the final squared Euclidean distance.*

Remark 6.3.2. *Equations* (29)-(34) *have been developed for a planar case, but the extension to the three-dimension case is straightforward, considering* $\boldsymbol{q}, \boldsymbol{q}^o \in \mathbb{R}^3$ *and* $k = x, y, z$.

The key idea behind the proposed approach lies in the possibility to separate the agent curve and the obstacle one both in space, and time, letting the algorithm to select the optimal trade-off between moving the position control points, or the timing law ones. The adopted Bézier parameterisation is crucial in this view since it allows to continuously keep separated the two curves, and straightforwardly join them when is required. A graphical representation of this property is depicted in Figure 23, where is reported a convex hull representation of the colliding curves shown in Figure 22. In Figure 23 the two-dimensional curves are depicted in $\mathbb{R}^3$, with the time being the third axis. This three-dimensional representation has been obtained enlarging the set of composed control points from Equation (29), with u_i and u_i^o. The resulting trajectories are still Bézier curves, thus the convex hull containment property still holds, and the obtained polyhedra can be used to verify collisions both in time and space.

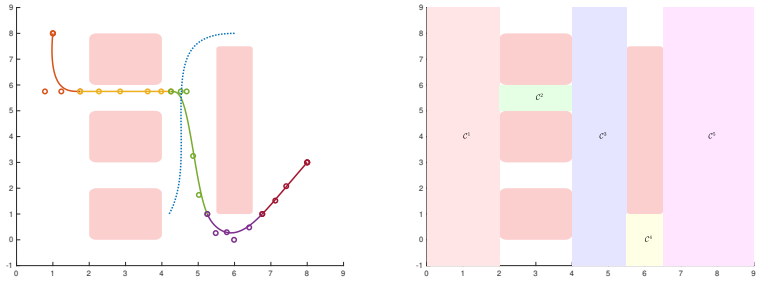
We conclude this section by recalling the piecewise Bézier structure defined in Section 4.2.1, used here to represent the final quadcopter trajectory.

$$\mathbf{r}(t) = \begin{cases} \sum_{i=0}^{p} B_i^p(u_1(\zeta_1))\boldsymbol{q}_i^1 & t \in [T_0, T_1], \\ \sum_{i=0}^{p} B_i^p(u_2(\zeta_1))\boldsymbol{q}_i^2 & t \in [T_1, T_2], \\ \vdots & \vdots \\ \sum_{i=0}^{p} B_i^p(u_l(\zeta_l))\boldsymbol{q}_i^l & t \in [T_{l-1}, T_l], \end{cases} \tag{35}$$

with $\zeta_i = \frac{t-T_{i-1}}{T_i-T_{i-1}}$ and $u_j(\zeta_j) = \sum_{i=0}^{p_u} B_i^{p_u}(\zeta_j)u_i^j$. From now on we suppose $p = 5$ and $p_u = 3$.

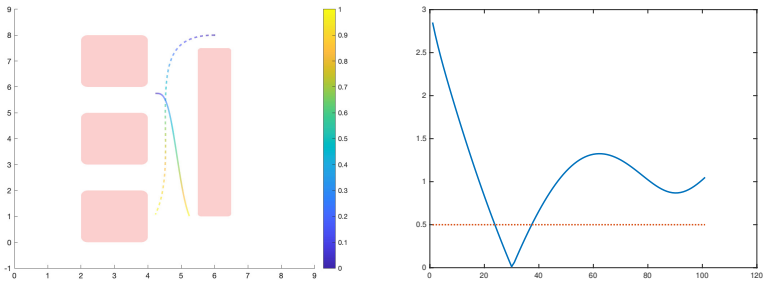
6.3.2 *Spatio-Temporal Separation*

With the previous analysis at hand, we propose here an optimisation-based approach to the replanning problem described in Section 6.1.2, in the specific case where the obstacle trajectory $\mathbf{r}_o(t)$ is known and parameterised as a Bézier curve with control points q_o^i for the path, and u_o^i for the timing law. In particular, let $\mathbf{r}^j(t)$ the segment of Equa-



(a) (b)

Figure 24: Experimental setting used to test the proposed approach. Figure (a) depicts the initial trajectory planning made using the corridords depicted in image (b). The colored lines represent the initial agent trajectory, along with the respective selected control points, while the dotted blue line is the moving obstacle path, chosen in order to get in collision with the third green piece. The red rectangles are environment obstacles assumed to be completely known.



(a) (b)

Figure 25: Initial setting extrapolated from Figure 24. In the left image, the two colliding pieces of trajectory are depicted with their behavior in time, normalized inside the interval $[0, 1]$, while in the right image the blue line represents the agent-obstacle distance in time. The red dotted line, in the right image, is the chosen threshold for the minimum safe distance.

tion (35) in (possible) collision with the obstacle trajectory $\mathbf{r}_o(t)$, then the problem of trajectory replanning can be solved as

$$\min_{\substack{\boldsymbol{q}_0,\ldots,\boldsymbol{q}_p;\\ u_0,\ldots,u_{p_u}}} \mathcal{J}\left(\boldsymbol{q}_0,\ldots,\boldsymbol{q}_p,u_0,\ldots,u_{p_u},\boldsymbol{q}_0^j,\ldots,\boldsymbol{q}_p^j,u_0^j,\ldots,u_{p_u}^j\right)$$

$$\text{subj. to} \quad \begin{array}{l}\text{Continuity constraint,}\\ \text{Dynamical constraint,}\\ \text{Collision constraint,}\end{array} \tag{36}$$

with the constraints set depending on $\boldsymbol{q}_0^o, \ldots, \boldsymbol{q}_p^o, u_0^o, \ldots, u_{p_u}^o$. The cost function $\mathcal{J}(\cdot)$ can be chosen so as described in Section 6.1.2, recalling the necessity to keep the replanned trajectory as close as possible to the original one

$$\mathcal{J}(\cdot) = \sum_{i=0}^{p} \left\| \boldsymbol{q}_i - \boldsymbol{q}_i^j \right\| + \sum_{i=0}^{p_u} \left| u_i - u_i^j \right|, \tag{37}$$

or so as described in Section 6.2.3, including some trajectory regularisation terms. In the following we briefly details and formulate the constraints in (36).

Continuity constraint.
Belongs to this family of constraints all those conditions required to continuously connect the replanned trajectory segment with $\mathbf{r}^{j-1}$ and $\mathbf{r}^{j+1}$. In this context, we enforce continuity up to the third derivate, since higher order differentiations may complex the problem unnecessarily. Position continuity is easily obtained via control points matching as

$$\begin{aligned} u_0 &= 0, \\ u_{p_u} &= 1, \\ \mathbf{r}_0 &= \mathbf{r}_0^j, \\ \mathbf{r}_p &= \mathbf{r}_p^j, \end{aligned}$$

where points denoted with apex ^{j} are the old control points. As regards velocity and acceleration continuity, we differentiate $\mathbf{r}^j(t)$, with an eye to the composed stucture.

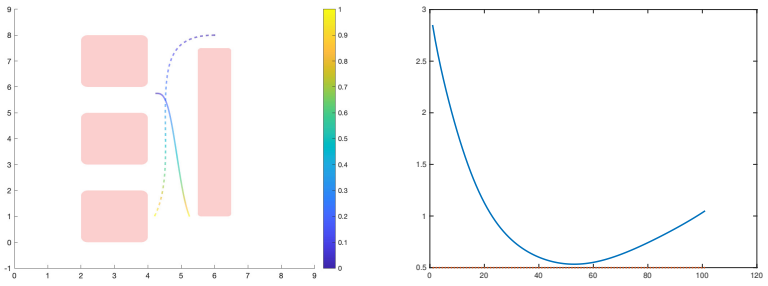
$$\begin{aligned} (\mathbf{r}_1 - \mathbf{r}_0)(u_1 - u_0) &= \left(\mathbf{r}_1^j - \mathbf{r}_0^j\right)\left(u_1^j - u_0^j\right), \\ \left(\mathbf{r}_p - \mathbf{r}_{p-1}\right)\left(u_{p_u} - u_{p_u-1}\right) &= \left(\mathbf{r}_p^j - \mathbf{r}_{p-1}^j\right)\left(u_{p_u}^j - u_{p_u-1}^j\right). \end{aligned}$$

$$\begin{aligned} &(\mathbf{r}_1 - \mathbf{r}_0)(u_2 - 2u_1 + u_0) + (\mathbf{r}_2 - 2\mathbf{r}_1 + \mathbf{r}_0)(u_1 - u_0)^2 = \\ &\left(\mathbf{r}_1^j - \mathbf{r}_0^j\right)\left(u_2^j - 2u_1^j + u_0^j\right) + \left(\mathbf{r}_2^j - 2\mathbf{r}_1^j + \mathbf{r}_0^j\right)\left(u_1^j - u_0^j\right)^2, \\ &\left(\mathbf{r}_p - \mathbf{r}_{p-1}\right)\left(u_{p_u} - 2u_{p_u-1} + u_{p_u-2}\right) + \left(\mathbf{r}_p - 2\mathbf{r}_{p-1} + \mathbf{r}_{p-2}\right)\left(u_{p_u} - u_{p_u-1}\right)^2 = \\ &\left(\mathbf{r}_p^j - \mathbf{r}_{p-1}^j\right)\left(u_{p_u}^j - 2u_{p_u-1}^j + u_{p_u-2}^j\right) + \left(\mathbf{r}_p^j - 2\mathbf{r}_{p-1}^j + \mathbf{r}_{p-2}^j\right)\left(u_{p_u}^j - u_{p_u-1}^j\right)^2. \end{aligned}$$

Dynamical constraint.
In the specific use case of quadcopter trajectory planning, differential flatness (Section 2.2) can be used to collapse the dynamics constraint to velocity and acceleration bounds.

$$\begin{aligned} pp_u \left(\mathbf{r}_{u_{i+1}} - \mathbf{r}_{u_i}\right) / \left(T_{j+1} - T_j\right) &\leq v_{\max} \quad \forall i = 0, \ldots, pp_u - 1, \\ pp_u \left(pp_u - 1\right)\left(\mathbf{r}_{u_{i+2}} - 2\mathbf{r}_{u_{i+1}} + \mathbf{r}_{u_i}\right) / \left(T_{j+1} - T_j\right)^2 &\leq a_{\max} \quad \forall i = 0, \ldots, pp_u - 2, \end{aligned}$$

with $\boldsymbol{q}_{u_i}$ be the ith control point of the composed curve (29). Here we are exploiting the properties of closure with respect



(a) (b)

Figure 26: Results of the proposed solution when used to optimise u_i only. In the left image is depicted the same path as Figure 25, but with the new time behavior, normalized inside the interval $[0, 1]$. The blue continuous line, in the right image, represents the new agent-obstacle distance, and the red dotted line is the fixed minimum safe distance.

to composition and derivation, along with the convex hull containment one. In this context, as well as in the next collision constraint, the Bézier parameterisation plays a crucial role in simplifying the constraint formulation and dimensionality, leading to a manageable problem.

Collision constraint.
Finally, collision constraints are formulated as n_c flght safe corridors for static obstacles avoidance, and as spatio-temporal distance for dynamic obstacle avoidance

$$\begin{aligned} d_i &\geq d_{\text{safe}} \quad \forall i = 0, \dots, 2pp_u, \\ A^k \mathbf{r}_i &\leq b^k \quad \forall i = 0, \dots, p, \quad \forall k = 0, \dots, n_c. \end{aligned}$$

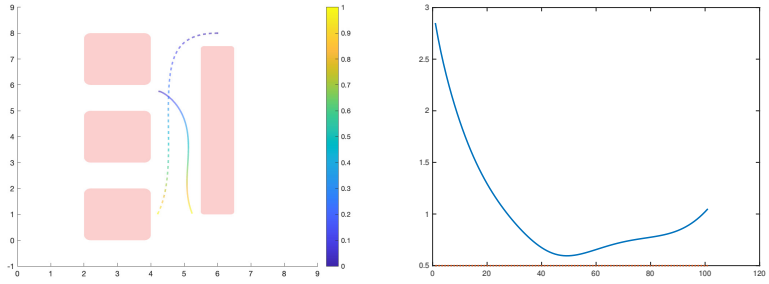
In the aforementioned relation d_i are the $2pp_u$ control points defining the squared agent-obstacle distance, defined in Equation (34), while A^k and b^k are n_c convex polyhedra defining the safe regions free of static obstacles. In this setting,

$$\mathcal{X}' = \bigcup_{k=0}^{n_c} \left\{ x \in \mathbb{R}^{n_x} : A^k x \leq b^k \right\}.$$

Remark 6.3.3. *The optimisation problem (36) has been formulated for one agent only, colliding with a moving obstacle, the extension to the multi-agents case can be straightforwardly obtained by adding a new set of optimisation variables, describing the new agent trajectory.*

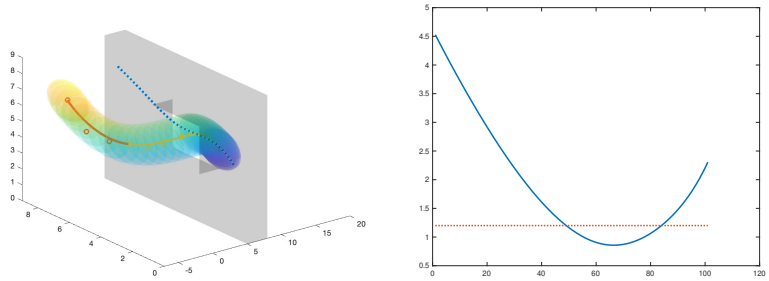
6.3.3 *Experimental Results*

The proposed approach has been successfully applied in two different meaningful scenarios representing a two-dimensional framework



(a) (b)

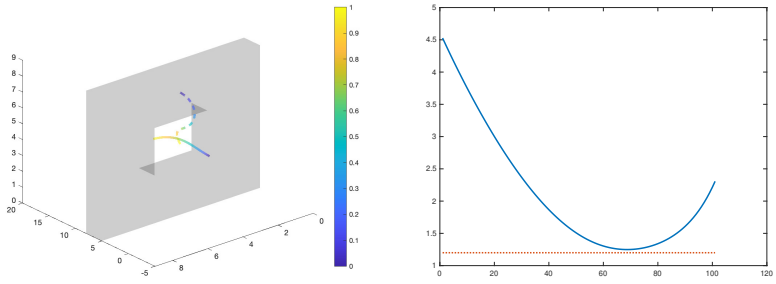
Figure 27: Results of the proposed solution when used to optimise both q_i and u_i. In the left image is depicted the new path with the new time behavior, normalized inside the interval $[0, 1]$. The blue continuous line, in the right image, represents the new agent-obstacle distance, and the red dotted line is the fixed minimum safe distance.



(a) (b)

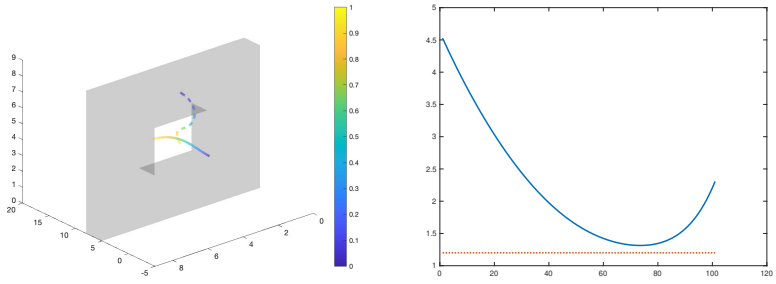
Figure 28: Initial three-dimensional testing scenario. In the left image, the two colliding pieces of trajectory are depicted with the considered collision sphere. In the right image, the blue line represents the agent-obstacle distance in time, and the red dotted line is the chosen threshold for the minimum safe distance.

(Figure 25), and a three-dimensional one (Figure 28). In both cases, the optimisation problem (36) has been solved for u_i only and for both q_i and u_i. The obtained results are reported in Figures 26 and 27 for the two-dimensional case, and in Figures 29 and 30 for the three-dimensional one. In all figures the (a) image reports the resulting trajectory along with its behavior in time, while image (b) depicts the agent-obstacle distance (blue line) with the fixed minimum safe distance (red dotted line). In all simulations, we choose the loss function as in Equation (37), $d_{\text{safe}} = 0.25$, $v_{\max} = 1.5m/s^2$, and $a_{\max} = 0.5m/s^2$. The proposed solution succeeds, in each proposed scenario, to solve the trajectory replanning problem. In the complete optimisation case, in Figures 27 and 30, the algorithm is left free to trade-off between



(a) (b)

Figure 29: Results of the proposed solution when used to optimise u_i only. In the left image is depicted the same path as Figure 28, but with the new time behavior, normalized inside the interval $[0, 1]$. The blue continuous line, in the right image, represents the new agent-obstacle distance, and the red dotted line is the fixed minimum safe distance.



(a) (b)

Figure 30: Results of the proposed solution when used to optimise both $\boldsymbol{q}_i$ and u_i. In the left image is depicted the new path with the new time behavior, normalized inside the interval $[0, 1]$. The blue continuous line, in the right image, represents the new agent-obstacle distance, and the red dotted line is the fixed minimum safe distance.

$\left\|\boldsymbol{q}_i - \boldsymbol{q}_i^j\right\|$ and $\left|u_i - u_i^j\right|$ leading to a local optima with an higher final minimum distance.

6.4 DATA-DRIVEN CONTROL BARRIER FUNCTIONS

In this section, we detail a novel control-oriented approach to the replanning problem described in Section 6.1.2, focusing on the particular simplified case where only static obstacles may appear in a partially known environment. The considered framework is well represented by Equation (26) with $\mathcal{X}$ representing free and navigable space, $\mathcal{U}$ the set of all possible inputs, and f the considered system

dynamics. In this settings, the autonomous robot is pursuing an initial motion planned with only partial information about the surrounding environment and concurrently is gathering information about it. The initial motion is then continuously refined with the collected new information in order to ensure the robot safety at each time instant. Motivated by this safety requirement, we ground the proposed solution on the notion of Control Barrier Function (CBF) [9] which has been successfully demonstrated on many safety-critical applications [9, 69, 154]. CBF applications ensure safety by enforcing the forward invariance of a *safe set* described as a superlevel set of a candidate function h which maps the system states to this notion of safety. In particular, the candidate function defines the hyper space where the system is considered in a safe condition. Traditionally, these safety functions have been hand-designed. However, depending on the application, designing such a candidate CBF is not straightforward in many practical settings. Consider, as an example, the obstacle avoidance scenario, where the safe space is simply the free space, design the map h in this context requires the complete environment knowledge, which is usually not completely a priori known. Due to the aforementioned problem, we enforce the run-time adaptation of h via Gaussian process inference exploiting real-time collected data, allowing for a constantly reshaping of the current motion with newly upcoming safe information. The idea of learning the safe set from data is not new, in this sense Support Vector Machines (SVMs) were used in [134] to parameterize CBFs with the help of sensor measurements. Neural networks were successfully used to regress the safety barrier function in [7, 51, 145, 169], while a second-order cone program was formulated in [21], with GPs used for modeling the control input and the system uncertainties. However, all these solutions are limited to offline training which limits their applicability in many practical scenarios. Gaussian processes have been successfully used in [77] and [78] where the candidate function h is directly learned out of the collected data. The latter works, completely detach from a possible parameterised CBF and exploit the GP regression in a completely data-driven scenario, without any a priori knowledge about the safe set. However the quantity of collected data, required to build run-time the safe set, increases a lot with the system relative degree, leading to poor, or even completely wrong, results. Here we focused on the latter problem by formalising a new CBF-based solution that implements a Gaussian process regression requiring a limited number of samples whatever the system relative degree is. Then, the developed safety tool is successfully implemented in the obstacle avoidance scenario.

We stress the fact that, unlike the approaches detailed in Section 6.2 and Section 6.3, the proposed solution does not require neither a resource consuming map building and maintenance, nor heavy random sampling procedure which may lead to very large set of samples,

or possible trajectories, hard to handle. The remain of this section unfolds as follow, in Section 6.4.1 we briefly review key results on CBFs, Section 6.4.2 describes a possible method to enforce the safe set learning, while in Section 6.4.3 and Section 6.4.4 we formulate the proposed regulator and show the obtained results.

6.4.1 *Control Barrier Functions*

Consider the nonlinear control affine system

$$\dot{x} = f(x) + g(x)\,u, \quad y = h(x) \tag{38}$$

where $x \in \mathbb{R}^{n_x}$ is the system state, $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ is the input, and $y \in \mathbb{R}^{n_y}$ the measurable output. Moreover let $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$ $g : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x \times n_u}$ and $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ Lipschitz continuous maps. Let the safety of (38) be encoded as the superlevel set $\mathcal{S}$ of the smooth function h

$$\mathcal{S} = \mathcal{X}' = \{x \in \mathbb{R}^{n_x} : h(x) \geq 0\},$$

in this context we recall a couple of results from [9, 109, 161].

Definition 6.4.1. *The map $h(x) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ is defined as a Control Barrier Function (CBF), if there exists a class $\mathcal{K}$ function α, i.e. with the features (a) $\alpha(0) = 0$ and (b) α stricly increasing, such that the relation*

$$\sup_{u \in \mathcal{U}} L_f h(x) + L_g h(x)\,u + \alpha(h(x)) \geq 0,$$

holds for any $x \in \mathcal{S}$.

In the previous definition the quantities $L_f h(x)$ and $L_g h(x)$ are the directional derivatives of h, along the flow defined by f and g, respectively (along the literature called *Lie derivatives*). With the aforementioned definition at hand, we can easily enforce the system safety, which in this context implies no collisions, by shaping the control input u of (38) over the designed CBF in Definition 6.4.1.

Theorem 6.4.1. *Given a nonlinear system as in Equation (38), with a defined safe set $\mathcal{S} \subset \mathbb{R}^{n_x}$, and a smooth control barrier function $h(x) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$, any Lipschitz continuous control law $u(t) \in \mathbb{R}^{n_u}$ satisfying*

$$L_f h(x(t)) + L_g h(x(t))\,u(t) + \alpha(h(x(t))) \geq 0, \tag{39}$$

for any $x \in \mathbb{R}^{n_x}$, makes the safe set $\mathcal{S}$ forward invariant for (38).

The reader can immediately conceive from Theorem 6.4.1 and Definition 6.4.1 the very limiting requirement to have relative degree (here referred to as p) equal to one, $p = 1$. For systems with $p > 1$, we require an extension to the CBF notion defined so far. In particular, referring to [109, 161], let's introduce the concept of Exponential Control Barrier Function (ECBF).

Definition 6.4.2. *The smooth function $h(x) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$, with relative degree p, is defined as an exponential control barrier function, if there exists a set of coefficients $\Lambda \in \mathbb{R}^p$ such that for any $x \in \mathbb{R}^{n_x}$*

$$\sup_{u \in \mathcal{U}} L_f^p h(x) + L_g L_f^{p-1} h(x)\, u + \Lambda^\top \mathcal{H} \geq 0 \tag{40}$$

holds with $\mathcal{H} = \left[h(x), L_f h(x), \ldots, L_f^{p-1} h(x)\right]^\top \in \mathbb{R}^p$ the Lie derivative vector for h, and $\Lambda = \left[\lambda_0, \lambda_1, \ldots, \lambda_{p-1}\right]^\top$ is a coefficient gain vector for $\mathcal{H}$ which can be conputed via standard linear control techniques.

The functions (39) or (40) can then be combined with Quadratic Programs (QPs) to achieve safety constrained control, in particular, for high relative degree systems, the forward invariance of the safe set $\mathcal{S} = \{x \in \mathbb{R}^{n_x} : h(x) \geq 0\}$ can be enforced solving the following optimisation problem

$$\begin{aligned} u = \arg\min_{v \in \mathcal{U}} & \frac{1}{2} \|v - u^\star\| \\ \text{subj. to } & L_f^p h(x) + L_g L_f^{p-1} h(x)\, v + \Lambda^\top \mathcal{H} \geq 0 \end{aligned} \tag{41}$$

with $u^\star \in \mathbb{R}^{n_u}$ previous planned control input, referred to as *nominal input*.

Remark 6.4.1. *The problem (41) resembles the replanning problem (26) borrowing the same loss and with the safe and dynamic constraints condensed in Equation (40).*

6.4.2 *Learning the Safe Set*

The major issue in using CBF to enforce safety lies in designing the candidate function h, whose superlevel set must encode all safe regions where the system state is allowed to evolve. This problem is mainly present also in obstacle avoidance scenarios where the environment in not a priori known making impossible the design of h. To overcome this limitation, [78] proposes to handle the construction of h following a complete data-driven approach: the key idea was to model this unknown map as a realisation of a Gaussian process. In particular, supposing to have access to a data-set $(\boldsymbol{x}, \boldsymbol{y}) = \{(x(t_1), y(t_1)), \ldots, (x(t_N), y(t_N))\}$ of N samples, with each pair $(x(t_h), y(t_h)) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ obtained as $y(t_h) = h(x(t_h)) + \varepsilon(t_h)$ with $\varepsilon(t_h) \sim \mathcal{N}(0, \sigma_n^2 I_{n_y})$ be a white Gaussian noise, then an approximation of h can be computed as

$$\begin{aligned} \mu(x) &= \boldsymbol{\kappa}(x)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{y}, \\ \sigma^2(x) &= \kappa(x, x) - \boldsymbol{\kappa}(x)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{\kappa}(x), \end{aligned} \tag{42}$$

where $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{N \times N}$ is the *Gram matrix* whose (k, h)th entry is $\boldsymbol{\mathcal{K}}_{k,h} = \kappa(\boldsymbol{x}_k, \boldsymbol{x}_h)$, with $\boldsymbol{x}_k$ the kth entry of $\boldsymbol{x}$, $\boldsymbol{\kappa}(x) \in \mathbb{R}^N$ is the kernel vector

whose kth component is $\boldsymbol{\kappa}_k(x) = \kappa(x, \boldsymbol{x})$, and $\kappa(x, x')$ is a custom kernel function [127]. For further information about Gaussian regression and about the kernel structure the reader is referred to Appendix **??**. Gaussian regression works by iteratively restricting the pool of possible functions, conditioning a prior Gaussian distribution on the collected dataset. In this sense, Equation (42) expresses the a posteriori mean (μ) and variance (σ^2) computed after the collection of N samples $(\boldsymbol{x}, \boldsymbol{y})$. In this context, the computed mean represents the "best" function approximation, while σ^2 quantifies the amount of uncertainties affecting the current estimation, as it strongly depends on the amount of data collected near the evaluating point x. Although the estimation μ of h can be directly used inside the optimisation problem (41) in place of h, we should be careful of possible errors and uncertainties in the reconstructed approximation as a wrong value of μ may make condition (40) holds even in unsafe conditions. As a matter of fact, evaluating an unknown function out of data may lead to wrong results, especially if the collected data are poorly informative or very scattered. To overtake this particular problem, [78] proposed a re-reading of the candidate function keeping into consideration also the provided a posteriori variance of the current estimation μ, thus h has been reformulated as

$$h_{\mathrm{GP}}(x) = \mu(x) - \rho\sigma^2(x),$$

with $\rho \in \mathbb{R}^+$ a tunable value, leading to the new optimisation problem

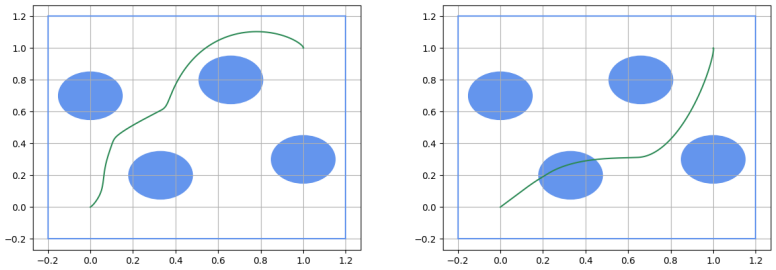
$$\begin{aligned} u = \arg\min_{v \in \mathcal{U}} \; & \frac{1}{2}\|v - u^\star\| \\ \text{s.t.} \; & L_f^p h_{\mathrm{GP}}(x) + L_g L_f^{p-1} h_{\mathrm{GP}}(x)\, v + \Lambda^\top \mathcal{H} \geq 0. \end{aligned}$$

The aforementioned approach is very promising under the perspective of adapt CBF tools to a very large number of use cases, especially when an analytical formulation of h is not accessible, but results to be very poor practically when dealing with systems with high relative degrees $p > 1$ because of loss of information during Gaussian process differentiation [65]. Furthermore, the parameter ρ results to be the key to obtain good performances even with $p = 1$, thus a very careful tuning is required to make the algorithm works correctly. The aforementioned problems motivated our research in the field of Gaussian control barrier functions toward more general and resilient solutions, especially tailored for high relative degree systems.

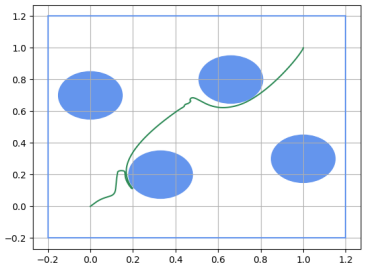
6.4.3 *Proposed Gaussian Control Barrier Function*

The proposed regulator reads as

$$\dot{z} = Az + B\left(L_f\mu(x) + L_g\mu(x)\, u\right) + G(l)\, H(y - Cz), \tag{43}$$



(a) (b)



(c)

Figure 31: Comparison between (a) classic ECBF, (b) GP-CBF proposed by [78], and (c) GP-CBF proposed here. In image (a) the safe set is designed using the complete knowledge of the environment, while in images (b) and (c) the candidate function is reconstructed run-time using the collected sensor data.

with output

$$u = \arg\min_{v \in \mathcal{U}} \frac{1}{2} \|v - u^\star\| + L_g \sigma^2(x)\, v \quad \text{subj. to } L_f \mu(x) + L_g \mu(x)\, v + \Lambda^\top \mathcal{H} \geq 0. \tag{44}$$

As emerges from Equation (43), the proposed regulator consists in a standard high-gain observer with A, B, and C matrices in prime form

$$A = \begin{pmatrix} 0_{(p-1)n_y \times n_y} & I_{(p-1)n_y} \\ 0_{n_y \times n_y} & 0_{n_y \times (p-1)n_y} \end{pmatrix}, \quad B = \begin{pmatrix} 0_{(p-1)n_y \times n_y} \\ I_{n_y} \end{pmatrix},$$

$$C = \begin{pmatrix} I_{n_y} & 0_{n_y \times (p-1)n_y} \end{pmatrix},$$

and $G(l) = \text{diag}\left(l I_{n_y}, l^2 I_{n_y}, \ldots, l^r I_{n_y}\right)$, $H = \text{diag}\left(H_1, \ldots, H_r\right)$, and $H_i = \text{diag}\left(h_i^1, \ldots \ldots, h_i^{n_y}\right)$ with $\{h_1^j, h_2^j, \ldots, h_{r+1}^j\}$ for all $j = 1, \ldots, n_y$ coefficients of a Hurwitz polinomial, where $l \in \mathbb{R}_{>0}$ is a control parameter. In this context, the proposed observer is meant to directly reconstruct $L_f^{p-1} h$, that in turn is feeded back to a Gaussian regressor which outputs an analytical and differentiable estimate

$$\mu(x) = \boldsymbol{\kappa}(x)^\top \left[\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right]^{-1} \boldsymbol{z},$$

with z extracted from the recostructed data-set

$$(\boldsymbol{x}, \boldsymbol{z}) = \{(x(t_1), z_{p-1}(t_1)), \ldots, (x(t_N), z_{p-1}(t_N))\}.$$

The regressed directional derivate is then used to reconstruct the observer consistency term [17]. Equation (44) reformulates the one-relative-degree problem (41) in this particular framework. Note that the used candidate function does not take into account any uncertainties in the retrieved estimation, which in turn are encode inside the loss function as $L_g\sigma^2(x)\,v$, with σ^2 the a posteriori Gaussian variance

$$\sigma^2(x) = \kappa(x,x) - \boldsymbol{\kappa}(x)^\top \left[\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right]^{-1} \boldsymbol{\kappa}(x).$$

Remark 6.4.2. *Although the new loss proposed in Equation (44) presents a slightly different structure with respect to classical CBF formulation (41) and with respect to the formulated problem in Section 6.1.2, it still succeed in solving the obstacle avoidance problem as the collision is encoded as optimisation hard constraint. Moreover, minimise $L_g\sigma^2(x)\,v$ directly implies a minimisation of the induced uncertainties affecting μ as the agent is forced to navigate toward already visited frontiers. As a matter of fact, the aforementioned term comes out from Equation (40) cosidering $h = \mu + \sigma^2$*

$$\sup_{u\in\mathcal{U}} L_f h(x) + L_g h(x)\,v + \Lambda^\top \mathcal{H} \geq 0,$$

$$L_f\left(\mu(x) + \sigma^2(x)\right) + L_g\left(\mu(x) + \sigma^2(x)\right) v.$$

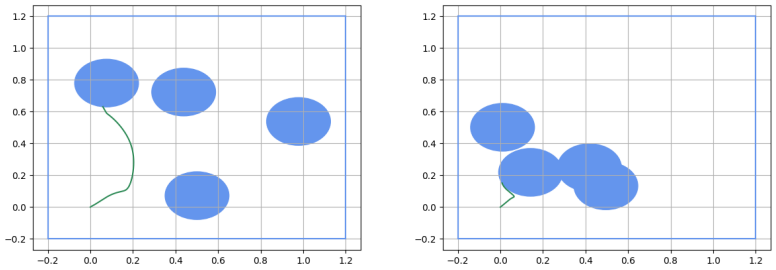
We stress the fact that encoding an uncertainties term inside the cost formulation drops the necessity to introduce a weight ρ inside the candidate function, as done in [78], whose wrong tuning may degrade the algorithm performance.

Remark 6.4.3. *The proposed solution (43)-(44) is well suited for high relative degree applications as the regressed map μ is differentiated only once, leading to a very few loss of information. In the context of Gaussian process, the latter property implies the possibility to keep the value of N low, leading to light real-time training procedures.*

We conclude this section by recalling two basic properties which the regressor should undergo to make the proposed solution works correctly. We stress the fact that the assumptions reported below are not restrictive under the practical point of view, and can be easily satisfied by all commonly used kernels. The following assumptions are borrowed from [17].

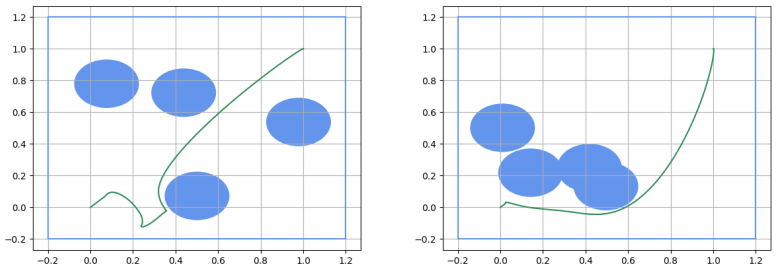
Assumption 6.4.1. *μ is Lipschitz continuous with Lipschitz constant L_μ, and its norm is bounded by μ_{max}.*

Assumption 6.4.2. *The kernel function $\kappa(\cdot,\cdot)$ is Lipschitz continuous with constant L_κ, with a locally Lipschitz derivative of constant $L_{d\kappa}$, and its norm is bounded by κ_{max}.*



(a) (b)

Figure 32: Results obtained in challenging environments when applying the proposed approach without taking into account estimation uncertainties carried by σ^2. As the reader can observe, the proposed solution does not make the agent colliding, but can stuck it close to obstacles.



(a) (b)

Figure 33: Results obtained in challenging environments when applying the overall proposed approach. The encoded uncertainties via σ^2 drive the agent in regions with low uncertainties, making the final solution working in all cases.

6.4.4 *Experimental Results*

The proposed approach has been applied to a synthetic scenario and tested against the classical ECBF, where the safe set design has been made using the full knowledge about the navigating environment, and the state-of-the-art Gaussian-based CBF proposed by [78]. In both the learning techniques, the training data are collected real time, while the agent is moving inside the environment. The testing scenario (see Figure 31) is equal in all experiments and consists of four shpere-like obstacles with random positions and four surrounding walls. The agent is modeled a simple double integrator with acceleration as control input, while the nominal control input $u^\star$ consists in a straight line from the initial to the goal position. In this settings, the candidate function h is modeled as the distance from the closest

obstacle, following the Euclidean Signed Distance Field (ESDF) principle [113]. The obtained results are reported in Figures 31, 32, and 33. In particular, Figure 31 depicts a comparison between the classical ECBF, the GP-CBF proposed by [78], and our approach when applied in the same settings described above. The classical ECBF, designed on the complete knowledge of the environment, always succeeds in safely steering the agent to the goal and its results are kept here as a baseline to follow in the case where no prior information about the obstacles is provided. As the reader can see, from Figure 31, the approach proposed by [78] sometimes fails, especially in highly cluttered environment, while our approach always wins. To highlight the fundamental role which plays the regulariser $L_g\sigma^2(x)v$ in (44), we reported here two simulations in Figures 32 and 33 where the proposed approach is applied without and with the regularisation term. In particular, in Figure 32 the term is not present, while in Figure 33 the optimisation problem is solved with the proposed complete loss function. It is worth to remark how in both cases the agent does not collide with obstacles, but in the first reported simulations the high uncertainties affecting the estimated candidate function made the agent stuck near to obstacles.

6.5 CONTRIBUTIONS

In this chapter we reviewed and discussed the problem of real-time trajectory replanning in front of new environmental information. The chapter develops firstly by reviewing a state-of-the-art solution to the problem at hand, adapted, re-implemented and tested in the specific case of the Leonardo drone contest. The selected solution performed well and succeed in replanning jerk-continuous trajectories well suited for quadcopter navigation. Then, two novel solutions have been described and analyzed, one primarily focused on the specific case of moving obstacles and multi-agent scenarios, and the second one focused on static environments. The proposed flow embraces a research direction especially focused on numeric and system theory-oriented solutions where the possibility of success can be easily assessed via analysis tools commonly used in the system theory field. As a future research direction, we will continue to follow this idea and try to encode Lyapunov theory in novel trajectory plaNning, or replanning, techniques yielding to very high reliable and robust solutions.

Part III

ADAPTIVE NONLINEAR OUTPUT REGULATION

7

DATA-DRIVEN NONLINEAR REGULATION

In this chapter, we collect the basic notions behind the concept of output regulation and data-driven output regulation for nonlinear systems. The section unfolds as follows, first we introduce, in the simplest possible terms, the problem of output regulation in the nonlinear context, with an eye to the taxonomy usually adopted in this field. Then, the next part is devoted to briefly presenting the state-of-the-art of the most consolidated approaches to the emergent field of *adaptive* nonlinear output regulation and how these techniques can be extended via learning tools, obtaining a highly robust and flexible solution, able to adapt to a very large class of systems. The same concepts, notation, and results can also be found in [13–15, 2, 3] thus we refer the reader to these works for the complete analysis of the presented results.

7.1 THE FRAMEWORK OF OUTPUT REGULATION

Consider a continuous-time nonlinear system of the form

$$\begin{aligned} \dot{x} &= f(w, x, u), \\ y &= h(w, x), \end{aligned} \tag{45}$$

with state $x \in \mathbb{R}^{n_x}$, control input $u \in \mathbb{R}^{n_u}$, measured output $y \in \mathbb{R}^{n_y}$, and with $w \in \mathbb{R}^{n_w}$ exogenous signal generated by an exosystem of the form

$$\dot{w} = s(w). \tag{46}$$

The exogenous signal can be treated, in this context, as references to be tracked or disturbances to be rejected. For this purpose, associated to Equation (45), there is a set of $n_e > 0$ regulation errors

$$e = h_e(w, x).$$

In this framework, we define the problem of ε-approximate output regulation as the problem to find an output feedback regulator of the form

$$\begin{aligned} \dot{x}_c &= \varphi(x_c, y), \\ u &= \gamma(x_c, y), \end{aligned} \tag{47}$$

possibly ε-dependent, with state $x_c \in \mathbb{R}^{n_{x_c}}$, such that:

Stability. The origin of the interconnection between Equation (45) and Equation (47), with $w = 0$, is asymptotically stable with a domain of attraction $\mathcal{X} \times \mathcal{X}_c \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_{x_c}}$ that is an open neighborhood of the origin.

Boundedness. There exists $\mathcal{W} \in \mathbb{R}^{n_w}$ such that the closed-loop system is uniformly bounded from $\mathcal{W} \times \mathcal{X} \times \mathcal{X}_c$.

Regulation. Each solution to the closed-loop system originating in $\mathcal{W} \times \mathcal{X} \times \mathcal{X}_c$ satisfies

$$\limsup_{t \to \infty} \| e(t) \| \leq \varepsilon.$$

If $\mathcal{X}$ coincides with $\mathbb{R}^{n_x}$, we say that the problem is solved *globally*, otherwise we say that the problem is solved *locally*. If given each $\mathcal{X} \subset \mathbb{R}^{n_x}$ it is possible to find a possibly $\mathcal{X}$-dependent regulator that solves the problem in $\mathcal{X}$, we say that the problem is solved *semi-globally*. If $\varepsilon = 0$, we refer to the problem as the *asymptotic* output regulation problem, while we talk about *practical* regulation problem whenever, given any $\varepsilon > 0$, there exists a regulator that solves the ε-approximate output regulation problem. An anchor point in the solution of the above problem is represented by the steady-state trajectories $(x^*(t), u^*(t))$, solution of the so-called *regulator equations*

$$\begin{aligned} \dot{w} &= s(w), \\ \dot{x}^* &= f(w, x^*, u^*), \\ 0 &= h_e(w, x^*), \end{aligned} \tag{48}$$

with x^* representing the ideal state trajectory associated with a zero regulation error and u^* the associated input (often referred to as "the friend" of x^*). Regulator structures proposed in the nonlinear context are typically composed of two units, an internal model unit, and a stabilising unit, with a neat, albeit limiting in many contexts, "role" conferred on the two at the design stage: the former is designed to generate the steady state input u^* required to keep the error at zero in steady state, while the latter is designed to steer the closed-loop trajectories of the system to x^*. What makes the design problem particularly challenging is, of course, the fact that $(x^*(t), u^*(t))$ are unknown as the initial conditions of Equation (48) are such. Moreover, the sufficient conditions under which asymptotic regulation is ensured are typically expressed by equations whose analytic solution becomes a hard task even for "simple" problems. Moreover, even if a regulator can be constructed, asymptotic regulation remains a fragile property that is lost at front of slightest plant's or exosystem's perturbation. The aforementioned problems motivate the researcher to move toward more robust solutions, introducing the concept of adaptive regulation. In the following we briefly present the two main adaptive approaches to nonlinear regulation designs that have influenced this

thesis most strongly, then we try to extend them via non-supervised learning techniques, to adapt the proposed structure on an ideally infinite class of systems. The reported regulators embed two different internal models, being a "post-processing" the first one and a "pre-processing" the second one, but both apply to multivariable systems. The proposed construction is not "friend-centric" but rather it is based on a "qualitative" information on the ideal error-zeroing steady state.

7.2 IDENTIFICATION-BASED POST-PROCESSING INTERNAL MODEL

In this section, we focus on a subclass of the general regulation problem presented in Section 7.1 by considering continuous-time nonlinear systems of the form

$$\begin{aligned} \dot{z} &= f_0(x,w) + b_0(x,w)u, \\ \dot{\chi} &= F\chi + H\zeta, \\ \dot{\zeta} &= q(x,w) + b(x,w)u, \\ e &= C\chi, \quad y = \operatorname{col}(\chi,\zeta), \end{aligned} \tag{49}$$

in which $z \in \mathbb{R}^{n_z}$, $y \in \mathbb{R}^{n_y}$, $e \in \mathbb{R}^{n_e}$, $\chi \in \mathbb{R}^{n_e}$, and, $u \in \mathbb{R}^{n_u}$ with $n_u \geq n_e$. The entire stack of states is denoted here by $x = \operatorname{col}(z,\chi,\zeta)$. Moreover, $\chi = \operatorname{col}(\chi^1,\dots,\chi^{n_e})$, with $\chi^i \in \mathbb{R}^{n_\chi^i}$, $i = 1,\dots,n_e$, and $\sum_{k=1}^{n_e} n_\chi^k = n_\chi$. The exogenous signal $w \in \mathbb{R}^{n_w}$ is generated by an exosystem of the same form of Equation (46). The matrices $F \in \mathbb{R}^{n_\chi \times n_\chi}$, $H \in \mathbb{R}^{n_\chi \times n_e}$, and $C \in \mathbb{R}^{n_e \times n_\chi}$ are defined as a block-diagonal matrices with entries

$$F_i = \begin{pmatrix} 0_{(n_\chi^i-1)\times 1} & I_{n_\chi^i-1} \\ 0 & 0_{1\times(n_\chi^i-1)} \end{pmatrix}, \quad H_i = \begin{pmatrix} 0_{(n_\chi^i-1)\times 1} \\ 1 \end{pmatrix},$$

$$C_i = \begin{pmatrix} 1 & 0_{1\times(n_\chi^i-1)} \end{pmatrix}.$$

Equation (49) frames the problem of output regulation on a particular class of systems that embraces a large number of use-cases addressed in literature. In particular, note that all systems presenting *(a)* a well-defined vector relative degree and admitting a canonical normal form, or that are *(b)* strongly invertible and feedback linearisable, with respect to the pair (u,e), fit inside the proposed framework. The regulator presented in this section is based on the following standing assumptions (see [14, Assumption A1, A2]).

Assumption 7.2.1. *There exist $\beta_0 \in \mathcal{KL}$, $\alpha_0 > 0$ and, for each solution w of (46), there exist $z^\star : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^{n_z}$ and $u^\star : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^{n_u}$ fulfilling*

$$\begin{aligned} \dot{z}^\star &= f_0(w,x^\star) + b_0(w,x^\star)u^\star, \\ 0 &= q(w,x^\star) + b(w,x^\star)u^\star, \end{aligned} \tag{50}$$

where $x^\star = (z^\star, 0, 0)$*, and for all* $t > 0$ *the following holds*

$$|z(t) - z^\star(t)| \leq \beta_0 \left(|z(0) - z^\star(0)|, t \right) + \alpha_0 \left| (\chi, \zeta) \right|_{[0,t)} .$$

Assumption 7.2.2. *There exists a full-rank matrix* $\mathcal{L} \in \mathbb{R}^{n_u \times n_e}$ *such that the matrix* $b(w,x)\mathcal{L}$ *is bounded, and*

$$\mathcal{L}^\top b(w,x)^\top + b(w,x)\mathcal{L} \geq I_{n_e}$$

holds for all $(w,x) \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_x}$*, and the map*

$$(w,x) \mapsto (b(w,x)\mathcal{L})^{-1} q(w,x)$$

is Lipschitz.

Equations (50) are the specialisation of the regulator equations (48) in this non-equilibrium context. As a consequence of the latter assumption, $u^\star$ in (50) is given by

$$u^\star = -b\left(w, x^\star\right)^\top \left(b\left(w, x^\star\right) b\left(w, x^\star\right)^\top \right)^{-1} q\left(w, x^\star\right).$$

In this framework, [14] proposes a post-processing internal model of the form

$$\dot{\eta} = \Phi(\eta) + Ge, \quad \eta \in \mathbb{R}^{dn_e},$$

with $d \in \mathbb{N}$, $\eta = (\eta_1, \ldots, \eta_d)^\top$, $\eta_i \in \mathbb{R}^{n_e}$, and

$$\Phi(\eta) = \begin{pmatrix} \eta_2 \\ \vdots \\ \eta_d \\ \psi(\eta, \vartheta) \end{pmatrix}, \quad G = \begin{pmatrix} g h_1 I_{n_e} \\ g^2 h_2 I_{n_e} \\ \vdots \\ g^d h_d I_{n_e} \end{pmatrix}. \tag{51}$$

In the aforementioned definition, the coefficients h_i, with $i = 1, \ldots, d$, are fixed so that the polynomial $s^d + h_1 s^{d-1} + \cdots + h_{d-1} s + h_d$ is Hurwitz, $g > 0$ is a parameter to be designed, and $\psi : \mathbb{R}^{dn_e} \times \mathbb{R}^{n_\vartheta} \mapsto \mathbb{R}^{n_e}$ is a function to be fixed and $\vartheta \in \mathbb{R}^{n_\vartheta}$, with $n_\vartheta \in \mathbb{N}$, is an adaptive parameter generated by the identifier subsystem, whose dynamics is described by

$$\begin{aligned} \dot{\varsigma} &= \mu\left(\varsigma, \eta, e\right), \\ \vartheta &= \omega\left(\varsigma\right), \end{aligned} \tag{52}$$

in which $\mu : \mathcal{S} \times \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \mapsto \mathcal{S}$ and $\omega : \mathcal{S} \mapsto \mathbb{R}^{n_\vartheta}$, with $\mathcal{S}$ a normed vector space of finite dimension, have to be fixed. Finally, the static stabiliser control action is chosen as

$$u = \mathcal{L}(K_\chi \chi + K_\zeta \zeta + K_\eta \eta_1 + K_w \nu(x^\star, w)),$$

where the matrices K_χ, K_ζ, and K_η take the form

$$K_\chi(l,\delta) = lK(\delta), \quad K_\zeta(l) = -lI_{n_e}, \quad K_\eta(l,\delta) = lK(\delta)C^\top,$$

with $K(\delta) = \text{blkdiag}\left(K^1(\delta),\dots,K^{n_e}(\delta)\right)$, where

$$K^i(\delta) = -\begin{pmatrix} c_1^i\delta^{n_\chi^i} & c_2^i\delta^{n_\chi^i-1} & c_{n_\chi^i}^i\delta \end{pmatrix},$$

for $i = 1,\dots,n_e$, in which the coefficients c_j^i are chosen so that the polynomial $s^{n_\chi^i} + c_{n_\chi^i}^i s^{n_\chi^i-1} + \cdots + c_2^i s + c_1^i$, $i = 1,\dots,n_e$, is Hurwitz, and $l,\delta > 0$ are design parameters to be fixed. Note that the matrix K_w and the function $\nu(\cdot,\cdot)$ are left as a degree of freedom. Indeed these quantities can be used to represent possible feedforward contributions added by the designer employing knowledge about w and $x^\star$ (Equation (50)). The degrees of freedom left to be fixed at this stage are the dimension d and function ψ of the internal model unit, the data $(\mathcal{S}, n_\vartheta, \mu, \omega)$ of the identifier, and the control parameters g, l, and δ. A key step in the regulator synthesis is the choice of the internal model (51) and of its adaptation through the design of the identifier (52). This should be chosen in order to achieve small, possibly zero, asymptotic regulation error, in spite of uncertainties involving the regulation equations and the system dynamics. From Equation (51) we can write

$$e(t) = \left(h_d g^d\right)^{-1}\left(\dot\eta_d(t) - \psi\left(\eta(t), \vartheta(t)\right)\right). \tag{53}$$

The proposed design strategy chooses (d,ψ) and the identifier pivoting around the idea that $\dot\eta_d(t) - \psi\left(\eta(t),\vartheta(t)\right)$ can be interpreted as a prediction error attained by the model ψ in relating the next derivative $\dot\eta_d(t)$ to the previous derivatives $\eta(t)$, and that, by minimising this prediction error, the actual regulation error is also minimised due to (53). In this context, the problem of choosing (d,ψ) is treated as an identification problem, where $\psi(\eta,\vartheta)$ is referred to as *prediction model* and the set

$$\mathcal{M} = \{\psi(\eta,\vartheta) : \vartheta \in \mathbb{R}^{n_\vartheta}\}$$

of all the possible candidate models as the corresponding *model set*. The selection of such quantities must be grounded on some preliminary knowledge about the class of signals to $\dot\eta_d$ and η are expected to belong. In this context, the steady-state signals $(x^\star, u^\star)$ resulting from the regulator equations are the anchor point on which that knowledge can be drawn. The original work [14] provides a constructive procedure to design the internal model quantities, as well as the identifier functions, showing how approximate regulation can be attained, thus for more details the reader is referred to it.

7.3 ADAPTING THE POST-PROCESSING INTERNAL MODEL

All the approaches that assume to known the membership model of the friend have the disadvantage of limiting the *class of friends* which we can deal with, leading to a degraded performance in all those cases in which the steady-state signals $(x^\star, u^\star)$ are highly uncertain and the chosen class $\mathcal{M}$ is inadequate to represent the ideal function $\psi^\star$. Unlike previous works in this field, we dropped the assumption of ψ belonging to a given model set on behalf of a more general and less conservative hypothesis. In particular, we let such a function be of whatever shape, with the only constraint to be *sufficiently smooth*. In this respect, we refer to $\psi(\eta, \vartheta)$ with $\psi(\eta)$, to highlight the generality of the proposed framework, being ψ not parameterised by any ϑ. In view of the latter, we recall [14, Assumption A3] under which the asymptotic stability results can be drawn.

Assumption 7.3.1. *The map $\psi(\eta)$ is Lipschitz and differentiable with a locally Lipschitz derivative, and the Lipschitz constants do not depend on δ and l. Moreover, there exists a compact set $H^\star \subset \mathbb{R}^{n_e} \times \mathbb{R}^{dn_e}$, independent from δ and l, such that every solution of*

$$\dot{\eta} = \Phi(\eta) + Ge,$$

satisfies $\left(\eta_d^\star(t), \eta^\star(t)\right) \in H^\star$ for all $t \in \mathbb{R}_{\geq 0}$.

7.3.1 *Gaussian Process Regression*

The key idea behind the proposed approach dwells in modeling the unknown function ψ as the realization of a Gaussian process. GPs are function estimators widely used because of the flexibility they offer in modeling nonlinear maps directly out from the collected data [127]. A GP model is fully described by a mean function $m : \mathbb{R}^{dn_e} \mapsto \mathbb{R}^{n_e}$ and a covariance function (aka *kernel*) $\kappa : \mathbb{R}^{dn_e} \times \mathbb{R}^{dn_e} \mapsto \mathbb{R}^{n_e}$. Whereas there are many possible choices of mean and covariance functions, in this work we keep the formulation of κ general, with the only constraint expressed by Assumption 7.3.3 below. Yet we force, without loss of generality, $m(\eta) = 0_{n_e}$ for any η. Thus we assume that

$$\psi(\eta) \sim \mathcal{GP}(0, \kappa(\cdot, \cdot)).$$

Supposing to have access to a data-set $(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}_{\boldsymbol{d}}) = \{(\eta(t_1), \dot{\eta}_d(t_1)), \ldots$ $\ldots, (\eta(t_N), \dot{\eta}_d(t_N))\}$ with each pair $(\eta(t_h), \dot{\eta}_d(t_h)) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e}$ obtained as $\dot{\eta}_d(t_h) = \psi(\eta(t_h)) + \varepsilon(t_h)$ with $\varepsilon(t_h) \sim \mathcal{N}\left(0, \sigma_n^2 I_{n_e}\right)$ be a white Gaussian noise, then the posterior distribution of ψ given the data-set is still a Gaussian process with mean μ and variance σ^2 given by [127]

$$\begin{aligned} \mu(\eta) &= \boldsymbol{\kappa}(\eta)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \dot{\boldsymbol{\eta}}_{\boldsymbol{d}}, \\ \sigma^2(\eta) &= \kappa(\eta, \eta) - \boldsymbol{\kappa}(\eta)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{\kappa}(\eta), \end{aligned} \tag{54}$$

where $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{N\times N}$ is the *Gram matrix* whose (k,h)th entry is $\boldsymbol{\mathcal{K}}_{k,h} = \kappa(\boldsymbol{\eta}_k, \boldsymbol{\eta}_h)$, with $\boldsymbol{\eta}_k$ the kth entry of $\boldsymbol{\eta}$, and $\boldsymbol{\kappa}(\eta) \in \mathbb{R}^N$ is the kernel vector whose kth component is $\boldsymbol{\kappa}_k(\eta) = \kappa(\eta, \boldsymbol{\eta}_k)$. The problem of inferring an unknown function ψ from a finite set of data can be seen as a special case of *ridge regression* where the prior assumptions (mean and covariance) are encoded in terms of *smoothness* of μ. In particular, let $\mathcal{H}$ be a RKHS associated with the kernel function κ, then the function ψ can be infered by minimizing the functional

$$\mathcal{J} = \frac{\lambda_s}{2}\|\mu\|^2_{\mathcal{H}} + Q(\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}, \mu(\boldsymbol{\eta})),$$

where the first term plays the role of *regularizer* and represents the smoothness assumptions on μ as encoded by a suitable RKHS, while the second one represents the data-fit term assessing the quality of the prediction $\mu(\boldsymbol{\eta})$ with respect to the observed data $\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}$ [127]. According to the *representer theorem* [114], each minimizer $\mu \in \mathcal{H}$ of $\mathcal{J}$ takes the form $\mu(\eta) = \boldsymbol{\kappa}(\eta)\,\alpha$. In the particular case in which $Q(\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}, \mu(\boldsymbol{\eta}))$ corresponds to a negative log-likelihood of a Gaussian model with variance σ_n^2, namely

$$Q(\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}, \mu(\boldsymbol{\eta})) = \frac{1}{2\sigma_n^2}\|\dot{\boldsymbol{\eta}}_{\boldsymbol{d}} - \mu(\boldsymbol{\eta})\|_2^2,$$

the value of α recovers the expression in Equation (54) as

$$\alpha = \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I\right)^{-1}\dot{\boldsymbol{\eta}}_{\boldsymbol{d}}.$$

From now on we suppose that the following standing assumptions hold (see [17, Assumption 2, Assumption 3])

Assumption 7.3.2. *μ is Lipschitz continuous with Lipschitz constant L_η, and its norm is bounded by μ_{max}.*

Assumption 7.3.3. *The kernel function $\kappa(\cdot,\cdot)$ is Lipschitz continuous with constant L_κ, with a locally Lipschitz derivative of constant $L_{d\kappa}$, and its norm is bounded by κ_{max}.*

Although any kernel fulfilling Assumption 7.3.3 can be a valid candidate, in the following, we exploit the commonly adopted *squared exponential kernel* as prior covariance function, which can be expressed as

$$\kappa(\eta,\eta') = \sigma_p^2 \exp\left(-(\eta-\eta')^\top \Lambda^{-1}(\eta-\eta')\right) \tag{55}$$

for all $\eta,\eta' \in \mathbb{R}^{dn_e}$, where $\Lambda = \mathrm{diag}(2\lambda_{\eta_1}^2,\ldots,2\lambda_{\eta_{dne}}^2)$, $\lambda_{\eta_i} \in \mathbb{R}_{>0}$ is known as *characteristic length scale* relative to the ith signal, and σ_p^2 is usually called *amplitude* [127]. We conclude this section by stating a constructive assumption, on which the main contribution of this chapter is built.

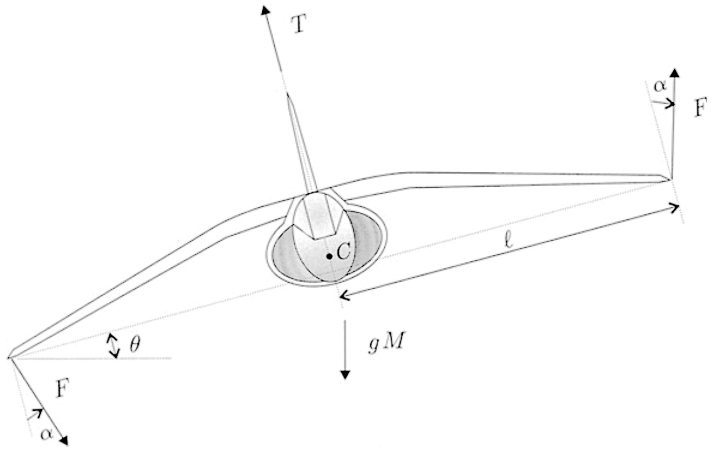


Figure 34: The vertical takeoff and landing aircraft considered in numerical simulations.

Lemma 7.3.1. *Let $\mathcal{H}$ be a RKHS induced by a positive-definite kernel κ, satisfying Assumption 7.3.3, and let $f \in \mathcal{H}$, then f is Lipschitz continuous with constant $L_f = \|f\|_{\mathcal{H}} c_{\mathcal{H}} L_\kappa$, where $c_{\mathcal{H}}$ is a positive real constant.*

The L^2 and $\mathcal{H}$ norms, being the latter induced by a positive definite kernel, are equivalent with constant $c_{\mathcal{H}}$ (see [29] for further details).

Proof.

$$\begin{aligned}
\left|f(x) - f(x')\right| &= \left|\langle f, \kappa(\cdot, x)\rangle_{\mathcal{H}} - \langle f, \kappa(\cdot, x')\rangle_{\mathcal{H}}\right| \\
&= \left|\langle f, \kappa(\cdot, x) - \kappa(\cdot, x')\rangle_{\mathcal{H}}\right| \\
&\leq \|f\|_{\mathcal{H}} \left\|\kappa(\cdot, x) - \kappa(\cdot, x')\right\|_{\mathcal{H}} \\
&\leq \|f\|_{\mathcal{H}} c_{\mathcal{H}} \left\|\kappa(\cdot, x) - \kappa(\cdot, x')\right\| \\
&\leq \|f\|_{\mathcal{H}} c_{\mathcal{H}} L_\kappa \left\|x - x'\right\| \\
&\leq L_f \left\|x - x'\right\|
\end{aligned}$$

□

Assumption 7.3.4. *The map ψ belongs to the RKHS associated to the kernel function $\kappa(\cdot, \cdot)$ in Equation (55).*

It is worth noting that the first statement of Assumption 7.3.1 is implied by Assumption 7.3.4 by means of Lemma 7.3.1.

7.3.2 *Gaussian Process-based Adaptive Regulation*

The proposed regulator reads as follows

$$
\begin{cases}
\dot{\zeta} & = 1 \\
\dot{\eta} & = \Phi(\eta, \mu(\eta, \varsigma, \alpha, \zeta)) + Ge \\
\dot{\xi}_1 & = \xi_2 - m_1 \rho (\xi_1 - \eta_d) \\
\dot{\xi}_2 & = \dot{\mu}(\eta, \varsigma, \alpha, \zeta) - m_2 \rho^2 (\xi_1 - \eta_d) \\
\dot{\varsigma} & = 0
\end{cases}
\quad (\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathcal{C},
$$

$$
\begin{cases}
\zeta^+ & = 0 \\
\eta^+ & = \eta \\
\xi_1^+ & = \xi_1 \\
\xi_2^+ & = \xi_2 \\
\varsigma^+ & = (S \otimes I_p)\, \varsigma + (B \otimes I_p) \begin{bmatrix} \eta & \xi_2 & \zeta \end{bmatrix}^\top
\end{cases}
\quad (\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathcal{D}, \tag{56}
$$

in which $\alpha = \gamma(\varsigma) \in \mathbb{R}^{N \times N}$, $p = dn_e + n_e + 1$, $(\xi_1, \xi_2) \in \mathbb{R}^{n_e} \times \mathbb{R}^{n_e}$, and $\varsigma \in \mathbb{R}^{n_\varsigma}$ with $n_\varsigma = pN$. The matrices $S \in \mathbb{R}^{N \times N}$ and $B \in \mathbb{R}^N$ have the "shift" form

$$
S = \begin{pmatrix} 0_{(N-1)\times 1} & I_{N-1} \\ 0 & 0_{1\times(N-1)} \end{pmatrix}, \quad B = \begin{pmatrix} 0_{(N-1)\times 1} \\ 1 \end{pmatrix},
$$

while Φ and G have the same structure described by Equation (51), $\mathcal{C} = \{(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \times \mathbb{R}^{n_e} \times \mathcal{S} \times \mathbb{R}_{\geq 0} : \sigma^2(\eta, \varsigma, \alpha, \zeta) \leq \sigma^2_{\text{thr}}\}$ represents the flow set, and $\mathcal{D} = \{(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \times \mathbb{R}^{n_e} \times \mathcal{S} \times \mathbb{R}_{\geq 0} : \sigma^2(\eta, \varsigma, \alpha, \zeta) \geq \sigma^2_{\text{thr}}\}$ is the jump set, with σ^2_{thr} arbitrary. The functions $\mu(\eta, \varsigma, \alpha, \zeta)$ and $\sigma^2(\eta, \varsigma, \alpha, \zeta)$ represent the a posteriori GP estimated mean and variance after the collection of N samples, and $\mathcal{S} \subseteq \mathbb{R}^{Nn_\varsigma}$. The proposed regulator is composed of *(a)* a purely continuous-time subsystem (η, ξ_1, ξ_2) whose dynamics depends on ς and α that are constant during flow, *(b)* a purely discrete-time subsystem ς updated ad each jump time, and *(c)* a hybrid clock ζ. Note that, due to the definition of the sets $\mathcal{C}$ and $\mathcal{D}$ the jumping law is not directly related to the clock variable ζ, thus at the moment it is not clear if (56) suffers from *zeno* or *chattering* issues. The subsystem η plays the role of internal model unit, and it is taken of the same form as (51). The subsystem (ξ_1, ξ_2) plays the role of *observer* of the quantity $\dot{\eta}_d$ required to build the data-set and not directly available. The dynamic equation of $(\dot{\xi}_1, \dot{\xi}_2)$ follows the canonical high-gain construction with the coefficients $m_1, m_2 > 0$ arbitrary and $\rho > 0$ left as

a design parameter. The quantities (ξ_2, η) act as proxy for the ideal $(\ddot{\eta}_d^\star, \eta^\star)$ required to build an approximation of $\psi^\star$; (ξ_2, η) are thus feeded to the discrete-time GP regressor represented by the subsystem ς and by the properly defined functions (μ, σ^2, γ). In particular, denoting $z = \mathrm{col}(\eta, \zeta)$, the latter functions read as

$$\begin{aligned}
\mu(z) &= \boldsymbol{\kappa}(z)^\top \gamma(\varsigma)\, \boldsymbol{\xi_2}, \\
\sigma^2(z) &= \kappa(z,z) - \boldsymbol{\kappa}(z)^\top \gamma(\varsigma)\, \boldsymbol{\kappa}(z), \\
\gamma(\varsigma) &= \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1},
\end{aligned}$$

where $\boldsymbol{\kappa}$ and $\boldsymbol{\mathcal{K}}$ are defined as in Section 7.3.1, with the only difference that the kernel function κ has been enhanced by adding a dependency from ζ. In this context, Equation (55) takes the form

$$\begin{aligned}
\kappa(z_i, z_j) = \sigma_p^2 \exp\left(-(\eta_i - \eta_j)^\top \Lambda^{-1} (\eta_i - \eta_j)\right) \\
\exp\left(-\sum_{k=\min(i,j)}^{\max(i,j)} \left\| \frac{\zeta_k}{2\lambda_\zeta} \right\|\right) \text{ if } z_i, z_j \in \boldsymbol{z},
\end{aligned}$$

$$\begin{aligned}
\kappa(z, z_j) = \sigma_p^2 \exp\left(-(\eta - \eta_j)^\top \Lambda^{-1} (\eta - \eta_j)\right) \\
\exp\left(-\frac{\zeta}{2\lambda_\zeta^2} + \sum_{k=j}^{m} \left\| \frac{\zeta_k}{2\lambda_\zeta} \right\|\right) \text{ if } z \notin \boldsymbol{z}, z_j \in \boldsymbol{z},
\end{aligned} \tag{57}$$

where λ_ζ is a parameter to be tuned and the quantity $(\boldsymbol{z}, \boldsymbol{\xi_2})$ represents the data-set constructed as discussed in Section 7.3.1 extended with the sampled clock ζ, and stored, as a state variable, inside the shift register ς. The addition of the clock as independent variable inside the GP regression is the key to obtain a good estimation of ψ starting from the noisy proxy (ξ_2, η). This choice is motivated by the fact that the introduction of ζ allows us to shape μ through the parameter λ_ζ whose inverse value can be interpreted as a *forgetting factor*, commonly used in identification.

The design parameters (g, l, δ, ρ) in (56) can be chosen so that the closed-loop system has an asymptotic regulation error bounded by a function of the best attainable prediction, namely

$$\limsup_{t\to\infty} \|e(t)\| = c_e \limsup_{t\to\infty} \|\ddot{\eta}_d^\star(t) - \mu(\eta^\star(t))\|,$$

with c_e constant that depends on the chosen parameters. Such a results directly follows from the adaptation of the arguments reported by [14] and [13] in the specific case in which (56) satisfies a set of properties known as *identifier requirements*. The following results are instrumental to verify that the proposed regulator fits inside the same framework of [14] and [13].

Lemma 7.3.2. *Let Assumptions 7.3.1, 7.3.2, and 7.3.3 hold. Moreover, suppose that the chosen σ^2_{thr} satisfies*

$$\frac{\sigma_p^2\sigma_n^2}{\sigma_p^2+\sigma_n^2} < \sigma^2_{thr} < \sigma_p^2, \tag{58}$$

then any solution $(\eta, \xi_1, \xi_2, \varsigma, \zeta)$ of (56) originating from the set $\chi_0 = \{(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \times \mathbb{R}^{n_e} \times \mathcal{S} \times \mathbb{R}_{\geq 0} : \zeta = 0\}$ satisfies a dwell time condition in the sense of [64], and a reverse dwell time condition in the sense of [63].

Proof. Using the same arguments proposed by [63], we shape the proof to show the existence of $\overline{\mathrm{T}}$ and $\underline{\mathrm{T}}$ such that $\mathcal{C} \subseteq \{(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \times \mathcal{S} \times \mathbb{R}^{mp} \times \mathbb{R}_{\geq 0} : 0 \leq \zeta \leq \overline{\mathrm{T}}\}$ and $\mathcal{D} \subseteq \{(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathbb{R}^{dn_e} \times \mathbb{R}^{n_e} \times \mathcal{S} \times \mathbb{R}^{mp} \times \mathbb{R}_{\geq 0} : \underline{\mathrm{T}} \leq \zeta \leq \overline{\mathrm{T}}\}$. First note that, in view of Assumption 7.3.1, from the definition of kernel in (57), the value of $\sigma^2(\eta, \varsigma, \alpha, \zeta)$ reaches σ_p^2 as long as t goes to infinty, i.e.

$$\lim_{t\to\infty} \sigma^2(\eta(t), \varsigma(t), \alpha(t), \zeta(t)) = \sigma_p^2. \tag{59}$$

Recalling $z = \mathrm{col}(\eta, \zeta)$, then the flow and jump dynamics of σ^2 are described by

$$\begin{aligned}(\dot{\sigma^2}) &= \dot{z}^\top \left(\frac{d\kappa(z,z)}{dz} - 2\frac{d\boldsymbol{\kappa}(z)}{dz}^\top \gamma(\varsigma)\boldsymbol{\kappa}(z) \right) \\ &= -2\dot{z}^\top \frac{d\boldsymbol{\kappa}(z)}{dz}^\top \gamma(\varsigma)\boldsymbol{\kappa}(z),\end{aligned} \tag{60}$$

when $(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathcal{C}$, and

$$(\sigma^2)^+ = \kappa(z^+, z^+) - \boldsymbol{\kappa}(z^+)\gamma(\varsigma^+)\boldsymbol{\kappa}(z^+), \tag{61}$$

when $(\eta, \xi_1, \xi_2, \varsigma, \zeta) \in \mathcal{D}$. The existence of $\overline{\mathrm{T}}$ follows from (59) and the second inequality of (58), in particular by choosing $\sigma^2_{\mathrm{thr}} < \sigma_p^2$, since $\sigma^2 \to_{\zeta\to\infty} \sigma_p^2$, there exists a real value $\overline{\mathrm{T}}$ such that $\sigma^2 \geq \sigma^2_{\mathrm{thr}}$ for any $\zeta \geq \overline{\mathrm{T}}$. This ensures persistency of jump intervals. Consider now Equations (60) and (61), the flow dynamics $(\dot{\sigma^2})$ results to be a continous function being it a product of Lipschitz functions (Assumptions 7.3.1, 7.3.2, 7.3.3). Moreover, its norm can be upper bounded as

$$\left\|(\dot{\sigma^2})\right\| \leq 2\|\dot{z}\| \left\|\frac{d\boldsymbol{\kappa}(z)}{dz}\right\| \|\gamma(\varsigma)\| \|\boldsymbol{\kappa}(z)\|,$$

with $\dot{z} = [\eta_2, \ldots, \eta_d, \psi(\eta), 1]^\top$. Since during flow the function κ takes the form of (57), rewriting the latter as

$$\kappa(z, z_j) = \sigma_p^2 \exp\left(-(z - z_j)^\top \bar{\Lambda}^{-1}(z - z_j)\right),$$

the quantity

$$\frac{d\boldsymbol{\kappa}(z)}{dz} = \begin{bmatrix} -\kappa(z, z^1)\,\bar{\Lambda}^{-1}(z - z^1) \\ -\kappa(z, z^2)\,\bar{\Lambda}^{-1}(z - z^2) \\ \vdots \\ -\kappa(z, z^m)\,\bar{\Lambda}^{-1}(z - z^m) \end{bmatrix},$$

where z^i denotes the ith training point stored inside ς, and with $\bar{\Lambda}$ properly defined. Assumption 7.3.1 ensures $\|\dot{z}\| \leq l_z$, with l_z dependent on the chosen $H^\star$, while Assumption 7.3.3 ensures boundedness of $\gamma(\varsigma)$ and $\boldsymbol{\kappa}(z)$, namely

$$\|\gamma(\varsigma)\| \leq l_\gamma, \quad \|\boldsymbol{\kappa}(z)\| \leq l_k.$$

From the previous arguments follows that $\|d\boldsymbol{\kappa}(z)/dz\| \leq l_{dk}$, yielding to $\left\|\dot{\sigma^2}\right\| \leq l_{\sigma^2}$ with $l_{\sigma^2} = l_z l_{dk} l_\gamma l_k$. Note that the bound l_{σ^2} does not depend neither on the chosen regulator parameters, nor on the initial conditions. Consider now the jump dynamics, under the Assumption 7.3.3 of Lipschitz continuous kernel, we can explicitly derive an upper bound on the value of $(\sigma^2)^+$ at each jump (see [86, Theorem 1])

$$\begin{aligned}(\sigma^2)^+ \leq \Big[\big|\mathcal{B}(\bar{z})\big|\,(\kappa(z^\star, z^\star) + 2L_k\varrho) + \sigma_n^2\Big]^{-1} \\ \Big[\big(2lb\big(\kappa\left(z^+, z^+\right) + \kappa\left(z^\star, z^+\right)\big) \\ - L_k^2\varrho^2\big)\big|\mathcal{B}(z^\star)\big| + \sigma_n^2\kappa\left(z^+, z^+\right)\Big],\end{aligned}$$

where $\mathcal{B}(z^\star)$ denotes the training data set restricted to a ball around $z^\star$ with radius $\varrho \in \mathbb{R}$. In our specific case $z^\star = z^+$, thus the aforementioned relation boils down to

$$(\sigma^2)^+ \leq \frac{\left(4L_k\varrho\sigma_p^2 - L_k^2\varrho^2\right) m_{\mathcal{B}(z^\star)} + \sigma_n^2\sigma_p^2}{\left(\sigma_p^2 + L_k\varrho\right) m_{\mathcal{B}(z^\star)} + \sigma_n^2}$$

with $\varrho \leq \sigma_p^2/L_k$ and $m_{\mathcal{B}(z^\star)} \leq m$ is the cadinality of $\mathcal{B}(z^\star)$. Clearly, the worst case scenario is represented by the case in wich $\mathcal{B}(z^\star)$ cannot embrace any training points, thus when $\varrho = 0$. In this particular case we get

$$(\sigma^2)^+ \leq \frac{\sigma_n^2\sigma_p^2}{\sigma_p^2 + \sigma_n^2} < \sigma_{\text{thr}}^2.$$

Finally, the existence of $\underline{\mathrm{T}}$ follows from the fact that the flow dynamics (60) is continuous with upper bounded norm and its initial condition, at each jump time, is lower than the chosen threshold σ_{thr}^2. This ensures persistency of flow intervals. □

Lemma 7.3.3. *Consider the hybrid subsystem*

$$
\begin{aligned}
&\begin{cases} \dot{\zeta} &= 1 \\ \dot{\eta} &= \Phi\left(\eta, \mu\left(\eta, \varsigma, \alpha, \zeta\right)\right) + Ge \\ \dot{\varsigma} &= 0 \end{cases} \\
&\left(\eta, \varsigma, \zeta\right) \in \mathcal{C}, \\
&\begin{cases} \zeta^{+} &= 0 \\ \eta^{+} &= \eta \\ \varsigma^{+} &= \left(S \otimes I_p\right) \varsigma + \left(B \otimes I_p\right) \\ & \qquad \begin{bmatrix} \eta + \delta_{\eta_{1:d-1}} & \eta_d + \delta_{\eta_d} & \zeta \end{bmatrix}^{\top} \end{cases} \\
&\left(\eta, \varsigma, \zeta\right) \in \mathcal{D},
\end{aligned}
\tag{62}
$$

with $\mathcal{C}$ and $\mathcal{D}$ defined as above and $\delta_\eta = \left(\delta_{\eta_{1:d-1}}, \delta_{\eta_d}\right) \in \mathbb{R}^{dn_e}$ hybrid input. Let Assumptions 7.3.2 and 7.3.3, and Lemma 7.3.2 hold, then the tuple $\left(\mathcal{S}, \mu, \gamma, \alpha\right)$ satisfies the identifier requirements [13] relative to $\mathcal{J}$, namely there exists a compact set $S^\star \subset \mathcal{S}$, $\beta_\varsigma \in \mathcal{KL}$, a Lipschitz function $\rho_\varsigma \in \mathcal{K}$, and for each solution $\left(w, x, \eta, \zeta\right)$ to Equations (46), (49), and (51), a hybrid arc $\varsigma^\star : dom\left(w, x, \eta, \zeta\right) \mapsto \mathcal{S}$ and a $j^\star \in \mathbb{N}$ such that $\left(\zeta, \eta, \varsigma^\star, d\right)$ with $\delta_\eta = 0$ is a solution pair to (62) satisfying $\varsigma^\star(j) \in S^\star$ for all $j \geq j^\star$, and the following holds:

1. ***Optimality:*** *For all $j \geq j^\star$, the function $\mu^\star(\cdot) = \mu\left(\eta, \varsigma^\star, \gamma\left(\varsigma^\star\right), \zeta\right)$ satisfies*

$$\mu_j^\star(\cdot) \in \arg\min \mathcal{J}_j.$$

2. ***Stability:*** *For every hybrid input δ_η, every solution pair $\left(\eta, \varsigma, \zeta, \delta\right)$ of the hybrid subsystem (62) satisfies for all $j \in Jmp\left(\eta, \varsigma, \zeta\right)$*

$$\left|\varsigma(j) - \varsigma^\star(j)\right| \leq \max\{\beta_\varsigma(\left|\varsigma(0) - \varsigma^\star(0)\right|, j), \rho_\varsigma(\left|\delta_\eta\right|_j)\}.$$

3. ***Regularity:*** *The map $\mu(\cdot)$ is Lipschitz and differentiable with a locally Lipschitz derivative.*

The proof of the latter lemma directly follows from the same arguments proposed by [13, Proposition 2].

7.3.3 *Numerical Simulations*

We consider, as a testbed, the problem of regulation the lateral (y_1, y_2) and angular (θ_1, θ_2) dynamics of a Vertical-TakeOff-and-Landing (VTOL) aircraft [72] subjected to lateral forces produced by the wind denoted

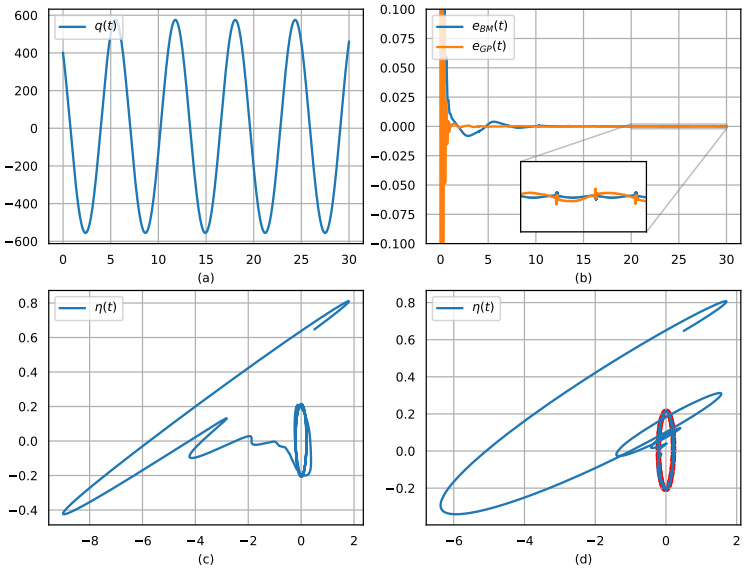


Figure 35: Results obtained comparing our approach (e_{GP}) versus [14] (e_{BM}) when the exogenous disturbance $d(w)$ is generated by (65). In both cases, the used regulator parameters are the same reported in Table 6. In the figure, image (a) depicts the injected noise, (b) compares the behavior of the regulation errors, while (c) and (d) shows the dynamics of (η_1, η_2) along the experiments in which the Bin-Marconi regulator [14] and ours is applied, respectively. In figure (c) the used samples (ς) during the last flow interval are shown as green dots. The reported quantities are plotted with respect to the time in seconds (abscissa).

by $d(w)$. A graphical representation of the considered system is reported in Figure 34. The VTOL dynamics reads as

$$\begin{aligned} \dot{y}_1 &= y_2, & \dot{\theta}_1 &= \theta_2, \\ \dot{y}_2 &= d(w) - g\tan(\theta_1) + v, & \dot{\theta}_2 &= 2lJ^{-1}u, \end{aligned} \tag{63}$$

N	λ_{η_1}	λ_{η_2}	λ_ς	σ_p^2	σ_n^2	σ_{thr}^2
100	0.1	0.1	2	1	0.01	0.1

Table 5: Gaussian process parameters used in simulations.

(c_1, c_2, c_3)	l	δ	$\mathcal{L}$	(h_1, h_2)	g	(m_1, m_2)	ρ
$(15, 75, 125)$	250	150	20	$(15, 70)$	2	$(20, 20)$	2

Table 6: Regulator parameters used in simulations.

M	J	l	g
$5 \cdot 10^4$	$1.25 \cdot 10^4$	2	9.81

Table 7: Model parameters used in simulations.

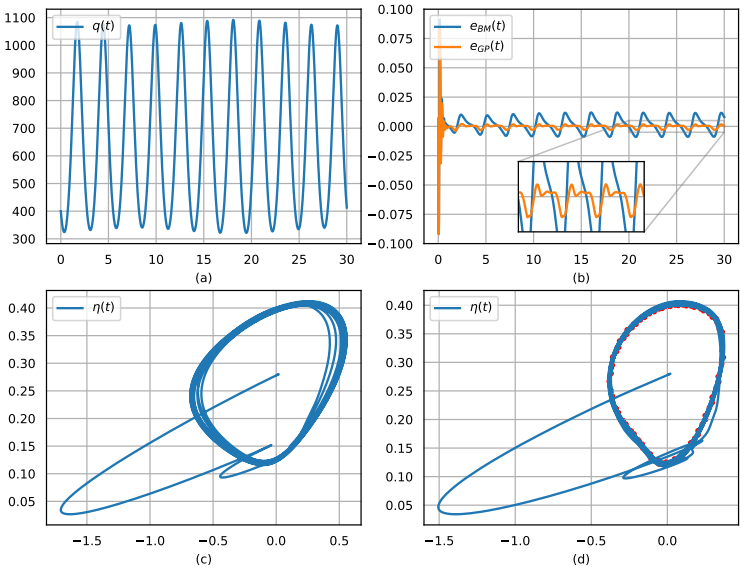


Figure 36: Results obtained comparing our approach (e_{GP}) versus [14] (e_{BM}) when the exogenous disturbance $d(w)$ is generated by (66). For a comprehensive explanation of Figures (a), (b), (c), and (d) please refer to Figure 35. The reported quantities are plotted with respect to the time in seconds (abscissa).

where $M > 0$ and $J > 0$ are the aircraft mass and inertia respectively, while $l > 0$ represents the wings lenght and $g > 0$ the gravitational constant. The input u is the force (F) on the wingtips, v is a vanishing input taking into account the (controlled) vertical dynamics (not considered here), and $d(w) := M^{-1}d_0(w)$, with $d_0(w)$ that is the lateral force produced by the wind. Considering as regulation error the aircraft lateral position $(e = y_1)$, the control objective is to remove the wind disturbance out from the lateral dynamics. Let $w(t)$ be generated by an exosystem of the form (46) and consider the following change of coordinates

$$\chi_1 = y_1, \quad \chi_3 = d(w) + g\tan(\theta_1),$$
$$\chi_2 = y_2, \quad \zeta = L_s d(w) - g\theta_2/\cos(\theta_1).$$

In the new coordinates, letting $x = \mathrm{col}\,(\chi, \zeta)$, the system (63) reads as follow

$$\begin{aligned} \dot{\chi}_1 &= \chi_2, \quad \dot{\chi}_3 = \zeta, \\ \dot{\chi}_2 &= \chi_3, \quad \dot{\zeta} = q(w,x) + b(w,x)u, \end{aligned} \tag{64}$$

with $q(w,x)$ and $b(w,x)$ given by

$$q(w,x) = L_s^2 d(w) - \frac{1}{g}\left(L_s d(w) - \zeta\right)^2 \sin\left(2\tan^{-1}\left(\frac{d(w) - \chi_3}{g}\right)\right),$$

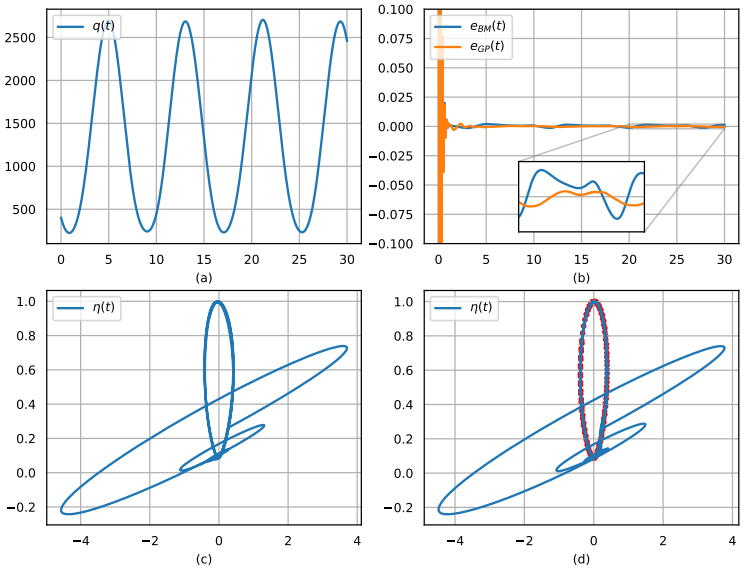


Figure 37: Results obtained comparing our approach (e_{GP}) versus [14] (e_{BM}) when the exogenous disturbance $d(w)$ is generated by (67). For a comprehensive explanation of Figures (a), (b), (c), and (d) please refer to Figure 35. The reported quantities are plotted with respect to the time in seconds (abscissa).

$$b(w,x) = -2gIJ^{-1}\cos\left(\tan^{-1}\left(\frac{d(w)-\chi_3}{g}\right)\right)^{-2}.$$

The system (64) is in the form (49) with Assumption 7.4.2 trivially fulfilled, since the z dynamics is absent, by $x^\star = 0$ and $u^\star = (gL_s^2 d(w) - 2d(w)(g^2+d(w)^2)(L_s d(w))^2)/2IJ^{-1}(g^2+d(w)^2)$, and Assumption 7.4.3 fulfilled on each compact set with $\mathcal{L}$ a negative number. With (c_1,c_2,c_3) the coefficients of a Hurwitz polynomial and $\delta, l > 0$ design parameters, we fix the control law as

$$u = \mathcal{L}\Big[c_1 l\delta^3(y_1+\eta_1) + c_2 l\delta^2 y_2 + c_3 l\delta(-g\tan(\theta_1)) + l(-g\theta_2/\cos^2(\theta_1))\Big].$$

Figures 35, 36, and 37 report the obtained results when the aircraft is perturbed with a lateral disturbance $d(w) = (2(10^7 w_1) + 10^6 w_3)/M$, where w_1 and w_3 are the states of three different exosystems. In particular, in Figure 35, $s(w)$ reads as

$$\begin{aligned} \dot{w}_1 &= w_2, & \dot{w}_3 &= w_4, \\ \dot{w}_2 &= -w_1, & \dot{w}_4 &= -4w_3, \end{aligned} \tag{65}$$

in Figure 36, $s(w)$ behaves as

$$\begin{aligned} \dot{w}_1 &= w_2, & \dot{w}_3 &= w_4, \\ \dot{w}_2 &= 4w_1 - w_1^3, & \dot{w}_4 &= -4w_3, \end{aligned} \tag{66}$$

while in Figure 37, the exosystem is described by

$$\begin{aligned} \dot{w}_1 &= w_2, & \dot{w}_3 &= w_4, \\ \dot{w}_2 &= 3\tan^{-1}(w_1) - w_1, & \dot{w}_4 &= -4w_3. \end{aligned} \tag{67}$$

In all simulations we exploit the same set of parameters, for both the regulator and the discrete-time identifier. The adopted parameters are reported in Table 5, Table 6, and Table 7.

7.4 IDENTIFICATION-BASED PRE-PROCESSING INTERNAL MODEL

In this section, we recall an adaptive pre-processing internal model design technique for a nonlinear system of the same kind of (49). In this second case, we set, without loss of generality, $b_0 = 0$, and restrict the analysis to those systems having an equal number of inputs and regulated outputs $n_u = n_e$. For seek of clarity, we report here the reference nonlinear system, with a slightly different notation

$$\begin{aligned} \dot{z} &= f_0(w, z, e), \\ \dot{e} &= Ae + B(q(w, z, e) + b(w, z, e)\, u), \\ y &= Ce, \end{aligned} \tag{68}$$

in which $z \in \mathbb{R}^{n_z}$ together with the error dynamics $e \in \mathbb{R}^{n_e}$ represent the overall state of the plant. The quantities $u \in \mathbb{R}^{n_y}$ and $y \in \mathbb{R}^{n_y}$ are the control input and the measured output respectively, while $w \in \mathbb{R}^{n_w}$ is an exogenous input, $f_0 : \mathbb{R}^{n_w} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_e} \mapsto \mathbb{R}^{n_z}$, $q : \mathbb{R}^{n_w} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_e} \mapsto \mathbb{R}^{n_y}$, $b : \mathbb{R}^{n_w} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_e} \mapsto \mathbb{R}^{n_y \times n_y}$ are continuous functions, and A, B, and C are defined as

$$A = \begin{pmatrix} 0_{(r-1)n_y \times n_y} & I_{(r-1)n_y} \\ 0_{n_y \times n_y} & 0_{n_y \times (r-1)n_y} \end{pmatrix}, \quad B = \begin{pmatrix} 0_{(r-1)n_y \times n_y} \\ I_{n_y} \end{pmatrix},$$

$$C = \begin{pmatrix} I_{n_y} & 0_{n_y \times (r-1)n_y} \end{pmatrix},$$

The presented regulator is based on the following set of standing assumptions.

Assumption 7.4.1. *The function f_0 is locally Lipschitz and the functions q and b are $\mathcal{C}^1$ functions, with local Lipschitz derivative.*

Assumption 7.4.2. *There exists a $\mathcal{C}^1$ map $\pi : \mathcal{C} \subset \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_z}$, with $\mathcal{C}$ an open neighborhood of $\mathcal{W}$, satisfying*

$$L^{(w)}_{s(w)} \pi(w) = f_0(w, \pi(w), 0),$$

with $L^{(w)}_{s(w)} \pi(w) = \frac{\partial \pi(w)}{\partial w} s(w)$, such that the system

$$\dot{w} = s(w), \quad \dot{z} = f_0(w, z, e),$$

is Input-to-State Stable (ISS) with respect to the input e, relative to the compact set $\mathcal{A} = \{(w, z) \in \mathcal{W} \times \mathbb{R}^{n_z} : z = \pi(w)\}$.

Assumption 7.4.3. *There exists a known constant nonsingular matrix* $\boldsymbol{b} \in \mathbb{R}^{n_y \times n_y}$ *such that the inequality*

$$\left\| (b(w,z,e) - \boldsymbol{b})\boldsymbol{b}^{-1} \right\| \leq 1 - \mu_0,$$

holds for some known scalar $\mu_0 \in (0,1)$*, and for all* $(w,z,e) \in \mathcal{W} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_e}$.

Remark 7.4.1. *Although not necessary (see [18]), Assumption 7.4.2 is a minimum-phase assumption customary made in the literature of output regulation (see [71], [119]). In particular, Assumption 7.4.2 is asking that the zero dynamics*

$$\dot{w} = s\left(w\right), \quad \dot{z} = f_0\left(w,z,0\right),$$

has a steady-state of the kind $z = \pi\left(w\right)$*, compatible with the control objective* $y = 0$*. As a consequence, the ideal input* $u^\star$ *making the set* $\mathcal{B} = \mathcal{A} \times \{0\}$ *invariant for (68) reads as*

$$u^\star\left(w, \pi\left(w\right)\right) = -b\left(w, \pi\left(w\right), 0\right)^{-1} q\left(w, \pi\left(w\right), 0\right).$$

The ability of the regulator to generate such an input is generally referred to as the internal model property*. With a little abuse of notation, from now on we refer to* $u^\star\left(w, \pi\left(w\right)\right)$ *with* $u^\star\left(w\right)$.

Remark 7.4.2. *Assumption 7.4.3 is a stabilizability assumption asking that* $b\left(w,z,e\right)$ *is always invertible whatever* (w,z,e) *is (see [156]). Moreover, the designer is required to have access to an estimate* $\boldsymbol{b}$ *of* $b\left(w,z,e\right)$ *which captures enough information about its behavior.*

In this framework, we now recall two results based on [94] and [13, Theorem 1].

Lemma 7.4.1. *Let previous assumptions hold and let* $n_\eta = 2(n_w + n_z + 1)$*. Then, for any choice of controllable pair* (F,G)*, with* F *a Hurwitz matrix, there exist two maps* $\tau : \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_\eta}$*, and* $\gamma : \mathbb{R}^{n_\eta} \mapsto \mathbb{R}^{n_y}$ *such that for all* w *in* $\mathcal{W}$

$$\begin{aligned} \gamma \circ \tau(w) &= u^\star(w), \\ L^{(w)}_{s(w)} \tau(w) &= F\tau(w) + Gu^\star(w), \end{aligned}$$

and the system

$$\begin{aligned} \dot{w} &= s(w), \\ \dot{z} &= f_0(w,z,e), \\ \dot{\eta} &= F\eta + Gu^\star(w) + \delta, \end{aligned}$$

is ISS relative to the set $\mathcal{E} = \left\{ (w,z,\eta) \in A \times \mathbb{R}^{n_\eta} : \eta = \tau(w) \right\}$ *and with respect to the input* (e,δ).

Let previous assumptions hold, and let $\mathcal{M} = \{\psi(\vartheta, \cdot) : \mathbb{R}^{n_\eta} \mapsto \mathbb{R}^{n_y} | \vartheta \in \Theta\}$, with Θ a finite-dimensional normed vector space, be a finite-dimensional model set where γ is supposed to range. Consider the following regulator structure

Same regulator proposed in [13], with the only difference in the definition of $\vartheta = \omega(\varsigma)$. It is, in fact, equivalent to set $\vartheta^+ = \omega(\varsigma)$ with $\dot{\vartheta} = 0$ and delay the optimality condition of one step, i.e. $\vartheta^\star \in \arg\min_{\vartheta \in \Theta} \mathcal{J}(j-1, \vartheta)$.

$$\begin{cases} \dot{\zeta} = 1 \\ \dot{\eta} = F\eta + Gu \\ \dot{\hat{e}} = A\hat{e} + B(\hat{\chi} + \boldsymbol{b}u) + \Lambda(l)H(y - \hat{e}_1) \\ \dot{\hat{\chi}} = -\boldsymbol{b}\psi(\vartheta, \eta, u) + l^{r+1}H_{r+1}(y - \hat{e}_1) \\ \dot{\varsigma} = 0 \end{cases}$$
$$(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in C_\zeta \times \mathbb{R}^{n_\eta + n_e + n_y} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_y},$$

$$\begin{cases} \zeta^+ = 0 \\ \eta^+ = \eta \\ \hat{e}^+ = \hat{e} \\ \hat{\chi}^+ = \hat{\chi} \\ \varsigma^+ = \mu(\varsigma, \eta, u) \end{cases}$$
$$(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in D_\zeta \times \mathbb{R}^{n_\eta + n_e + n_y} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_y},$$

with $\vartheta = \omega(\varsigma)$ and output $u = \boldsymbol{b}^{-1}\text{sat}(-\hat{\chi} + \kappa_s(\hat{e}))$. Where A, B, $\boldsymbol{b}$ are the same in (68) and Assumption 7.4.3, while (F, G) and n_η are the same of Lemma 7.4.1, and $\mathcal{S}$ a finite-dimensional normed vector space. The sets C_ζ, D_ζ are defined as

$$C_\zeta = [0, \overline{\mathrm{T}}], \quad D_\zeta = [\underline{\mathrm{T}}, \overline{\mathrm{T}}],$$

with $\overline{\mathrm{T}}$, $\underline{\mathrm{T}} \in \mathbb{R}_+$, satisfying $0 < \underline{\mathrm{T}} \leq \overline{\mathrm{T}}$. Furthermore, fix $\Lambda(l) = \text{diag}(lI_{n_y}, l^2 I_{n_y}, \ldots, l^r I_{n_y})$, $H = \text{diag}(H_1, \ldots, H_r)$, and $H_i = \text{diag}(h_i^1, \ldots, h_i^{n_y})$ with $\{h_1^j, h_2^j, \ldots, h_{r+1}^j\}$ for all $j = 1, \ldots, n_y$ coefficients of a Hurwitz polinomial, and $l \in \mathbb{R}_{>0}$ is a control parameter. Let the tuple $(\mathcal{M}, \mathcal{S}, \psi, \Theta, \omega)$ be such that the *identifier requirements*, relative to a given cost function $\mathcal{J}$, are satisfied [13]. Namely there exist $\beta_\varsigma \in \mathcal{KL}$, locally Lipschitz $\rho_\varsigma, \rho_\vartheta \in \mathcal{K}$, a compact set $\mathcal{S}^\star \subset \mathcal{S}$ and, for each solution pair $((\zeta, w, \varsigma, \vartheta), (d_\eta, d_y))$ to

$$\begin{cases} \dot{\zeta} = 1 \\ \dot{w} = s(w) \\ \dot{\varsigma} = 0 \end{cases} \tag{69}$$
$$(\zeta, w, \varsigma, \vartheta, d_\eta, d_y) \in C_\zeta \times \mathcal{W} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_\eta} \times \mathbb{R}^{n_y},$$

$$\begin{cases} \zeta^+ = 0 \\ w^+ = w \\ \varsigma^+ = \mu\left(\varsigma, \tau(w) + d_\eta, \gamma(\tau(w)) + d_y\right) \end{cases}$$
$$\left(\zeta, w, \varsigma, \vartheta, d_\eta, d_y\right) \in D_\zeta \times \mathcal{W} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_\eta} \times \mathbb{R}^{n_y},$$

with $\vartheta = \omega(\varsigma)$, there exists a pair $(\varsigma^\star, \vartheta^\star)$ and a $j^\star \in \mathbb{N}$, such that $((\zeta, w, \varsigma^\star, \vartheta^\star), (0,0))$ is a solution pair to (69) satisfying $\varsigma^\star(j) \in \varsigma^\star$ for all $j \geq j^\star$ and the following properties hold:

1. *Optimality:* For each $j \geq j^\star$

$$\vartheta^\star \in \arg\min_{\vartheta \in \Theta} \mathcal{J}(j, \vartheta).$$

2. *Stability:* For each j

$$|\varsigma(j) - \varsigma^\star(j)| \leq \max\left\{\beta_\varsigma\left(\varsigma(0) - \varsigma^\star(0), j\right), \rho_\varsigma\left(\left|\left(d_\eta, d_y\right)\right|\right)\right\}.$$

3. *Regularity:* The function ω satisfies

$$|\omega(\varsigma) - \omega(\varsigma^\star)| \leq \rho_\vartheta\left(|\varsigma - \varsigma^\star|\right),$$

for all $(\varsigma, \varsigma^\star) \in \mathcal{S} \times \mathcal{S}^\star$, the map $\psi(\vartheta, \eta, u)$ is $\mathcal{C}^1$ with locally Lipschitz derivative in the argument η.

Then, for each compact sets $\mathcal{Z} \subset \mathbb{R}^{n_z}$, $\mathcal{E} \subset \mathbb{R}^{n_e}$, and $\mathcal{B} \subset \mathbb{R}^{n_e} \times \mathbb{R}^{n_y}$ of initial conditions for z, e, and $(\hat{e}, \boldsymbol{b})$ respectively, there exists $l_s^\star > 0$ such that if $l > l_s^\star$ then the aggregate state $x = (\zeta, w, z, e, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta)$ of the closed-loop system is bounded. Moreover, there exists a $\alpha_x > 0$ and for each $\underline{\mathrm{T}} > 0$, an $l_\varepsilon^\star > 0$, such that if $l > l_\varepsilon^\star(\underline{\mathrm{T}})$ then

$$\limsup_{t \to \infty} |y(t)| \leq \alpha_x \limsup_{t+j \to \infty} |u^\star(w) - \psi(\vartheta^\star, \tau(w))|.$$

7.5 ADAPTING THE PRE-PROCESSING INTERNAL MODEL

Building an identifier satisfying the requirements presented in Section 7.4 is necessary linked to a specific choice of the model set $\mathcal{M}$, which is the space of functions where γ is supposed to range. Due to implementation constraints, it is customary to focus on finite-dimensional sets, which allows the parametrization of γ by a parameter ϑ ranging in a finite-dimensional vector space Θ. This, in turn, limits the flexibility of the proposed approach, especially when the structure of the friend is not a priori known. For this reason, we drop the assumption about $\mathcal{M}$ by performing regression in the space of *universal approximators* made by Gaussian processes.

Remark 7.5.1. *Assumption 7.4.1 asks for some Lipschitz conditions on maps that play a fundamental role in the stability analysis. In particular, Lipschitz continuity is required as long as high-gain-based observers are employed inside the regulator structure, later detailed in* (**??**)*. Furthermore, even if here we deal with data-driven adaptive control techniques, that ideally require smoothness assumptions on the function to be identified $u^\star$, in practice the adaptation of the internal model structure proposed by [94] makes the problem solvable without any further assumption. The details about this issue are more deeply discussed in Section* **??**.

7.5.1 *Gaussian Process Inference*

As in Section 7.3.1, the key idea behind the proposed approach consists in modeling the unknown function γ as the realization of a Gaussian process defined by a prior mean function $m : \mathbb{R}^{n_\eta} \mapsto \mathbb{R}^{n_y}$ and the kernel $\kappa : \mathbb{R}^{n_\eta} \times \mathbb{R}^{n_\eta} \mapsto \mathbb{R}$ [127]. While there are many possible choices of mean and covariance functions, we force $m(\eta) = 0_{n_y}$ for any $\eta \in \mathbb{R}^{n_\eta}$, and we keep the formulation of κ general, with the only constraint expressed by Assumption 7.5.2 below. Thus, we assume that

$$\gamma \sim \mathcal{GP}(0, \kappa(\cdot, \cdot)).$$

Supposing to have access to a data-set of samples collected at different time instants $t_i \in \mathbb{R}_{>0}$, $\mathcal{DS} = \{(\eta, u) \in \mathbb{R}^{n_\eta} \times \mathbb{R}^{n_y} : \eta = \eta(t_i), u = u(t_i) \text{ with } i = 1, \dots, N\}$, with each pair $(\eta, u) \in \mathcal{DS}$ obtained as $u(t_i) = \gamma(\eta(t_i)) + \varepsilon(t_i)$ with $\varepsilon(t_i) \sim \mathcal{N}(0, \sigma_n^2 I_{n_y})$ white Gaussian noise with known variance σ_n^2, the regression is performed by conditioning the prior GP distribution on the training data $\mathcal{DS}$ and a test point η. Denoting $\boldsymbol{\eta} = (\eta(t_1), \dots, \eta(t_N))^\top$ and $\boldsymbol{u} = (u(t_1), \dots, u(t_N))^\top$, the conditional posterior distribution given the data-set is still a Gaussian process with mean μ and variance σ^2 given by [127]

$$\begin{aligned} \mu(\eta) &= \boldsymbol{\kappa}(\eta)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{u}, \\ \sigma^2(\eta) &= \kappa(\eta, \eta) - \boldsymbol{\kappa}(\eta)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{\kappa}(\eta), \end{aligned} \tag{70}$$

where $\boldsymbol{\mathcal{K}} \in \mathcal{R}^{N \times N}$ is the Gram matrix whose (k, h)-th entry is $\boldsymbol{\mathcal{K}}_{k,h} = \kappa(\boldsymbol{\eta}_k, \boldsymbol{\eta}_h)$, with $\boldsymbol{\eta}_k$ the k-th entry of $\boldsymbol{\eta}$, and $\boldsymbol{\kappa}(\eta) \in \mathbb{R}^N$ is the kernel vector whose k-th component is $\boldsymbol{\kappa}_k(\eta) = \kappa(\eta, \boldsymbol{\eta}_k)$.

Remark 7.5.2. *The assumption of measurements perturbed by Gaussian noise is commonly used in learning-based control since it is caused, for example, by numerical differentiation (see [146]).*

From now on we suppose that the following standing assumptions hold (see [17], [86])

Assumption 7.5.1. *The unknown function γ has a bounded norm in the RKHS $\mathcal{H}$ generated to the kernel κ.*

Assumption 7.5.2. *The kernel function κ is* isotropic *and Lipschitz continuous with constant L_κ, with a locally Lipschitz derivative of constant $L_{d\kappa}$.*

Isotropic kernels are functions depending only on the Euclidean distance of their arguments. In this respect, the compact notation $\kappa(x, x') = \kappa(\|x - x'\|)$ is commonly used.

Although any kernel fulfilling Assumption 7.5.2 can be a valid candidate, in the following, we exploit the commonly adopted squared exponential kernel as prior covariance function, which can be expressed as

$$\kappa(\eta, \eta') = \sigma_p^2 \exp\left(-(\eta - \eta')^\top \Lambda^{-1} (\eta - \eta')\right) \tag{71}$$

for all $\eta, \eta' \in \mathbb{R}^{n_\eta}$, where $\Lambda = \text{diag}(2\lambda_{\eta_1}^2, \dots, 2\lambda_{\eta_{n_\eta}}^2)$, $\lambda_{\eta_i} \in \mathbb{R}_{>0}$ is the characteristic length scale relative to the i-th signal, and σ_p^2 is the amplitude [127].

Remark 7.5.3. *Assumption 7.5.2 is asking some Lipschitz continuity property of the unknown function that makes it well-representable by means of a Gaussian process prior. Nevertheless, it represents a very strong assumption, difficult to be checked even if the unknown function is known. Assumption 7.5.2 can be relaxed to the condition that γ is a sample from the Gaussian process $\mathcal{GP}(0, \kappa(\cdot, \cdot))$, which, in turn, leads to a larger pool of posssible unkown functions and it is easier to be check. As an example, the pool generated by the squared exponenial kernel Equation (71) is equal to the space of continuous functions.*

Remark 7.5.4. *The isotropic kernel structure is a customary (although not necessary) assumption in the literature of Gaussian process regression. In this respect, the following results can be generalized for any Lipschitz continuous kernel by means of well-known arguments (see [86]).*

We conclude this section by recalling two results based on [86].

Lemma 7.5.1. *Consider a zero-mean Gaussian process defined through a kernel $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$, satisfying Assumption 7.5.2 on a compact subset $\mathcal{X}$ of $\mathbb{R}^{n_\eta}$, and $N \in \mathbb{N}$ observations $\mathcal{DS} = \{(x^1, y^1), \dots, (x^N, y^N)\}$, with $y^i = f(x^i) + \varepsilon^i$, where $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2 I_{n_y})$. Then, the posterior variance is bounded as*

$$\sigma^2(x) \leq \kappa(0) - \frac{\kappa(\rho)^2}{\kappa(0) + \frac{\sigma_n^2}{|\mathcal{B}_\rho(x)|}} \qquad \forall x \in \mathcal{X},$$

where $\mathcal{B}_\rho(x) = \{x' \in \mathcal{DS} : \|x - x'\| \leq \rho\}$ denotes the training data-set restricted to a ball around x with radius $\rho \in \mathbb{R}_{>0}$, and $|\cdot|$ denotes the cardinality.

Lemma 7.5.2. *Consider a zero-mean Gaussian process defined through a kernel $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, satisfying Assumption 7.5.2 on the compact set $\mathcal{X}$. Furthermore, consider a continuous unknown function $f : \mathcal{X} \mapsto \mathbb{R}$ with Lipschitz constant L_f, and $N \in \mathbb{N}$ observations $y^i = f(x^i) + \varepsilon^i$,*

with $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2 I_{nn_y})$. *Then, there exists* $\rho \in \mathbb{R}_{>0}$ *such that the posterior mean* μ *and posterior variance* σ^2 *conditioned on the training data* $\mathcal{DS} = \{(x^1, y^1), \ldots, (x^N, y^N)\}$ *are continuous with Lipschitz constants* L_μ *and* L_{σ^2} *on* $\mathcal{X}$, *respectively, satisfying*

$$L_\mu \leq L_\kappa \sqrt{N} \left\| \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{y} \right\|,$$
$$L_{\sigma^2} \leq 2\rho L_\kappa \left(1 + N \left\| \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \right\| \max_{x,x' \in \mathcal{X}} \kappa(x, x') \right),$$

with $\boldsymbol{x} = (x^1, \ldots, x^N)^\top$ *and* $\boldsymbol{y} = (y^1, \ldots, y^N)^\top$. *Moreover, pick* $\delta \in (0,1)$ *and set*

$$\beta(\rho) = 2 \log\left(\frac{M(\rho, \mathcal{X})}{\delta}\right),$$
$$\alpha(\rho) = (L_f + L_\mu)\rho + \sqrt{\beta(\rho) L_{\sigma^2} \rho},$$

with $M(\rho, \mathcal{X})$ *the* ρ*-covering number related to the set* $\mathcal{X}$. *Then, the bound*

$$|f(x) - \mu(x)| \leq \sqrt{\beta(\rho)}\sigma^2(x) + \alpha(\rho) \qquad \forall x \in \mathcal{X}$$

holds with probability al least $1 - \delta$.

The ρ*-covering number related to the set* $\mathcal{X}$ *is the minimum number satisfying* $\min_{x \in \mathcal{X}} \max_{x' \in \mathcal{DS}} \|x - x'\| \leq \rho$.

7.5.2 *The Proposed Regulator*

The proposed regulator reads as follows

$$\begin{cases} \dot{\zeta} = 1 \\ \dot{\eta} = F\eta + Gu \\ \dot{\hat{e}} = A\hat{e} + B(\hat{\chi} + \boldsymbol{b}u) + \Lambda(l) H (y - \hat{e}_1) \\ \dot{\hat{\chi}} = -\boldsymbol{b}(\dot{\mu}(\eta, \varsigma, \vartheta)) + l^{r+1} H_{r+1} (y - \hat{e}_1) \\ \dot{\varsigma} = 0 \end{cases}$$
$$(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in \mathcal{C},$$
$$\begin{cases} \zeta^+ = 0 \\ \eta^+ = \eta \\ \hat{e}^+ = \hat{e} \\ \hat{\chi}^+ = \hat{\chi} \\ \varsigma^+ = (S \otimes I_N)\varsigma + (B \otimes I_N) \begin{bmatrix} \eta & u \end{bmatrix}^\top \end{cases} \tag{72}$$
$$(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in \mathcal{D},$$

with $\vartheta = \omega(\varsigma)$ and output $u = \boldsymbol{b}^{-1}\text{sat}(-\hat{\chi} + \kappa_s(\hat{e}))$. Where A, B, and $\boldsymbol{b}$ are the same as in (68) and Assumption 7.4.3, F, G, and n_η are the same as Lemma 7.4.1, and $\Lambda(l)$, H are defined as in Section 7.4

with $l \in \mathbb{R}_{>0}$ a free control parameter fixed later to a sufficiently large number, while the matrices $S \in \mathbb{R}^{N(n_\eta+n_y)\times N(n_\eta+n_y)}$ and $B \in \mathbb{R}^{N(n_\eta+n_y)}$ have the shift form, denoting $n_\varsigma = N(n_\eta + n_y)$

$$S = \begin{pmatrix} 0_{(n_\varsigma-1)\times 1} & I_{n_\varsigma-1} \\ 0 & 0_{1\times(n_\varsigma-1)} \end{pmatrix}, \quad B = \begin{pmatrix} 0_{(n_\varsigma-1)\times 1} \\ 1 \end{pmatrix}.$$

The flow and jump set are defined as $\mathcal{C} = \left\{(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in \mathbb{R}_{>0} \times \mathbb{R}^{n_\eta+n_e+n_y} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_y} : 0 \leq \zeta \leq \overline{\mathrm{T}}, \sigma^2(\eta, \varsigma, \vartheta) \leq \sigma^2_{\text{thr}}\right\}$ and $\mathcal{D} = \left\{(\zeta, \eta, \hat{e}, \hat{\chi}, \varsigma, \vartheta, y) \in \mathbb{R}_{>0} \times \mathbb{R}^{n_\eta+n_e+n_y} \times \mathcal{S} \times \Theta \times \mathbb{R}^{n_y} : \underline{\mathrm{T}} \leq \zeta \leq \overline{\mathrm{T}}, \sigma^2(\eta, \varsigma, \vartheta) \geq \sigma^2_{\text{thr}}\right\}$ respectively, where $\mathcal{S} \subset \mathbb{R}^{N(n_\eta+n_y)}$, $\Theta \subset \mathbb{R}^N$ with $N \in \mathbb{N}_{>0}$, and $\underline{\mathrm{T}}, \overline{\mathrm{T}}, \sigma^2_{\text{thr}} \in \mathbb{R}_{>0}$ satisfying $\underline{\mathrm{T}} \leq \overline{\mathrm{T}}$ and $\sigma_p^2\sigma_n^2(\sigma_p^2 + \sigma_n^2)^{-1} < \sigma^2_{\text{thr}} \leq \sigma_p^2$. The functions $\mu(\eta, \varsigma, \vartheta)$ and $\sigma^2(\eta, \varsigma, \vartheta)$ are the a posteriori GP estimate mean and variance, respectively, after the collection of N samples. According to Section 7.5.1, denoting $\varsigma = (\varsigma_\eta, \varsigma_u)^\top$, the latter functions read to

$$\begin{aligned} \mu(\eta, \varsigma, \vartheta) &= \boldsymbol{\kappa}(\eta)^\top \vartheta, \\ \sigma^2(\eta, \varsigma, \vartheta) &= \kappa(\eta, \eta) - \boldsymbol{\kappa}(\eta)^\top \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \boldsymbol{\kappa}(\eta) \end{aligned}$$

with $\vartheta = \left(\boldsymbol{\mathcal{K}} + \sigma_n^2 I_N\right)^{-1} \varsigma_u$. In this settings, $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{\kappa}$ are evaluated with respect to the data-set $\mathcal{DS}$ as defined in Section 7.5.1.

Claim 7.5.1. *Let Assumption 7.5.1 and 7.3.3 hold, then the tuple $(\mathcal{S}, \mu, \omega)$ satisfies the identifier requirements relative to the functional*

$$\mathcal{J} = \frac{\lambda_s}{2} \|\mu\|^2_{\mathcal{H}} + Q(\boldsymbol{u}, \mu(\boldsymbol{\eta})).$$

Claim 7.5.2. *Let Assumption 7.4.1-7.5.2 hold and consider the regulator (72), then for each compact sets Z_0, E_0, and S_0 there exists $\alpha_x > 0$ and $l^\star_\epsilon > 0$, and for any choice of $\underline{\mathrm{T}}, \sigma^2_{thr} \in \mathbb{R}_{>0}$, and $N \in \mathbb{N}_{>0}$, and for each initial condition $w_0 \in \mathcal{W}$, a $\rho^\star(w_0) > 0$ such that if $l > l^\star_\epsilon$, then the bound*

$$\limsup_{t\to\infty} |y(t)| \leq \alpha_x \left| \sqrt{\beta} \left[\kappa(0) - \frac{\kappa(\rho^\star)^2}{\kappa(0) + \sigma_n^2} \right] + \alpha(\rho^\star) \right|$$

with β and α defined as

$$\beta = 2\log\left(\tfrac{N}{\delta}\right), \quad \alpha(\rho^\star) = \left(L_f + L_\mu\right)\rho^\star + \sqrt{\beta L_{\sigma^2}\rho^\star},$$

holds with probability at least $1 - \delta$.

Remark 7.5.5. *The quantity $\rho^\star$ in Claim 7.5.2 represents a notion of coverage of the set $\mathcal{E}$ by the collected data-set. In particular, the lower $\rho^\star$ is, the better the set $\mathcal{E}$ is covered. As long as it approaches to zero, the regulation error approaches the lower bound*

$$\limsup_{t\to\infty} |y(t)| \leq \alpha_x \left| \sqrt{\beta} \left[\frac{\sigma_n^2}{\kappa(0) + \sigma_n^2} \right] \right|,$$

driven by the measurement noise σ_n^2.

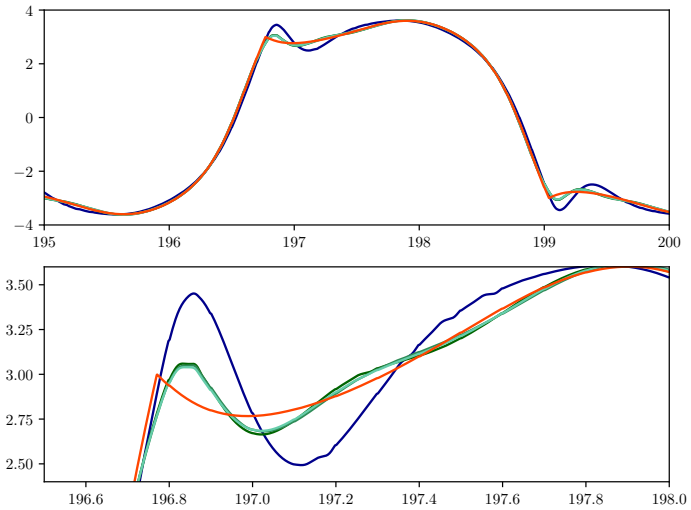


Figure 38: **Top:** Value of the real feedforward term $u^\star(w(t))$ (orange line) and of its approximations $\hat{\gamma}(\eta(t))$, along the system trajectory. The blue line shows the steady-state friend provied by the linear identifier, while the green lines report the gaussian-based identifier estimation with $N = 50$ (dark green line), $N = 100$ (green line), and $N = 200$ (light green line). **Bottom:** zoom-in to highlight the difference in the case of gaussian-based identifier with different number of samples. The reported quantities are plotted with respect to the time in seconds (abscissa).

7.5.3 *Numerical Simulation*

To test the proposed regulator performances against state-of-the-art output regulation solutions, we consider the same problem proposed by [13] where the output of a Van der Pol oscillator, with unknown parameter, must be synchronized with a triangular wave with unknown frequency. The forced Van der Pol oscillator is described by the following equations

$$\begin{aligned} \dot{\chi}_1 &= \chi_2, \\ \dot{\chi}_2 &= -\chi_1 + a\left(1 - \chi_1^2\right)\chi_2 + u, \end{aligned} \tag{73}$$

with a scalar unknown parameter regulating the system damping. Furthermore, a triangular wave can be generated by an exosystem of the form

$$\dot{w}_1 = w_2, \quad \dot{w}_2 = -\varrho w_1,$$

with output

$$\chi^\star(w) = \frac{2\sqrt{w_1^2 + w_2^2}}{\pi} \arcsin\left(\frac{w_1}{\sqrt{w_1^2 + w_2^2}}\right),$$

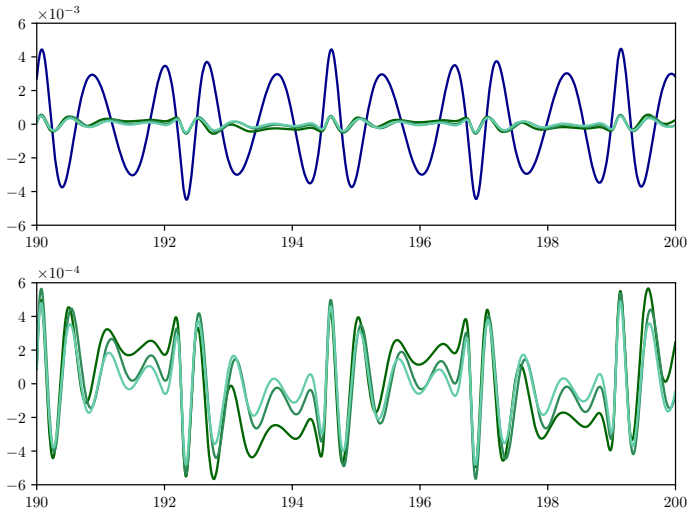


Figure 39: **Top:** steady-state evolution of the tracking error $y(t)$ in the two cases obtained by employing a linear identifier (blue line), and a gaussian-based identifier with $N = 50$ (dark green line), $N = 100$ (green line), and $N = 200$ (light green line). **Bottom:** zoom-in to highlight the error behavior. The reported quantities are plotted with respect to the time in seconds (abscissa).

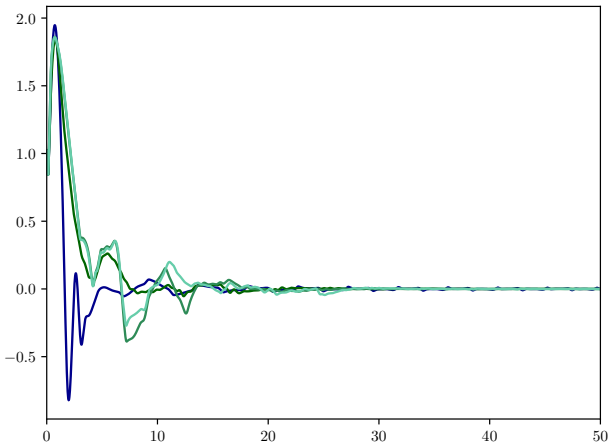


Figure 40: The transient evolution of the tracking error $y(t)$ in the two cases obtained by employing a linear identifier (blue line), and a gaussian-based identifier with $N = 50$ (dark green line), $N = 100$ (green line), and $N = 200$ (light green line). The reported quantities are plotted with respect to the time in seconds (abscissa).

with scalar parameter ϱ the unknown oscillating frequency. The goal is to steer the output χ_1 of (73) to the reference $\chi^\star(w)$. The error coordinates e are thus defined as

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} \chi_1 - \chi^\star(w) \\ \chi_2 - L_{s(w)}\chi^\star(w) \end{pmatrix},$$

and the error system reads as

$$\begin{aligned}\dot{e}_1 &= e_2,\\ \dot{e}_2 &= -e_1 - \chi^\star - L^2_{s(w)}\chi^\star \\ &\quad + a\left(1 - (e_1 + \chi^\star(w))^2\right)\left(e_2 + L_{s(w)}\chi^\star(w)\right).\end{aligned} \tag{74}$$

The system (74) is in the same form of (68) with Assumption 7.4.2 trivially fulfilled since the z dynamics is absent. Furthermore, Assumption 7.4.1 and Assumption 7.4.3 hold with $\boldsymbol{b} = 1$ and any $\mu \in (0,1)$. To be compliant with the results presented by [13] we exploit the same controller parameters

1. $\kappa_s(\hat{e}) = K\hat{e}$ with K such that $\sigma(A - BK) = \{-1, -2\}$, and the input u has been saturated inside the interval $[-100, 100]$.
2. The internal model dimension is $n_\eta = 2(n_w + 1) = 6$, and the matrices F and G has been fixed as
$$F = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0\\ 0 & -1 & 1 & 0 & 0 & 0\\ 0 & 0 & -1 & 1 & 0 & 0\\ 0 & 0 & 0 & -1 & 1 & 0\\ 0 & 0 & 0 & 0 & -1 & 1\\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}, \quad G = \begin{pmatrix} 0\\ 0\\ 0\\ 0\\ 0\\ 1 \end{pmatrix}.$$
3. The control parameters has been chosen as $l = 20$, $h_1 = 6$, $h_2 = 11$, $h_3 = 6$, and $\underline{\mathrm{T}} = \overline{\mathrm{T}} = 0.1$.

The simulations reported in Figures 39, 40, and 38 show the proposed regulator applied with $a = \varrho = 2$ in three cases with $N = 50$, $N = 100$, and $N = 200$. The obtained results are then compared with the regulator proposed by [13] where the identifier is chosen as a least-squares identifier working on the model set $\mathcal{M} = \{\psi(\vartheta, \eta) : \mathbb{R}^{n_\eta} \mapsto \mathbb{R}^{n_y} | \psi(\vartheta, \eta) = \vartheta^\top \eta, \vartheta \in \Theta \subset \mathbb{R}^{n_\eta}\}$. In all simulations the GP parameters has been kept fixed at $\sigma_n^2 = 0.01$ and $\sigma_{\text{thr}}^2 = \sigma_p^2 = 1$, while the kernel hyperparameters $\boldsymbol{\lambda} = (\lambda_{\eta_1}, \ldots, \lambda_{\eta_{n_\eta}})$ has been estimated via log-likelihood minimization [127] yielding to the values of $\boldsymbol{\lambda} = (7.7, 34.3, 19.9, 0.4, 133.6, 1.2)$. As emerges from Figure 39, the proposed approach reduces the maximum error of more than 100 times compared to the case with least-square identifier.

7.6 CONTRIBUTIONS

We presented two learning-based techniques to design internal model-based regulators for a large class of nonlinear systems. The proposed techniques fit in the general frameworks recently proposed in [14] and [13], and shows how the identification of the optimal steady state

control input can be performed by using Gaussian process models. The flexibility of the proposed approaches make the regulator able to deal with highly uncertain shapes of the optimal steady-state control input useful to make zero the output error. Thanks to the fact that only coarse and qualitative knowledge about the friend is required, the proposed designs may be employed as solution to many of the output regulation problems addressed in literature. The previous analysis also derives probabilistic bounds on the attained performances and presents numerical simulations showing how the proposed methods outperforms state-of-the-art approaches when the regulated plant or the exogenous disturbances are subject to unmodeled perturbations. Future research directions will be aimed at exploring deeper the Gaussian process flexibility by focusing on the injection of possibly a priori knowledge of the friend structure, and at investigating how the proposed performance bound changes. We also aim to investigate if Gaussian process-based internal models may deal with non minimum-phase systems.

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis we approached the problem of autonomous navigation by smoothly going through a sequence of issues that arise when approaching this field of study for the first time. Although the discussion mainly focused on quadcopter control and planning, the presented algorithms easily extend to a very large class of problems involving several different autonomous platforms, such as ground robots, fixed-wing UAVs, or even underwater robots. We started by reviewing the "golden standards" of quadcopter modeling and control in Chapter 2, representing in some way the baseline from which we started our work, then we moved to the problem of localisation and mapping in Chapter 3, trajectory planning (Chapter 5) and re-planning (Chapter 6), and environments exploration in Chapter 4. We especially focused on the re-planning and exploration problems, where beyond reviewing, developing, and testing state-of-the-art solutions, we contribute by proposing new fast and light approaches, with an eye to the system theory field from which we aim to borrow analysis tools to prove the algorithms stability. The touched problems, as they are reported in the thesis, are exactly the same issues that we faced during the development of a fully autonomous platform able to cope with the requirements proposed by the Leonardo drone contest.

The thesis ends with a window on a completely new *output regulation* control paradigm which although demonstrated very impressive results, it is not been used on real-scenario applications yet. The reason behind that lies mainly in its difficult design and poor generalisation in front of different exosystems and system uncertainties, in this sense we try to steer the research toward completely data-driven designs able to cope with a very large number of model sets. Besides that, the necessity to bound the computed control inputs to real actuator saturations is a fundamental issue that has not been deeply studied yet. The generality of the presented control technique, jointly with the possibility to extend the proposed algorithms to a very large number of different applications and robots motivate the chosen thesis title. The material presented is far from being a complete answer to the problem of motion planning and control of highly nonlinear robots which is definitely an open and challenging research field. As a matter of fact, many research directions are open by the proposed vision.

Part IV

APPENDICES

A

B-SPLINES, NURBS AND BÉZIER CURVES

This appendix briefly collects the central notions and properties of B-Splines, NURBS, and Bézier curves, borrowing the formalism and the framework from [11]. We start with a general overview of the three curve parameterizations mentioned above, then go deep through their properties and correlations. For further details, the reader is referred to [11, 46, 123].

A.1 B-SPLINE CURVES

A *B-spline*, or *Basic-spline*, is a computationally efficient technique to implement interpolating splines or to approximate functions, curves, and surfaces. Any generic spline can be expressed in *B-form* by linearly combining a proper number of B-splines basis functions $B_i^p(u)$,

$$\boldsymbol{s}(u) = \sum_{i=0}^{m} \boldsymbol{q}_i B_i^p(u) \quad \text{with} \quad u_{\min} \leq u \leq u_{\max},$$

where p is the curve order, while the coefficients $\boldsymbol{q}_i \in \mathbb{R}^n$, $i = 0, \ldots, m$, are known as *control points* and can be computed by imposing some approximation or interpolation conditions on a given set of data points. Let $\boldsymbol{u} = \left[u_0, \ldots, u_{p+m+1}\right]$ be a nondecreasing vector of real numbers, in this framework called *knots*, then the jth B-spline basis function of order p can be recursively computed via the De-Boor formula [33]

$$B_i^0(u) = \begin{cases} 1 & \text{if} \quad u_i \leq u < u_{i+1}, \\ 0 & \text{otherwise}, \end{cases}$$

$$B_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} B_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1}^{p-1}(u).$$

An example of B-spline basis functions of order 5 is reported in Figure 41, notice that each basis $B_i^p(u)$ is equal to zero everywhere except in the interval $\left[u_i, u_{i+p+1}\right)$, it results that in every knot span only $p+1$ basis functions are not null, yielding a dependency on only $p+1$ control points. The latter property is usually referred to as *locality property* since the curve is locally bent only by $p+1$ control points, this opens the possibility of deforming the curve without necessarily recomputing the overall B-spline. Moreover, from the aforementioned relation emerges that the basis functions enjoy a partition of the unity property (i. e. $\sum_{i=0}^{m} B_i^p(u) = 1$ for any u), the latter implies that the B-spline curve $\boldsymbol{s}(u)$ is entirely contained inside the convex hull generated by the given control points, being a convex combination of these.

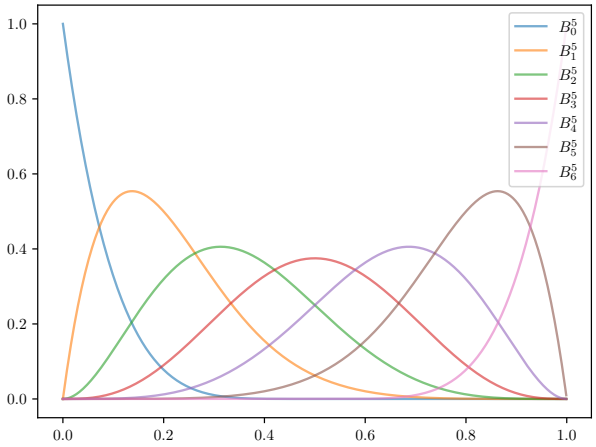


Figure 41: Example of B-spline basis functions of order 5.

A common choice during the design of a B-spline curve is to implement a *non-uniform* knot vector of the form

$$\boldsymbol{u} = \left[u_{\min}, \ldots, u_{\min}, u_{p+1}, \ldots, u_m, u_{\max}, \ldots, u_{\max}\right],$$

this particular form allows for endpoints interpolation, i. e. $\boldsymbol{s}(u_{\min}) = \boldsymbol{q}_0$ and $\boldsymbol{s}(u_{\max}) = \boldsymbol{q}_m$, an example of this type of curve is depicted in Figure 42. Moreover, the non-uniform knots vector allows shaping the endpoints of the k derivate only acting on the first and last $k+1$ control points. A B-spline curve is in fact differentiable infinite times in the interior of the knot intervals, and it is $p-j$ times continuously differentiable at a knot of multiplicity j. The curve derivate can be still represented in B-form with an order $p-1$ and control points

$$\boldsymbol{q}_i' = p\frac{\boldsymbol{q}_{j+1} - \boldsymbol{q}_j}{u_{j+p+1} - u_{j+1}}.$$

A.2 NURBS CURVES

The Non Uniform Rational B-Spline (NURBS) is a generalization of the classic B-spline curves that implements a set of m additional parameters encoded as control points *weight*. A NURBS curve of order p is defined as

$$\boldsymbol{s}(u) = \frac{\sum_{i=0}^{m} \boldsymbol{q}_i w_i B_i^p(u)}{\sum_{i=0}^{m} w_i B_i^p(u)} \quad \text{with} \quad u_{\min} \leq u \leq u_{\max},$$

where $\boldsymbol{q}_i$ is the ith control point, w_i the ith weight and B_i^p the B-spline basis function. An example of a NURBS curve of order 3 is reported in Figure 43, where the weight associated with the 4th control point

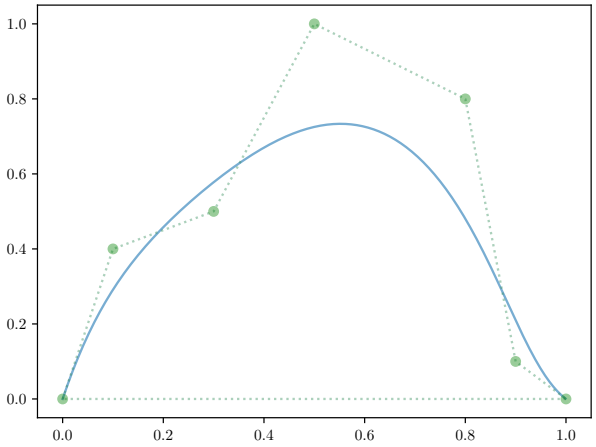


Figure 42: Example of B-spline curve of order 5.

is left free to span the range $(0, 5]$ while all others are fixed at 1, note that the introduce degree of freedom allows to locally shape the curve toward the control point just by increasing the associated weight. In this context, the absolute value of the single weight does not play a role in the final curve shape, which instead is governed by the value of the relative weight. As a matter of fact, if the weights are constant and equal the NURBS degenerate to a B-spline curve. The aforementioned relation can be rewritten in B-form by setting

$$N_i^p(u) = \frac{w_i B_i^p(u)}{\sum_{i=0}^m w_i B_i^p(u)} \quad \Longrightarrow \quad \boldsymbol{s}(u) = \sum_{i=0}^m \boldsymbol{q}_i N_i^p(u),$$

in this setting, N_i^p are piecewise rational functions, called *rational basis functions*. All properties stated for the B-splines hold also for NURBS curves with the only exception that the NURBS derivates cannot be represented in B-form. This limits their application in motion planning due to the poor practical derivates bounds evaluation and implementation.

A.3 RELATION BETWEEN B-SPLINE & NURBS CURVES

Even if NURBS curves can be represented in B-form, which resembles the B-spline notation, the obtained curve is not a *non-rational* B-spline, leading to inefficient representation and evaluation, jointly with the loss of some good properties. An efficient way to represent NURBS curves is based on homogeneous coordinates. In the three-dimensional case, for a given set of control points $\boldsymbol{q}_i = \left[q_{x,j}, q_{y,j}, q_{z,j}\right]^\top$ and weights w_i, it is possible to construct the weighted control points

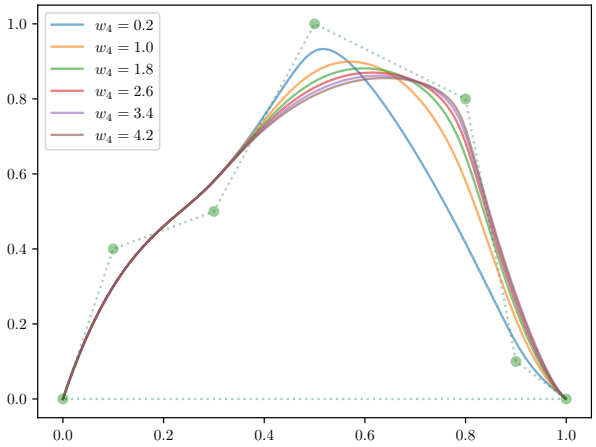


Figure 43: Example of NURBS curve of order 3.

$\boldsymbol{q}_i^w = \left[w_i q_{x,j}, w_i q_{y,j}, w_i q_{z,j}, w_i\right]^\top \in \mathbb{R}^4$, which define the nonrational B-spline

$$\boldsymbol{s}^w(u) = \sum_{i=0}^{m} \boldsymbol{q}_i^w B_i^p(u).$$

The latter representation and the NURBS curve formulation are equivalent in the sense that they are related to a bijective map.

A.4 BÉZIER CURVES

While NURBSs are a generalization of B-splines, Bézier curves represent a specialization of them. In particular, a Bézier curve can be obtained from a B-spline by setting the control points number equal to the curve order (i. e. $m = p$) and letting the knots vector degenerate to

$$\boldsymbol{u} = \left[u_{\min}^0, \ldots, u_{\min}^p, u_{\max}^0, \ldots, u_{\max}^p\right].$$

Usually, in this framework, the spline variable bounds are chosen as $u_{\min} = 0$ and $u_{\max} = 1$. The obtained curve is still in B-form, but allows for a more efficient evaluation

$$\boldsymbol{s}(u) = \textstyle\sum_{i=0}^{m} \boldsymbol{q}_i B_i^p(u) \quad \text{with} \quad 0 \leq u \leq 1, \tag{75}$$

where the coefficients $\boldsymbol{q}_i$ are the control points, and the basis functions $B_i^p(u)$ are pth order *Bernstein* polynomials defined by

$$B_i^p(u) = \binom{p}{i} u^i (1-u)^{p-i}.$$

In this context, the possibility to evaluate the curve without a recursive formula allows for a more efficient implementation. The Bézier curves, unlike B-splines and NURBSs, enjoy a set of new properties [46], summarized in the following.

1. *Derivate*. The Bézier space is closed with respect to the operation of derivation, as a matter of fact, the basis function satisfies

$$\frac{d}{du}B_i^p(u) = p\left(B_{i-1}^{p-1}(u) - B_i^{p-1}(u)\right),$$

that, jointly with Equation (75), yields to

$$\frac{d}{du}\boldsymbol{s}(u) = \sum_{i=0}^{m-1} p\left(\boldsymbol{q}_{i+1} - \boldsymbol{q}_i\right) B_i^{p-1}(u),$$

which in turn still represents a Bézier curve of lower degree.

2. *Arithmetic operations*. Two curves can be added, subtracted, or multiplied in a new Bézier curve by acting only on the respective control points. In particular, as regards addition and subtraction, the operation can be done by simply adding or subtracting the control points element-wise. If the two curves have different degrees, the lowest must be elevated to match the other one (see next properties). In case of product, two Bézier curves with coefficients $\boldsymbol{q}_0^1, \ldots, \boldsymbol{q}_{m_1}^1$ and $\boldsymbol{q}_0^2, \ldots, \boldsymbol{q}_{m_2}^2$, can be combined in a new curve with control points

$$\boldsymbol{q}_i^3 = \sum_{j=\max(0,i-m_2)}^{\min(m_1,i)} \frac{\binom{m_1}{j}\binom{m_2}{i-j}}{\binom{m_1+m_2}{i}} \boldsymbol{q}_j^1 \boldsymbol{q}_{i-j}^2.$$

3. *Curve composition*. Two curves $s(u)$ and $u(\bar{u})$ with Bernstein coefficients $\boldsymbol{q}_0^s, \ldots, \boldsymbol{q}_{m_s}^s$ and $q_0^u, \ldots, q_{m_u}^u$ can be composed in a new Bézier curve $\bar{s}(\bar{u}) = s(u(\bar{u}))$ via the recoursive formula

$$\lambda_{i,j}^k = \binom{km_u}{j}^{-1} \sum_{l=\max(0,j-m_u)}^{\min(j,km_u-m_u)} \binom{km_u - m_u}{l}\binom{m_u}{j-l} \left[\left(1 - q_{j-l}^u\right)\lambda_{i,l}^{k-1} + q_{j-l}^u \lambda_{i+1,l}^{k-1}\right]$$

for $k = 1, \ldots, m_s$, $i = 0, \ldots, m_s - k$, and $j = 0, \ldots, km_u$, and by setting $\lambda_{i,0}^0 = \boldsymbol{q}_i^s$. Finally the Bernstein coefficients of the composed curve are specified by

$$\boldsymbol{q}_i^{\bar{s}} = \lambda_{0,j}^{m_s}, \quad j = 0, \ldots, m_s m_u$$

4. *Degree elevation*. A curve of order p and with coefficients $\boldsymbol{q}_0, \ldots, \boldsymbol{q}_m$ can be expressed in the Bernstein basis of degree $p + r$, for all $r > 0$, as $\boldsymbol{s}^{p+r}(u) = \sum_{i=0}^{m+r} \boldsymbol{q}_i^{p+r} B_i^{p+r}(u)$ where the coefficients can be computed as

$$\boldsymbol{q}_i^{p+r} = \sum_{j=\max(0,i-r)}^{\min(m,k)} \frac{\binom{r}{i-j}\binom{m}{j}}{\binom{m+r}{i}} \boldsymbol{q}_j$$

5. *Scaling the independent variable*. The change of variable $u \mapsto \lambda u$ maps the interval $[0, 1]$ to $[0, \lambda]$, without changing the curve path, and scaling the kth derivate of λ^{-k}.

The latter properties are not completely for free, as a matter of fact, the Béziez curve requires a higher number of parameters to express the same spline with respect to B-splines or NURBs. On the other hand, the Béziez parameterization leads to a less conservative containment property, since the obtained convex hull is tighter over the curve itself [140]. Notice how going from general formulations (NURBS) to specialized forms (B-spline, Bézier), the number of properties increases a lot, but loses curve flexibility and degrees of freedom.

A.5 RELATION BETWEEN B-SPLINE & BÉZIER CURVES

The relation linking B-spline and Bézier curves is purely algebraic and goes through their matrix representation. To give an insight to the reader, both parameterizations can be represented in a form

$$\boldsymbol{s}(u) = \begin{bmatrix} 1 & u & u^2 & \cdots & u^p \end{bmatrix}^\top \boldsymbol{B} \begin{bmatrix} \boldsymbol{q}_0 & \boldsymbol{q}_1 & \cdots & \boldsymbol{q}_m \end{bmatrix}^\top,$$

with $\boldsymbol{B}$ being different depending on if $\boldsymbol{s}(u)$ represents a B-spline or a Bézier curve. Let $\boldsymbol{B}_{\text{BS}}$ and $\boldsymbol{B}_{\text{BC}}$ the two matrices representing a B-spline and a Bézier curve, respectively, moreover let $\boldsymbol{q}_{\text{BS}}$ and $\boldsymbol{q}_{\text{BC}}$ the associated control points, then the following relation holds

$$\begin{bmatrix} \boldsymbol{q}_{\text{BS}_0} & \boldsymbol{q}_{\text{BS}_1} & \cdots & \boldsymbol{q}_{\text{BS}_m} \end{bmatrix}^\top = \boldsymbol{B}_{\text{BS}}^{-1} \boldsymbol{B}_{\text{BC}} \begin{bmatrix} \boldsymbol{q}_{\text{BC}_0} & \boldsymbol{q}_{\text{BC}_1} & \cdots & \boldsymbol{q}_{\text{BC}_m} \end{bmatrix}^\top.$$

For further information about the construction of matrices $\boldsymbol{B}_{\text{BS}}$ and $\boldsymbol{B}_{\text{BC}}$, the reader is referred to [123].

BIBLIOGRAPHY

[7] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. "Formal synthesis of Lyapunov neural networks." In: *IEEE Control Systems Letters* 5.3 (2020), pp. 773–778.

[8] S. Agarwal, K. Mierle, and The Ceres Solver Team. *Ceres Solver*. Version 2.1. Mar. 2022. URL: https://github.com/ceres-solver/ceres-solver.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. "Control barrier function based quadratic programs for safety critical systems." In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[10] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt. "Multi-robot coordination for estimation and coverage of unknown spatial fields." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 7740–7746.

[11] L. Biagiotti and C. Melchiorri. *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[12] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi. "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs." In: *Journal of Intelligent & Robotic Systems* 100 (2020), pp. 1213–1247.

[13] M. Bin, P. Bernard, and L. Marconi. "Approximate nonlinear regulation via identification-based adaptive internal models." In: *IEEE Transactions on Automatic Control* 66.8 (2020), pp. 3534–3549.

[14] M. Bin and L. Marconi. ""Class-Type" Identification-Based Internal Models in Multivariable Nonlinear Output Regulation." In: *IEEE Transactions on Automatic Control* 65.10 (2019), pp. 4369–4376.

[15] M. Bin and L. Marconi. "Model identification and adaptive state observation for a class of nonlinear systems." In: *IEEE Transactions on Automatic Control* 66.12 (2020), pp. 5621–5636.

[16] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. "Receding horizon" next-best-view" planner for 3d exploration." In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1462–1468.

[17] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio. "Joint state and dynamics estimation with high-gain observers and Gaussian process models." In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 4027–4032.

[18] C. I Byrnes and A. Isidori. "Limit sets, zero dynamics, and internal models in the problem of nonlinear output regulation." In: *IEEE Transactions on Automatic Control* 48.10 (2003), pp. 1712–1723.

[19] F. Caballero and L. Merino. "DLL: Direct LIDAR Localization. A map-based localization approach for aerial robots." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 5491–5498.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam." In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[21] F. Castañeda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath. "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics." In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 3683–3690.

[22] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. "Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping." In: *Robotics: Science and Systems*. Vol. 11. Rome. 2015, pp. 3–12.

[23] J. Chen, K. Su, and S. Shen. "Real-time safe trajectory generation for quadrotor flight in cluttered environments." In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2015, pp. 1678–1685.

[24] Y. Chen and G. Medioni. "Object modelling by registration of multiple range images." In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[25] R. Choe, J. Puig, V. Cichella, E. Xargay, and N. Hovakimyan. "Trajectory generation using spatial pythagorean hodograph bézier curves." In: *AIAA Guidance, Navigation, and Control Conference*. 2015, p. 0597.

[26] C. Choy, J. Park, and V. Koltun. "Fully convolutional geometric features." In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 8958–8966.

[27] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. "Rapid exploration with multi-rotors: A frontier selection method for high speed flight." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 2135–2142.

[28] C. Connolly. "The determination of next best views." In: *Proceedings. 1985 IEEE international conference on robotics and automation*. Vol. 2. IEEE. 1985, pp. 432–435.

[29] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M. F. F. Balcan, and L. Song. "Scalable kernel methods via doubly stochastic gradients." In: *Advances in neural information processing systems* 27 (2014).

[30] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis. "Graph-based path planning for autonomous robotic exploration in subterranean environments." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 3105–3112.

[31] T. Dang, C. Papachristos, and K. Alexis. "Autonomous exploration and simultaneous object search using aerial robots." In: *2018 IEEE Aerospace Conference*. IEEE. 2018, pp. 1–7.

[32] T. Dang, C. Papachristos, and K. Alexis. "Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2526–2533.

[33] C. De Boor. *A practical guide to splines*. Vol. 27. springer-verlag New York, 1978.

[34] D. Deng, R. Duan, J. Liu, K. Sheng, and K. Shimada. "Robotic Exploration of Unknown 2D Environment Using a Frontier-based Automatic-Differentiable Information Gain Measure." In: *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2020, pp. 1497–1503.

[35] W. Ding, W. Gao, K. Wang, and S. Shen. "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7595–7602.

[36] W. Ding, W. Gao, K. Wang, and S. Shen. "An efficient b-spline-based kinodynamic replanning framework for quadrotors." In: *IEEE Transactions on Robotics* 35.6 (2019), pp. 1287–1306.

[37] W. Dong, G. Y. Gu, X. Zhu, H. Ding, et al. "Modeling and control of a quadrotor UAV with aerodynamic concepts." In: *World Academy of Science, Engineering and Technology* 7.5 (2013), pp. 901–906.

[38] R. Dubé, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto. "Incremental-segment-based localization in 3-d point clouds." In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1832–1839.

[39] M. Elhousni and X. Huang. "A survey on 3d lidar localization for autonomous vehicles." In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 1879–1884.

[40] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration." In: *Journal of Software Engineering for Robotics* 3.1 (2012), pp. 2–12.

[41] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer. 2014, pp. 834–849.

[42] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories." In: *IEEE Robotics and Automation Letters* 3.2 (2017), pp. 620–626.

[43] J. Faigl and P. Váňa. "Surveillance planning with Bézier curves." In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 750–757.

[44] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. "PAMPC: Perception-aware model predictive control for quadrotors." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–8.

[45] D. Falanga, K. Kleber, and D. Scaramuzza. "Dynamic obstacle avoidance for quadrotors with event cameras." In: *Science Robotics* 5.40 (2020), eaaz9712.

[46] R. T. Farouki. "The Bernstein polynomial basis: A centennial retrospective." In: *Computer Aided Geometric Design* 29.6 (2012), pp. 379–419.

[47] D. Floreano and R. J. Wood. "Science, technology and the future of small autonomous drones." In: *nature* 521.7553 (2015), pp. 460–466.

[48] P. Foehn, A. Romero, and D. Scaramuzza. "Time-optimal planning for quadrotor waypoint flight." In: *Science Robotics* 6.56 (2021), eabh1221.

[49] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 15–22.

[50] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. "RotorS—A modular gazebo MAV simulator framework." In: *Robot operating system (ROS)*. Springer, 2016, pp. 595–625.

[51] N. Gaby, F. Zhang, and X. Ye. "Lyapunov-net: A deep neural network architecture for Lyapunov function approximation." In: *arXiv preprint arXiv:2109.13359* (2021).

[52] F. Gao, Y. Lin, and S. Shen. "Gradient-based online safe trajectory generation for quadrotor flight in complex environments." In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 3681–3688.

[53] F. Gao and S. Shen. "Online quadrotor trajectory generation and autonomous navigation on point clouds." In: *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2016, pp. 139–146.

[54] F. Gao, W. Wu, W. Gao, and S. Shen. "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments." In: *Journal of Field Robotics* 36.4 (2019), pp. 710–733.

[55] F. Gao, W. Wu, Y. Lin, and S. Shen. "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 344–351.

[56] J. Gardner, G. Pleiss, R. Wu, K. Weinberger, and A. Wilson. "Product kernel interpolation for scalable Gaussian processes." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1407–1416.

[57] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. "Automatic generation and detection of highly reliable fiducial markers under occlusion." In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[58] R. Gomez-Ojeda, F. A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez. "PL-SLAM: A stereo SLAM system through the combination of points and line segments." In: *IEEE Transactions on Robotics* 35.3 (2019), pp. 734–746.

[59] H. H. González-Banos and J. C. Latombe. "Navigation strategies for exploring indoor environments." In: *The International Journal of Robotics Research* 21.10-11 (2002), pp. 829–848.

[60] S. Granger and X. Pennec. "Multi-scale EM-ICP: A fast and robust approach for surface registration." In: *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part IV 7*. Springer. 2002, pp. 418–432.

[61] M. A. Graule, P. Chirarattananon, S. B. Fuller, N. T. Jafferis, K. Y. Ma, M. Spenko, R. Kornbluh, and R. J. Wood. "Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion." In: *Science* 352.6288 (2016), pp. 978–982.

[62] L. Han, F. Gao, B. Zhou, and S. Shen. "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4423–4430.

[63] J. P. Hespanha, D. Liberzon, and A. R. Teel. "On input-to-state stability of impulsive systems." In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE. 2005, pp. 3992–3997.

[64] J. P. Hespanha and A. S. Morse. "Stability of switched systems with average dwell-time." In: *Proceedings of the 38th IEEE conference on decision and control (Cat. No. 99CH36304)*. Vol. 3. IEEE. 1999, pp. 2655–2660.

[65] T. Holsclaw, B. Sansó, H. K. H. Lee, K. Heitmann, S. Habib, D. Higdon, and U. Alam. "Gaussian process modeling of derivative curves." In: *Technometrics* 55.1 (2013), pp. 57–67.

[66] H. Hong and B. H. Lee. "Probabilistic normal distributions transform representation for accurate 3D point cloud registration." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3333–3338.

[67] A. Hornung, Kai M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[68] S. Hrabar. "Reactive obstacle avoidance for rotorcraft uavs." In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 4967–4974.

[69] S. C. Hsu, X. Xu, and A. D. Ames. "Control barrier function based quadratic programs with application to bipedal robotic walking." In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 4542–4548.

[70] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K. T. Cheng. "Act: An autonomous drone cinematography system for action scenes." In: *2018 ieee international conference on robotics and automation (icra)*. IEEE. 2018, pp. 7039–7046.

[71] A. Isidori. *Lectures in feedback design for multivariable systems*. Springer, 2017.

[72] A. Isidori, L. Marconi, and A. Serrani. *Robust autonomous guidance: an internal model approach*. Springer Science & Business Media, 2003.

[73] M. W. Jones, J. A. Baerentzen, and M. Sramek. "3D distance fields: A survey of techniques and applications." In: *IEEE Transactions on visualization and Computer Graphics* 12.4 (2006), pp. 581–599.

[74] M. Juliá, A. Gil, and O. Reinoso. "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments." In: *Autonomous Robots* 33.4 (2012), pp. 427–444.

[75] J. M. Kai, G. Allibert, M. D. Hua, and T. Hamel. "Nonlinear feedback control of quadrotors exploiting first-order drag effects." In: *IFAC-PapersOnLine* 50.1 (2017), pp. 8189–8195.

[76] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. "STOMP: Stochastic trajectory optimization for motion planning." In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[77] M. Khan, T. Ibuki, and A. Chatterjee. "Safety uncertainty in control barrier functions using gaussian processes." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6003–6009.

[78] M. Khan, T. Ibuki, and A. Chatterjee. "Gaussian Control Barrier Functions: A Non-Parametric Paradigm to Safety." In: *arXiv preprint arXiv:2203.15474* (2022).

[79] S. Khatoon, D. Gupta, and L. Das. "PID & LQR control for a quadrotor: Modeling and simulation." In: *2014 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2014, pp. 796–802.

[80] G. Kim, S. Choi, and A. Kim. "Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments." In: *IEEE Transactions on Robotics* 38.3 (2021), pp. 1856–1874.

[81] S. Kim and J. Kim. "Continuous occupancy maps using overlapping local gaussian processes." In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 4709–4714.

[82] Y. Kompis, L. Bartolomei, R. M. Palliser, L. Teixeira, and M. Chli. "Informed Sampling Exploration Path Planner for 3D Reconstruction of Large Scenes." In: *IEEE Robotics and Automation Letters* (2021).

[83] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui. "Micro air vehicle link (mavlink) in a nutshell: A survey." In: *IEEE Access* 7 (2019), pp. 87658–87680.

[84] S. M. LaValle. "Rapidly-exploring random trees: A new tool for path planning." In: (1998).

[85] S. Lagüela, L. Díaz, D. Roca, H. Lorenzo, et al. "Aerial thermography from low-cost UAV for the generation of thermographic digital terrain models." In: *Opto-Electronics Review* 23.1 (2015), pp. 78–84.

[86] A. Lederer, J. Umlauft, and S. Hirche. "Uniform error and posterior variance bounds for Gaussian process regression with application to safe control." In: *arXiv preprint arXiv:2101.05328* (2021).

[87] T. Lee, M. Leok, and N. H. McClamroch. "Geometric tracking control of a quadrotor UAV on SE (3)." In: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, pp. 5420–5425.

[88] H. Liu, Q. Ye, H. Wang, L. Chen, and J. Yang. "A precise and robust segmentation-based lidar localization system for automated urban driving." In: *Remote Sensing* 11.11 (2019), p. 1348.

[89] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar. "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments." In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695.

[90] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. "Learning high-speed flight in the wild." In: *Science Robotics* 6.59 (2021), eabg5810.

[91] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza. "Dronet: Learning to fly by driving." In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1088–1095.

[92] T. Madani and A. Benallegue. "Backstepping control for a quadrotor helicopter." In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3255–3260.

[93] M. Magnusson, A. Lilienthal, and T. Duckett. "Scan registration for autonomous mining vehicles using 3D-NDT." In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.

[94] L. Marconi, L. Praly, and A. Isidori. "Output stabilization via nonlinear Luenberger observers." In: *SIAM Journal on Control and Optimization* 45.6 (2007), pp. 2277–2298.

[95] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. "The office marathon: Robust navigation in an indoor office environment." In: *2010 IEEE international conference on robotics and automation*. IEEE. 2010, pp. 300–307.

[96] J. Maver and R. Bajcsy. "Occlusions as a guide for planning the next view." In: *IEEE transactions on pattern analysis and machine intelligence* 15.5 (1993), pp. 417–433.

[97] A. A. Meera, M. Popović, A. Millane, and R. Siegwart. "Obstacle-aware adaptive informative path planning for uav-based target search." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 718–724.

[98] S. B. Mehdi, R. Choe, V. Cichella, and N. Hovakimyan. "Collision avoidance through path replanning using Bézier curves." In: *AIAA Guidance, navigation, and control conference*. 2015, p. 0598.

[99] S. B. Mehdi, R. Choe, and N. Hovakimyan. "Collision avoidance in cooperative missions: Bézier surfaces for circumnavigating uncertain speed profiles." In: *Journal of Guidance, Control, and Dynamics* 42.8 (2019), pp. 1779–1796.

[100] L. Meier, D. Honegger, and M. Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms." In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 6235–6240.

[101] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. "Pixhawk: A system for autonomous flight using onboard computer vision." In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2992–2997.

[102] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors." In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 2520–2525.

[103] M. Ww. Mueller, M. Hehn, and R. D'Andrea. "A computationally efficient motion primitive for quadrocopter trajectory generation." In: *IEEE transactions on robotics* 31.6 (2015), pp. 1294–1310.

[104] M. Mueller, G. Sharma, N. Smith, and B. Ghanem. "Persistent aerial tracking system for uavs." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 1562–1569.

[105] M. Muja and D. G. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration." In: *VISAPP (1)* 2.331-340 (2009), p. 2.

[106] M. Muja and D. Lowe. "Flann-fast library for approximate nearest neighbors user manual." In: *Computer Science Department, University of British Columbia, Vancouver, BC, Canada* 5 (2009).

[107] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system." In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[108] R. Mur-Artal and J. D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[109] Q. Nguyen and K. Sreenath. "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints." In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 322–328.

[110] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart. "A UAV system for inspection of industrial facilities." In: *2013 IEEE Aerospace Conference*. IEEE. 2013, pp. 1–8.

[111] K. Ok, S. Ansari, B. Gallagher, W. Sica, F. Dellaert, and M. Stilman. "Path planning with uncertainty: Voronoi uncertainty fields." In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 4596–4601.

[112] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran. "Continuous-time trajectory optimization for online UAV replanning." In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2016, pp. 5332–5339.

[113] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1366–1373.

[114] F. O'sullivan, B. S. Yandell, and W. J. Raynor Jr. "Automatic smoothing of regression functions in generalized linear models." In: *Journal of the American Statistical Association* 81.393 (1986), pp. 96–103.

[115] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li. "MULLS: Versatile LiDAR SLAM via multi-metric linear least square." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11633–11640.

[116] C. Papachristos, S. Khattak, and K. Alexis. "Uncertainty-aware receding horizon exploration and mapping using aerial robots." In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 4568–4575.

[117] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim. "Online Distributed Trajectory Planning for Quadrotor Swarm with Feasibility Guarantee using Linear Safe Corridor." In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4869–4876.

[118] J. Park, J. Kim, I. Jang, and H. J. Kim. "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 434–440.

[119] A. Pavlov, N. Van De Wouw, and H. Nijmeijer. *Uniform output regulation of nonlinear systems: a convergent dynamics approach*. Vol. 205. Springer, 2006.

[120] B. Penin, P. R. Giordano, and F. Chaumette. "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions." In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3725–3732.

[121] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto. "An informative path planning framework for UAV-based terrain monitoring." In: *Autonomous Robots* 44.6 (2020), pp. 889–911.

[122] M. Popović, T. Vidal-Calleja, G. Hitz, I. Sa, R. Siegwart, and J. Nieto. "Multiresolution mapping and informative path planning for UAV-based terrain monitoring." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1382–1388.

[123] K. Qin. "General matrix representations for B-splines." In: *Proceedings Pacific Graphics' 98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No. 98EX208)*. IEEE. 1998, pp. 37–43.

[124] K. Qin. "General matrix representations for B-splines." In: *The Visual Computer* 16.3-4 (2000), pp. 177–186.

[125] T. Qin, P. Li, and S. Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator." In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[126] J. Quinonero-Candela and C. E. Rasmussen. "A unifying view of sparse approximate Gaussian process regression." In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[127] C. E. Rasmussen. "Gaussian processes in machine learning." In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[128] C. Richter, A. Bry, and N. Roy. "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments." In: *Robotics Research: The 16th International Symposium ISRR*. Springer. 2016, pp. 649–666.

[129] C. Rösmann, F. Hoffmann, and T. Bertram. "Integrated online trajectory planning and optimization in distinctive topologies." In: *Robotics and Autonomous Systems* 88 (2017), pp. 142–153.

[130] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto. "An efficient sampling-based method for online informative path planning in unknown environments." In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1500–1507.

[131] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. "Motion planning with sequential convex optimization and convex collision checking." In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[132] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt. "Efficient autonomous exploration planning of large-scale 3-D environments." In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1699–1706.

[133] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames. "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8129–8136.

[134] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela. "Synthesis of control barrier functions using a supervised machine learning approach." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 7139–7145.

[135] C. Stachniss, G. Grisetti, and W. Burgard. "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters." In: *Robotics: Science and systems*. Vol. 2. 2005, pp. 65–72.

[136] J. A Stork and T. Stoyanov. "Ensemble of sparse Gaussian process experts for implicit surface mapping with streaming data." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10758–10764.

[137] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza. "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight." In: *IEEE Transactions on Robotics* 38.6 (2022), pp. 3357–3373.

[138] E. Tal and S. Karaman. "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness." In: *IEEE Transactions on Control Systems Technology* 29.3 (2020), pp. 1203–1218.

[139] L. Tang, H. Wang, Z. Liu, and Y. Wang. "A real-time quadrotor trajectory planning framework based on B-spline and nonuniform kinodynamic search." In: *Journal of Field Robotics* (2020).

[140] J. Tordesillas and J. P. How. "MINVO basis: Finding simplexes with minimum volume enclosing polynomial curves." In: *arXiv preprint arXiv:2010.10726* (2020).

[141] J. Tordesillas and J. P. How. "MADER: Trajectory Planner in Multiagent and Dynamic Environments." In: *IEEE Transactions on Robotics* (2021).

[142] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. "Data-driven MPC for quadrotors." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3769–3776.

[143] B. Tovar, L. Munoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. "Planning exploration strategies for simultaneous localization and mapping." In: *Robotics and Autonomous Systems* 54.4 (2006), pp. 314–331.

[144] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis. "A review on UAV-based applications for precision agriculture." In: *Information* 10.11 (2019), p. 349.

[145] H. Tsukamoto and S. J. Chung. "Neural contraction metrics for robust estimation and control: A convex optimization approach." In: *IEEE Control Systems Letters* 5.1 (2020), pp. 211–216.

[146] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche. "Feedback linearization using Gaussian processes." In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 5249–5255.

[147] V. Usenko, L. Von Stumberg, A. Pangercic, and D. Cremers. "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 215–222.

[148] M. J. Van Nieuwstadt and R. M. Murray. "Real-time trajectory generation for differentially flat systems." In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 8.11 (1998), pp. 995–1020.

[149] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Hybrid, frame and event based visual inertial odometry for robust, autonomous navigation of quadrotors." In: *arXiv preprint arXiv:1709.06310* (2017).

[150] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios." In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 994–1001.

[151] T. F. Villa, F. Gonzalez, B. Miljievic, Z. D. Ristovski, and L. Morawska. "An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives." In: *Sensors* 16.7 (2016), p. 1072.

[152] C. Virágh, M. Nagy, C. Gershenson, and G. Vásárhelyi. "Self-organized UAV traffic in realistic environments." In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2016, pp. 1645–1652.

[153] H. Voos. "Nonlinear control of a quadrotor micro-UAV using feedback-linearization." In: *2009 IEEE International Conference on Mechatronics*. IEEE. 2009, pp. 1–6.

[154] L. Wang, A. D. Ames, and M Egerstedt. "Safe certificate-based maneuvers for teams of quadrotors using differential flatness." In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3293–3298.

[155] L. Wang, A. D. Ames, and M. Egerstedt. "Safety barrier certificates for collisions-free multirobot systems." In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674.

[156] L. Wang, A. Isidori, Z. Liu, and H. Su. "Robust output regulation for invertible nonlinear MIMO systems." In: *Automatica* 82 (2017), pp. 278–286.

[157] A. Wilson and H. Nickisch. "Kernel interpolation for scalable structured Gaussian processes (KISS-GP)." In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1775–1784.

[158] C. Witting, M. Fehr, R. Bähnemann, H. Oleynikova, and R. Siegwart. "History-aware autonomous exploration in confined environments using mavs." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[159] R. J. Wood, B. Finio, M. Karpelson, K. Ma, N. O. Pérez-Arancibia, P. S. Sreetharan, H. Tanaka, and J. P. Whitney. "Progress on 'pico'air vehicles." In: *The International Journal of Robotics Research* 31.11 (2012), pp. 1292–1302.

[160] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja. "Faithful Euclidean distance field from log-Gaussian process implicit surfaces." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2461–2468.

[161] W. Xiao and C. Belta. "Control barrier functions for systems with high relative degree." In: *2019 IEEE 58th conference on decision and control (CDC)*. IEEE. 2019, pp. 474–479.

[162] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. "Fast-lio2: Fast direct lidar-inertial odometry." In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073.

[163] W. Xu and F. Zhang. "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324.

[164] Z. Xu, D. Deng, and K. Shimada. "Autonomous UAV Exploration of Dynamic Environments Via Incremental Sampling and Probabilistic Roadmap." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2729–2736.

[165] B. Yamauchi. "A frontier-based approach for autonomous exploration." In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. IEEE. 1997, pp. 146–151.

[166] H. Yang, J. Shi, and L. Carlone. "Teaser: Fast and certifiable point cloud registration." In: *IEEE Transactions on Robotics* 37.2 (2020), pp. 314–333.

[167] J. Yang, H. Li, D. Campbell, and Y. Jia. "Go-ICP: A globally optimal solution to 3D ICP point-set registration." In: *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2015), pp. 2241–2254.

[168] J. Zhang and S. Singh. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[169] H. Zhao, X. Zeng, T. Chen, and Z. Liu. "Synthesizing barrier certificates using neural networks." In: *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. 2020, pp. 1–11.

[170] B. Zhou, F. Gao, J. Pan, and S. Shen. "Robust real-time uav replanning using guided gradient-based optimization and topological paths." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1208–1214.

[171] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen. "Robust and efficient quadrotor trajectory generation for fast autonomous flight." In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3529–3536.

[172] B. Zhou, J. Pan, F. Gao, and S. Shen. "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight." In: *IEEE Transactions on Robotics* (2021).

[173] B. Zhou, Y. Zhang, X. Chen, and S. Shen. "FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 779–786.

[174] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. "Chomp: Covariant hamiltonian optimization for motion planning." In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1164–1193.