University of Verona

Department of Computer Science

Graduate School of Natural Sciences and Engineering

Doctoral Program in Computer science

Cycle xxxvii

Stochastic Constrained Optimal Control and Planning via Monte Carlo Tree Search for Autonomous Aircraft Systems

Ph.D. Candidate:
Francesco Trotti

Coordinator: Prof. Ferdinando Cicalese

Advisor: Prof. Riccardo Muradore

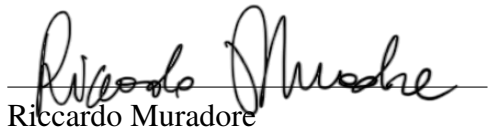
Co-Advisor: Prof. Alessandro Farinelli

S.S.D. ING-INF/05

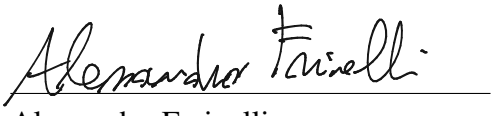
Stochastic Constrained Optimal Control and Planning via Monte Carlo Tree Search for Autonomous Aircraft Systems
Francesco Trotti
PhD Thesis
Verona

Coordinator: 

Ferdinando Cicalese

Advisor: 

Riccardo Muradore

Co-Advisor: 

Alessandro Farinelli

Declaration of Authorship

I, Francesco Trotti, declare that this thesis titled “Stochastic Constrained Optimal Control and Planning via Monte Carlo Tree Search for Autonomous Aircraft Systems” and the works presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

“Alla base di ogni scrittura c’è un paziente, scrupoloso ed estenuante lavoro di rifinitura, di correzione, di messa a fuoco, di puntualizzazione, di calibratura che costituisce la qualità e la forza del buon artigiano.”

“At the heart of every piece of writing lies a patient, meticulous, and exhausting process of refinement, correction, focus, precision, and calibration, which defines the quality and strength of a skilled craftsman.”

– Andrea Camilleri

Abstract

Stochastic constrained control and planning of autonomous systems under uncertainty are key challenges in modern control theory, driven by the growing demand for reliable and efficient autonomy. This thesis addresses these challenges by proposing a novel hierarchical control architecture for the constrained optimal control and planning of stochastic systems, with a focus on complex physical platforms such as aircraft and coordinated fleets.

The core innovation lies in the integration and enhancement of decision-making frameworks—specifically, the use of Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs)—within a hierarchical structure that separates low-level control and high-level planning. The low-level controller is tailored for handling nonlinear aircraft dynamics, while the high-level planner utilizes an online Monte Carlo Tree Search (MCTS) algorithm to solve MDPs under stochastic uncertainty.

A key contribution is the adaptation of MCTS to operate effectively within the MDP/POMDP frameworks, including the development of formal guarantees for its optimality and bounded sub-optimality. This enables robust planning in environments with noisy sensor measurements and partial observability. The use of the POMDP framework for integrated state estimation and uncertainty-aware decision-making represents an advancement in applying probabilistic planning methods to continuous, safety-critical domains.

The thesis also contributes to the design of the overall control architecture, which allows for modularity, scalability, and real-time feasibility across a diverse set of scenarios—including autonomous navigation, multi-agent coordination, and warehouse planning. By explicitly exploiting knowledge of the system dynamics in the planning process, the approach enhances planning accuracy and efficiency compared to traditional black-box or reactive methods.

Through these contributions, the thesis advances both the theoretical foundations and practical implementations of robust control for autonomous systems, offering a unified framework that bridges the gap between planning under uncertainty and nonlinear control.

Abstract (Italian)

Il controllo stocastico vincolato e la pianificazione di sistemi autonomi in presenza di incertezza rappresentano alcune delle principali sfide della teoria del controllo moderna, spinte dalla crescente domanda di autonomia affidabile ed efficiente. Questa tesi affronta tali sfide proponendo una nuova architettura gerarchica di controllo per il controllo ottimo vincolato e la pianificazione di sistemi stocastici, con particolare attenzione a piattaforme fisiche complesse come velivoli singoli o in flotta.

L'innovazione principale risiede nell'integrazione e nel perfezionamento dei framework decisionali MDP (Markov Decision Process) e POMDP (Partially Observable Markov Decision Process) all'interno di una struttura gerarchica che separa il controllo a basso livello dalla pianificazione ad alto livello. Il controllore di basso livello è progettato per gestire le dinamiche non lineari dei velivoli, mentre il pianificatore di alto livello utilizza un algoritmo Monte Carlo Tree Search (MCTS) eseguito online per risolvere problemi MDP in condizioni di incertezza stocastica.

Un contributo rilevante della tesi è l'adattamento dell'algoritmo MCTS al contesto MDP/POMDP, insieme allo sviluppo di garanzie formali di ottimalità e di sub-ottimalità limitata, che assicurano la robustezza delle soluzioni anche in presenza di rumore nelle misure e osservabilità parziale. L'impiego del framework POMDP consente una integrazione efficace tra stima dello stato e pianificazione consapevole dell'incertezza, rappresentando un progresso nell'applicazione di tecniche probabilistiche a domini continui e critici per la sicurezza.

La tesi apporta inoltre contributi alla progettazione dell'architettura complessiva di controllo, caratterizzata da modularità, scalabilità e fattibilità in tempo reale, applicabile a diversi scenari quali navigazione autonoma, coordinamento multi-agente e pianificazione logistica in magazzini. Sfruttando in modo esplicito la conoscenza del modello dinamico del sistema durante la pianificazione, l'approccio migliora la precisione e l'efficienza rispetto ai metodi reattivi o non modellati.

Attraverso questi contributi, la tesi avanza sia le basi teoriche sia le applicazioni pratiche del controllo robusto per sistemi autonomi, offrendo un framework unificato che colma il divario tra pianificazione sotto incertezza e controllo non lineare.

Contents

List of Figures

List of Tables

List of Symbols

The following list describes the meaning of symbols and notations commonly used in the theory of mechanical control systems, differential geometry, and Lie groups (see [2], [3], and [4]).

Aircraft

α	Angle of attack, $\alpha = \tan^{-1}(\frac{w}{u})$.
$\bar{c}$	Chord.
$\bar{q}$	Pressure.
β	Sideslip angle, $\beta = \sin^{-1}(\frac{v}{V_T})$.
$\boldsymbol{\eta}^e$	Aircraft orientation in Earth-frame, $\boldsymbol{\eta}^e = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$.
$\boldsymbol{\omega}_b$	Angular velocity in body-frame, $\boldsymbol{\omega}_b = \begin{bmatrix}p & q & r\end{bmatrix}^T$.
$\boldsymbol{p}^e$	Aircraft position in Earth-frame, $\boldsymbol{p}^e = \begin{bmatrix}p_x & p_y & p_z\end{bmatrix}^T$.
$\boldsymbol{V}_b$	Translational velocity in body-frame, $\boldsymbol{V}_b = \begin{bmatrix}u & v & w\end{bmatrix}^T$.
$\delta_e, \delta_a, \delta_r$	Elevator, aileron, and rudder deflection commands.
B	Wingspan.
C_l, C_m, C_n	Aerodynamic coefficients for roll, pitch, and yaw moments.
C_x, C_y, C_z	Aerodynamic coefficients along the x, y, z body axes.
g	Gravitational acceleration.
I_{xz}	xz body-axis product of inertia.
I_x, I_y, I_z	x, y, z body-axis moments of inertia.
m	Aircraft mass.
S	Planform area.
T	Thrust command.
V_T	Aircraft airspeed, $V_T = \sqrt{u^2 + v^2 + w^2}$.

Control system

$\mathcal{U}$	Action space with $x \in \mathcal{U} \in \mathbb{R}^m$.
$\mathcal{X}$	State space with $x \in \mathcal{X} \in \mathbb{R}^n$.
A	Linear dynamic matrix $A \in \mathbb{R}^{n \times n}$.
B	Linear control matrix $B \in \mathbb{R}^{m \times n}$.
$f(x)$	Nonlinear drift function.

$g(x)$	Nonlinear control function.
m	Input space dimension.
n	State space dimension.
Lie group	
G	A connected Lie group.
g	An element of G.
$\mathfrak{g}$	The Lie algebra associated with G.
$[\cdot, \cdot]$	The Lie bracket of $\mathfrak{g}$.
$\mathfrak{g}^*$	The dual of the Lie algebra $\mathfrak{g}$.
$L_g : G \to G$	The left translation $L_g h = gh$.
$T_h L_g$	The tangent map of L_g at $h \in G$.
gX	Shorthand for $T_e L_g(X) \in T_g G$.
$\langle \cdot, \cdot \rangle$	The duality pairing $\langle \mu, X \rangle = \mu(X)$.
V	A finite-dimensional vector space.
$f : G \to V$	A differentiable map $f : G \to V$.
$\mathrm{d}f(g)$	The differential of f at g, $\mathrm{d}f(g) : T_g G \to V$, identifying $T_{f(g)} V$ with V.
$T(X, Y) \in \mathfrak{g}$	The torsion function associated with ω.
$T_X : \mathfrak{g} \to \mathfrak{g}$	A partial torsion function $T_X Y = T(X, Y)$.
$I : \mathfrak{g} \times \mathfrak{g} \to \mathbb{R}$	The inner product on $\mathfrak{g}$.
$I^\sharp : \mathfrak{g}^* \to \mathfrak{g}$	The $\sharp$-map associated to the inner product I.
$I^\flat : \mathfrak{g} \to \mathfrak{g}^*$	The $\flat$-map associated to the inner product I.
$\mathrm{ad} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$	The adjoint map on $\mathfrak{g}$.
$\mathrm{ad}^* : \mathfrak{g} \times \mathfrak{g}^* \to \mathfrak{g}^*$	The dual adjoint map.
$0_{n \times n}$	The null matrix of dimension $n \times n$.
$I_{n \times n}$	The identity matrix of dimension $n \times n$.
$\omega : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$	The connection function associated with ∇.
$\omega_X : \mathfrak{g} \to \mathfrak{g}$	A map $\mathfrak{g} \to \mathfrak{g}$, defined as $\omega_X(Y) = \omega(X, Y)$.
$(\phi)^W : L(W, U) \to L(W, V)$	The exponential functor $(\cdot)^W$ applied to a linear map $\phi : U \to V$, lifting ϕ to $\phi^W : L(W, U) \to L(W, V)$.
$\mathrm{Hess} f(g) : T_g G \to L(T_g G, V)$	The Hessian operator of a twice differentiable function $f : G \to \mathbb{R}$ or a map $f : G \to V$.
Markov Decision Process	
γ	The discount factor.
$\mathcal{A}$	The action space with $a \in \mathcal{A}$.
$O(s, a)$	The observation probability $O^a_{s'o} = \mathbb{P}(o \mid s', a)$.
$\mathcal{R}(s, a)$	The reward function.
$\mathcal{S}$	The state space with $s \in \mathcal{S}$.
$\mathcal{T}(s, a)$	The transition function $\mathcal{T}^a_{s,s'} = \mathbb{P}(s_{t+1} = s' \mid s_t = s, a_t = a)$.
$\mathcal{Z}$	The observation space $z \in \mathcal{Z}$.

Chapter 1

INTRODUCTION

"The important thing is not to stop questioning. Curiosity has its own reason for existing."

Albert Einstein

The stochastic constrained control and planning of autonomous systems in optimal or suboptimal scenarios are topics of paramount importance that have received significant attention in recent years, especially in response to the growing demand for autonomy. Achieving autonomy requires control systems capable of handling different levels of uncertainty arising from noisy sensor measurements or partial knowledge of the plant state, as well as unexpected behavior of external systems, while at the same time minimizing or maximizing a cost function. Considering these aspects, the control and planning problems for an autonomous or a fleet of autonomous agents become challenging and complex, mostly if applied to complex physical systems such as aircraft or fleets of aircraft.

In control theory, the design of a strategy able to consider the uncertainty is a complex task that sometimes requires some assumption concerning the nature of the uncertainty (i.e., boundness or convexity). In the literature, a lot of strategies try to take into account different kinds of uncertainty: model uncertainty, noisy measurements and partial knowledge of the state. The design of a control strategy able to mitigate the effects of the additive model uncertainty on nonlinear systems relies on a model like

$$\begin{cases} x_{k+1} = f(x_k) + g(x_k)u_k + \omega_k \\ y_k = h(x_k) + v_k \end{cases} \tag{1.1}$$

where $f(x)$, $g(x)$ and $h(x)$ are nonlinear functions, x_k is the state, u_k is the control input, ω_k is the unknown exogenous signal modeling the uncertainty, y_k are the measurements and v_k is the exogenous signal modeling the measurement uncertainty. The estimation of this uncertainty becomes crucial for model-based strategies, where a horizon is projected into the future exploiting an internal dynamic model. Also, for the "classical" control strategy, i.e., the exact linearization via feedback, it is quite important to estimate this kind of uncertainty in order to stabilize the system and increase the performance. A strategy able to estimate and compensate for the noisy measurement is important for the design of a robust control strategy. For this reason, a branch of control systems literature is focused on the design of observers to filter out the noise and estimate the unknown state variables.

In control theory, hierarchical control architectures [5, 6] are commonly used to consider these different kinds of uncertainty, where the environmental uncertainty is taken into account by the planner, while model uncertainty is considered by controllers/observers closer to the physical systems as shown in Figure 1.1.

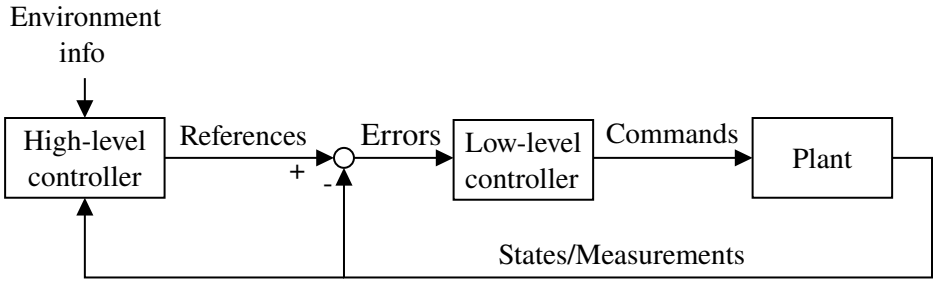


Fig. 1.1: Hierarchical control architecture

Another important aspect of working with cyber-physical systems, and in particular on aircraft systems, is to consider some constraints on the control variables (e.g., the deflection surfaces do not exceed the maximum/minimum angle) and on the state (e.g., the angle-of-attack). A common strategy to guarantee the constraints on the control variables is the design of the controller solving a constrained optimization problem.

In literature, controlling and planning in an optimal way to satisfy state and control variables constraints of stochastic systems while minimizing or maximizing a cost function is a very complex task that classical methods tend to solve in approximate ways or by increasing the computational complexity. Such solutions are often unusable on cyber-physical systems such as aircraft. More recent approaches based on Reinforcement Learning (RL) are able to manage the uncertainty, learning the internal model during the training phase. However, these approaches fail on the guarantees that can be provided concerning the optimality and stability of the solutions. Other branches of Reinforcement Learning do not exploit the training phase but work online, alternating simulation and execution. These approaches combine Markov Decision Process formalism, heuristic strategy to explore the state/action space efficiently, and internal models with uncertainty. Therefore, in these strategies, it is possible to manage the uncertainty of both model and measurements, minimizing a cost function (reward function for the Reinforcement Learning community).

1.1 Related works

Constrained optimal control is a critical challenge in nonlinear dynamic systems, where both the system state and the control inputs must satisfy strict physical, operational, and safety constraints; moreover, the problem becomes more challenging when model uncertainty is considered. In recent years, Model Predictive Control (MPC) has emerged as a widely adopted solution for constrained optimal control problems due to its ability to predict future states and optimize control actions over a finite horizon [7]. MPC operates by solving an optimization problem at each time step, using a dynamic model of the system to predict future behavior [8]. Other approaches exploit the MPC as planner [9] find the best control sequence to guide a fixed-wing UAV into a deep stall for landing. Approaches based on Nonlinear Model Predictive Control (NMPC), as discussed in [10], leverage a polytopic description of each robot's shape and formulate collision avoidance as a dual optimization problem. Mixed approaches, as presented in [11], combine decentralized Model Predictive Control (MPC) formalization with consensus control strategies to address cooperative formation control with collision avoidance. Nevertheless, these approaches often fail to account for various types of uncertainty, such as model errors, sensor measurement errors, or environmental dynamics. Traditional MPC assumes that the model of the system is known and reliable, but in practice, the system dynamics may change or contain unknown disturbances [12]. To address these challenges, robust MPC and stochastic MPC have been developed, incorporating worst-case scenario modeling [13], probabilistic techniques to mitigate uncertainty [14–16] or stochastic stabilization of linear systems affected by multiplicative disturbances [17]. Other probabilistic MPC formalizations [18] allows the computation of optimal control sequences to ensure that linear constraints are satisfied with a certain probability. Extending this concept, the authors in [19] formalize the problem as a Disjunctive Linear Program, considering the probabilistic distribution of the aircraft state to maintain the obstacle collision probability under a defined threshold.

However, these methods increase computational complexity and lead to conservative solutions, as they overcompensate potential uncertainties. Beyond MPC, alternative strategies have been explored to tackle the complexity of constrained optimal control under uncertainty, such as the stochastic optimal control (SOC) [20,21], which models the uncertainty explicitly in the form of stochastic processes. SOC aims to optimize a control policy by minimizing expected costs over time while accounting for randomness in system dynamics. Another promising approach is Deep Reinforcement Learning (DRL) that combines Reinforcement Learning (RL) [22,23] with Deep Neural Networks to handle high-dimensional state spaces, making it possible to apply RL techniques to complex control problems in robotics and autonomous systems [24–26]. Recent works focused on the Formal Verification (FV) of Deep Neural Networks in order to get a provable guarantee on the output of the neural network [27]. However, reinforcement learning requires large amounts of data and can suffer from instability in learning, particularly in environments with safety-critical constraints. Meanwhile, the NP-completeness of formal verification [28] makes the computation intractable during the training due to the significant overhead of verification tools [29]. Recently, alternative approaches have gained attention for addressing uncertainty in optimal control. One method is to formalize the optimal control

problem as a Markov Decision Process (MDP) [30–32]. Particular attention is also given to Monte Carlo Tree Search (MCTS), which integrates tree-based planning with random sampling to explore large state and action spaces. Initially developed for game theory applications [33], MCTS has proven to be highly adaptable for solving sequential decision-making problems, especially under uncertainty, balancing the exploration-exploitation trade-off by iterative refining. This process makes it suitable for systems where the model dynamics are either partially or entirely unknown [34].

The Markov Decision Processes find wide use in optimal trajectory and path planning [35] and collision avoidance with and without uncertainty. To ensure the success of the task, the path planner for an autonomous system has to consider different levels of uncertainty and also has to be optimal with respect to certain metrics (e.g., fuel consumption or travel time). Taking into account these aspects, the planning problem becomes more challenging and complex since different types of uncertainty must be considered, such as model errors or measurement errors [36]. Some approaches, [37], [38], attempt to solve the path planning as an optimization problem using Mixed-Integer Linear Programming (exploiting the Branch and Bound technique) to design the best trajectory while minimizing certain metrics (e.g., fuel consumption). The optimization approaches that consider uncertainty in their formalization struggle to account for both model errors, measurement errors, and uncertainty due to a dynamic environment. For these reasons, formalizing the optimization problem as a decision-making problem (Markov Decision Process (MDP)) could bring some advantages, such as the Markovian properties, the ability to perform multiple simulations in the future to optimize a cost function, and the inherent characteristic of MDPs to consider model errors.

One way to face the problem of having a good estimation of the state is to formulate a filter/observer as an optimization problem. In [39], the author exploits the dynamic programming principle to provide optimal estimates of a system with nonlinear dynamics, nonlinear measurement equations, and a cost functional (energy) that weighs the contribution of model and measurement errors. This idea has been exploited in [40], where the authors present a second-order optimal minimum-energy filter constructed on Lie groups. The effectiveness of this filter has been studied considering free rigid bodies [41], rigid bodies with nonholonomic constraints (see [42]), and articulated vehicles [43,44]. A comparison with the extended Kalman filter is presented in [45], while an application of the filter in the case of switching dynamics is provided in [46]. Another way to take into account the uncertainty due to the environment and sensor measurement errors is the Partially Observable Markov Decision Processes (POMDPs). Significant attention has been given to the path planning problem for aircraft formalized as a POMDP [47, 48]. In [49], the POMDP with nominal belief-state optimization (NBO) is used to solve the tracking problem of static or dynamic targets. In [50] and [51], POMDP-based solutions are proposed for UAV multiple target tracking problems considering sensor uncertainty. However, in most of the path planners formalized as POMDP, the relationship with the low-level controller, which provides the commands to the aircraft flaps, is not considered. This aspect is quite important during the exploration phase of the POMDP to guarantee the feasibility of the actions (e.g., the physical constraints are satisfied). Achieving autonomy in formation flight and fleet coordination

requires control systems capable of handling different levels of uncertainty arising from the unexpected behavior of agents within the fleet due to the limited or absent knowledge of the states of the other UAVs or environmental conditions.

Hierarchical approaches, as proposed in [52], aim to address formation control by managing issues like leader-following, behavioral, and virtual structure problems through control-theoretic methods. Additionally, leveraging control-theoretic techniques, as discussed in [53], helps coordinate a fleet of mobile robots using a feedback control law and a reactive control framework. However, these approaches often require a centralized component, limiting scalability to a large number of robots. A significant number of approaches in the literature are based on the consensus theorem, e.g., [54], [55], and [56]. Some collision avoidance control algorithms for multi-UAV systems designed around consensus-based algorithms and leader-follower control strategies are presented in [57] and [58]. However, these approaches primarily focus on the leader-following consensus problem, assuming that only the parameters of followers are uncertain. Differently, [59] addressed the issue of parametric uncertainties and unknown external disturbances for both leaders and followers by employing a multivariable model reference adaptive control (MRAC) and the consensus theory.

1.2 Motivation

The core motivation of this thesis is to design a robust and scalable control architecture capable of operating effectively in the presence of model uncertainty, sensor noise, and physical constraints inherent in dynamic systems. These challenges are especially critical in real-world scenarios, where perfect models and precise measurements are rarely available. To address them, this work leverages the model-based capabilities of Reinforcement Learning (RL) within a hierarchical control framework, combining principled decision-making with robust low-level control strategies.

At the heart of the proposed architecture is a two-layered structure: a high-level *planner* and a low-level *controller*, each tailored to confront specific aspects of uncertainty. The planner, responsible for long-horizon decision-making under uncertainty, is formulated within the Markov Decision Process (MDP) framework and solved online via the Monte Carlo Tree Search (MCTS) algorithm. This approach enables planning under uncertainty by modeling system evolution as a constrained stochastic process, capturing the probabilistic effects of both dynamics and environmental variability.

However, noisy sensor readings and partial observability introduce a significant challenge to reliable planning and autonomous navigation. These factors can severely degrade decision quality if not properly addressed. To counter this, we extend our planning approach to the Partially Observable Markov Decision Process (POMDP) framework, allowing the system to perform state estimation and noise filtering simultaneously during decision-making. This provides a principled method to handle measurement uncertainty without relying solely on external filters.

On the control side, we design a low-level control strategy specifically suited for highly nonlinear systems, such as aircraft dynamics. This controller ensures

that real-world actuation respects physical constraints while executing the high-level planner's decisions.

The thesis further investigates the theoretical guarantees provided by the MDP-MCTS combination, especially regarding the optimality and sub-optimality of the selected local actions. Building upon these insights, we evaluate two complementary approaches for robust control under uncertainty: one that integrates noisy observations directly into the planner using a POMDP model, and another that employs a geometric filter to separately estimate the system state before planning.

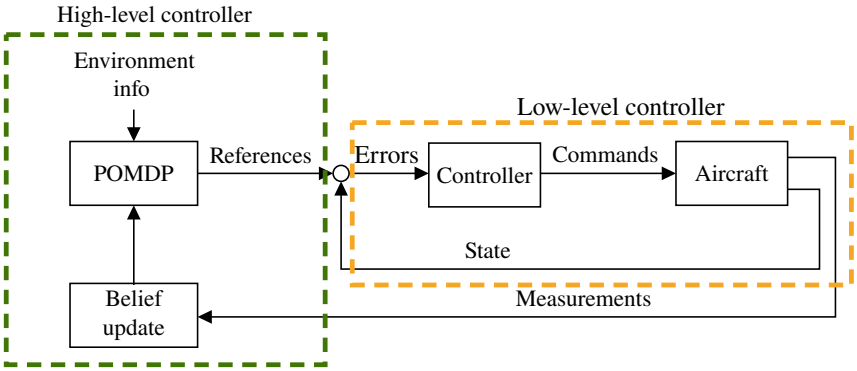
Finally, the developed hierarchical architecture is extended to multi-agent systems, addressing the decentralized fleet navigation problem. Each agent operates with its own local planner, modeled as an MDP, which predicts and adapts to the behavior of neighboring agents—enabling coordination and collision avoidance in uncertain, dynamic environments.

Stochastic Constrained Optimal Control

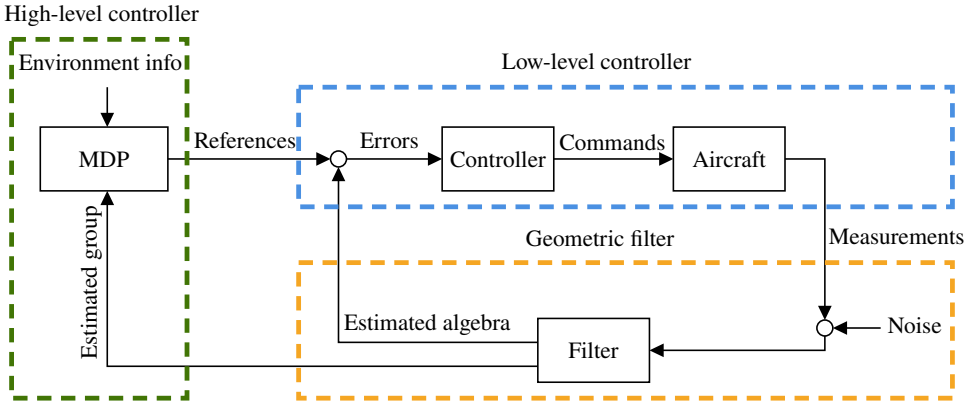
In Chapter 3, we provide some formal guarantees concerning Monte Carlo Tree Search. We use Monte Carlo Tree Search to solve an optimization problem considering state and control constraints and also additive model uncertainty. We focus on the convergence to optimality with an infinite number of simulations. Given the infeasibility of the infinite simulations on cyber-physical systems, we compute the number of simulations needed to reach an ε optimality with a δ confidence in the cost function without model uncertainty. After that, we introduce the model uncertainty, computing the minimum number of simulations needed to compensate for the uncertainty and reach the ε optimality with a δ confidence. A comparison with the Linear Quadratic Regulator (LQR) controller on the linearized aircraft dynamic model is made to support the formal statements. Additionally, considering the aircraft nonlinear systems and the relative constraints, we compare the solution of MCTS with the Nonlinear Model Predictive Control (NMPC) in order to show the behavior of the two controllers.

Planning via Markov Decision Process

Solving the planning problem in an optimal way considering the model and environmental uncertainty is challenging. In Chapters 4 and 5, we focus on the planning and control of 6 degrees of freedom nonlinear aircraft dynamics considering noisy measurements. In both chapters, the hierarchical architecture is applied but with two different control and planning strategies. In Chapter 4, we design a Partially Observable Markov Decision Process as a planner (high-level controller) in order to take into account the noisy measurements and compute the reference values for the low-level controller, designed using the exact linearization via feedback. In Chapter 5, we formalize the problem from the geometric point of view in order to exploit a minimum-energy optimal geometric observer to estimate the state filtering out the noisy measurements on the Lie Groups. We design a Markov Decision Process (high-level controller) as a planner using the estimated state as input. The low-level controller is designed to control the left-trivialized components of the dynamics. In Figure 1.2, the control architectures used in these two Chapters are shown.



(a) Hierarchical control architecture in Chapter 4



(b) Hierarchical control architecture in Chapter 5

Fig. 1.2: Hierarchical control architectures

We test the two approaches in different simulated scenarios in order to evaluate the ability of the architecture to plan and control the aircraft in the right way.

Fleet Coordination via Markov Decision Process

In Chapter 6, we address the problem of formation planning and controlling a fleet of aircraft in order to reach a target position, avoiding some no-fly zones. We design a decentralized hierarchical control architecture (Figure 1.3) for each aircraft, considering the uncertainty of the positions of the other aircraft.

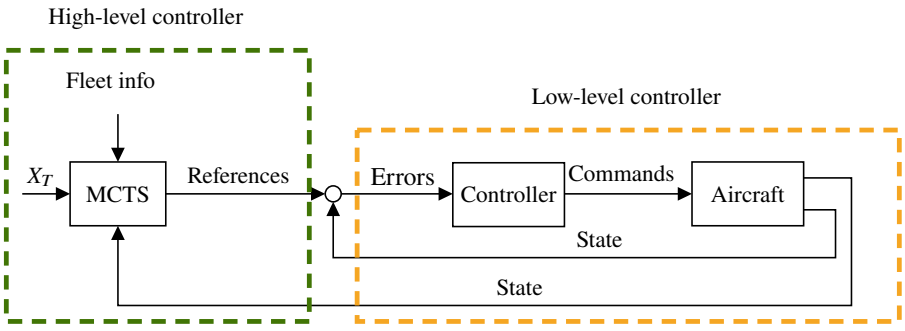


Fig. 1.3: Hierarchical control architecture Chapter 6

In particular, we formalize the MDP as a planner considering the aircraft dynamics and the future trajectories of the other agents, while the low-level controller is designed with exact linearization via feedback. We tested our approach in different simulated scenarios, showing the ability of the planner and the controller to drive the fleet to the target.

Planning and Re-planning via Markov Decision Process

In Chapter 7, we investigate a possible application of Monte Carlo Tree Search as a strategy to plan and re-plan the path of mobile robots in the presence of unexpected obstacles during the assigned paths in a warehouse scenario. A stochastic description of the lifespan of the obstacles is modeled, and the path of the other agents is considered as a constraint on the state. We tested our strategy with quantitative simulations, and we also conducted a real experiment in our laboratory.

1.3 Outline of the thesis

The thesis is organized as follows

- Chapter 2 introduces the backgrounds concerning MDP, POMDP, and the dynamic model of the aircraft.
- Chapter 3 states the formal guarantees of Monte Carlo Tree Search (MCTS) used as a stochastic constrained optimal controller with and without model uncertainty.
- Chapter 4 designs a hierarchical control architecture that considers an optimal/-suboptimal planner based on the Partially Observable Markov Decision Process (POMDP) with a dynamic model and measurement uncertainty.
- Chapter 5 uses the hierarchical control architecture exploiting the Markov Decision Process (MDP) as a planner and an optimal minimum-energy filter to estimate the state.
- Chapter 6 formalizes a strategy based on the hierarchical architecture to control in a decentralized way a fleet of aircraft.
- Chapter 7 designs a decentralized high-level planning and re-planning strategy for multi-agent systems in warehouse scenarios.

Chapter 2

BACKGROUND

"The more I learn, the more I realize how much I don't know."

Albert Einstein

Abstract In this chapter, we provide some background that will be used in this thesis. In particular, we report some definitions concerning the Markov Decision Process (MDP) and the state-of-the-art version [34] of MDP solver Monte Carlo Tree Search (MCTS). After that, we illustrate the Partially Observable Markov Decision Process (POMDP) and the corresponding state-of-the-art solver Partially Observable Monte Carlo Tree Process (POMCP) [60]. Finally, we recall Hoeffding's Inequality and the nonlinear dynamic model of the aircraft used in the simulations. Note that in this chapter, for the definition of the MDP/POMDP and MCTS/POMCP, we use the common formalism of the reinforcement learning community, while in the other chapters, we will use the formalism affine with the control theory.

2.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a sequential model-based decision-making framework that considers actions and internal model stochasticity [61].

Definition 2.1: *A MDP $\mathcal{M}$ is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where:*

1. *$\mathcal{S}$ is a finite state space with $s \in \mathcal{S}$*
2. *$\mathcal{A}$ is a finite action space with $a \in \mathcal{A}$*
3. *$\mathcal{T}$ is the transition model $\mathcal{T}^{a}_{s,s'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$ that defines the probability to evolve from the current state s at the time t to the future state s' at the time $t+1$ by taking the action a*
4. *$\mathcal{R}(s, a)$ is the reward function from the current state $s \in \mathcal{S}$ taking the action $a \in \mathcal{A}$*
5. *γ is the discount factor ($\gamma \in (0, 1)$)*

2.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search is a common solver for the Markov Decision Processes that alternates a *training phase*, exploring the state-action spaces, with the *execution phase*, where the optimal or suboptimal action is applied to the environment. MCTS is a tree-based algorithm that explores the state space using a heuristic and simulates the possible evolution of the state, exploiting an internal model (model-based algorithm). This chapter provides only the basic definition of Monte Carlo Tree Search [62], which will be explained in more detail in Chapter 3. The algorithm consists of four phases shown in Figure 2.1.

- *Selection*: The algorithm traverses the existing search tree from the root to a leaf node by selecting actions according to the Upper Confidence Bound applied to Trees (UCT) heuristic [63], [34]. At each decision node, the action a^* is chosen as follows

$$a^* = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \left(Q(s,a) + C\sqrt{\frac{\log N(s)}{N(s,a)}} \right) \tag{2.1}$$

  where $Q(s,a)$ is the estimated value of taking action a in state s, $N(s)$ is the total number of times state s has been visited, $N(s,a)$ is the number of times action a has been selected from state s, and C is a constant balancing exploration and exploitation. The first term of UCT ($Q(s,a)$) favors actions with high estimated value (exploitation), while the second term $\left(\sqrt{\frac{\log N(s)}{N(s,a)}}\right)$ encourages trying less-visited actions (exploration).
- *Expansion*: Once a leaf node (a node that is not fully expanded or has not been visited) is reached during the selection phase, it is expanded by adding one or more child nodes, corresponding to the possible actions available from that state.
- *Simulation*: From the newly expanded node, a simulation—also called a rollout—is conducted to estimate the potential outcome. The system evolves using an internal model, where the next state and reward are generated as $(s_{t+1}, r_{t+1}) \sim f(s,a)$. The rollout policy selects actions randomly (or using a predefined heuristic) until a terminal state is reached or a maximum depth is exceeded. This process helps estimate the long-term value of the decision path.
- *Backpropagation*: After the simulation concludes, the result (i.e., the accumulated reward) is propagated back through the selected path, from the expanded node up to the root. The visit counts $N(s)$ and $N(s,a)$ are incremented accordingly, and the $Q(s,a)$ value is updated as

$$Q(s,a) \leftarrow Q(s,a) + \frac{R}{N(s,a)} \tag{2.2}$$

  where R is the reward obtained from the simulation. This update rule incrementally refines the value estimate based on cumulative experience.

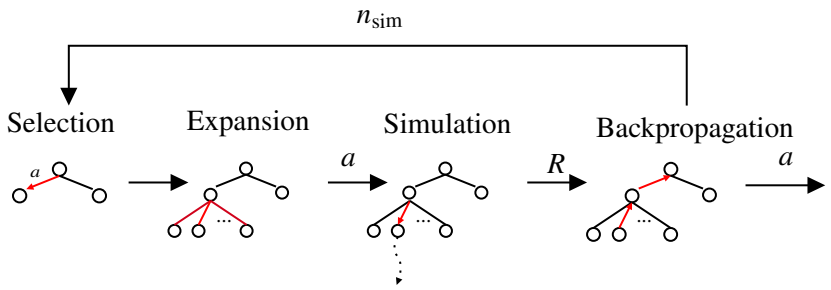


Fig. 2.1: Monte Carlo Tree Search

2.3 Partially Observable Markov Decision Process (POMDP)

A Partially Observable Markov Decision Process (POMDP) is a sequential model-based decision-making framework to generate a control policy (i.e., a mapping between the belief state and the actions) that considers not only the stochasticity of the actions but also the noise of the measurements [60].

Definition 2.2: *A POMDP is defined by the tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O}, \gamma \rangle$*, where:*

- $\mathcal{S}$ *is the state space with* $s \in \mathcal{S}$
- $\mathcal{A}$ *is the actions space with* $a \in \mathcal{A}$
- $\mathcal{T}$ *is the transition probability* $\mathcal{T}^{a}_{s,s'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$ *that defines the probability to evolve from the current state* s *at the time* t *to the future state* s' *at the time* $t+1$ *by taking the action* a
- $\mathcal{R}(s, a)$ *is the reward function that maximize the reward value* $\mathcal{R}$ *from the current state* $s \in \mathcal{S}$ *taking the action* $a \in \mathcal{A}$
- $\mathcal{Z}$ *is the observations space*
- $\mathcal{O}$ *is the probability* $\mathcal{O}^{a}_{s'o} = \mathbb{P}(o_{t+1} = o | s_t = s, a_t = a)$ *to obtain the observation* o *at the time* $t+1$ *by taking action* a *in the state* s *at the time* t
- γ *is the discount factor (i.e., a real number between 0 and 1).*

In POMDPs, states are not directly observable by the agent; hence it is necessary to define a belief $\mathcal{B}$ that provides the probability of being in the state s at the time t, $\mathcal{B}(s, h) = \mathbb{P}(s_t = s | h_t = h)$, where h is the history composed of previous actions and observations $h_t = \{a_1, o_1, ..., a_t, o_t\}$.

2.4 Partially Observable Monte Carlo Tree Process (POMCP)

Partially Observable Monte Carlo Tree Process [60] is an online solver for the Partially Observable Markov Decision Processes. POMCP is an algorithm based on a probability distribution of the state and on Monte Carlo Tree Search; however, since the state is partially observable, the four phases of MCTS can no longer be based on the state. The algorithm bases the state-action exploration on a history of actions and observations $(h_t = a_0, o_0, \ldots, a_t, o_t)$, and then the four phases of MCTS are updated accordingly. Therefore, the POMCP phases are

- *Selection*: The algorithm following the Upper Confidence Bound applied to the Tree (UCT) heuristic crosses the tree to reach the leaf node

$$a^* = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \left(Q(h,a) + C \sqrt{\frac{\log N(h)}{N(h,a)}} \right) \tag{2.3}$$

where $N(h,a)$ is the number of times action a has been taken from the history h, $N(s)$ is the number of times that the history h has been visited, C is the exploration-exploitation constant and $Q(h,a)$ is the history-action values of the current node. UCT balances the exploration ($\sqrt{\frac{\log N(h)}{N(h,a)}}$) and exploitation ($Q(h,a)$) of the tree, selecting a new node or an already visited node to analyze the score better.

- *Expansion*: In this phase, the algorithm expands the leaf node with a number of nodes equal to the number of actions.
- *Simulation*: The algorithm simulates the system evolution exploiting the internal model, computing the next uncertain state and the observation given a sampled state from the belief and an action $(s_{t+1}, o_{t+1}, r_{t+1}) \sim f(s,a)$. In this phase, the rollout strategy simulates random actions from the current state until the target or the maximum depth is reached. This strategy helps to increase the history of the node with a random exploration.
- *Backpropagation*: In this phase, the UCT path to the leaf node is retraced backward until the root node is reached, increasing the number of history visits $N(h)$ and $N(h,a)$ and updating the history-action values $Q(h,a)$ in the following way

$$Q(h,a) \leftarrow Q(h,a) + \frac{R - Q(h,a)}{N(h)} \tag{2.4}$$

where R is the immediate reward of the current node.

- *Belief update*: The algorithm updates the belief state given the observations from the environment. Specifically, by exploiting the Bayes theorem and using observations from the environment and the previous belief as a prior distribution, it is possible to update the belief

$$\mathcal{B}(s') = \frac{\mathcal{O}^a_{s',o} \sum_{s \in \mathcal{S}} \mathcal{T}^a_{s,s'} b(s)}{\sum_{s' \in \mathcal{S}} \mathcal{O}^a_{s',o} \sum_{s \in \mathcal{S}} \mathcal{T}^a_{s,s'} b(s)} \tag{2.5}$$

where $\mathcal{B}(s)$ is the prior belief over the state $s \in \mathcal{S}$, $\mathcal{B}(s')$ is the updated belief over the new state $s' \in \mathcal{S}$.

Figure 2.2 shows the block schema of POMCP algorithm.

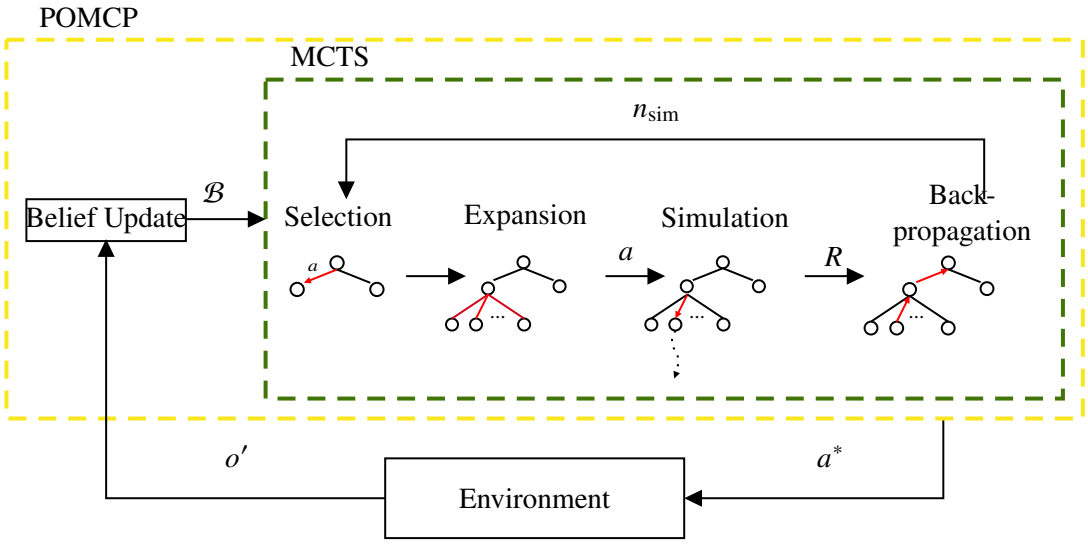


Fig. 2.2: Partially Observable Monte Carlo Tree Process

2.5 Hoeffding's Inequality

Hoeffding's inequality (see [64, Section 4.5]) provides a bound on the probability that the sum of bounded random variables deviates from its expected value. Consider a sequence of n independent random variables $X_1, X_2, \ldots, X_n$ where each X_i belongs to the interval $[a, b]$. Hoeffding's inequality quantifies how the empirical mean of these random variables deviates from the true mean. More specifically, for any $\varepsilon > 0$, Hoeffding's inequality is given by

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right]\right| \geq \varepsilon\right) \leq 2e^{-\frac{2n\varepsilon^2}{(b-a)^2}}. \tag{2.6}$$

This inequality states that the probability that the absolute difference between the empirical mean $\frac{1}{n}\sum_{i=1}^{n} X_i$ and the expected mean $\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n} X_i]$ is at least ε is upper bounded by $2e^{-\frac{2n\varepsilon^2}{(b-a)^2}}$. Hoeffding's inequality does not assume any specific distribution for the random variables X_i, except that they are independent and belong to the interval $[a, b]$. The bound $(b - a)$ reflects the variable range and influences the inequality's tightness; narrower ranges yield stronger concentration around the mean.

2.6 Multi-Agent Path Finding

The Multi-Agent Path Finding (MAPF) problem concerns the computation of collision-free paths for a group of agents operating within a shared environment, typically represented as a graph. The objective is to optimize a performance metric—such as total travel time or makespan—while ensuring that agents do not interfere with one another during execution [65]). Formally, the MAPF problem is defined as follows

- Let M denote the total number of agents, and let m_i represent the i-th agent with $i \in M$.
- Let the environment be modeled as an undirected graph $G = (V, E)$, where V is the set of nodes (locations), $E \subseteq V \times V$ is the set of edges (allowable movements between nodes).

Each agent may either wait at its current node or move to an adjacent node at each discrete time step. A path for agent m_i is defined as a sequence of actions

$$\pi_i = \{a_t, a_{t+1}, \ldots, a_{t+K}\} \tag{2.7}$$

where t represents the starting time, and K is the number of steps required to reach the goal. A plan is the set of individual paths for all agents

$$\pi = \{\pi_1, \pi_2, \ldots, \pi_M\} \tag{2.8}$$

A valid plan must be collision-free, meaning that it avoids the following types of conflicts

- Vertex collision: occurs when two or more agents occupy the same node at the same time step t.
- Edge collision: occurs when two agents traverse the same edge in opposite directions at the same time step t.

A MAPF solution is a collision-free path that minimizes the number of steps for each agent. Conflict-Based Search (CBS) is an optimal and complete MAPF algorithm [66]. This algorithm is based on the definition of a set of constraints defined as a tuple (m_i, v, t) where m_i is the i-th agent and v is the prohibited vertex at time step t. When the algorithm solves the planning problem for each agent complying with the constraints, collision-free paths are guaranteed. If there is a collision, a new constraint is added. This approach iterates until collision-free plans are generated.

2.7 Aircraft dynamics

In this Section, we recall the aircraft dynamics used in simulations: a supersonic 6 Degrees of Freedom (DoF) nonlinear dynamic model [67, 68]. The aircraft state vector $\boldsymbol{x}$ and the control variables $\boldsymbol{u}$ are

$$\begin{aligned} \boldsymbol{x} &= [V_T \quad \alpha \quad \beta \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r \quad p_x \quad p_y \quad p_z]^T \\ \boldsymbol{u} &= [T \quad \delta_e \quad \delta_a \quad \delta_r] \end{aligned} \tag{2.9}$$

where, V_T is the airspeed of the aircraft, α and β the angle of attack and the sideslip angle, ϕ, θ, ψ are the Euler angles in Earth-frame, while p, q, r are the rate of the angle in body frame. Finally, p_x, p_y, p_z are the aircraft position in the Earth-frame (see Figures 2.4, 2.5, 2.6). Concerning the commands T is the thrust, $\delta_e, s\delta_a, \delta_r$ are the elevator, aileron and rudder (see Figure 2.3)

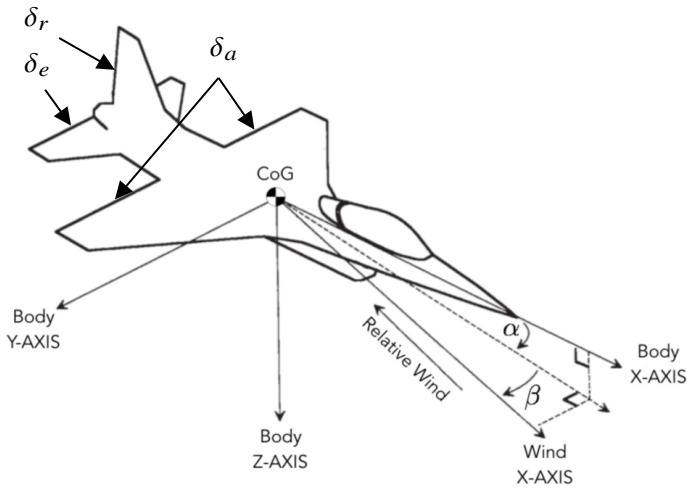


Fig. 2.3: Body-fixed reference frames. Courtesy of [1].

The time derivatives of the navigation equations in the Earth-frame for each axis are given by

$$\begin{aligned} \dot{p}_x &= u\cos\psi\cos\theta + v(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) \\ &\quad + w(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \\ \dot{p}_y &= u\sin\psi\cos\theta + v(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi) \\ &\quad + w(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \\ \dot{p}_z &= -u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi \end{aligned} \tag{2.10}$$

where u, v, w are defined by

$$\begin{aligned} u &= V_T\cos\alpha\cos\beta \\ v &= V_T\sin\beta \\ w &= V_T\sin\alpha\cos\beta. \end{aligned} \tag{2.11}$$

The following equations express the linear accelerations along the three axes in the body-frame

$$\begin{aligned}\dot{u} &= -qw + rv + \frac{\bar{q}SC_x + T}{m} - g\sin\theta \\ \dot{v} &= pw - ru + \frac{\bar{q}SC_y}{m} + g\sin\phi\cos\theta \\ \dot{w} &= -pv + qu + \frac{\bar{q}SC_z}{m} + g\cos\phi\cos\theta,\end{aligned} \tag{2.12}$$

where C_x, C_y, C_z are the aerodynamic coefficients given by

$$\begin{aligned}C_x &= C_{x_0}(\alpha, \delta_e) + \frac{\bar{c}}{2V_T}C_{x_q}(\alpha)q \\ C_y &= C_{y_0}(\beta, \delta_a, \delta_r) + \frac{B}{2V_T}(C_{y_p}(\alpha)q + C_{y_r}(\alpha)r) \\ C_z &= C_{z_0}(\alpha, \beta, \delta_e) + \frac{\bar{c}}{2V_T}C_{z_q}(\alpha)q.\end{aligned} \tag{2.13}$$

The angular accelerations in body-frame are

$$\begin{aligned}\dot{p} &= (c_1 r + c_2 p + c_4 h_{eng})q + \bar{q}SB(c_3 C_l + c_4 C_n) \\ \dot{q} &= (c_5 p - c_7 h_{eng})r - c_6(p^2 - r^2) + \bar{q}S\bar{c}C_m c_7 \\ \dot{r} &= (c_8 p - c_2 r + c_9 h_{eng})q + \bar{q}SB(c_4 C_l + c_9 C_n),\end{aligned} \tag{2.14}$$

with

$$\begin{aligned} c_1 = \tfrac{(I_y - I_z)I_z - I_{xz}^2}{I_x I_z - I_{xz}^2},\ c_2 = \tfrac{(I_x - I_y + I_z)Ixz}{I_x I_z - I_{xz}^2},\ c_3 = \tfrac{I_z}{I_x I_z - I_{xz}^2} \\ c_4 = \tfrac{I_{xz}}{I_x I_z - I_{xz}^2}, \qquad c_5 = \tfrac{I_z - I_x}{I_y}, \qquad c_6 = \tfrac{I_{xz}}{I_y} \\ c_7 = \tfrac{1}{I_y}, \qquad c_8 = \tfrac{(I_x - I_y)I_x - I_{xz}^2}{I_x I_z - I_{xz}^2},\ c_9 = \tfrac{I_x}{I_x I_z - I_{xz}^2}.\end{aligned} \tag{2.15}$$

The aerodynamics coefficients C_l, C_m, C_n are

$$\begin{aligned}C_l &= C_{l_0}(\alpha, \beta) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\ &\quad + \frac{\bar{B}}{2V_T}(C_{l_p}(\alpha)p + C_{l_r}(\alpha)r) \\ C_m &= C_{m_0}(\alpha, \delta_e) + \frac{\bar{c}}{2V_T}C_{m_q}(\alpha)q + C_z(x_{c.g.ref} - x_{c.g.}) \\ C_n &= C_{n_0}(\alpha, \beta) + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \\ &\quad + \frac{\bar{B}}{2V_T}(C_{n_p}(\alpha)p + C_{n_r}(\alpha)r) - \frac{\bar{c}}{B}C_y(x_{cg_{ref}} - x_{cg}).\end{aligned} \tag{2.16}$$

where $C_{l_0}(\alpha, \beta)$, $C_{l_{\delta_a}}$, $C_{l_{\delta_r}}$, C_{l_p} and C_{l_r} allow to calculate the aerodynamic components considering different flaps and attitude configurations (the meaning is the same for the components of C_m and C_n). The relationships between the angular velocities in body-frame and the Euler angular velocities in Earth-frame are

$$\begin{aligned}\dot{\phi} &= p + \tan\theta(q\sin\phi + r\cos\phi) \\ \dot{\theta} &= q\cos\phi - r\sin\phi \\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}.\end{aligned} \tag{2.17}$$

Concerning the time derivatives of the airspeed, we have

$$\dot{V}_T = \frac{C_x T \cos\alpha \cos\beta}{m} - \frac{C_y}{m} - g \sin\gamma \tag{2.18}$$

where γ is the flight path angle

$$\gamma = \sin^{-1}(\cos\alpha \cos\beta \sin\theta - \sin\alpha \cos\beta \cos\phi \cos\theta - \sin\beta \sin\phi \cos\theta). \tag{2.19}$$

Finally, the angle of attack and the sideslip are governed by

$$\begin{aligned} \dot{\alpha} &= \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \\ \dot{\beta} &= \frac{V_T \dot{v} - v\dot{V}_T}{V_T^2 \sqrt{1 - (\frac{v}{V_T})^2}}. \end{aligned} \tag{2.20}$$

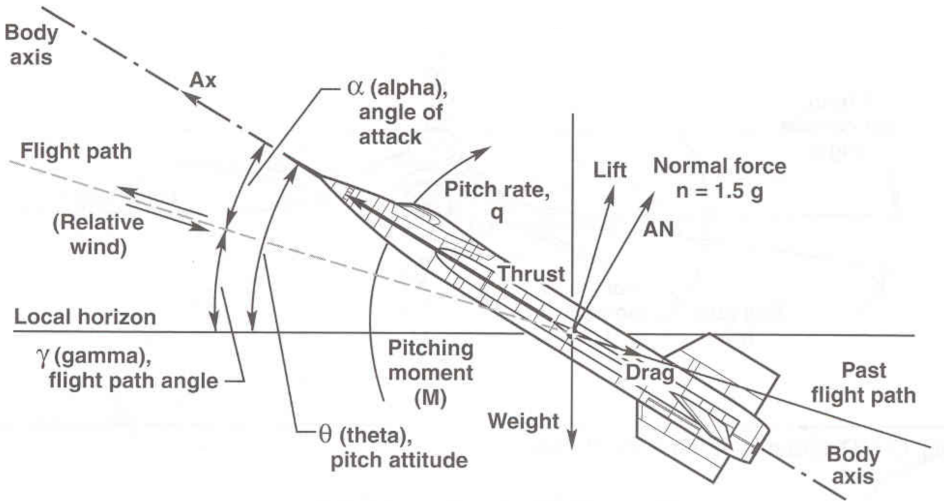


Fig. 2.4: Pitch angle, angle-of-attack, and flight-path angle; Pitching moment, lift, drag and weight. Courtesy of [1].

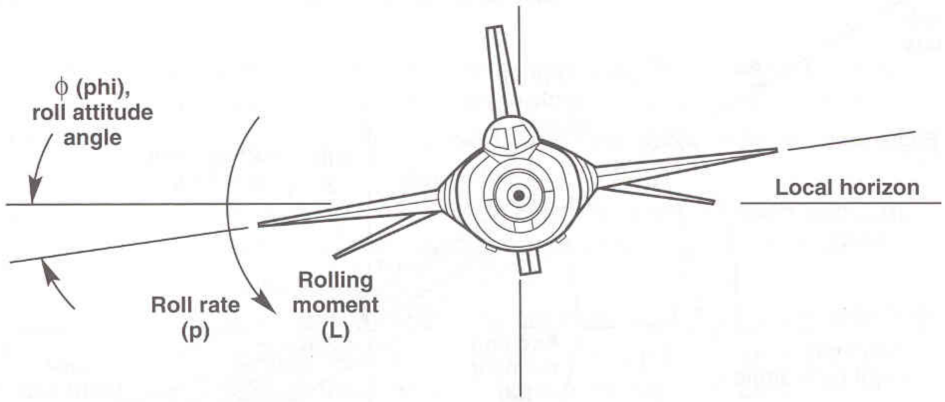


Fig. 2.5: Roll angle and rolling moment. Courtesy of [1].

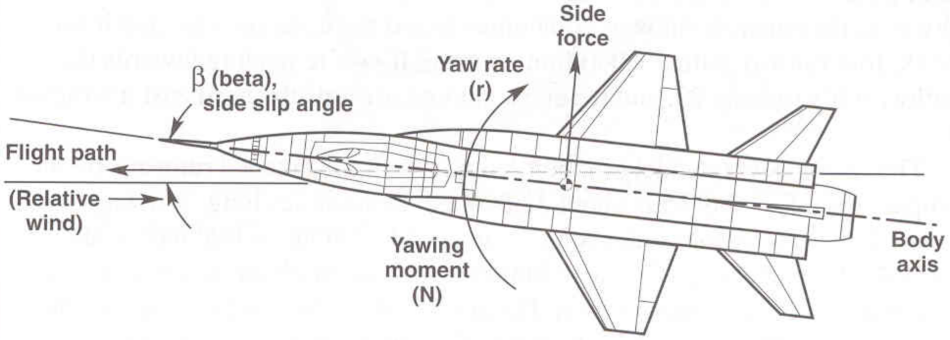


Fig. 2.6: Sideslip angle, yaw rate, side force, and yawing moment. Courtesy of [1].

Chapter 3

STOCHASTIC CONSTRAINED OPTIMAL CONTROL

"The greatest challenge to any thinker is stating the problem in a way that will allow a solution."

Bertrand Russell

Abstract[1] In this chapter, we investigate the formal guarantees that Monte Carlo Tree Search provides when it is designed as a controller, focusing on convergence, optimality analysis, and constraint compliance. We use Monte Carlo Tree Search (MCTS) to solve constrained optimal control problems with/without model uncertainty. We start investigating the formalization of the constrained optimal control problem as MDP. First, we analyze the guarantees that MCTS provides when addressing constrained optimal problems without model uncertainty. We then prove the convergence of MCTS to the optimal value for an infinite horizon. After that, given the infeasibility of infinite iterations, we compute the number of iterations/actions needed to reach an ε-optimality with a δ confidence exploiting Hoeffding's inequality. Second, introducing the model uncertainty, we prove the boundness of the state/action value function error by computing the number of simulations needed to compensate for the uncertainty and reach again the ε-optimality with δ confidence. We conducted two experiments to confirm the theoretical results. In the first one, we compare MCTS with the Linear Quadratic Regulator (LQR) controller on a linear system (e.g., one of the most popular optimal controllers on linear systems able to find the optimal value in closed form). In the second one, we compare the MCTS controller with the Nonlinear Model Predictive Controller (e.g., one of the most used optimal controllers for nonlinear systems) on a nonlinear system to show the capability of MCTS to manage nonlinear systems.

[1] The content of this chapter is based on Trotti, Francesco, Alessandro Farinelli, and Riccardo Muradore. "Stochastic Constrained Optimal Control via Monte Carlo Tree Search", submitted

Summarizing, in this chapter we address:

- The formalization of Monte Carlo Tree Search to solve constrained optimal control problem considering model uncertainty
- The proof of convergence with infinite iterations
- The computation of the number of iterations needed for the ε-optimality with δ confidence without uncertainty
- The robustness against model uncertainty proving the boundness of the state/action value function error by computing the number of iterations needed to reach the ε-optimality with δ confidence.

The chapter is organized as follows: in Section 3.1, we state the problems that we want to solve. In Section 3.2, we explain how Monte Carlo Tree Search works and its formalization to solve an optimal control problem, while in Section 3.3, we present the algorithm convergence. In Section 3.4, we prove the robustness of MCTS against model uncertainty. Finally, we show some simulated results in Section 3.5.

3.1 Problem statement

We consider the Monte Carlo Tree Search (MCTS) strategy to solve the Markov Decision Process (MDP) and the related constrained optimization problem to compute an optimal control for dynamic systems with/without model uncertainty.

Definition 3.1: *The Markov Decision Process (MDP) $\mathcal{M}$ stated in Definition 2.1 is formulated as a constrained optimal control problem where*

1. *States: The state $x \in \mathcal{X}$ represents the state of the system, and $x_{\min}$, $x_{\max} \in \mathcal{X}$ represents the constraints on the state space (component-wise).*
2. *Actions: The actions $u \in \mathcal{U}(x) \subseteq \mathcal{U}$ represents the set of admissible actions or controls available in state x and $u_{\min}$, $u_{\max} \in \mathcal{U}$ represents the control constraints (component-wise).*
3. *Transition model: The transition model is a dynamic system affected by stochastic noise*
$$x_{k+1} = f(x_k) + g(x_k)u_k + B\omega_k \tag{3.1}$$
where $x_k \in \mathbb{R}^n$ is the state at time $t_k = k\Delta t$ where Δt is the sample time, $u_k \in \mathbb{R}^m$ is the control input, and $\omega_k \in \mathcal{W} \in \mathbb{R}^{n_\omega}$ is a stochastic input modeling uncertainty and disturbances (i.e. exogenous input) with $|\omega_k| \leq \kappa$, and $B \in \mathbb{R}^{n \times n_\omega}$ is a constant matrix to select the channels where the disturbance is active.
4. *Cost function: $l(x_k, u_k)$ represents the immediate cost received after taking action u_k in state x_k.*

Let us state the following assumptions about the transition model and cost function:

Assumption 3.1 (State, Control and Disturbance Sets) *The sets $\mathcal{X}, \mathcal{U}$ and $\mathcal{W}$ are closed, bounded and connected subsets of $\mathbb{R}^n, \mathbb{R}^m$ and $\mathbb{R}^{n_\omega}$, respectively.*

Assumption 3.2 (Bound on cost functions) *The cost function $l(x_k, u_k)$ is bounded on $\mathcal{X} \times \mathcal{U}$, i.e., there exists $M > 0$ such that $|l(x_k, u_k)| \leq M$.*

Solving the optimization problem means to compute the optimal pair (x_k^*, u_k^*) over a finite time horizon $[k, k+H]$ at discrete time $t_k = k\Delta t, k \in \mathbb{N}$.

Optimization Problem MDP

$$
\begin{aligned}
&\min_{\boldsymbol{u}} \qquad J(\boldsymbol{x}, \boldsymbol{u}) \\
&\text{subject to:} \\
&x_{k+i+1|k} = f(x_{k+i|k}) + g(x_{k+i|k})u_{k+i|k} + B_\omega w_{k+i|k} \\
&x_{k+i|k} \in \mathcal{X} \qquad x_{\min} \leq x_{k+i|k} \leq x_{\max} \\
&u_{k+i|k} \in \mathcal{U} \qquad u_{\min} \leq u_{k+i|k} \leq u_{\max} \\
&\qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \{0, \dots, H\} \\
&x_{k|k} = x_k
\end{aligned}
\tag{3.2}
$$

where x_k are the initial conditions at the time t_k, $\boldsymbol{x} = (x_{k|k}, \dots, x_{k+H|k})$ and $\boldsymbol{u} = (u_{k|k}, \dots, u_{k-1+H|k})$ are the predicted states and actions time series up to the horizon H, and the cost function J is defined as

$$
J(\boldsymbol{x}, \boldsymbol{u}) = \mathbb{E}\left[\sum_{j=0}^{H-1} \gamma^j l(x_{k+j|k}, u_{k+j|k}) + \phi(x_{k+H|k})\right] \tag{3.3}
$$

with $\phi(\cdot)$ the terminal cost. We indicate the predicted evolution of the state with $k+i|k$ where the right k is the actual time, and the left $k+i$ is the time in the prediction horizon. Given the previous MDP optimal problem formalization, we are interested in studying how the MCTS algorithm solves the optimization problem and, after that, studying the guarantees of the solution provided without model error. The following problems will be addressed

Problem 1: (Convergence to optimality with infinite iterations) Let MCTS be formalized to solve an optimization problem, verify that the estimated state/action value function $\hat{Q}_n(x, u)$ converges to the optimal ones $Q(x, u)$ when the number of iterations n of MCTS goes to infinity.

Given the unfeasibility of infinite iterations, we are interested in studying the number of iterations needed to reach a weak form of optimality.

Definition 3.2 (ε-Optimality with δ-Confidence): *Let $\hat{Q}_n(x, u)$ denote the estimated state/action value function after n iterations of the Monte Carlo Tree Search (MCTS) algorithm, and let $Q(x, u)$ denote the true optimal state-action value function. For a given tolerance $\varepsilon > 0$ and confidence level $1 - \delta$, with $0 < \delta < 1$, we say that the MCTS policy u^* is ε-optimal with δ-confidence if, for all $x \in \mathcal{X}$ and $u \in \mathcal{U}(x)$, the following holds:*

$$
\mathbb{P}\left(\left|\hat{Q}_n(x, u) - Q(x, u)\right| \leq \varepsilon\right) \geq 1 - \delta. \tag{3.4}
$$

This implies that with probability at least $1 - \delta$, the cost associated with the action selected by MCTS is within ε of the optimal cost. The minimum number of iterations n required to satisfy this condition will be addressed in the following problems

Problem 2: (ε-Optimality with δ Confidence) Let MCTS be formalized to solve an optimization problem, we compute the minimum number of iterations such that the estimated cost of the state/action value function converges to the ε-optimal value with δ confidence.

After that, we will study the behaviors of MCTS when model uncertainty is introduced in the optimization problem.

Problem 3: (Boundness of the cost functions error due to model uncertainty) Let MCTS be formalized to solve an optimization problem, we prove that the error between the real and estimated state/action value function in the presence of model uncertainty is bounded by a finite value.

Problem 4: (ε-Optimality with δ Confidence with model uncertainty) Let MCTS be formalized to solve an optimization problem, compute the minimum number of iterations such that the estimated cost of the state/action value function converges to the ε-optimal values with δ confidence.

3.2 Optimization Problem with Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm that balances exploration and exploitation through a combination of random sampling, and tree-based search [33], making it a useful method for exploring large and uncertain state space. The algorithm operates iteratively and can be broken down into four distinct phases, each refining the search towards optimal or near-optimal actions [34]. These four phases, Selection, Expansion, Simulation, and Backpropagation, have been explained in Section 2.2 and will be reformulated in this Section following the control theory formalism. Each phase plays a crucial role in guiding the search process, balancing the exploration of new areas of the state space, and the exploitation of known promising paths to compute the optimal action. The following subsections explain and formalize how the MCTS solves the optimal control problem, but firstly, we will detail how the action space is represented.

3.2.1 Actions space

The action space is represented through a multivariate Gaussian distribution of the control variables limited by the control constraints $(u_{\min}, u_{\max})$. Let's define $\mu \in \mathbb{R}^m$ as the mean vector of the control variables and let $\Sigma \in \mathbb{R}^{m\times m}$ be its covariance matrix. The probability density function (PDF) of the multivariate Gaussian distribution $u \sim \mathcal{N}(\mu, \Sigma)$ is

$$f(u) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp\left(-\frac{1}{2}(u-\mu)^T \Sigma^{-1}(u-\mu)\right) \tag{3.5}$$

and the PDF of its truncated version is given by

$$f_T(u) = \begin{cases} \frac{f(u)}{\int_{u_{\min}}^{u_{\max}} f(u)du}, & u \in [u_{\min}, u_{\max}] \\ 0, & u \notin [u_{\min}, u_{\max}] \end{cases} \tag{3.6}$$

Update rule for action space

The new optimal control variable u^* (how u^* is computed will be discussed later) is used to update the mean and covariance matrix of the multivariate distribution. This policy gives more weight to the action near the optimal value rather than the others. A gradient-based update [62, 69] for the mean and covariance is used

$$\begin{aligned} \mu' &= \mu + \eta(u^* - \mu) \\ \Sigma' &= \Sigma + \eta((u^* - \mu)(u^* - \mu)^T - \Sigma) \end{aligned} \tag{3.7}$$

where μ' and Σ' are the updated mean and covariance matrix of the distribution, and η is the learning rate controlling how much the mean and covariance should shift toward the optimal action.

Relation between η and ε

Defining n the number of iterations needed to reach an ε-optimality with and without model uncertainty, the next proposition correlates the learning rate η of the multivariate distribution with ε. Since ε controls how close the value of $\hat{Q}(\hat{x}, u)$ is to the optimal value $Q(x, u)$, and η controls the magnitude of updates to the distribution, we want to adjust η so that the updates become more conservative as the algorithm approaches the ε-optimal solution avoiding overshoots.

Proposition 3.1: *Let $\mu^{(n)}$ represent the mean of the multivariate distribution at iteration n and let u^* denote the optimal action. Let the learning rate $\eta(n)$ for updating $\mu^{(n)}$ be define equal to*

$$\eta(n) = \frac{\varepsilon}{n}. \tag{3.8}$$

For finite n, we have

$$|\mu^{(n)} - u^*| \leq O\left(\frac{\varepsilon}{n}\right). \tag{3.9}$$

Proof: Since the harmonic series $\sum_{n=1}^{\infty} \frac{1}{n}$ diverges, so

$$\sum_{n=1}^{\infty} \eta(n) = \varepsilon \sum_{n=1}^{\infty} \frac{1}{n} \to \infty. \tag{3.10}$$

Moreover, considering that the sum of squared terms of the harmonic series converges

$$\sum_{n=1}^{\infty} \eta(n)^2 = \varepsilon^2 \sum_{n=1}^{\infty} \left(\frac{1}{n}\right)^2 < \infty, \tag{3.11}$$

then the mean updating rule that controls how much the mean $\mu^{(n)}$ moves toward the optimal action u^* is

$$\mu^{(n+1)} = \mu^{(n)} + \frac{\varepsilon}{n}(u^* - \mu^{(n)}). \tag{3.12}$$

The error $|\mu^{(n)} - u^*|$ can be bounded as

$$|\mu^{(n+1)} - u^*| \leq \left(1 - \frac{\varepsilon}{n}\right) |\mu^{(n)} - u^*|. \tag{3.13}$$

As n increases, the term $(1 - \frac{\varepsilon}{n})$ becomes smaller, decreasing the error. When the number of iterations n is finite, the cumulative effect leads to an error bound

$$|\mu^{(n)} - u^*| \leq O\left(\frac{\varepsilon}{n}\right). \tag{3.14}$$

indicating the error decreases proportionally to $O\left(\frac{\varepsilon}{n}\right)$. ■

3.2.2 Optimization problem

The MCTS algorithm solves a constrained optimal control problem by iterating n times on the four phases (the number of needed iterations n to reach the optimality will be discussed later). The optimization problem formulated in (3.2) can be rewritten in the following way so that it can be solved by the MCTS algorithm at any discrete time $t_k = k\Delta t, k \in \mathbb{N}$.

Optimization Problem using MCTS

$$\min_{\boldsymbol{u}} \quad \text{UCT}(x_{k+i|k}, u_{k+i|k}) \tag{3.15a}$$

subject to:

$$x_{k+i+1|k} = f(x_{k+i|k}) + g(x_{k+i|k})u_{k+i|k} + B_\omega w_{k+i|k} \tag{3.15b}$$

$$x_{k+i|k} \in \mathcal{X} \qquad x_{\min} \leq x_{k+i|k} \leq x_{\max} \tag{3.15c}$$

$$u_{k+i|k} \in \mathcal{U} \qquad u_{\min} \leq u_{k+i|k} \leq u_{\max}$$

$$w_{k+i|k} \in \mathcal{W} \tag{3.15d}$$

$$i \in \{0, \ldots, H-1\}$$

$$x_{k|k} = x_k$$

where x_k is the initial condition, and $\text{UCT}(\cdot,\cdot)$ will be explained below. The pair $(x_{k+i|k}, u_{k+i|k})$ are the state $x_{k+i|k}$ and action $u_{k+i|k}$ at time $k+i$, while $\boldsymbol{u}$ is the trajectory of the actions along the horizon H ($\boldsymbol{u} = (u_{k|k}, u_{k+1|k}, \ldots, u_{k+H-1|k})$). The four phases of MCTS are based on the exploration and expansion of a tree and solve the previous optimization problem in the following way.

Selection. In the selection phase, the algorithm starts from the root node (initial condition of the system) and recursively selects child nodes that minimize a specific selection policy until it reaches a leaf node. The selection process is based on the Upper Confidence Bound for Trees (UCT) [34], [33], which balances exploration and exploitation. The UCT equation for a given action $u_{k+i|k}$ from state $x_{k+i|k}$ at the time $k+i$ is

$$\text{UCT}(x_{k+i|k}, u_{k+i|k}) = \hat{Q}_n(x_{k+i|k}, u_{k+i|k}) + C\sqrt{\frac{\log N(x_{k+i|k})}{N(x_{k+i|k}, u_{k+i|k})}} \tag{3.16}$$

where C is a trade-off constant between exploration and exploitation, $N(x_{k+i|k})$ is the number of times the state $x_{k+i|k}$ has been visited at the time $k+i$ in the horizon, $N(x_{k+i|k}, u_{k+i|k})$ is the number of times action $u_{k+i|k}$ has been taken from state

$x_{k+i|k}$ at the time $k+i$ in the horizon; $\log N(x_{k+i|k})$ ensures that less frequently visited states/actions are explored, and the square root term ensures exploration diminishes over time as more iterations are run. $\hat{Q}_n(x_{k+i|k}, u_{k+i|k})$ is the state/action value function that is computed as the average cost based on all previous iterations that have passed through this particular state-action pair and are given by

$$\hat{Q}_n(x_{k+i|k}, u_{k+i|k}) = \frac{1}{N(x_{k+i|k}, u_{k+i|k})} \sum_{j=1}^{N(x_{k+i|k}, u_{k+i|k})} J_j(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k}) \tag{3.17}$$

where $J_j(\cdot,\cdot)$ will be explained in detail in the simulation phase. This process is repeated until the algorithm reaches a leaf node x_{leaf}, where no further actions have been explored. When the algorithm reaches the defined number of iterations n the trajectories $(\boldsymbol{x}, \boldsymbol{u})$ are generated

$$\begin{aligned} \boldsymbol{x} &= (x_{k|k}, x_{k+1|k}, \dots, x_{k+H|k}) \\ \boldsymbol{u} &= (u_{k|k}, u_{k+1|k}, \dots, u_{k+H-1|k}), \end{aligned} \tag{3.18}$$

Therefore, the selection phase using UCT computes the optimal action u^* that solves the minimization problem

$$u^* = \underset{\boldsymbol{u}}{\operatorname{argmin}} \quad \text{UCT}(x, u). \tag{3.19}$$

Expansion. In this phase, the algorithm expands the search tree by adding one or more child nodes by taking ν samples from the multivariate distribution defined in (5.30)-(3.6). Formally, for each action u sampled from the distribution $\mathcal{N}(\mu, \Sigma)$, a new state x is added to the tree following the dynamic model (3.15b). This phase allows the algorithm to explore new regions of the state space. The constraints on the states are also checked (3.15c), if the new state computed by the sampling action does not satisfy the constraints, this state is discarded, and another action is sampled to check another new state. This approach enables us to focus solely on states that satisfy the constraints, effectively eliminating actions that would lead the system into infeasible or invalid states.

Simulation. The simulation phase, or rollout, starting from a state $x_{k+i|k}$ of the tree involves alternating a sequence of state $\tilde{x}$ and random action $\tilde{u} \sim \mathcal{N}(\mu, \Sigma)$ exploiting the transition model (3.15b). This process is repeated until the terminal state or a predefined rollout depth (D) is reached, yielding a sequence of states and actions

$$\begin{aligned} \tilde{\boldsymbol{x}}_{k+i|k} &= (\tilde{x}_{k+i+d|k}, \dots, \tilde{x}_{k+i+D|k}) \\ \tilde{\boldsymbol{u}}_{k+i|k} &= (\tilde{u}_{k+i+d|k}, \dots, \tilde{u}_{k+i+D-1|k}) \\ d &\in \{0, \dots, D-1\} \end{aligned} \tag{3.20}$$

where for $d = 0$ we set $\tilde{x}_{k+i|k} = x_{k+i|k}$ and $\tilde{u}_{k+i|k} = u_{k+i|k}$. At the end of the simulation, the algorithm computes a cumulative cost $J(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k})$ (used in equation (3.17)) based on the outcome of the simulated trajectory $(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k})$. The discounted cumulative cost is defined as

$$J(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k}) = \mathbb{E}\left[\sum_{d=0}^{D-1} \gamma^d l(\tilde{x}_{k+i+d|k}, \tilde{u}_{k+i+d|k}) + \phi(\tilde{x}_{k+i+D|k})\right] \tag{3.21}$$

where $\gamma \in (0, 1]$ is a discount factor, as defined in Section 2.1, $l(\tilde{x}_{k+i+d|k}, \tilde{u}_{k+i+d|k})$ is the immediate cost defined in Section 3.1 and $\phi(\tilde{x}_{k+i+D|k})$ is the final cost. Note that during the rollout process, the MCTS tree is frozen, and when the rollout is terminated, the cumulative cost is computed, and the explored nodes in the rollout are pruned.

Backpropagation. In the backpropagation phase, the state/action values of the node $\hat{Q}(x_{k+i|k}, u_{k+i|k})$ is updated using the new cumulative cost $J(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k})$ obtained from the rollout. The visit counts $N(\cdot)$ and $N(\cdot, \cdot)$ of the nodes of the selected path from the root to the leaf are incremented. This process ensures that the tree is incrementally improved as new iterations are run, with actions leading to higher rewards becoming increasingly favored.

Summarizing, we can say that MCTS works using two types of horizons: prediction and evaluation. The *prediction horizon* is the depth of the tree (H) and defines the length of the state/action trajectories (3.18), while the *evaluation horizon* is the depth of the rollouts (D) that allows the exploration of the evolution of a state in the future starting from a state $x_{k+i|k}$ (3.20).

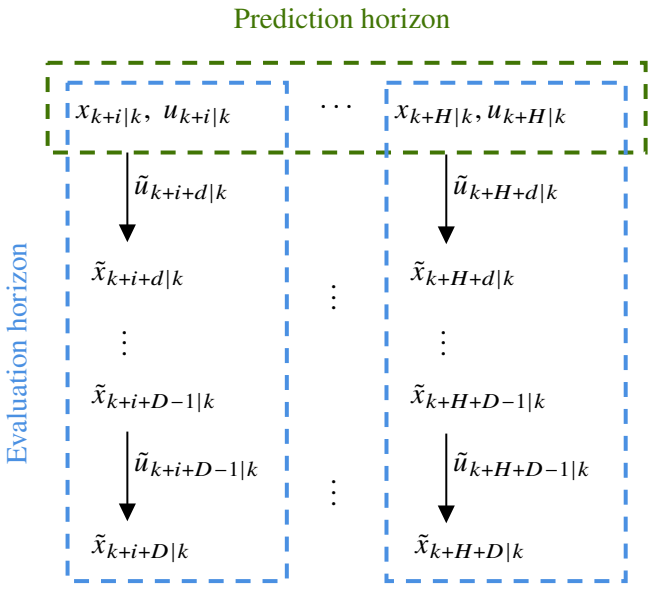


Fig. 3.1: Monte Carlo Tree Search horizons time evolution.

Figure 3.1 summarizes the evolution of the prediction and evaluation horizons of MCTS during the optimization process. The green box represents the prediction horizon where, for each pair $(x_{k+i|k}, u_{k+i|k})$ at time $k+i$, starts an evaluation horizon (blue box).

Figure 3.2 shows the horizons from the tree point of view. The green path represents the prediction horizon with depth H, while the blue path shows the evaluation horizon with depth D. The figure shows a snapshot of the MCTS algorithm where UCT iteratively selects the green path, and the simulation phase starts from

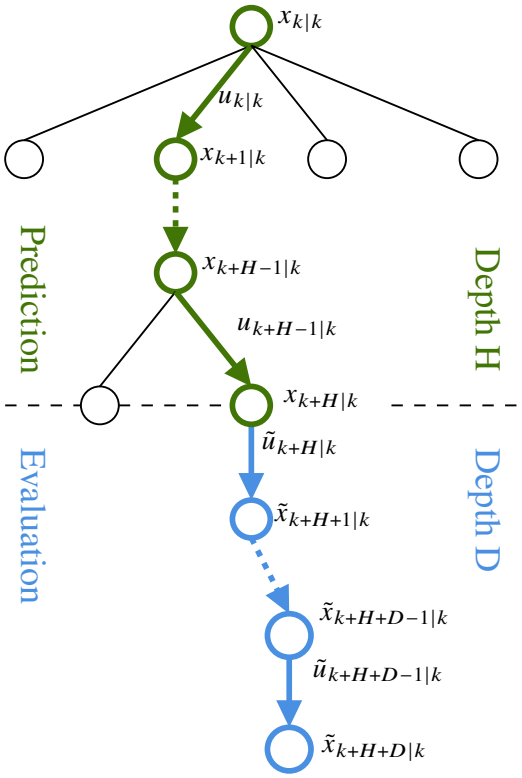


Fig. 3.2: Monte Carlo Tree Search horizons tree evolution.

the node with state $x_{k+H|k}$. In other words, the depth of the tree corresponds to different timestamps in the future (e.g., layer 2 is equal to the timestamp $k+2$) while the amplitude of the layer represents the state/action space exploration.

Note that, from each node of MCTS different rollouts are iteratively executed. The cumulative cost of each rollout only depends on the sequence of the randomly taken actions and on the underlying dynamics model. As demonstrated in [34], we can notice that the result of each rollout $J(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k})$, i.e., the cumulative cost of the path executed, only depends on the sum of the immediate cost of each randomly explored node in that rollout. Hence, we have independent cumulative costs. To see this in practice, considering the example in Figure 3.2, the cumulative cost obtained from the blue path is not influenced by the outcome of the cumulative cost of other nodes; for this reason, we can say that each cumulative cost $J(\cdot,\cdot)$ is independent. Also, since the dynamic model and the cost formulation are identical in each rollout, we can also say that the cumulative costs are identically distributed. Therefore, following this formulation of MCTS, we can say that the cumulative costs are independent and identically distributed (i.i.d)

3.3 Convergence analysis without model uncertainty

We start analyzing the convergence criteria to the optimal value considering infinite iterations. To prove the convergence, it is important to consider two main aspects: the convergence to the true action-value function $Q(x, u)$ and the sufficient exploration to guarantee that UCT (3.16) selects the optimal action u_k^*. Given the infeasibility of infinite iterations in real applications, it is relevant to compute the number of actions per node needed to reach an ε-optimality with defined confidence δ and then compute the number of iterations needed to obtain the ε-optimality with δ confidence for the entire tree. For the current analysis, the model uncertainty is not considered; therefore, no disturbance ω affects the system.

3.3.1 Optimality with infinite iterations

Let $Q(x, u)$ be the state/action value function representing the expected total reward from state x taking action u, and let $\hat{Q}_n(x, u)$ be the estimated value of $Q(x, u)$ after n iterations. Let $N(x)$ be the number of times state x has been visited, and $N(x, u)$ be the number of times action u has been taken from state x. The following propositions and theorem prove the convergence to the true state/action value function $Q(x, u)$ and the sufficient exploration of UCT.

Proposition 3.2 (Convergence to $Q(x, u)$): *For any state $x \in \mathcal{X}$ and action $u \in \mathcal{U}$, suppose that the cost $J(x, u)$ are independent and identically distributed (i.i.d.) random variables with finite expectation and variance. Then the estimate $\hat{Q}_n(x, u)$ converges to $Q(x, u)$ almost surely as $n \to \infty$.*

Proof: The estimate $\hat{Q}_n(x, u)$ is defined as

$$\hat{Q}_n(x, u) = \frac{1}{N(x, u)} \sum_{i=1}^{N(x,u)} (J_i(x, u)) \tag{3.22}$$

where $J_i(x, u)$ represents the cost obtained in the i-th rollout starting from (x, u). Since the cumulative costs $J_i(x, u)$ are i.i.d with finite expectation $Q(x, u) = \mathbb{E}[J(x, u)]$ and variance $\sigma^2 = \text{Var}[J(x, u)] < \infty$, we can apply the Strong Law of Large Numbers (SLLN) [70, 71]. The SLLN states that the sample mean of an i.i.d. sequence with finite expectation converges almost surely to the true expectation. Recalling the i.i.d property, the cost for each rollout depends only on the sequence of actions taken and the underlying dynamic model, the results of each rollout are independent. Furthermore, since all rollouts from a given state-action pair (x, u) share the same dynamic model and cost structure, they are identically distributed. Therefore, the cost sequences are independent and identically distributed (i.i.d.). Thus, by the SLLN

$$\hat{Q}_n(x, u) = \frac{1}{N(x, u)} \sum_{i=1}^{N(x,u)} J_i(x, u) \xrightarrow{a.s.} Q(x, u) \quad \text{as} \quad n \to \infty \tag{3.23}$$

■

Proposition 3.3 (Sufficient Exploration): *For any state $x \in \mathcal{X}$ and action $u \in \mathcal{U}$, suppose that the number of times that the action u is taken from state x, denoted by $N(x,u)$, grows indefinitely as $N(x) \to \infty$. Then*

$$N(x,u) = \Omega(\log N(x)). \tag{3.24}$$

Proof: : We prove by contradiction. Suppose that $N(x,u)$ does not grow at least logarithmically,

$$N(x,u) = o(\log N(x)) \tag{3.25}$$

By the definition of little-o notation [2], this means that

$$\lim_{N(x)\to\infty} \frac{N(x,u)}{\log N(x)} \to 0. \tag{3.26}$$

This implies that $N(x,u)$ grows strictly slower than $\log N(x)$. The exploration term in UCT (3.16) for action u is

$$C\sqrt{\frac{\log N(x)}{N(x,u)}} \tag{3.27}$$

where $C > 0$ is a constant. Substituting $N(x,u) = o(\log N(x))$, we get that the denominator $N(x,u)$ grows strictly slower than $\log N(x)$, which implies

$$\lim_{N(x)\to\infty} C\sqrt{\frac{\log N(x)}{N(x,u)}} \to \infty. \tag{3.28}$$

Thus, the exploration term remains large indefinitely, forcing action u to be selected more frequently in future iterations. This contradicts our assumption that $N(x,u)$ grows sub-logarithmically. Hence, we must have

$$N(x,u) = \Omega(\log N(x)) \tag{3.29}$$

so the ratio $\frac{\log N(x)}{N(x,u)}$ remains finite for $N(x) \to \infty$. This ensures that every action is explored sufficiently often so that the exploration term remains bounded

$$C\sqrt{\frac{\log N(x)}{N(x,u)}} < \infty. \tag{3.30}$$

■

[2] Concerning the O, o, Ω and the asymptotic notations, we refer to [72].
$f(n) = O(g(n))$ means that $f(n)$ grows at most as fast as $g(n)$ asymptotically, i.e., there exist positive constants c and n_0 such that $|f(n)| \leq cg(n)$ for all $n \geq n_0$.
$f(n) = o(g(n))$ means that $f(n)$ grows strictly slower than $g(n)$ asymptotically, i.e., for any positive constant c, there exists an n_0 such that $|f(n)| < cg(n)$ for all $n \geq n_0$.
$f(n) = \Omega(g(n))$ means that $f(n)$ grows at least as fast as $g(n)$ asymptotically, i.e., there exist positive constants c and n_0 such that $|f(n)| \geq cg(n)$ for all $n \geq n_0$

Theorem 3.1: *As the number of iterations $n \to \infty$, the probability that the action selected by UCT is the optimal action converges to* 1*,*

$$\lim_{n\to\infty} \mathbb{P}\left(\operatorname*{argmin}_u \hat{Q}_n(x,u) = \operatorname*{argmin}_u Q(x,u)\right) = 1. \tag{3.31}$$

Proof: We establish this result using the following steps:

1. *Convergence*: From Proposition 1, we know that

$$\hat{Q}_n(x,u) \xrightarrow{a.s.} Q(x,u) \quad \forall x \in \mathcal{X}, \forall u \in \mathcal{U} \tag{3.32}$$

   This means that, as $n \to \infty$, the empirical estimates $\hat{Q}_n(x,u)$ become arbitrarily close to their true values $Q(x,u)$ almost surely.
2. *Sufficient Exploration*: From Proposition 2, we know that

$$N(x,u) = \Omega(\log N(x)), \quad \forall x \in \mathcal{X}, \forall u \in \mathcal{U}. \tag{3.33}$$

   This guarantees that every action is explored infinitely often, preventing premature bias toward suboptimal actions.

Since $\hat{Q}_n(x,u)$ converges almost surely to $Q(x,u)$, the argmin function ensures that

$$\operatorname*{argmin}_u \hat{Q}_n(x,u) \xrightarrow{a.s.} \operatorname*{argmin}_u Q(x,u). \tag{3.34}$$

Thus, the probability that UCT selects the optimal action converges to 1

$$\lim_{n\to\infty} \mathbb{P}\left(\operatorname*{argmin}_u \hat{Q}_n(x,u) = \operatorname*{argmin}_u Q(x,u)\right) = 1. \tag{3.35}$$

Consequently, the UCT (3.16) algorithm selects the optimal action with probability one as the number of iterations becomes large. ■

3.3.2 Boundness of iterations number

Stated the asymptotic convergence to the optimal action with infinite iterations, the following theorem addresses the problem of computing the number of actions and iterations needed to reach an ε-optimality with a δ confidence.

Theorem 3.2 (ε-Optimality with δ Confidence): *Let ν be the number of actions to be sampled per node, H be the depth of the tree (i.e., the prediction horizon), ε be the admissible error to the optimal value, δ be the desired confidence level of the ε error, and α be the exponential decay factor representing the effective reduction in the number of nodes explored at greater depths of the tree. The total number of MCTS iterations n_{itr} required to ensure that the selected path $\boldsymbol{u}$ via UCT is ε-optimal with δ confidence is given by*

$$n_{itr} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\alpha i} \right) \frac{\log(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.36}$$

Proof: We begin by applying Hoeffding's inequality (Section 2.5) to determine the number of iterations required to ensure, with probability at least δ, that the estimated action-value function $\hat{Q}_n(x,u)$ is within ε of the true value $Q(x,u)$ for each action u at a given node x. The independent random variables X_i of the Hoeffding definition correspond to the cumulative costs (3.21) ($J(\tilde{\boldsymbol{x}}_{k+i|k}, \tilde{\boldsymbol{u}}_{k+i|k})$ at the time $k+i$ with $i \in \{0,\ldots,H-1\}$) obtained from simulations in the rollout phases. In particular, we can state that the cumulative costs are i.i.d because the cumulative cost for each rollout depends only on the sequence of actions taken, and the cumulative cost of each rollout is not influenced by the outcomes of previous rollouts and, therefore, are independent. All rollouts that start from a state/action pair $(x_{k+i|k}, u_{k+i|k})$ at time $k+i$ share the same cost function and dynamics, and so, the cumulative costs are identically distributed. Therefore, we can state that the cumulative costs are independent and identically distributed (i.i.d.). Moreover, by Assumption 3.2, these costs can be normalized within 0 and 1 knowing the minimum and maximum bound ($a=0, b=1$). Therefore, it is possible to state

$$\mathbb{P}(|\hat{Q}_n(x,u) - Q(x,u)| \geq \varepsilon) \leq 2e^{-\frac{2n\varepsilon^2}{(b-a)^2}}. \tag{3.37}$$

To ensure that the estimate of $\hat{Q}(x,u)$ for each action u at a node x is within ε of the true values $Q(x,u)$ with δ confidence, we set

$$2e^{-\frac{2n\varepsilon^2}{(b-a)^2}} \leq \delta. \tag{3.38}$$

Solving for n, the total number of iterations required per node is

$$n_{\text{node}} \geq \frac{\log(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.39}$$

Since n_{node} represents the number of actions sampled (ν) and tested at a given node, we extend this bound to the entire tree by considering the impact of the UCT (3.27) selection mechanism. Given a tree depth H, the total number of nodes

in a fully expanded tree would be $N = \sum_{i=0}^{H} \nu^i$. However, due to the logarithmic selective exploration nature of UCT, the effective number of explored nodes follows an exponential decay model

$$N = \sum_{i=0}^{H} \nu^i e^{-\alpha i} \tag{3.40}$$

where α is the decay factor. Consequently, the total number of iterations n_{itr} required to achieve ε-optimality with δ confidence across the entire MCTS tree is

$$n_{\text{itr}} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\alpha i} \right) \frac{\log(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.41}$$

■

Given the tree-like nature of the algorithms and modeling the exploration as an exponential decay factor, the following remark provides the relationship between the decay rate α and the exploration-exploitation constant C of UCT.

Remark 1 (Relationship α - C): The exploration - exploitation trade-off parameter C in the UCT formula (3.16) directly influences the decay rate α of node exploration in MCTS. This relationship can be expressed as

$$\alpha \propto \frac{1}{C}. \tag{3.42}$$

and stems from the role of the exploration term in the UCT algorithm

$$C\sqrt{\frac{\log N(x)}{N(x,u)}} \tag{3.43}$$

The term encourages visiting under-explored actions by increasing their estimated values. A larger C amplifies the exploration component, leading to a more thorough search of the tree and, consequently, a slower decay in the probability of visiting deeper nodes (i.e., smaller α). Conversely, by reducing C, the emphasis on exploration diminishes, resulting in faster decay in node exploration as the depth increases. Using this relationship, the total number of MCTS iterations n_{itr} required to achieve a ε error with δ confidence can be bounded as

$$n_{\text{itr}} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\frac{i}{C}} \right) \frac{\log(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.44}$$

Here, the exponential decay term $e^{-\frac{i}{C}}$ captures the depth-dependent exploration determined by C, while ν^i represents the number of actions considered at depth i.

3.4 Robustness to model uncertainty

This section considers the dynamic system with model error in the transition model. We are interested in analyzing the convergence of the algorithm to the ε-optimality with a δ confidence. For the analysis, we focus on two aspects: 1) computation of the bound due to the model uncertainty on the state/action value function error using the recursive Bellman equation, and 2) computation of the number of iterations needed to compensate for the uncertainty and reach an ε-optimality with a δ confidence.

3.4.1 Boundness of the value functions error

Let the transition model of equation (3.1) be rewritten as $x' = T(x, u)$ without model error (i.e., $B = 0$); then let $\hat{x}' = \hat{T}(\hat{x}, u) = T(\hat{x}, u) + \omega_k$ be the transition model affected by the model error, where x' and $\hat{x}'$ are the next states provided by $T(x, u)$ and $\hat{T}(\hat{x}, u)$, respectively. Let $J(x, u)$ be the reward function defined in (3.3) and $\gamma \in (0, 1]$ the discount factor.

Theorem 3.3: *Let $Q(x, u)$ and $\hat{Q}(\hat{x}, u)$ be the state/action value functions generated using $T(x, u)$ and $\hat{T}(\hat{x}, u)$ respectively, and let κ be the bound of the model uncertainty. Then, the error in the value function due to the transition model error for a finite horizon H is bounded by*

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \frac{\kappa(1 - \gamma^H)}{1 - \gamma}. \tag{3.45}$$

Proof: The proof is done by induction. Let's use the recursive Bellman equation to write the state/action value functions using the nominal end stochastic models

$$\begin{aligned} Q(\hat{x}, u) &= l(\hat{x}, u) + \gamma \mathbb{E}_{\substack{x' \sim T(\hat{x}, u) \\ u' \sim \mathrm{UCT}(x)}} [Q(x', u')] \\ \hat{Q}(\hat{x}, u) &= l(\hat{x}, u) + \gamma \mathbb{E}_{\substack{\hat{x}' \sim \hat{T}(\hat{x}, u) \\ u' \sim \mathrm{UCT}(\hat{x})}} [\hat{Q}(\hat{x}', u')] \end{aligned} \tag{3.46}$$

where $'$ represents the next step of the recursion. Focusing only on the expectation of the next state for now, the transition model $\hat{T}(\hat{x}, u)$ is the stochastic model, where the additive noise ω_k introduces stochasticity, leading to a distribution over possible next states $\hat{x}'$ [73–75]. This means the expectation over $\hat{x}'$ can be written as a weighted sum

$$\mathbb{E}_{\hat{x}' \sim \hat{T}(\hat{x}, u)}[\hat{Q}(\hat{x}', u')] = \sum_{\hat{x}'} \hat{T}(\hat{x}, u) \hat{Q}(\hat{x}', u') \tag{3.47}$$

where u' represents the selected action for state x'. Substituting into the second equation of (3.46), we have

$$\hat{Q}(\hat{x}, u) = l(\hat{x}, u) + \gamma \sum_{\hat{x}'} \hat{T}(\hat{x}, u) \mathbb{E}_{u' \sim \mathrm{UCT}(\hat{x})}[\hat{Q}(\hat{x}', u')] \tag{3.48}$$

The action $u' \sim \mathrm{UCT}(\hat{x})$ is policy-dependent. This means that u' is determined entirely by the UCT algorithm acting on $\hat{x}'$. This allows the expectation over u' to be considered in the summation over $\hat{x}'$, obtaining the equation (3.47). We can

rewrite the equation of the estimated value function in order to express explicitly the transition model in the following way

$$\hat{Q}(\hat{x}, u) = l(\hat{x}, u) + \gamma \sum_{\hat{x}'} \hat{T}(\hat{x}, u)\hat{Q}(\hat{x}', u') \tag{3.49}$$

and then by substituting the stochastic transition model $\hat{T}(\hat{x}, u)$, we end up with

$$\hat{Q}(\hat{x}, u) = l(\hat{x}, u) + \gamma \sum_{\hat{x}'} (T(\hat{x}, u) + \omega)\hat{Q}(\hat{x}', u'). \tag{3.50}$$

Finally, expanding the terms we obtain

$$\hat{Q}(\hat{x}, u) = \hat{Q}(\hat{x}, u) + \gamma \sum_{\hat{x}'} T(\hat{x}, u)\hat{Q}(\hat{x}', u') + \gamma \sum_{\hat{x}'} \omega\hat{Q}(\hat{x}', u'). \tag{3.51}$$

Proceed defining the error between the true value function and the estimated one as

$$\begin{aligned} \hat{Q}(\hat{x}, u) - Q(\hat{x}, u) = &\left[l(\hat{x}, u) + \gamma \sum_{\hat{x}'} T(\hat{x}, u)\hat{Q}(\hat{x}', u') + \gamma \sum_{\hat{x}'} \omega\hat{Q}(\hat{x}', u') \right] \\ &- \left[l(\hat{x}, u) + \gamma \sum_{x'} T(x, u)Q(x', u') \right]. \end{aligned} \tag{3.52}$$

We now use the triangle inequality and rearrange the terms. Hence, the error equation becomes

$$\begin{aligned} |\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq &\gamma \sum_{x'} T(x, u)|\hat{Q}(\hat{x}', u') - Q(x', u')| \\ &+ \gamma \sum_{x'} |\omega||\hat{Q}(\hat{x}', u')|. \end{aligned} \tag{3.53}$$

Let's proceed with the recursive error equation for the next step. We can identify in (3.53) the propagated error component

$$\gamma \sum_{x'} T(x, u)|\hat{Q}(\hat{x}', u') - Q(x', u')| \tag{3.54}$$

which depends on the error at the previous step and the immediate error caused by the model error

$$\gamma \sum_{x'} |\omega||\hat{Q}(\hat{x}', u')|. \tag{3.55}$$

Recalling that $|\omega| \leq \kappa$ and assuming a bound on the state/action value function $|Q(x', u')| \leq Q_{\max} = \frac{l_{\max}(x,u)}{1-\gamma}$, the immediate error term becomes

$$\gamma \sum_{x'} |\omega||\hat{Q}(\hat{x}', u')| \leq \gamma\kappa Q_{\max}. \tag{3.56}$$

This term reflects the direct contribution of model uncertainty. The propagated error is the contribution of the difference $|\hat{Q}(\hat{x}', u') - Q(x', u')|$ from the next state to the current state. Using the recursive structure of the Bellman equation

$$|\hat{Q}(\hat{x}', u') - Q(x', u')| \leq \gamma \sum_{x''} T(x', u') |\hat{Q}(\hat{x}'', u'') - Q(x'', u'')| + \gamma \kappa Q_{\max} \quad (3.57)$$

where $''$ represents the second step of the recursion. Analyzing the recursion on multiple steps, the recursive error at the current step will depend on the error at the next step, which will depend on the error at the step after, and so on. Expanding this in the finite horizon H, we get

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \gamma \kappa Q_{\max} + \gamma^2 \kappa Q_{\max} + \gamma^3 \kappa Q_{\max} + \cdots + \gamma^H \kappa Q_{\max} \quad (3.58)$$

This is a geometric series, where each term is scaled by γ. Summing up the series gives

$$\text{Total propagated error} = \sum_{i=0}^{H-1} \gamma^i \kappa Q_{\max} = \kappa Q_{\max} \frac{1 - \gamma^H}{1 - \gamma}. \quad (3.59)$$

Therefore, combining the current error and the total propagated error, we have

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \frac{\kappa Q_{\max}(1 - \gamma^2)}{1 - \gamma} + \gamma \kappa Q_{\max} \leq \frac{\kappa Q_{\max}(1 - \gamma^H)}{1 - \gamma}. \quad (3.60)$$

For $\gamma \in (0, 1]$ and combining $Q_{\max}$ in κ since they are both constants, the induction hypotheses hold. Therefore, the error bound is

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \frac{\kappa(1 - \gamma^H)}{1 - \gamma}. \quad (3.61)$$

This bound is a function of the depth H. As $H \to \infty$ the term $1 - \gamma^H$ approaches 1, recovering the following infinite depth bound

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \frac{\kappa}{1 - \gamma}. \quad (3.62)$$

■

3.4.2 Boundness of iterations number with model uncertainty

The following theorem provides the number of iterations needed to MCTS to compensate for the model uncertainty, ensuring an ε-optimality with a δ confidence.

Theorem 3.4: *Let ε_T be an admissible error bound that accounts for both model uncertainty and finite MCTS iterations. The total number of MCTS iterations required to ensure that the selected actions are ε_T-optimal with δ confidence is*

$$n_{itr} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\frac{i}{C}} \right) \frac{\nu \log(\frac{2}{\delta})}{2 \left(\varepsilon_T - \frac{\kappa(1-\gamma^H)}{1-\gamma} \right)^2}. \quad (3.63)$$

Proof: : Following the result of *Theorem* 3.2, we have that the number of iterations for an ε-optimal with δ confidence action with no uncertainty is given by

$$n_{\text{itr}} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\frac{i}{C}} \right) \frac{\nu \log(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.64}$$

Thanks to *Theorem* 3.3, the error between the value function in the presence of a model error is bounded by

$$|\hat{Q}(\hat{x}, u) - Q(\hat{x}, u)| \leq \frac{\kappa(1 - \gamma^H)}{1 - \gamma} =: \varepsilon_m \tag{3.65}$$

To ensure the overall error remains within the admissible bound ε_T, the sum of the MCTS tolerance ε and the model error ε_m must satisfy

$$\varepsilon_m + \varepsilon \leq \varepsilon_T \tag{3.66}$$

and then rearranging the terms to highlight ε, we have

$$\varepsilon \leq \varepsilon_T - \varepsilon_m. \tag{3.67}$$

This shows that the MCTS simulation error ε is bounded by the total admissible error ε_T minus the model error ε_m. So, if the model uncertainty is large (i.e., ε_m is large), the allowable error ε will be smaller, which means that more iterations are needed to achieve the desired accuracy. Substituting this inequality in (3.64) we finally obtain

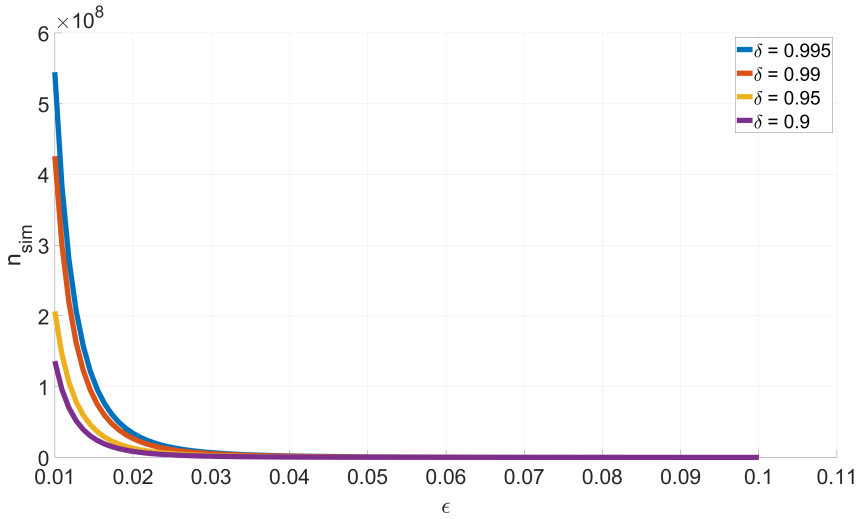
$$n_{\text{itr}} \geq \left(\sum_{i=0}^{H} \nu^i e^{-\frac{i}{C}} \right) \frac{\nu \log(\frac{2}{\delta})}{2\left(\varepsilon_T - \frac{\kappa(1-\gamma^H)}{1-\gamma} \right)^2}. \tag{3.68}$$

■

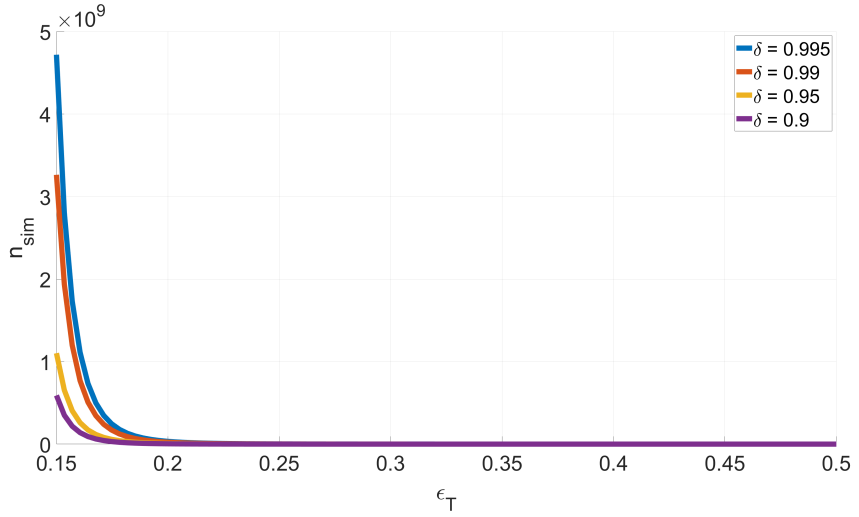
3.5 Simulations

Monte Carlo Tree Search is tested in simulations to validate the theoretical results and show that it can solve the optimal control problem with/without uncertainty. We conduct two types of experiments:

1. Solve the problem of attitude rate regulation (reach a desired attitude rate) [76]. We use the linearized model of the aircraft (Section 2.7) around its equilibrium and the Linear Quadratic Regulator (LQR) [77] as a baseline for the optimality.
2. Solve the regulation problem on the 6 degrees of freedom (DoF) nonlinear system of the aircraft presented in Section 2.7 comparing our approach with the Nonlinear Model Predictive Control (NMPC). NMPC is quite similar at the formalization level [30, 31] and also is one of the most widely used frameworks for solving optimal control problems [78].



(a) Evolution of n_{itr} vs ε for fixed δ without uncertainty



(b) Evolution of n_{itr} vs ε_T for fixed δ with uncertainty

Fig. 3.3: Evolution of n_{itr}

Figure 3.3 shows the evolution of the number of iterations with different δ confidence. In particular, in Figure 3.3a, the evolutions of the n_{itr} modifying ε and fixing different δ are plotted following the results of *Theorem* 3.2. In Figure 3.3b, the evolution of the n_{itr} modifying ε and fixing different confidence are shown following the result of *Theorem* 3.4. The parameters for each experiment are listed in Tables 5.2 and 3.2, respectively.

3.5.1 Attitude rate regulation

In this experiment, we want to regulate the attitude rate of the aircraft controlling the deflection surfaces (elevator δ_e, aileron δ_a and rudder δ_r) to show the convergence to the desired reference values assuming a constant thrust (T). Given the results of the LQR optimal controller, we show that by computing the proper number of iterations, MCTS computes the commands reaching the chosen ε-optimality in the cost function. We first consider a system without uncertainty and compute the number of iterations of MCTS according to *Theorem* 3.2. Then, according to [79, Theorem 11.3] for the LQR controller, we will introduce the model uncertainty, and exploiting *Theorem* 3.4, we recompute the number of iterations n_{itr} for MCTS.

	Parameters
ε	0.05
ε_m	0.11
ε_T	0.3
δ	95%
Q	diag(100, 100, 100)
R	diag(0.1, 0.1, 0.1)
H	2
D	2
κ	5
γ	0.1
μ	diag(0.0, 0.0, 0.0)
Σ	diag(1, 1, 1)

Table 3.1: Parameters used in the attitude rate stabilization experiment

Table 5.2 lists the parameters and the number of simulations in this experiment.

We linearize the nonlinear system $\dot{x}_{k+1} = f(x_k, u_k)$ of Section 2.7 around the equilibrium (x_e, u_e)

$$\begin{aligned} x_e &= [0 \;\; 0 \;\; 15000 \;\; 0 \;\; 0.077 \;\; 0 \;\; 550 \;\; 0.077 \;\; 0 \;\; 0 \;\; 0 \;\; 0] \\ u_e &= [2120.62 \;\; -2.460 \;\; 0 \;\; 0] \end{aligned} \tag{3.69}$$

by computing the first-order Taylor expansion

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{\substack{x=x_e \\ u=u_e}} \in \mathbb{R}^{n\times n}, B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{\substack{x=x_e \\ u=u_e}} \in \mathbb{R}^{n\times m}. \tag{3.70}$$

We define the error as $e_k = x_k - x_e$. The infinite-horizon cost function for the LQR controller has the usual quadratic form

$$J = \sum_{k=0}^{\infty} l(e_k, u_k) \tag{3.71}$$

where

$$l(e_k, u_k) = e_k^T Q e_k + u_k^T R u_k \tag{3.72}$$

and $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive semi-definite matrix and $R \in \mathbb{R}^{m \times m}$ is symmetric positive definite matrix. The steady-state optimal control input u_k^* is given by the state-feedback law

$$u_k^* = -K e_k \tag{3.73}$$

where $K = (R + B^T P B)^{-1} B^T P A$ and P is the solution of the Discrete Algebraic Riccati Equation (DARE)

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A. \tag{3.74}$$

The linearized system is also used as a transition model in the MCTS algorithm, and the cumulative cost function is

$$J_k = \sum_{i=0}^{k} l(e_i, u_i^*) \tag{3.75}$$

where the optimal control u_i^* is computed solving the MCTS optimization problem (3.15). Concerning the experiment without model uncertainty, we want to control the deflation surfaces $(\delta_e, \delta_a, \delta_r)$ to regulate the attitude rate (p, q, r) to a desired configuration. The initial condition of the attitude rate is the vector $[0.3, 0.3, 0.3]$ $\frac{\text{rad}}{s}$, and the desired attitude rate (x_e) is $[0, 0, 0]$ $\frac{\text{rad}}{s}$ (only the p, q, r components of the state vector are reported). The simulation time is $50s$, and the control frequency is $100Hz$. Thanks to *Theorem* 3.2, setting the constant number $C = 0.3$, the number of iterations used in MCTS is $n_{\text{itr}} = 531221$.

Figure 3.4 shows how both techniques achieve the regulation of the attitude rate on the three axes. Figure 3.5 shows the costs of the optimal action u_k^* computed by the two different controllers, and it is possible to see that the MCTS algorithm cost converges at the value that is at distance ε (black dashed lines) to the optimal value (confirming *Theorem* 3.2), while the dashed red line is the optimal value computed by the LQR on the infinite horizon. Additionally, to prove the δ confidence of the algorithm, we compute the error between the cost of the LQR and MCTS controllers for each timestamp since *Theorem* 3.2 guarantees that, for each action provided by MCTS, the relatives cost is within ε distance. Therefore, we count the number of times that the error is under or equal to the ε value, and we obtain that MCTS with the ε error maintains a 0.96% of confidence that is closer to the defined confidence value.

The uncertainty ω is modeled as a Gaussian noise (zero mean and variance 0.1) in the dynamic model as explained in (3.1). According to *Theorem* 3.4, setting the

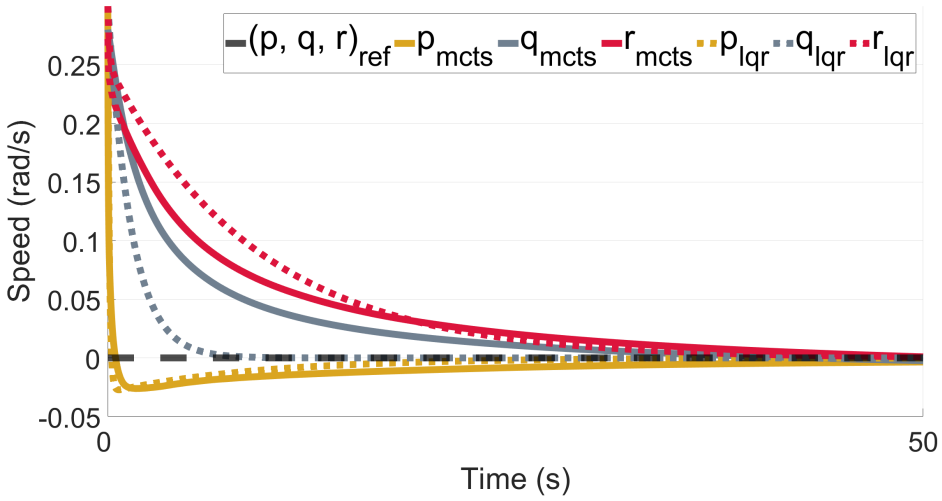


Fig. 3.4: Attitude rate regulation (p, q, r).

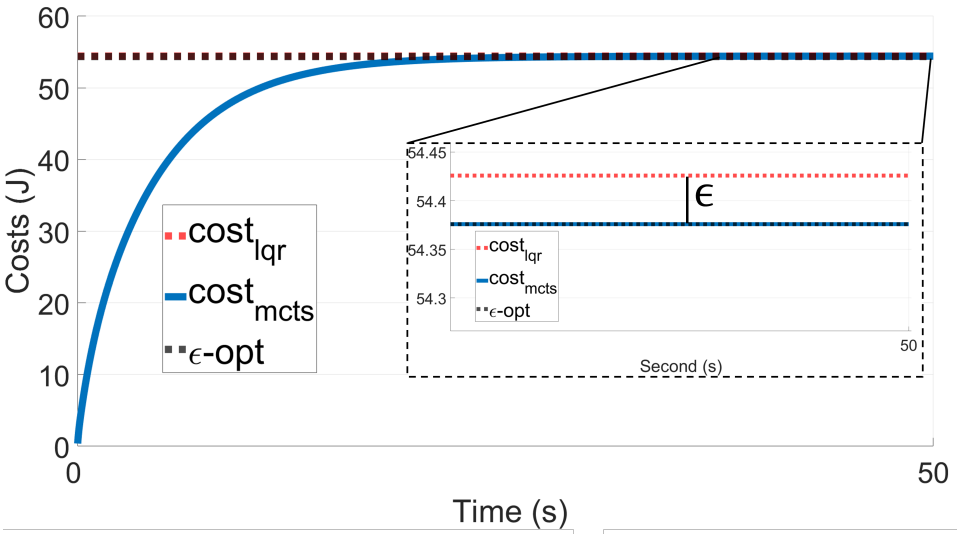


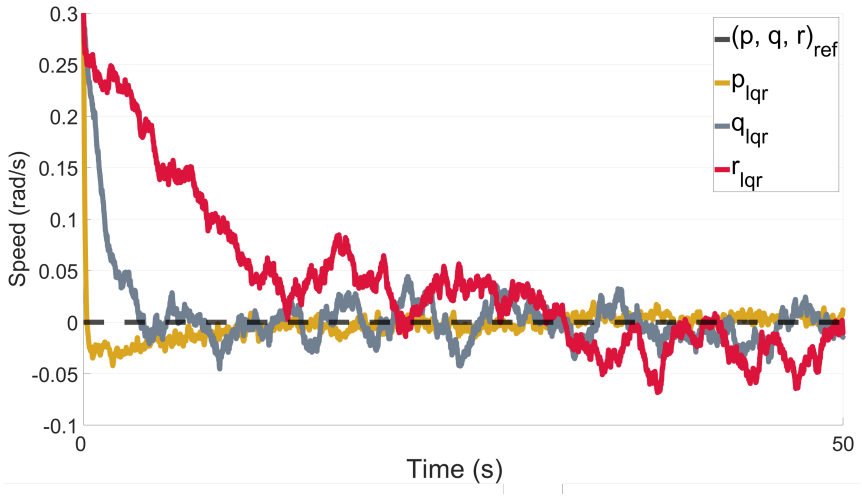
Fig. 3.5: Costs (J) for MCTS and LQR on the linearized dynamic model without uncertainty.

constant $C = 0.6$ to improve the exploration, the number of iterations is $n_{\text{itr}} = 984261$. In the presence of stochasticity, the LQR cost function is

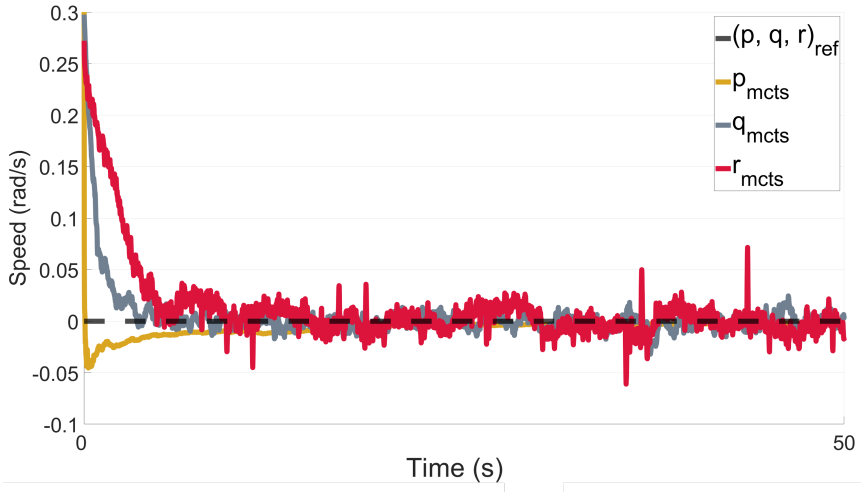
$$J = \mathbb{E} \sum_{k=0}^{\infty} l(e_k, u_k). \tag{3.76}$$

where $l(e_k, u_k)$ is modeled as (3.72), the cumulative cost function for MCTS is defined in (3.75), and u_k^* is computed solving the optimization problem (3.15).

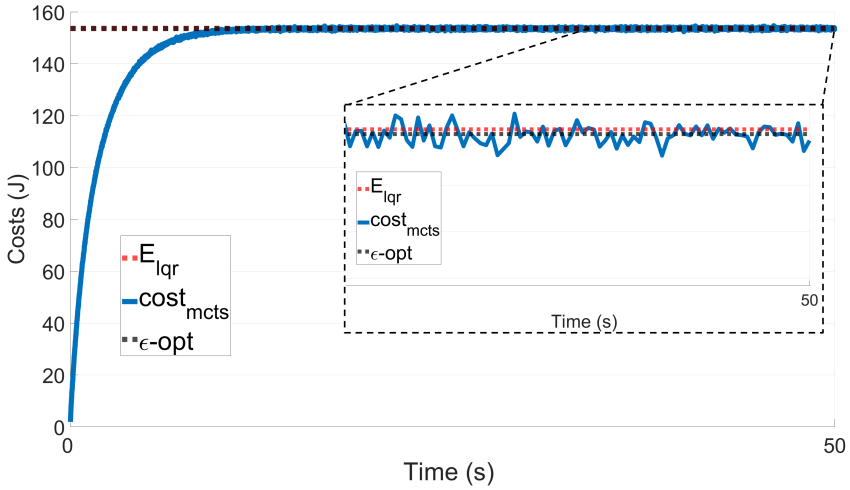
Considering model uncertainty, Figure 3.6a and Figure 3.6b show the convergence of the LQR and MCTS controller respectively. In both cases, the controllers stabilize the attitude. In order to confirm the results of the *Theorem* 3.4, Figure 3.6c shows the cost of both controllers. In particular, concerning the LQR cost, the figure shows the expectation value of the cumulative cost (orange dashed line). For the MCTS



(a) LQR attitude regulation (p, q, r).



(b) MCTS attitude regulation (p, q, r).



(c) Costs (J).

Fig. 3.6: MCTS and LQR on linearized dynamic model with uncertainty.

controller, the figure shows the cumulative cost (blue line), and it is possible to note that the expectation of the cumulative cost is around the ε distance (black dashed line) to the expectation of the LQR optimal value, confirming *Theorem* 3.4.

3.5.2 Airspeed and attitude regulation

In this experiment, we solve the regulation problem on the airspeed and attitude rate of the aircraft controlling the deflection surfaces (elevator δ_e, aileron δ_a and rudder δ_r) to stabilize the attitude rate, and the thrust (T) of the aircraft to reach the desired airspeed. We compare our approach with a Nonlinear Model Predictive Control (NMPC). For both approaches, we use the full nonlinear dynamic model of Section 2.7. We formalize the NMPC in the following way

$$
\begin{aligned}
u^*_{k+i|k}\Big|_{i=0}^{H} = \underset{u}{\operatorname{argmin}} \quad & J_k^{\text{NMPC}} \\
\text{subject to:} & \\
x_{k+i+1|k} &= f(x_{k+i|k}) + g(x_{k+i|k})u_{k+i|k} \\
x_{k+i|k}, &\in \mathcal{X} \qquad x_{\min} \leq x_{k+1|k} \leq x_{\max} \\
u_{k+i|k}, &\in \mathcal{U} \qquad u_{\min} \leq u_{k+1|k} \leq u_{\max} \\
& \qquad\qquad\qquad\qquad \forall i \in \{0, \dots, H-1\} \\
x_{k|k} &= x_k
\end{aligned}
\tag{3.77}
$$

where $x_{k|k}$ are the initial conditions at time k equal to the actual state of the plant x_k. The error is $e_{k+i|k} = x_{k+i|k} - x_e$ and the cost function J is defined as

$$
J_k^{\text{NMPC}} = \sum_{i=0}^{H} l(e_{k+i|k}, u_{k+i|k}) \tag{3.78}
$$

where H is the horizon of NMPC and $l(e_{k+i|k}, u_{k+i|k})$ is defined as (3.72). We also constrain the state and the control in order to guarantee that the attitude angle rate does not exceed the feasible values. The optimal control applied at time k is

$$
u^*_k = u^*_{k|k}, \tag{3.79}
$$

and the cumulative cost is defined in (3.75), where u^*_k is the optimal control of the NMPC. Concerning MCTS, the immediate cost function $l(e_k, u_k)$ is modeled as (3.72), and we use the nonlinear systems as a transition model. The optimal command u^*_k is computed solving the optimization problem defined in (3.15) and the cumulative cost is defined as (3.75), where u^*_k is the control computed by the MCTS.

The initial attitude rate is $[0.4, 0.4, 0.4]\frac{\text{rad}}{s}$ and the desired attitude rate is $[0, 0, 0]\frac{\text{rad}}{s}$, while the current airspeed is $500\frac{m}{s}$ and the desired one is $550\frac{m}{s}$. Table 3.2 lists the parameters. The simulation time is $50s$, the control frequency is $100Hz$, and the horizon of the NMPC is set to 10. First, we do not consider model uncertainty, and the goal is to minimize the error with respect to the reference ($[0, 0, 0]\frac{\text{rad}}{s}$). The controls are the deflation surfaces ($\delta_e, \delta_a, \delta_r$) to regulate the attitude rate (p, q, r) and the thrust (T) to reach a desired airspeed (V_T) using the

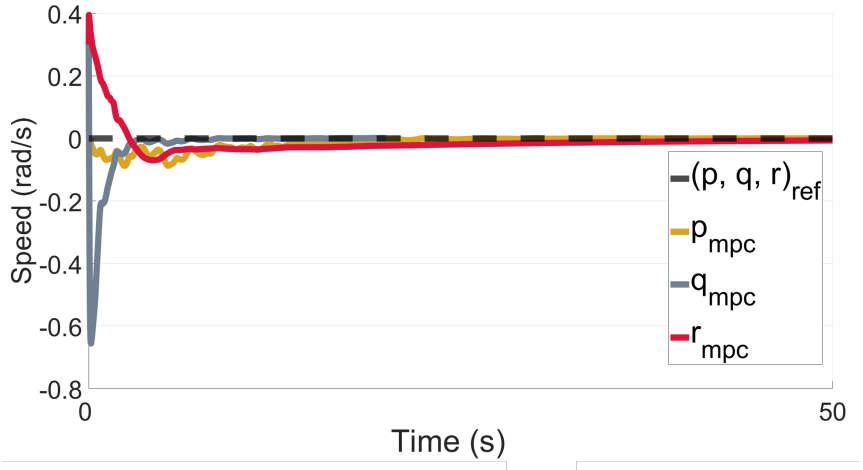
	Parameters
ε	0.05
ε_T	0.3
δ	95%
Q	diag(5000, 100, 100, 100)
R	diag(0.1, 0.1, 0.1, 0.1)
H	2
D	2
γ	0.1
κ	5
μ	diag(5000, 0.0, 0.0, 0.0)
Σ	diag(100, 1, 1, 1)
u_{min}	(1000, −20, −30, −20)
u_{max}	(19000, 20, 30, 20)
x_{min}	(−1, −1, −1)
x_{max}	(1, 1, 1)

Table 3.2: Parameters used in the airspeed and attitude regulation experiment

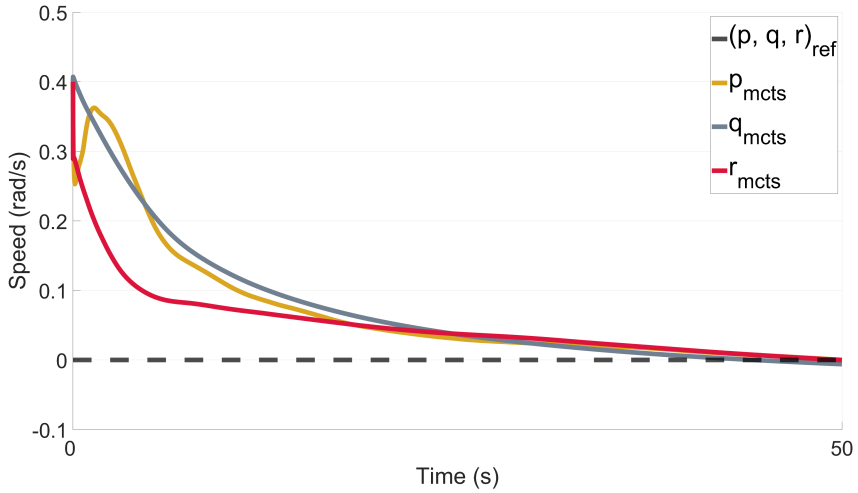
nonlinear model. Setting the constant number $C = 0.3$, the number of iterations used for MCTS is $n_{\text{itr}} = 531221$.

Figure 3.7 shows the trajectories of the attitude rate computed by the NMPC and MCTS, respectively, while the airspeed trajectories are shown in Figure 3.8. Both approaches reach the zero steady-state error for attitude rate and airspeed. Figure 3.9 shows the costs of the NMPC (orange line) and the cost of MCTS (blue line). Over time, both methods converge to the same cumulative cost. The MCTS cost starts lower than the MPC cost. This means that, initially, MCTS explores better trajectories or takes less costly actions compared to NMPC. This reflects differences in the optimization process and in the constraint evaluation.

Finally, we conduct the last experiment by adding a Gaussian model uncertainty (zero mean and variance 0.1 concerning the angular velocities, while the airspeed variance is set to 10) in the nonlinear dynamic model, maintaining the same initial and desired values as in the previous experiment; in this stochastic case, the constant is $C = 0.6$, and then the number of iterations is $n_{\text{itr}} = 984261$. Figure 3.10a and 3.10b show the time series of the attitude regulation of the NMPC and MCTS controllers with additive noise using the nonlinear model. Both controllers reach the reference values on average. Concerning the airspeed control, Figure 3.11a shows the action of the NMPC, while Figure 3.11b shows the MCTS control. In both cases, the controllers reach the desired airspeed on average. Finally, Figure 3.12 shows the costs of both controllers that converge in expectation to the same cumulative costs.



(a) NMPC controller for the attitude rate (p, q, r).

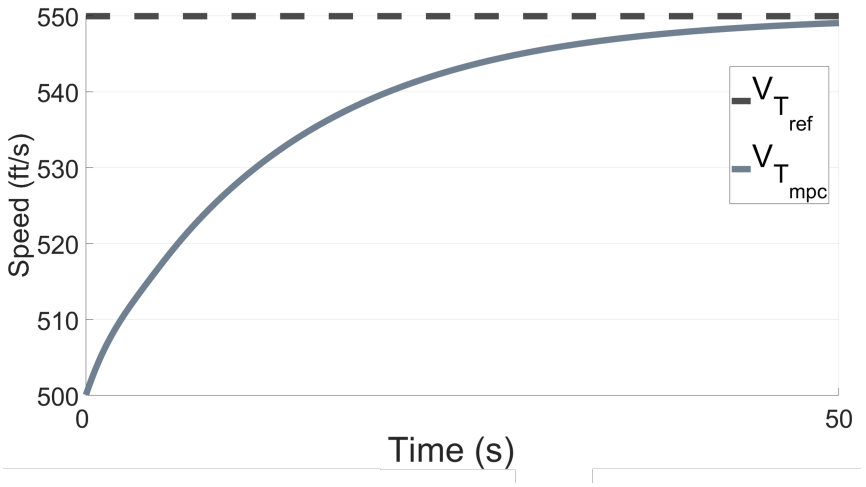


(b) MCTS controller for the attitude rate (p, q, r).

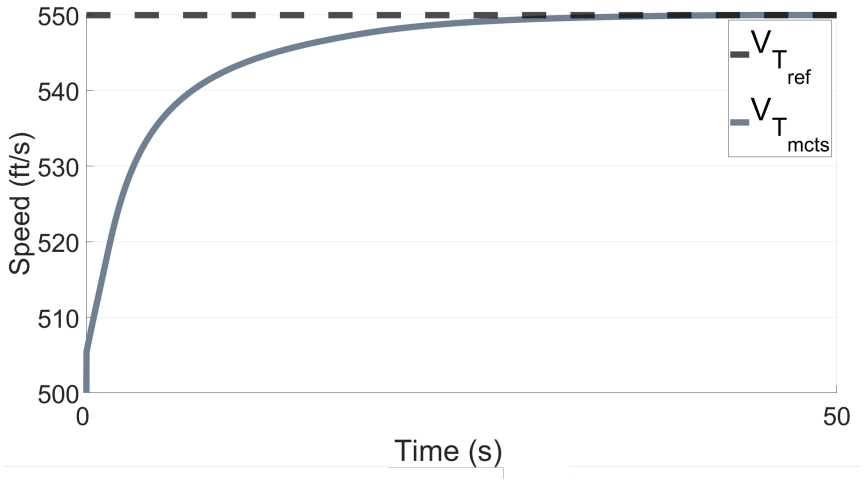
Fig. 3.7: NMPC and MCTS on a nonlinear model for the attitude regulation

3.6 Discussion

In this chapter, we established a formal framework for using Monte Carlo Tree Search (MCTS) as a controller to solve constrained optimal control problems, both in deterministic settings and under model uncertainty. We demonstrated that MCTS, when properly formulated, converges to the optimal policy in the infinite iteration limit, and we derived finite-sample bounds to guarantee ε-optimality with δ confidence using Hoeffding's inequality. Importantly, we extended these guarantees to settings with model uncertainty, showing how additional iterations can compensate for approximation errors. Through comparative iterations with established linear and nonlinear control methods, we highlighted the flexibility and robustness of MCTS algorithm.



(a) NMPC controller for the airspeed (V_T).



(b) MCTS controller for the airspeed (V_T).

Fig. 3.8: NMPC and MCTS on a nonlinear model for the airspeed regulation

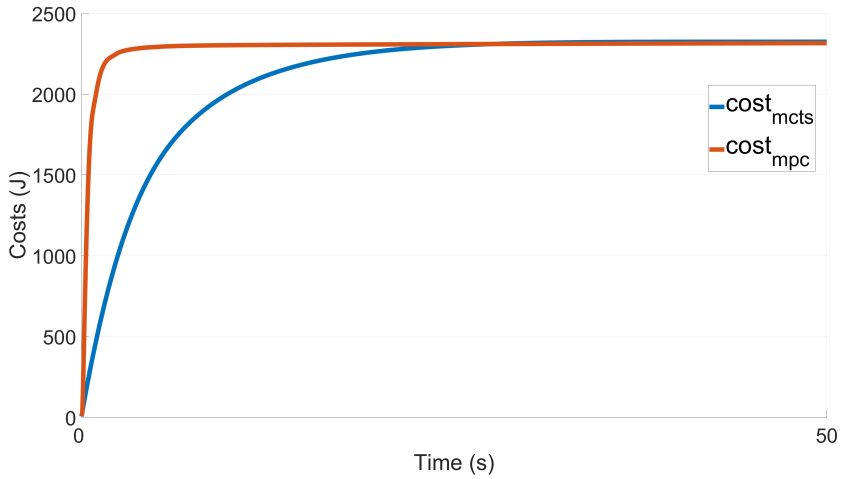
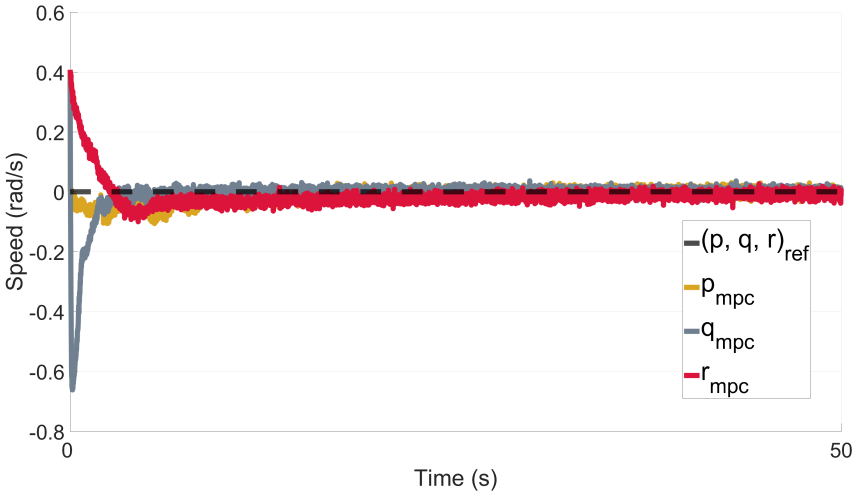
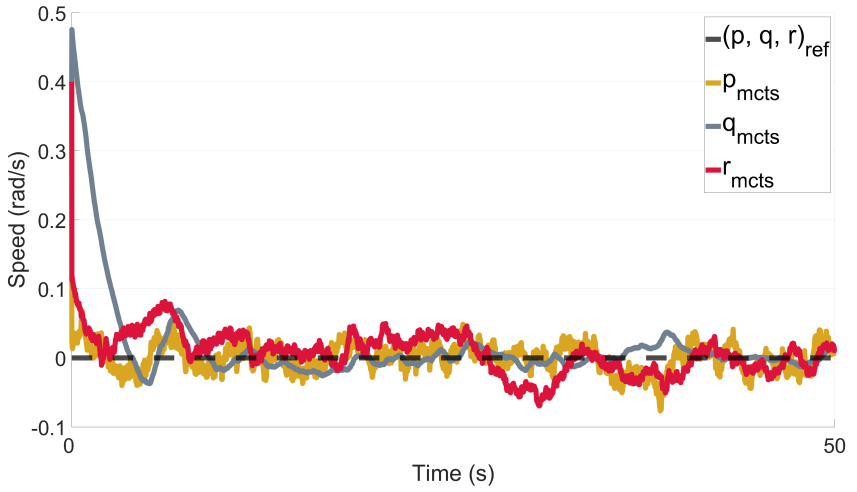


Fig. 3.9: Costs of the NMPC and MCTS controller

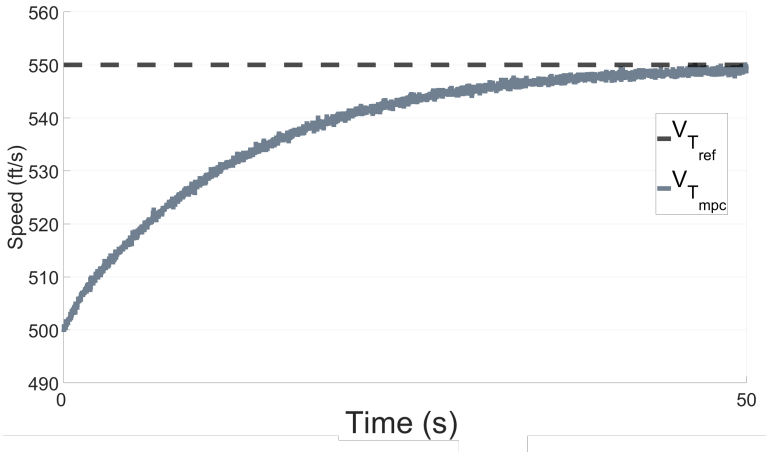


(a) NMPC attitude regulation (p, q, r).

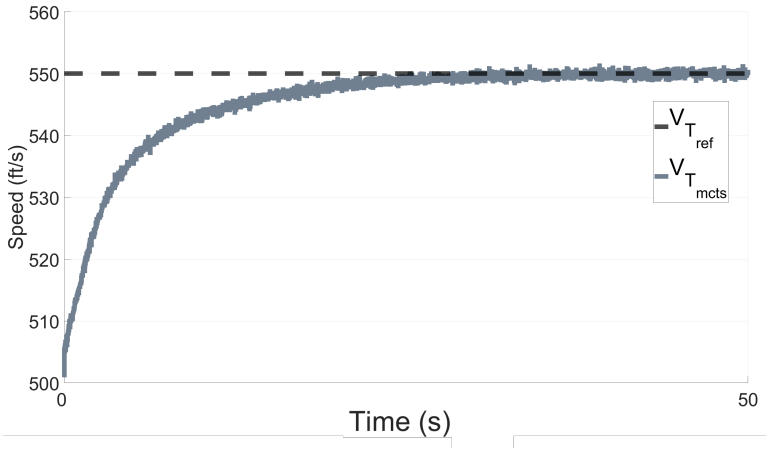


(b) MCTS attitude regulation (p, q, r).

Fig. 3.10: NMPC and MCTS on a nonlinear model for the attitude regulation with model uncertainty



(a) NMPC airspeed regulation (V_T).



(b) MCTS airspeed regulation (V_T).

Fig. 3.11: NMPC and MCTS on a nonlinear model for the airspeed regulation with model uncertainty

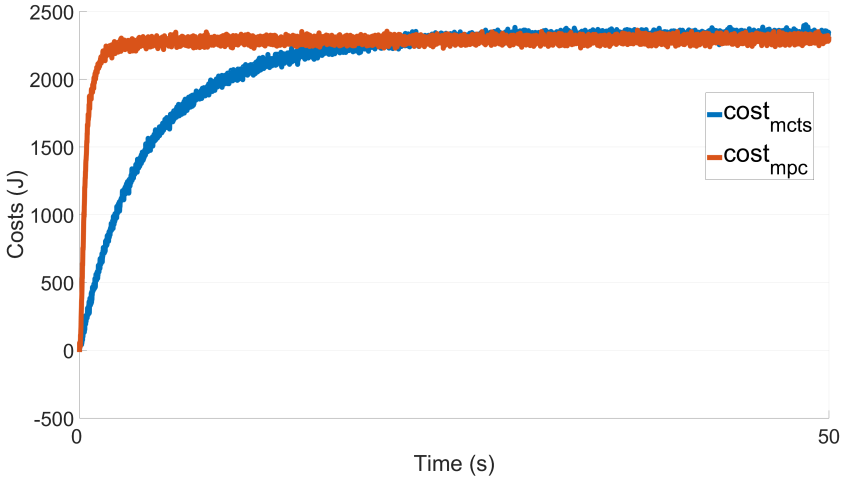


Fig. 3.12: Costs of the NMPC and MCTS with noise

Chapter 4

PATH-PLANNING FOR AIRCRAFT AUTONOMY WITH POMDP

"Science is not only a disciple of reason but also one of romance and passion."

Stephen Hawking

Abstract[1] Building on the formal guarantees of Monte Carlo Tree Search (MCTS) for solving constrained optimal control problems, this chapter extends those results to trajectory planning under measurement uncertainty. Specifically, we address how to plan and control an aircraft autonomously when relying on noisy sensor data (e.g., GPS, IMU).

To tackle this, we design a trajectory planner based on the Partially Observable Markov Decision Process (POMDP) framework, which models both environmental uncertainty and sensor measurement errors. Although the system is governed by complex nonlinear dynamics, we demonstrate that a linearized closed-loop model can effectively support high-level decision-making within the POMDP formulation.

The aircraft is controlled using a two-layer architecture: a low-level controller based on inverse dynamics ensures accurate tracking, while a high-level POMDP planner selects optimal local reference actions at each time step. Within this planner, a belief distribution over states is maintained and updated via Bayes' theorem, enabling robust decision-making despite partial observability. The POMDP is solved online using the Partially Observable Monte Carlo Planning (POMCP) algorithm to provide real-time guidance that avoids no-fly zones and minimizes a cost function.

[1] The content of this chapter is based on the following publications:
Trotti, Francesco, Alessandro Farinelli, and Riccardo Muradore. "Towards Aircraft Autonomy Using a POMDP-Based Planner." *2024 American Control Conference (ACC).* IEEE, 2024.
Trotti, Francesco, Alessandro Farinelli, and Riccardo Muradore. "An online path planner based on POMDP for UAVs." *2023 European Control Conference (ECC).* IEEE, 2023.

The key contributions of this chapter include:

- A method to control a complex nonlinear system using a POMDP framework built upon a linearized closed-loop dynamic model.
- A high-level POMDP planner that accounts for measurement noise and environmental uncertainty to generate optimal reference values for a low-level controller.

The chapter is structured as follows: Section 4.1 introduces the low-level controller. Section 4.2 discusses the high-level controller (POMDP) formulation, and Section 4.3 presents simulation results.

4.1 Low-level controller

A low-level inverse dynamics controller is implemented to control the aircraft. The linearized system via feedback is used within the high-level controller.

4.1.1 Exact linearization via feedback

The aircraft is controlled using an inverse dynamics controller [80], [81] to control the deflection surfaces (δ_e, δ_a, and δ_r) and thrust T of the aircraft defining the desired attitude rate (p_d, q_d, r_d) and the desired airspeed value (V_{T_d}).

Let

$$\dot{x} = f(x) + g(x)u \tag{4.1}$$

be the nonlinear system, the controller is obtained as

$$u = \frac{1}{g^{-1}(x)}(-f(x) + v) \tag{4.2}$$

where v is an auxiliary control signal, and we assume that the function $g(x)$ is invertible.

Thrust controller

Using the equation (2.18)

$$\dot{V}_T = \frac{C_x T \cos\alpha \cos\beta}{m} - \frac{C_y}{m} - g \sin\gamma \tag{4.3}$$

the aircraft thrust (T) to track the desired velocity (V_{T_d}) is computed according to the control law

$$T = \frac{m}{C_x \cos\alpha \cos\beta}[K_V(V_{T_d} - V_T) + \frac{C_y}{m} + g \sin\gamma]. \tag{4.4}$$

Therefore, the feedback linearized model of the airspeed is

$$\dot{V}_T = K_V(V_{T_d} - V_T), \tag{4.5}$$

where K_V is a positive scalar gain.

Surface deflections controller

The controller is designed to track the angular velocities ($\omega_{b_d} = [p\ q\ r]_d$) of the aircraft, computing the deflection surfaces (δ_e, δ_a, δ_r). Let the equation (2.14) be expressed in matrix form

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (c_1 r + c_2 p + c_4 h_{eng})q \\ (c_5 p - c_7 h_{eng})r - c_6(p^2 - r^2) \\ (c_8 p - c_2 r + c_9 h_{eng}))q \end{bmatrix} + \begin{bmatrix} \bar{q}SBc_3 & 0 & \bar{q}SBc_4 \\ 0 & \bar{q}S\bar{c}c_7 & 0 \\ \bar{q}SBc_4 & 0 & \bar{q}SBc_9 \end{bmatrix} \begin{bmatrix} C_l(\delta_a, \delta_r) \\ C_m(\delta_e) \\ C_n(\delta_a, \delta_r) \end{bmatrix} \tag{4.6}$$

Therefore, the control law is

$$\begin{aligned} \begin{bmatrix} C_l(\delta_a, \delta_r) \\ C_m(\delta_e) \\ C_n(\delta_a, \delta_r) \end{bmatrix} = \begin{bmatrix} \bar{q}SBc_3 & 0 & \bar{q}SBc_4 \\ 0 & \bar{q}S\bar{c}c_7 & 0 \\ \bar{q}SBc_4 & 0 & \bar{q}SBc_9 \end{bmatrix}^{-1} \Bigg(K_{p,q,r} \left(\begin{bmatrix} p \\ q \\ r \end{bmatrix}_d - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \\ - \begin{bmatrix} (c_1 r + c_2 p + c_4 h_{eng})q \\ (c_5 p - c_7 h_{eng})r - c_6(p^2 - r^2) \\ (c_8 p - c_2 r + c_9 h_{eng}))q \end{bmatrix} \Bigg) \end{aligned} \tag{4.7}$$

where $K_{p,q,r} \in \mathbb{R}^{3\times3}$ is a positive definite matrix gains. Note the inversion is made maintaining $C_l(\cdot,\cdot)$, $C_m(\cdot)$, and $C_n(\cdot,\cdot)$ in the function of the controls since they represent the aerodynamic coefficient moments in a particular configuration of the control surfaces (equation (2.16)). A bivariate spline interpolation over the aerodynamic values is made using the lookup table provided in [82] to extrapolate the deflection values of the flaps from the aerodynamic coefficients. Finally, a linear approximation over the coefficient is made to obtain the deflection angles. Therefore, the linearized attitude acceleration, expressed in matrix form, results

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = K_{p,q,r} \left(\begin{bmatrix} p \\ q \\ r \end{bmatrix}_d - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{4.8}$$

4.2 High-level controller

The high-level controller based on POMDP has to choose the best reference values (i.e., V_{T_d}, p_d, q_d, r_d) for the low-level controller to accomplish the mission. The goal is to reach the target position while avoiding no-fly zones, minimizing the cost function to reduce fuel and/or travel time, and achieving the desired attitude during the approach to the target.

Figure 4.1 shows the block diagram of the two-level control architecture. In particular, $p_{x_T}, p_{y_T}, p_{z_T}, \phi_T, \theta_T, \psi_T$ are the position and orientation of the target, V_{T_d}, p_d, q_d, r_d are the desired airspeed and angular velocities input to the low-level controller that provides the controls $T, \delta_e, \delta_a, \delta_r$. The plant provides via feedback the measurements of the positions and orientations of the aircraft $p_x^o, p_y^o, p_z^o, \phi^o, \theta^o, \psi^o$ to the POMDP, while $V_T, \alpha, \beta, \phi, \theta, \psi, p, q, r, p_x, p_y, p_z$ are the state feedback. In the POMDP transition function, the linearized model of the low-level controller is included to ensure:

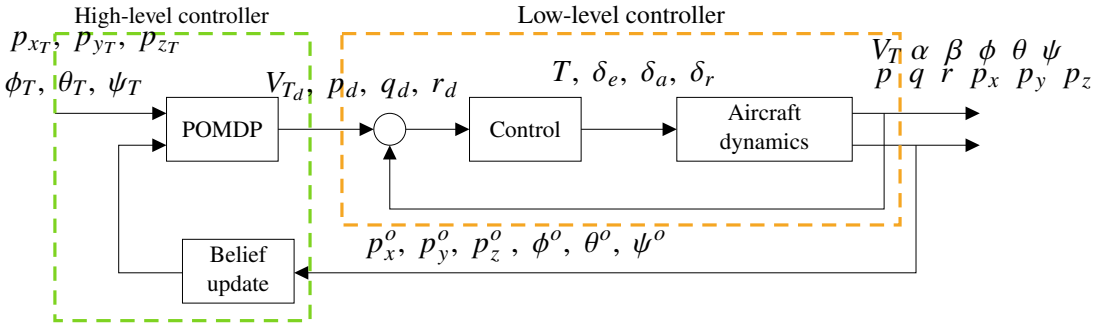


Fig. 4.1: Block diagram of the two-layer architecture

- the coherence with the physical model of the aircraft
- a low computational load due to the simpler dynamic model
- the feasibility of chosen actions for the aircraft.

4.2.1 POMDP formalization

In the proposed approach, the state space $\mathcal{S}$, action space $\mathcal{A}$, and observation space $\mathcal{Z}$ are discretized $t_k = k\Delta t, k \in \mathbb{N}$.

State: The POMDP state vector represents the system variables that evolve over time. The state vector of the aircraft dynamic model is

$$\boldsymbol{s} = [V_T \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; p \;\; q \;\; r \;\; p_x \;\; p_y \;\; p_z]^T. \tag{4.9}$$

The aircraft's sensing system provides the noisy measurements used as the initial conditions at any iteration of the POMDP. The uncertainty over the aircraft state is represented by the *belief space* that allows considering the environment uncertainty.

Action: The action set used in the POMDP are

$$\boldsymbol{a} = [V_{T_d} \;\; p_d \;\; q_d \;\; r_d]^T. \tag{4.10}$$

The POMDP chooses the best local action that minimizes the cost function and will be set to the low-level controller as reference values.

Observations: The observation set is composed of values obtained from the onboard aircraft sensor system, such as a GPS for positions (p_x^o, p_y^o, p_z^o) and a gyroscope for attitude $(\phi^o, \theta^o, \psi^o)$

$$\boldsymbol{o} = [p_x^o \;\; p_y^o \;\; p_z^o \;\; \phi^o \;\; \theta^o \;\; \psi^o]^T. \tag{4.11}$$

A multi-variate white Gaussian process simulates the measurement sensor noises.

Transition function: The transition function defines the next state, given the current state and the action. The linearized dynamic model obtained from the inverse dynamics controller of the aircraft is used as the transition function in the POMDP. In particular, the equations (4.5) and (4.8) define the airspeed and attitude rate derivatives. While using the equations

$$\begin{aligned} \dot{\phi} &= p + \tan\theta(q\sin\phi + r\cos\phi) \\ \dot{\theta} &= q\cos\phi - r\sin\phi \\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}, \end{aligned} \tag{4.12}$$

the time derivative of the Euler angle in the Earth-frame can be obtained. The time derivatives of the angle of attack and the sideslip angle can be obtained using the equations

$$\begin{aligned} \dot{\alpha} &= \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \\ \dot{\beta} &= \frac{V_T\dot{v} - v\dot{V}_T}{V_T^2\sqrt{1 - (\frac{v}{V_T})^2}} \end{aligned} \tag{4.13}$$

The linear velocities along the body axis by

$$\begin{aligned} u &= V_T \cos\alpha \cos\beta \\ v &= V_T \sin\beta \\ w &= V_T \sin\alpha \cos\beta, \end{aligned} \tag{4.14}$$

and, finally, the linear velocity in the Earth frame by

$$\begin{aligned} \dot{p}_x &= u\cos\psi\cos\theta + v(\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi) \\ &\quad + w(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \\ \dot{p}_y &= u\sin\psi\cos\theta + v(\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi) \\ &\quad + w(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \\ \dot{p}_z &= -u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi \end{aligned} \tag{4.15}$$

By formalizing the transition function in this way, the state vector of the dynamic model simulated within the POMDP is the same as the state vector of the aircraft dynamic model. Specifically, in this phase, the evolution of the low-level controller on the linearized model is simulated with the full nonlinear dynamics. This internal simulation allows the reward function to evaluate the state reached with corresponding reference actions.

Cost function: The cost function is used to evaluate how good the chosen action is in the current state. For our purposes, the cost function is a composition of different sub-functions that minimize different metrics to reach the target position while complying with several constraints (e.g., avoiding no-fly zones, optimizing fuel consumption, minimizing attitude error relative to the target). The reward component that allows to reach the target position is calculated as the Euclidean distance between the aircraft actual position X_a and the target position X_T

$$r_{T,a} = \|X_T - X_a\| . \tag{4.16}$$

The function to avoid the no-fly zones is modeled as a repulsive force [83]

$$r_{A_i,a} = \begin{cases} \nu\left(\left(\frac{1}{d_{A_i,a}} - \frac{1}{d_0}\right)^2\right), & \text{if } d_{A_i,a} < d_0 \\ 0, & \text{otherwise} \end{cases} \tag{4.17}$$

where d_{A_i} is the distance between the aircraft and the i-th no-fly zone, d_0 is the repulsive threshold and ν is the repulsive coefficient. Therefore, the reward value with N no-fly zones is

$$r_{A,a} = \sum_{i=1}^{N} r_{A_i,a}. \tag{4.18}$$

The cost function that minimizes the fuel is

$$r_{f_a} = f - c\left(\frac{V_T}{V_{T_{max}}}\right), \tag{4.19}$$

where f is the current fuel level, c is the fuel consumption ratio and $V_{T_{max}}$ is the maximum airspeed.

The cost function concerning the attitude error is defined as the difference between the target attitude $At_T = (\phi_T, \theta_T, \psi_T)$ and the aircraft actual attitude $At_a = (\phi_a, \theta_a, \psi_a)$

$$r_{At} = ||At_T - At_a||. \tag{4.20}$$

Finally, the function to be minimized is given by the sum of their costs

$$R = -r_{T,a} - r_{A,a} - \sigma_f(r_{f_a}) - \sigma_{At}(r_{At}) \tag{4.21}$$

where $\sigma_f, \sigma_{At} \in [0, 1]$ are weights to trade off the minimization of one component over another.

Belief state: The belief state represents the probability distribution over the aircraft states. The states in the belief are modeled as a particle filter [60], which is continuously updated through a particle re-invigoration system and an update belief function. Additionally, the belief update function, exploiting Bayes' theorem, fits the probability distribution over the states by considering the observations provided by the sensors. Therefore, given δ_{s,B_t^i} the Kronecker delta, the belief state is represented as a sum of particles

$$\mathcal{B}(s, h) = \frac{1}{K}\sum_{i=1}^{K} \delta_{sB_k^i} \tag{4.22}$$

$$\delta_{s,\mathcal{B}_k^i} = \begin{cases} 0, & \text{if } s \neq \mathcal{B}_k^i \\ 1, & \text{if } s = \mathcal{B}_k^i. \end{cases} \tag{4.23}$$

where $\mathcal{B}_k^i$ the i-th particles and K is the number of particles.

4.2.2 POMDP controller

The solver for the POMDP is based on the MCTS algorithm. In the first iteration, the algorithm receives as observation the pose of the aircraft in the Earth frame. Using this information, the algorithm samples a state from the belief state, which is then elected as the root node for Partially Observable Monte Carlo Process (POMCP) exploration.

The POMCP algorithm expands a tree to evaluate future actions, minimizing a cost function. The algorithm is based on five phases: selection, expansion, simulation, back-propagation, and belief update.

Selection and expansion: The algorithm chooses the best action to obtain the already-visited next node in the tree. The evaluation of the best action is made using the Upper Confidence bounds applied to the Trees (UCT) strategy [63]

$$a^* = \text{argmax}_{a \in \mathcal{A}(s)} \left(V(h,a) + C \sqrt{\frac{\log N(h)}{N(h,a)}} \right) \tag{4.24}$$

where $\mathcal{A}(s)$ represents the set of possible actions, $V(h,a)$ is the value of the history after executing action a, C is the exploration-exploitation constant, $N(h)$ is the number of visits to history h, and $N(h,a)$ is the number of visits to history h under action a.

The UCT strategy guides the algorithm along the tree, evaluating the scores obtained from previous iterations until the leaf node is reached. At this point, the algorithm expands the tree, generating a number of new nodes equal to the number of possible actions.

Simulation: The simulation phase allows for a more in-depth exploration of the future evolution of the current state by trying different actions. In this phase, a simulator provides the next state, observation, and reward value of the action applied to the current state. In our case, the simulator is the linearized dynamic model of the aircraft that guides exploration through feasible evolutions. The rollout procedure is executed by trying random actions from the current state; in this way, it is possible to have a more in-depth exploration of the future. Once the maximum depth or rollout timeout is reached, the cumulative reward value of the exploration is $R^s = \gamma \sum_i^N r_i^s$, where r_i^s is the reward value for one rollout iteration, γ is the discount factor and N is the horizon of the rollout. The states explored during this phase are continuously added to the belief space to increase the number of possible future states. Meanwhile, the actions and observations obtained from the simulator are used to generate a history for the current node.

Back-propagation: This phase allows the update of the node score ($V(h,a)$), after the expansion and exploration phases, starting from the leaf node and reaching the root node

$$V(h,a) = V(h,a) + \frac{R^s - V(h,a)}{N(h)} \tag{4.25}$$

where R^s is the reward. During the back-propagation, the number of history visits of the node is incremented, and the current state is added to the belief.

Belief update: When the maximum number of POMCP simulations is reached, the node with the highest $V(h,a)$ is chosen, and the corresponding action (the reference values) is provided to the low-level controller. The aircraft sends a new noisy observation to the POMCP. Such aircraft observation is compared with the states in the belief space. A state similar to the aircraft observation is selected as the new root for future exploration. According to the new root, the particles from the past are pruned to ensure that only states in the future are present in the belief. These states were added during the POMCP exploration phase, making them reachable and feasible by the dynamic model. Finally, the probability distribution is updated using Bayes' theorem, considering the probability distribution over the observations and the states.

	Parameters
K_V	10
$K_{p,q,r}$	diag(10, 20, 20)
γ	0.85
C	0.5
Max-depth	10

Table 4.1: Low and High-level controller parameters

4.3 Simulations

The proposed approach has been tested and validated in different scenarios. We analyzed how the choice of actions for controlling the aircraft, and then the trajectory, is influenced by the cost function. Therefore, the σ_f, σ_A values of the cost function (equation ((4.21))) are modified to showcase different behaviors:

- optimization of fuel (where σ_f has the higher weight)
- optimization of both fuel and attitude error (where σ_f and σ_{At} have equal contribution)
- optimization of attitude error (where σ_{At} has the higher weight)

In all cases, the aircraft must reach the target position while avoiding the no-fly zones. The target position is known to the aircraft, while the no-fly zones are unknown but discovered when the aircraft is less than 300 meters close to them (i.e., the threshold value d_0). The control parameters used in the proposed approach are $K_V = 10$ and $K_{p,q,r} = \text{diag}(10, 20, 20)$, while the hyper-parameters used in the POMCP are $\gamma = 0.85$, $C = 0.5$ and the Max-depth = 10.

	Cost functions			
	Baseline $\sigma_f = 0, \sigma_{At} = 0$	Fuel Optimization $\sigma_f = 1, \sigma_{At} = 0$	Fuel and Attitude Optimization $\sigma_f = 0.5, \sigma_{At} = 0.5$	Attitude error minimization $\sigma_f = 0, \sigma_{At} = 1$
Fuel consumption (l)	15.52 [±0.26]	13.80 [±0.21]	14.27 [±0.23]	16.46 [±0.29]
Avg. airspeed (m/s)	171.92 [±3.2]	150.62 [±2.6]	160.79 [±2.93]	168.34 [±3.67]
ϕ error (rad)	-	-	0.12 [±0.13]	0.03 [±0.05]
θ error (rad)	-	-	0.03 [±0.07]	0.04 [±0.06]
ψ error (rad)	-	-	0.83 [±0.07]	0.001 [±0.02]

Table 4.2: Average simulations results obtained from 50 runs (in square bracket the standard deviation error)

The aircraft must reach the target position ($p_{x_T} = 400\text{km}, p_{y_T} = 400\text{km}, p_{z_T} = 1.6Km\text{m}$) with the attitude ($\phi_T = 0.0, \theta_T = 0.14, \psi_T = 1.57$ rad), avoiding no-fly zones, and optimizing fuel consumption, attitude error, or both. The initial position of the aircraft is in $p_{x_T} = 0\text{km}, p_{y_T} = 0\text{km}, p_{z_T} = 1500\text{m}$ and the attitude is $\phi_T = 0.0, \theta_T = 0.14, \psi_T = 0.0$ rad. In the results, the altitude is not shown since the cost function does not influence it, and the aircraft always reaches the desired

altitude. A baseline path ($\sigma_f = 0, \sigma_{At} = 0$) is used to compare different behavior modifying the cost function.

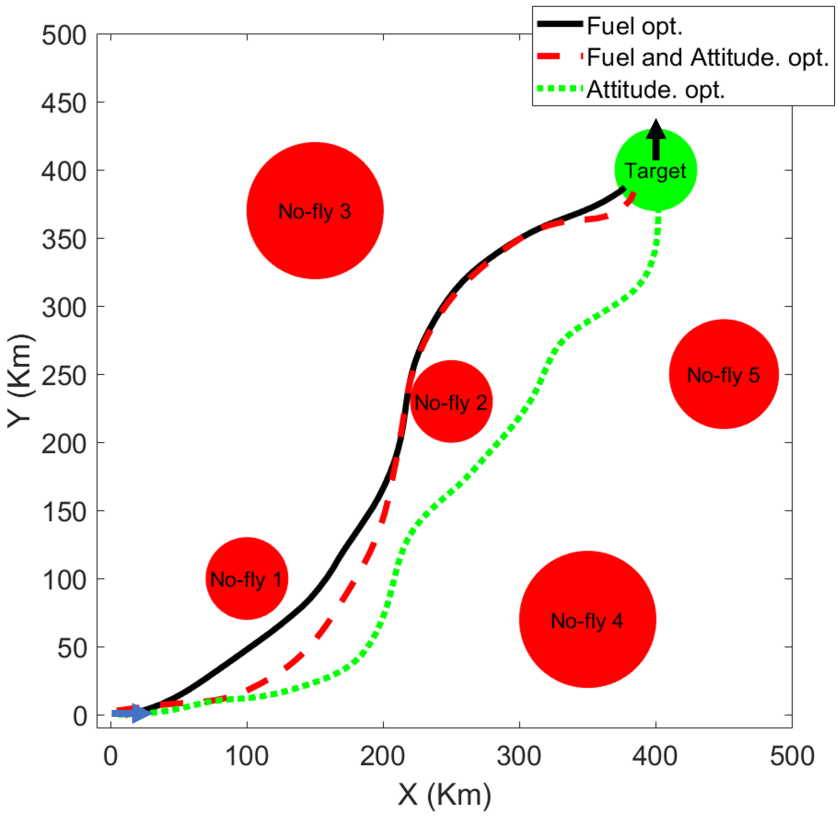


Fig. 4.2: Aircraft paths. The black and blue arrows are, respectively, the desired target yaw angle and aircraft initial yaw angle

Fuel Optimization: To optimize the fuel consumption, the fuel coefficient (σ_f) in the cost function is set to 1, while the attitude error component (σ_{At}) is set to 0. As expected, the lowest fuel consumption is obtained, as shown in the second column of Table 4.2. However, this optimization has an impact on the speed performance of the aircraft, resulting in the slowest of all simulations. This behavior results from selecting reference values that minimize fuel consumption. This involves actions that reduce airspeed and optimize the path, illustrated by the black line in Figure 4.2, representing the shortest trajectory to the target.

Fuel Optimization and Attitude Error Minimization: In this case, the fuel and the attitude error coefficient are considered of equal importance, $\sigma_f = 0.5$ and $\sigma_{At} = 0.5$. The fuel consumption and the attitude error are optimized by finding a trade-off between them, as shown in the third column of Table 4.2. As it is possible to see in Figure 4.2 (the dashed red line), the path is quite similar to the previous case, but near the target, the attitude error component in the cost function becomes prevalent over fuel consumption, and then the best reference values are those that turn the aircraft to minimize the attitude error. Therefore, the fuel is higher than in the previous case due to this variation on the path, which minimizes the contribution of attitude error. It is worth mentioning that the attitude is minimized only at the end

of the path because if other paths (not the shortest) had been followed, the component of the fuel consumption would have been higher. For this reason, the attitude error in this case is not completely zero.

Attitude Error Minimization: Finally, only the attitude error is minimized by setting $\sigma_{At} = 1$ and $\sigma_f = 0$. The behavior is quite similar to the base case (the aircraft has to reach only the target position, avoiding the no-fly zones, the first column of Table 4.2) in terms of fuel and airspeed since these components are not optimized, in this case. The results are shown in the fourth column of Table 4.2. However, the path is completely different compared to the previous cases, as it is possible to see with the green dotted line in Figure 4.2. The trajectory passes on the right side of the second no-fly zone to facilitate the angle of approach to the target. This occurs because, during the POMCP simulation in the future, the reference values that minimize attitude error (which involves passing to the right of the second no-fly zone) receive a higher reward compared to other reference values for minimizing the cost function. Therefore, the aircraft approaches the target with the correct attitude, obtaining a minimum attitude error.

4.4 Discussion

In this chapter, we demonstrated how the POMDP framework can be effectively employed to plan and control the trajectory of a nonlinear aircraft system under measurement and environmental uncertainty. By leveraging a linearized closed-loop model within the high-level POMDP planner, we enabled robust decision-making despite the complexity of the system dynamics and the partial observability of the environment.

The proposed two-layer control architecture—combining a belief-based POMDP planner with an inverse dynamics low-level controller—successfully integrates state estimation and action selection into a unified planning process. Using POMCP, we computed optimal reference commands in real-time, guiding the aircraft to its target while satisfying constraints and avoiding no-fly zones.

The main takeaway is that model-based planning under uncertainty, when structured through a POMDP and solved with sampling-based methods, provides a scalable and reliable approach to autonomous control in complex, uncertain environments.

Chapter 5

PATH-PLANNING FOR AIRCRAFT AUTONOMY WITH MDP ON LIE-GROUP

"If I were again beginning my studies, I would follow the advice of Plato and start with mathematics."

Galileo Galilei

Abstract[1] In this chapter, we develop a trajectory planning framework based on a Markov Decision Process (MDP), where state estimation is performed using an optimal filter defined on the Lie group structure of the system. Specifically, we represent the aircraft dynamics on the tangent bundle of the Special Euclidean group in $\mathbb{R}^3$, denoted as TSE(3), and design a minimum-energy observer to estimate the system state.

Building on the two-layer architecture introduced in Chapter 4, we now exploit the geometric structure of the dynamics. The low-level controller is tasked with tracking left-trivialized velocities defined on the Lie algebra, while the high-level MDP planner operates on the Lie group to minimize the tracking error based on the estimated state. Unlike the POMDP framework used in the previous chapter, which explicitly modeled measurement uncertainty, here the uncertainty is managed by the observer, and the planner assumes full state observability.

To compute reference commands for the low-level controller in real-time, we solve the MDP using Monte Carlo Tree Search (MCTS). The planner guides the aircraft to its target while avoiding no-fly zones and minimizing a predefined cost function.

The key contributions of this chapter are:

- A geometric formalization of aircraft dynamics on the Lie group TSE(3), capturing both position and velocity within a unified framework.
- The design of a low-level controller for tracking left-trivialized velocities on the Lie algebra.
- The development of an MDP-based high-level planner that exploits the estimated velocities to generate optimal reference trajectories on the Lie group.

The chapter is structured as follows: Section 5.1 formalizes the nonlinear system on the Lie group. Section 5.3 states the minimum-energy optimal filter on the Lie group. Section 5.4 formalizes the low-level controller on the left-trivialized

[1] The content of this chapter is based on Francesco Trotti, Damiano Rigo, and Riccardo Muradore, "Geometric methods for aircraft planning and control", submitted

velocities, while Section 5.5 discusses the high-level controller (MDP) on the Lie group. Section 5.6 presents simulation results.

5.1 Geometric formulation

In this section, we describe the geometric structure of the nonlinear dynamics model presented in Section 2.7 that will be useful for our planning, control, and filtering purposes. We will characterize the structure of the Lie group and identify the elements of Lie groups and Lie algebras with their matrix representations.

The mechanical system that we consider evolves on TSE(3), the tangent bundle of the Special Euclidean group in $\mathbb{R}^3$. For the sake of simplicity and for reasons that will be clearer in the context of filtering and control, we consider the following vector state

$$\mathcal{X} := [p, q, v, \omega]^T = [p_x, p_y, p_z, q_0, q_1, q_2, q_3, v_1, v_2, v_3, \omega_1, \omega_2, \omega_3]^T, \tag{5.1}$$

where $p = \left[p_x, p_y, p_z\right]^T$ is the aircraft position in Earth frame, $q = [q_0, q_1, q_2, q_3]^T$ describes the aircraft orientation in Earth frame, $v = [v_1, v_2, v_3]^T$ and $\omega = [\omega_1, \omega_2, \omega_3]^T$ are the linear and angular velocities in body frame, respectively. The quaternion description for the orientation is exploited to avoid singularity problems that arise with the use of Euler angles [84]. Moreover, some computations become easier due to their simple formulation (see e.g. [85]). The first seven elements of the state vector describe the 3D pose of the aircraft. In particular, $\bar{q} = (q_1, q_2, q_3)$ represents the vector component of a unitary quaternion, i.e. $q = (q_0, \bar{q})$ such that $||q|| = 1$. Given two quaternions $q = (p_0, \bar{q})$ and $b = (b_0, \bar{b})$ we define the Hamiltonian quaternion product as

$$q \circ_{\mathbb{H}} b = b_0 q_0 - \bar{b} \cdot \bar{q} + b_0 \bar{q} + q_0 \bar{b} + \bar{b} \times \bar{q} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

The conjugate of $q = (q_0, \bar{q})$ is $q^* = (q_0, -\bar{q})$, while its norm $||q||$ and inverse q^{-1} are given respectively by

$$||q|| = \sqrt{q \circ_{\mathbb{H}} q^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \quad q^{-1} = \frac{q^*}{||q||^2}.$$

The configuration space of an aerial vehicle is SE(3), which considers rotations and translations in $\mathbb{R}^3$. A generic element $Q \in \mathrm{SE}(3)$ can be written as

$$Q = \begin{bmatrix} R_{eb} & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & p_x \\ & R_{eb} & & p_y \\ & & & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where R_{eb} represents an orthogonal matrix (i.e. $R_{eb}^T R_{eb} = R_{eb} R_{eb}^T = I$) that, with the use of the quaternion q, can be conveniently written as

$$R_{eb} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & 1-2(q_1^2+q_2^2) \end{bmatrix}.$$

The linear and angular velocities v and ω admit an additive Lie group structure portrayed by the matrix

$$V = \begin{bmatrix} I_{3,3} & 0_{3,3} & v \\ 0_{3,3} & I_{3,3} & \omega \\ 0_{1,3} & 0_{1,3} & 1 \end{bmatrix}, \; v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \; \omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \tag{5.2}$$

With these choices, the configuration space will be described by the matrix Lie group G whose generic element $g \in G$ has the form

$$g = \begin{bmatrix} Q & 0_{4,7} \\ 0_{7,4} & V \end{bmatrix}. \tag{5.3}$$

The operation of two matrix Lie groups is the matrix multiplication, the neutral element is the identity matrix, while the inverse of g is the matrix inversion. We denote with $\mathfrak{g}$ the Lie algebra of the Lie group G. A generic element $\lambda \in \mathfrak{g}$ has the matrix representation

$$\lambda = \begin{bmatrix} \lambda_Q & 0_{4,7} \\ 0_{7,4} & \lambda_V \end{bmatrix}, \tag{5.4}$$

with

$$\lambda_Q = \begin{bmatrix} \lambda_R & \lambda_p \\ 0_{1,3} & 0 \end{bmatrix}, \lambda_R = \begin{bmatrix} 0 & -\lambda_{q_3} & \lambda_{q_2} \\ \lambda_{q_3} & 0 & -\lambda_{q_1} \\ -\lambda_{q_2} & \lambda_{q_1} & 0 \end{bmatrix}, \lambda_V = \begin{bmatrix} 0_{1,6} & \lambda_v \\ 0_{1,6} & \lambda_\omega \\ 0_{1,6} & 0 \end{bmatrix},$$
$$\lambda_p = \begin{bmatrix} \lambda_{p_x} \\ \lambda_{p_y} \\ \lambda_{p_z} \end{bmatrix}, \lambda_q = \begin{bmatrix} \lambda_{q_1} \\ \lambda_{q_2} \\ \lambda_{q_3} \end{bmatrix}, \lambda_v = \begin{bmatrix} \lambda_{v_1} \\ \lambda_{v_2} \\ \lambda_{v_3} \end{bmatrix}, \lambda_\omega = \begin{bmatrix} \lambda_{\omega_1} \\ \lambda_{\omega_2} \\ \lambda_{\omega_3} \end{bmatrix}.$$

In our case, the Lie bracket operation is given by the matrix commutator i.e., $[A, B] = AB - BA$ where $A, B \in \mathfrak{g}$. To facilitate computations, we introduce two Lie algebra isomorphisms. The first one is defined as the "wedge map"

$$\cdot^\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$$
$$\begin{bmatrix} \lambda_{q_1} \\ \lambda_{q_2} \\ \lambda_{q_3} \end{bmatrix} \mapsto \begin{bmatrix} \lambda_{q_1} \\ \lambda_{q_2} \\ \lambda_{q_3} \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\lambda_{q_3} & \lambda_{q_2} \\ \lambda_{q_3} & 0 & -\lambda_{q_1} \\ -\lambda_{q_2} & \lambda_{q_1} & 0 \end{bmatrix}$$

from the Lie algebra $(\mathbb{R}^3, \wedge)$ to the Lie algebra $(\mathfrak{so}(3), [\cdot, \cdot])$ where $\wedge$ is the wedge product and $[\cdot, \cdot]$ is the matrix commutator. The second is the isomorphism $\cdot^\star$

$$\cdot^\star : \mathbb{R}^{13} \to \mathfrak{g}$$
$$\begin{bmatrix} \lambda_p \\ \lambda_q \\ \lambda_v \\ \lambda_\omega \end{bmatrix}^\star = \begin{bmatrix} \lambda_Q & 0_{4,7} \\ 0_{7,4} & \lambda_V \end{bmatrix}$$

from the Lie algebra $(\mathbb{R}^{12}, \star)$ to the Lie algebra $(\mathfrak{g}, [\cdot,\cdot])$ where $\star$ is the Lie bracket operation defined by

$$\begin{bmatrix}\lambda_p\\ \lambda_q\\ \lambda_V\\ \lambda_\omega\end{bmatrix} \star \begin{bmatrix}\nu_p\\ \nu_q\\ \nu_V\\ \nu_\omega\end{bmatrix} = \begin{bmatrix}\lambda_q^\wedge \nu_p + \nu_q^\wedge \lambda_p\\ \lambda_q^\wedge \nu_q\\ 0\\ 0\end{bmatrix} = \begin{bmatrix}\lambda_q^\wedge \nu_p - \lambda_p^\wedge \nu_q\\ \lambda_q^\wedge \nu_q\\ 0\\ 0\end{bmatrix},$$

that expanded becomes

$$\begin{bmatrix}\lambda_{p_x}\\ \lambda_{p_y}\\ \lambda_{p_z}\\ \lambda_{q_1}\\ \lambda_{q_2}\\ \lambda_{q_3}\\ \lambda_{v_1}\\ \lambda_{v_2}\\ \lambda_{v_3}\\ \lambda_{\omega_1}\\ \lambda_{\omega_2}\\ \lambda_{\omega_3}\end{bmatrix} \star \begin{bmatrix}\nu_{p_x}\\ \nu_{p_y}\\ \nu_{p_z}\\ \nu_{q_1}\\ \nu_{q_2}\\ \nu_{q_3}\\ \nu_{v_1}\\ \nu_{v_2}\\ \nu_{v_3}\\ \nu_{\omega_1}\\ \nu_{\omega_2}\\ \nu_{\omega_3}\end{bmatrix} = \begin{bmatrix}\lambda_{q_2}\nu_{p_z} - \lambda_{q_3}\nu_{p_y} + \lambda_{p_y}\nu_{q_3} - \lambda_{p_z}\nu_{q_2}\\ \lambda_{q_3}\nu_{p_x} - \lambda_{q_1}\nu_{p_z} - \lambda_{p_x}\nu_{q_3} + \lambda_{p_z}\nu_{q_1}\\ \lambda_{q_1}\nu_{p_y} - \lambda_{q_2}\nu_{p_x} + \lambda_{p_x}\nu_{q_2} - \lambda_{p_y}\nu_{q_1}\\ \lambda_{q_2}\nu_{q_3} - \lambda_{q_3}\nu_{q_2}\\ \lambda_{q_3}\nu_{q_1} - \lambda_{q_1}\nu_{q_3}\\ \lambda_{q_1}\nu_{q_2} - \lambda_{q_2}\nu_{q_1}\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\end{bmatrix}.$$

Given $g \in G$ the adjoint action $\mathrm{Ad}_g : G \times \mathfrak{g} \to \mathfrak{g}$ evaluated in $\lambda \in \mathfrak{g}$ takes the form

$$\mathrm{Ad}_g \lambda = g \lambda g^{-1},$$

while the adjoint operator $\mathrm{ad}_\lambda \in L(\mathfrak{g}, \mathfrak{g})$ admits the following matrix representation

$$\mathrm{ad}_\lambda = \begin{bmatrix}\lambda_q^\wedge & -\lambda_p^\wedge & 0_{3,3} & 0_{3,3}\\ 0_{3,3} & \lambda_q^\wedge & 0_{3,3} & 0_{3,3}\\ 0_{3,3} & 0_{3,3} & 0_{3,3} & 0_{3,3}\\ 0_{3,3} & 0_{3,3} & 0_{3,3} & 0_{3,3}\end{bmatrix}. \tag{5.5}$$

The dual adjoint operator $\mathrm{ad}^*_\lambda \in L(\mathfrak{g}^*; \mathfrak{g}^*)$ satisfies $\mathrm{ad}^*_\lambda = \mathrm{ad}^T_\lambda$. Given the unitary quaternion $q = (q_0, \bar{q})$, it can be proven that the "quaternion" angular velocity $\omega = (0, \bar{\omega})$ is equivalent to $\omega = 2q^* \circ_{\mathbb{H}} \dot{q}$ (see [86]). This leads to the quaternion differentiation

$$\dot{q} = \frac{1}{2} q \circ_{\mathbb{H}} \omega$$

which provides us the tangent of the left quaternion multiplication map relative to the vector q

$$T_e L_q = \frac{1}{2}\begin{bmatrix}q_0 & -q_3 & q_2\\ q_3 & q_0 & -q_1\\ -q_2 & q_1 & q_0\end{bmatrix},$$

where e is the identity map. It follows that the left multiplication map relative to a generic element $g \in G$ is given by

$$T_eL_g = \begin{bmatrix} R_{eb} & 0_{3,3} & 0_{3,3} & 0_{3,3} \\ 0_{3,3} & T_eL_q & 0_{3,3} & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & I_{3,3} & 0_{3,3} \\ 0_{3,3} & 0_{3,3} & 0_{3,3} & I_{3,3} \end{bmatrix}.$$

Since we are considering a differentiable structure, it is necessary to choose an affine connection on the state space manifold. A left-invariant affine connection ∇ on G is characterized by its bilinear connection function $\Omega : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ through the identity $\nabla_{gX}(gY) = g\Omega(X, Y)$ for all $X, Y \in \mathfrak{g}$ (see, e.g., [87], [88]). Thus $\nabla_X Y \in \mathfrak{g}$, and, since this map is $\mathbb{R}$-linear, it is a multiplication in $\mathfrak{g}$. We choose a skew-symmetric connection function of the form $\nabla_X Y = \lambda[X, Y]$, $\lambda \in \mathbb{R}$ (see e.g. [89], [90]). The choices $\lambda = 0, \frac{1}{2}, 1$ define the (-), (0), and (+) connections that have negative, null, and positive torsion, respectively.

One of the advantages of using the Lie groups theory is that one can consider the trivialized dynamics defined on the Lie algebra. Using the items previously defined, the dynamic equation (2.10)-(2.20) can be rewritten as

$$\dot{g} = g\lambda$$

where $\lambda \in \mathfrak{g}$ is the left-trivialized dynamics with components:

$$\begin{aligned}
\lambda_{p_x} &= v_1, \quad \lambda_{p_y} = v_2, \quad \lambda_{p_z} = v_3, \\
\lambda_{q_1} &= \omega_1, \quad \lambda_{q_2} = \omega_2, \quad \lambda_{q_3} = \omega_3, \\
\lambda_{v_1} &= \omega_3 v_2 - \omega_2 v_3 + \frac{T + C_x}{m} + g_r R_{eb}(1,3), \\
\lambda_{v_2} &= \omega_1 v_3 - \omega_3 v_1 + \frac{C_y}{m} + g_r R_{eb}(2,3), \\
\lambda_{v_3} &= \omega_2 v_1 - \omega_1 v_2 + \frac{C_z}{m} + g_r R_{eb}(3,3), \\
\lambda_{\omega_1} &= (c_2\omega_1 + c_1\omega_3)\omega_2 + \delta_a, \\
\lambda_{\omega_2} &= (c_5\omega_1)\omega_3 + c_6(\omega_3\omega_3 - \omega_1\omega_1) + \delta_e, \\
\lambda_{\omega_3} &= (c_8\omega_1 - c_2\omega_1)\omega_2 + \delta_r,
\end{aligned} \tag{5.6}$$

where the external commands T, δ_e, δ_a, and δ_r are the thrust and the elevator, aileron, and rudder deflections, $R_{eb}(i, j)$ is the i-th row and j-th column of the matrix R_{eb}, $[c_1, \ldots, c_8]$ are the inertial parameters and C_x, C_y, C_z are the aerodynamic coefficients.

5.2 Problem Statement

The problem that we want to solve consists of autonomous spatial aircraft guidance considering measurements obtained by noisy sensors. The fixed-wing aircraft has to reach a target position, avoiding some no-fly zones in a 3D scenario. A hierarchical control architecture (two-layer system) is used to plan the path and control the aircraft. Moreover, a Lie group formalization is used to exploit the geometrical properties of the dynamic model. The hierarchical architecture is composed of different control levels: a high-level controller formalized as a Markov Decision Process, a low-level controller to manage the aircraft deflection surfaces, and a minimum energy optimal filter to estimate the state given noisy measurements. The outline of the entire control architecture is illustrated in Figure 5.1.

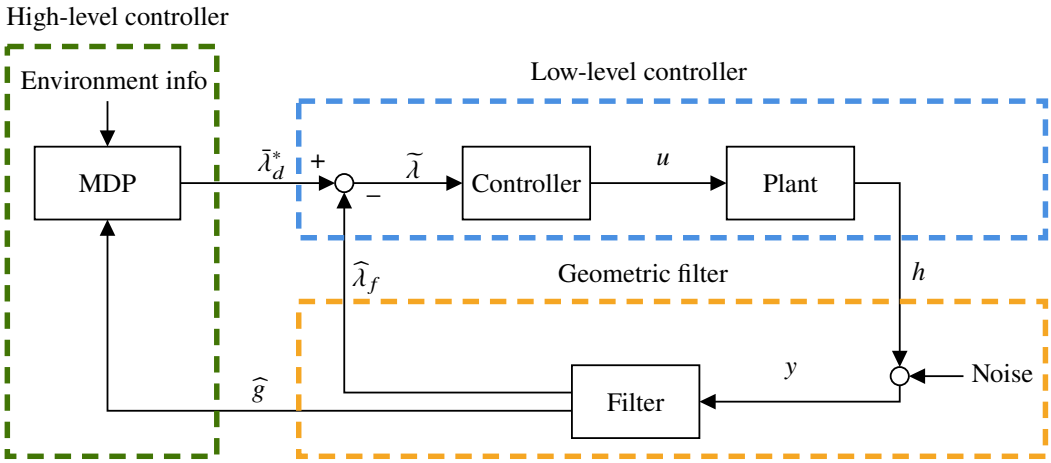


Fig. 5.1: Hierarchical control architecture.

Minimum energy optimal filter

The measurements y of the state $g \in G$ are corrupted by noises. An optimal minimum energy filter is designed to take into account the noisy measurements and estimate the state. The estimated state $\widehat{g} \in G$, where G is the Lie group, is the input for the high-level controller. The left-trivialized estimated velocities $\widehat{\lambda} \in \mathfrak{g}$, where $\mathfrak{g}$ is the Lie algebra associated with G, are the inputs for the low-level controller.

Low-level controller

The low-level controller takes as inputs the optimal reference values $\bar{\lambda}_d^* \in \mathbb{R}^4$ provided by the high-level controller and the estimated left-trivialized velocities $\widehat{\lambda}_f \in \mathbb{R}^4$ computed by the filter. Based on the error $\tilde{\lambda} = \bar{\lambda}_d^* - \widehat{\lambda}$, the low-level controller computes the control u.

High-level controller

We consider the Monte Carlo Tree Search (MCTS) strategy to solve the Markov Decision Process (MDP) and the related planning problem to compute the optimal or suboptimal reference values $\bar{\lambda}_d^*$, taking into account the estimated state $\widehat{g}$.

Definition 5.1: *The Markow Decision Process (MDP) $\mathcal{M}$ is formulated as*

1. *States: The state $\bar{g} \in G$ represents the state of the system.*
2. *Actions: The actions $\lambda_d \in \mathcal{U} \subset \mathfrak{g}$ represent the set of admissible reference values.*
3. *Transition model: The transition model is the dynamic system*

$$\dot{\bar{g}} = \bar{g}\lambda(\bar{g}, u). \tag{5.7}$$

   where λ is the left-trivialized dynamics, $u \in \mathbb{R}^4$ the control signal, and $\bar{g}$ is the state of the MDP with the initial condition the estimated state $\hat{g}$.
4. *Cost function: $l(\bar{g}, \lambda_d)$ represents the immediate cost received after taking action λ_d in state $\bar{g}$.*

We are interested in designing the optimal minimum-energy filter to estimate the state; after that, we study the controller exploiting the left-trivialized estimated velocities, and finally, we analyze how the MCTS algorithm plans the trajectory of the aircraft, taking into account the controlled plant.

The following problems will be tracked in this chapter:

1. Given the noisy measurements y and commands u, compute the estimated state $\widehat{g}$ and left-trivialized velocities $\widehat{\lambda}$ of the aircraft.
2. Given the estimated state $\widehat{g}$, compute optimal desired left-trivialized velocities $\bar{\lambda}_d^*$ to plan the trajectory of the aircraft (high-level controller).
3. Given the reference velocities $\bar{\lambda}_d^*$ and estimated left-trivialized velocities $\widehat{\lambda}$, compute the command u to minimize the tracking error (low-level controller).

5.3 Minimum-energy filter

In this section, we recall the second-order optimal filter designed on Lie groups proposed in [40]. In the filter setting, we consider the modified dynamics, which also take into account the model error

$$\dot{g}(t) = g(t)\big(\lambda(g(t), u(t)) + B\xi(t)\big), \quad g(t_0) = g_0, \tag{5.8}$$

where $g(t) \in G$ is the state, $u(t) = [T, \delta_a, \delta_e, \delta_r] \in \mathbb{R}^4$ is the command input, $\xi(t)$ is the unknown model error, $\lambda : G \times \mathbb{R}^4 \times \mathbb{R} \to \mathfrak{g}$ is the left trivialized dynamics and $B : \mathbb{R}^6 \to \mathfrak{g}$ is a linear map. It is sufficient to add model errors only on the "velocity" component; thus, the matrix form B is

$$B = \begin{bmatrix} 0_{7\times 6} \\ B_2 \end{bmatrix}, \quad B_2 = \left[\operatorname{diag}(\xi_{v_1}, \xi_{v_2}, \xi_{v_3}, \xi_{\omega_1}, \xi_{\omega_2}, \xi_{\omega_3})\right]. \tag{5.9}$$

The measurement equation is

$$y(t) = h(g(t)) + D\varepsilon(t), \tag{5.10}$$

where $h : G \to \mathbb{R}^{13}$ is the output map

$$h(g(t)) := [p_x, p_y, p_z, q_0, q_1, q_2, q_3, v_1, v_2, v_3, \omega_1, \omega_2, \omega_3],$$

$\varepsilon \in \mathbb{R}^{13}$ is the unknown measurement error and $D : \mathbb{R}^{13} \to \mathbb{R}^{13}$ is the invertible linear map

$$D = \text{diag}(\varepsilon_{p_x}, \varepsilon_{p_y}, \varepsilon_{p_z}, \varepsilon_{q_0}, \varepsilon_{q_1}, \varepsilon_{q_2}, \varepsilon_{q_3}, \varepsilon_{v_1}, \varepsilon_{v_2}, \varepsilon_{v_3}, \varepsilon_{\omega_1}, \varepsilon_{\omega_2}, \varepsilon_{\omega_3}). \tag{5.11}$$

For structure constraint, we have to impose that the vector of the quaternion measurements has a unitary norm.

Given the external command $u(t)$ and the measurement output $y(t)$, the goal of the filter is to find the best estimate of the state $g(t)$ minimizing the cost functional

$$J(\xi, \varepsilon, g_0, t, t_0) := m(g_0, t, t_0) + \int_{t_0}^{t} l(\xi(\tau), \varepsilon(\tau), t, \tau)\mathrm{d}\tau, \tag{5.12}$$

where $l : \mathbb{R}^6 \times \mathbb{R}^{13} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the incremental cost defined by

$$l(\xi, \varepsilon, t, \tau) := \frac{1}{2} e^{-c(t-\tau)}(\mathcal{R}(\xi) + \mathcal{Q}(\varepsilon)), \tag{5.13}$$

with c a non-negative scalar (to simplify the computation, we have chosen $c = 0$) and

$$\mathcal{R} : \mathbb{R}^6 \to \mathbb{R}, \quad \mathcal{Q} : \mathbb{R}^{13} \to \mathbb{R},$$

two quadratic forms that weigh the contribution of model and measurement errors. The function $m : G \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the initial cost and takes the form

$$m(g_0, t, t_0) := \frac{1}{2} e^{-c(t-t_0)} m_0(g_0) = m_0(g_0), \tag{5.14}$$

with $m_0 : G \to \mathbb{R}$ the bounded smooth function

$$m_0(g_0) = ||I - g^{-1}g_0||_F^2 = \text{trace}((I - g^{-1}g_0)^T(I - g^{-1}g_0)),$$

where $||\cdot||_F$ is the Frobenius norm.

We recall now the main result in [40].

Lemma 5.1: *([40, Theorem 4.1]) Consider the system defined by* (5.8) *and* (5.10) *with the energy cost functional* (5.12)-(5.14)*. Then the second-order-optimal minimum-energy filter yields the estimation state $\widehat{g}$ given by*

$$\widehat{g}^{-1}\dot{\widehat{g}} = \lambda(\widehat{g}, u) + K(t)r(\widehat{g}), \quad \widehat{g}(t_0) = \widehat{g}_0,$$

where $K(t) : \mathfrak{g}^* \to \mathfrak{g}$ is a (time-varying) second-order-optimal symmetric gain operator satisfying the perturbed Riccati operator (5.15) *given below,*

$$\widehat{g}_0 = \underset{g \in G}{\text{argmin}}\, m_0(g),$$

and the residual $r(\widehat{g}) \in \mathfrak{g}^*$ *is computed as*

$$r(\widehat{g}) = T_e L^*_{\widehat{g}}[\big((D^{-1})^* \circ Q \circ D^{-1}(y - h(\widehat{g}))\big) \circ \mathrm{d}h(\widehat{g})].$$

The perturbed Riccati equation for K *is*

$$\dot{K} = -\alpha \cdot K + A \circ K + K \circ A^* - K \circ E \circ K + B \circ R^{-1} \circ B^* - \Omega_{Kr} \circ K - K \circ \Omega^*_{Kr} \tag{5.15}$$

with initial condition $K(t_0) = X_0^{-1}$. *The operators* $X_0 : \mathfrak{g} \to \mathfrak{g}^*$, $A(t) : \mathfrak{g} \to \mathfrak{g}$, *and* $E(t) : \mathfrak{g} \to \mathfrak{g}^*$ *are given by*

$$\begin{aligned}
X_0 &= T_e L^*_{\hat{g}_0} \circ \mathrm{Hess}\, m_0(\hat{g}) \circ T_e L_{\hat{g}_0},\\
A(t) &= \mathrm{d}_1\lambda(\widehat{g}, u) \circ T_e L_{\widehat{g}} - \mathrm{ad}_{\lambda(\widehat{g},u)} - T_{\lambda(\widehat{g},u)},\\
E(t) &= -T_e L^*_{\widehat{g}} \circ \\
&\quad \big[\big((D^{-1})^* \circ Q \circ D^{-1}(\mathrm{y} - h(\widehat{g}))\big)^{T_{\widehat{g}}G} \circ \mathrm{Hess}\, h(\widehat{g})\\
&\quad - (\mathrm{d}h(\widehat{g}))^* \circ (D^{-1})^* \circ Q \circ D^{-1} \circ \mathrm{d}h(\widehat{g})\big] \circ T_e L_{\widehat{g}}.
\end{aligned}$$

To simplify the computation, we chose the affine connection associated with the choice of the Cartan-Schauten $(-)$connection: $\Omega(X, Y) = 0$. With this option, the torsion function is the opposite of the adjoint operator (see, e.g. [3], [89], [90]), and thus the operator $A(t)$ simplifies to

$$A(t) = \mathrm{d}_1\lambda(\widehat{g}, u) \circ T_e L_{\widehat{g}},$$

where d_1 indicates the differentiation with respect to the first argument. The term Ω_{Kr} in the Riccati equation is the contraction of the two-form Ω with respect to the vector field Kr

$$\Omega_{Kr}(\cdot) := \Omega(Kr, \cdot) = 0,$$

thus the last two addends of (5.15) disappear. Finally, given gX, $gY \in T_g G$, the Hessian operator $\mathrm{Hess}\, h(g) : T_g G \to T^*_g G$ at a point $g \in G$, defined as

$$\begin{aligned}
\mathrm{Hess}\, h(g)(gX)(gY) &= \mathrm{d}(\mathrm{d}h(g)(gY))(gX) - \mathrm{d}h(g)(\nabla_{gX}(gY))\\
&= \mathrm{d}(\mathrm{d}h(g)(gY))(gX) - \mathrm{d}h(g)(g\Omega(X, Y)),
\end{aligned}$$

becomes zero since the double differentiation of h provides the null tensor, and thus, the operator $E(t)$ modifies to

$$E(t) = T_e L^*_{\widehat{g}} \circ \big[(\mathrm{d}h(\widehat{g}))^* \circ (D^{-1})^* \circ Q \circ D^{-1} \circ \mathrm{d}h(\widehat{g})\big] \circ T_e L_{\widehat{g}}.$$

5.4 Low-level Controller

The low-level controller is designed to compute the control $u = [T, \delta_e, \delta_a, \delta_r]$ to reach an optimal desired velocity vector $\bar{\lambda}_d^* = [V_{T_d}, \omega_{1_d}, \omega_{2_d}, \omega_{3_d}]^T \in \mathbb{R}^4$. We can solve the control problem using the estimated state and the left-trivialized velocities provided by the filter presented in Section 5.3 in two different ways. The first approach uses the feedback estimated velocities $\widehat{V}_f = [\widehat{V}_T, \widehat{\omega}_1, \widehat{\omega}_2, \widehat{\omega}_3]^T \in \mathbb{R}^4$ to define the tracking error

$$\widetilde{\lambda}_1 := \bar{\lambda}_d^* - \widehat{V}_f = \begin{bmatrix} V_{T_d} - \widehat{V}_T \\ \omega_{1_d} - \widehat{\omega}_1 \\ \omega_{2_d} - \widehat{\omega}_2 \\ \omega_{3_d} - \widehat{\omega}_3 \end{bmatrix}, \tag{5.16}$$

where

$$\widehat{V}_T = \sqrt{\widehat{v}_1^2 + \widehat{v}_2^2 + \widehat{v}_3^2}.$$

Note that the estimated linear $[\widehat{v}_1, \widehat{v}_2, \widehat{v}_3]^T$ and angular velocities $[\widehat{\omega}_1, \widehat{\omega}_2, \widehat{\omega}_3]^T$ are defined in the tangent bundle of $\mathrm{SE}(3)$. The second approach considers the feedback left-trivialized velocities

$$\widehat{\lambda}_f = [\widehat{\lambda}_{V_T}, \widehat{\lambda}_{q_1}, \widehat{\lambda}_{q_2}, \widehat{\lambda}_{q_3}]^T \in \mathbb{R}^4$$

where

$$\widehat{\lambda}_{V_T} = \sqrt{\widehat{\lambda}_{p_x}^2 + \widehat{\lambda}_{p_y}^2 + \widehat{\lambda}_{p_z}^2}.$$

The tracking error is

$$\widetilde{\lambda}_2 := \bar{\lambda}_d^* - \widehat{\lambda}_f = \begin{bmatrix} V_{T_d} - \widehat{\lambda}_{V_T} \\ \omega_{1_d} - \widehat{\lambda}_{q_1} \\ \omega_{2_d} - \widehat{\lambda}_{q_2} \\ \omega_{3_d} - \widehat{\lambda}_{q_3} \end{bmatrix}, \tag{5.17}$$

Note that the left-trivialized velocities $[\widehat{\lambda}_{p_x}, \widehat{\lambda}_{p_y}, \widehat{\lambda}_{p_z}, \widehat{\lambda}_{q_1}, \widehat{\lambda}_{q_2}, \widehat{\lambda}_{q_3}]$ are defined in $\mathfrak{se}(3)$.

Therefore, the commands are computed using the control laws

$$\begin{aligned} T =& m(-K_1\widetilde{\lambda}_i(1) - (-\widehat{\omega}_2\widehat{v}_3 + \widehat{\omega}_3\widehat{v}_2 + g_r R_{eb}(1,3))) - C_x, \\ \delta_a =& -K_2\widetilde{\lambda}_i(2) - (c_2\widehat{\omega}_1 + c_1\widehat{\omega}_3)\widehat{\omega}_2, \\ \delta_e =& -K_3\widetilde{\lambda}_i(3) - c_5\widehat{\omega}_1\widehat{\omega}_3 - c_6(\widehat{\omega}_3^2 - \widehat{\omega}_1^2), \\ \delta_r =& -K_4\widetilde{\lambda}_i(4) - (c_8\widehat{\omega}_1 - c_2\widehat{\omega}_3)\widehat{\omega}_2 \end{aligned}$$

where $i \in \{1, 2\}$ for the two possible approaches and K_j with $j \in \{1, \ldots, 4\}$ are positive scalar gains.

5.5 MDP planner

Given the geometric formalization of the optimal filter to estimate the state $\widehat{g}$, we now formalize a sub-optimal geometric planner to compute the reference values for the low-level controller. The planner is formalized as a Markov Decision Process (MDP) solved with the Monte Carlo Tree Search (MCTS) algorithm. It has to drive the aircraft to the target point with a defined attitude, avoiding some no-fly zones and minimizing a cost function.

5.5.1 MDP formalization

We formalize the MDP as a planner as follows.

State: The MDP state vector represents the system variables that evolve over time ($\bar{g}$). The initial state vector used in the MDP is the estimated state $\widehat{g}$ provided by the optimal filter.

Action. The action set $\mathcal{U}$ used in the MDP consists of the reference values for the low-level controller

$$\bar{\lambda}_d = (V_{T_d}, \omega_{1_d}, \omega_{2_d}, \omega_{3_d}). \tag{5.18}$$

Transition function. The transition function defines the evolution of the system, given the current state and the action. The simulated dynamics is

$$\dot{\bar{g}} = \bar{g}\lambda(\bar{g}, u) \quad \text{where} \quad \bar{g}_0 = \widehat{g} \tag{5.19}$$

where the initial condition $\bar{g}_0$ is provided by the filter. Additionally, the low-level controller is simulated to drive the evolution of the dynamics, providing the right control u to the aircraft. In this way, simulating the controlled plant, it is possible to evaluate the impact of the selected actions in the cost function (i.e., references for the low-level controller).

Cost function. The cost function evaluates the chosen action in the current state. It is composed of sub-functions that minimize different metrics to reach the target position while complying with several constraints, such as avoiding no-fly zones and minimizing attitude error relative to the target. We define the desired reference and the estimated Lie group trajectories as $g_d \in G$ and $\widehat{g} \in G$, respectively. Therefore, the tracking error on Lie groups is

$$\widetilde{g} = g_d^{-1}\bar{g} = \begin{bmatrix} Q_d^{-1} & 0_{4,7} \\ 0_{7,4} & V_d^{-1} \end{bmatrix} \begin{bmatrix} \bar{Q} & 0_{4,7} \\ 0_{7,4} & \bar{V} \end{bmatrix} = \begin{bmatrix} Q_d^{-1}\bar{Q} & 0_{4,7} \\ 0_{7,4} & V_d^{-1}\bar{V} \end{bmatrix}. \tag{5.20}$$

where $Q_d, V_d, \bar{Q}, \bar{V}$ are the matrices in (5.3) evaluated in the desired trajectory and during the MDP process, respectively. Note that we can not define the error in the following way $\widetilde{g} = g_d^{-1}g$ because g is unknown. To solve the tracking problem, we want to drive the tracking error $\widetilde{g}$ to the identity $I \in G$. To minimize the pose error of the aircraft, we focus only on the position and orientation components of the estimated state error $\widetilde{g}$

$$Q_d^{-1}\bar{Q} = \begin{bmatrix} R_{eb_d}^T & -R_{eb_d}p_d \\ 0_{1,3} & 1 \end{bmatrix} \begin{bmatrix} \bar{R}_{eb} & \bar{p} \\ 0_{1,3} & 1 \end{bmatrix}. \tag{5.21}$$

The cost to minimize the pose error is computed with the Frobenius norm

$$r_{\text{pos}}(g) = ||Q_d^{-1}\bar{Q}||_F = \sqrt{\text{trace}((I - Q_d^{-1}\bar{Q})^T(I - Q_d^{-1}\bar{Q}))}. \tag{5.22}$$

The function to avoid the no-fly zones is modeled as a repulsive force [83]

$$r_{A_i}(g) = r_{\text{nofly}_i}(g) = \begin{cases} \nu\left(\frac{1}{d_{A_i}} - \frac{1}{d_0}\right)^2, & \text{if } d_{A_i} < d_0 \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

where d_{A_i} is the distance between the aircraft and the i-th no-fly zone, d_0 is the repulsive threshold and ν is the repulsive coefficient. The total cost concerning the N no-fly zones is thus

$$r_{\text{nofly}}(g) = \sum_{i=1}^{N} r_{A_i}(g). \tag{5.24}$$

The overall cost function to be minimized is given by the sum of the single costs

$$l(g) = r_{\text{pos}}(g) + r_{\text{nofly}}(g). \tag{5.25}$$

5.5.2 Monte Carlo Tree Search Planner

Given the MDP, we recall the formalization of the Monte Carlo Tree Search, and we explain how MCTS solves an optimization problem to generate optimal reference values for the controlled aircraft.

Optimization problem using MCTS

The MCTS algorithm solves the following discrete optimization problem with the discretization step Δt. We indicate the predicted evolution of the state with $k+i|k$ where the right k is the actual time, and the left $k+i$ is the time in the prediction horizon.

Optimization Problem MCTS:

$$\begin{aligned}
&\min_{\bar{\lambda}_d} \; U_n(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}}) + C\sqrt{\frac{\log N(\bar{g}_{k+i|k})}{N(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})}} \\
&\text{s. t.:} \\
&\quad \bar{g}_{k|k} = \widehat{g}_k \\
&\quad \bar{g}_{k+i+1|k} = \bar{g}_{k+i|k} \exp[\Delta t\, \bar{\lambda}(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})] \qquad (5.26a) \\
&\quad \lambda_{d_{k+i|k}} \in \mathcal{U} \\
&\qquad\qquad i \in \{0, \dots, H-1\} \\
&\quad \bar{g}_{k+i+j+1|k+i} = \bar{g}_{k+i+j|k+i} \exp[\Delta t\, \bar{\lambda}(\bar{g}_{k+i+j|k+i}, \bar{\lambda}_{d_{k+i+j|k+i}})] \qquad (5.26b) \\
&\quad \bar{\lambda}_{d_{k+i+j|k+i}} \sim \mathcal{N}(\mu, \Sigma) \\
&\qquad\qquad j \in \{0, \dots, D-1\}
\end{aligned}$$

where the cost function is the Upper Confidence Bound for Trees (UCT) [34], [33]

$$\mathrm{UCT}(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}}) = \left(U_n(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}}) + C\sqrt{\frac{\log N(\bar{g}_{k+i|k})}{N(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})}} \right) \tag{5.27}$$

with $N(\bar{g}_{k+i|k})$ the number of times the state $\bar{g}_{k+i|k}$ has been visited, and $N(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})$ the number of times that the action $\bar{\lambda}_{d_{k+i|k}}$ has been chosen in the state $\bar{g}_{k+i|k}$, while $U_n(\bar{g}, \bar{\lambda}_d)$ is the state/action value function defined as

$$\begin{aligned} U_n(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}}) = & \frac{1}{N(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})} \\ & \sum_{r=1}^{N(\bar{g}_{k+i|k}, \bar{\lambda}_{d_{k+i|k}})} J_r(\bar{\boldsymbol{g}}_{k+i}, \bar{\boldsymbol{\lambda}}_{d_{k+i}}) \end{aligned} \tag{5.28}$$

where $J_r(\cdot, \cdot)$ will be discussed in more detail below. Therefore, selecting iteratively via UCT (3.16), the optimal control sequence state/action $(\bar{g}^*, \lambda_d^*)$ is computed along the tree. More in detail, the optimization problem is solved by MCTS in four phases.

Selection: Given the initial state $\bar{g}_{k|k} = \widehat{g}_k$, the selection phase exploits Upper Confidence Bound for Trees (UCT) [34], [33] to select the optimal action and cross the tree iteratively exploiting (5.26a) in order to reach the leaf node. Note that we use the notation $\bar{\lambda}(\bar{g}, \bar{\lambda}_d)$ meaning that $\bar{\lambda} \in \mathfrak{g}$ depends on the group element $\bar{g} \in G$ and on the reference value $\bar{\lambda}_d \in \mathbb{R}^4$ choosing during MCTS optimization.

The optimal/sub-optimal reference values are given by

$$\bar{\lambda}_d^* = \underset{\bar{\lambda}_d \in \mathcal{U}(\bar{g})}{\operatorname{argmin}} \quad \mathrm{UCT}(\bar{g}, \bar{\lambda}_d). \tag{5.29}$$

Expansion: If the leaf node has never been visited, the algorithm proceeds with the Simulation phase. Otherwise, it expands the search tree by adding one or more child nodes to the leaf node, and the selection phase is executed again. The action space $\mathcal{U}$ is modeled iteratively using the multivariate Gaussian distribution where $\mu \in \mathbb{R}^m$ is the mean vector of the reference values and $\Sigma \in \mathbb{R}^{m \times m}$ is the covariance matrix. The probability density function (PDF) of the multivariate Gaussian distribution $\bar{\lambda}_d \sim \mathcal{N}(\mu, \Sigma)$ is

$$f(\bar{\lambda}_d) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp\left(-\frac{1}{2}(\bar{\lambda}_d - \mu)^T \Sigma^{-1} (\bar{\lambda}_d - \mu) \right). \tag{5.30}$$

The mean vector and the covariance matrix of the multivariate distribution are updated using gradient-based rules [62, 69]

$$\begin{aligned} \mu' &= \mu + \eta(\bar{\lambda}_d^* - \mu) \\ \Sigma' &= \Sigma + \eta((\bar{\lambda}_d^* - \mu)(\bar{\lambda}_d^* - \mu)^T - \Sigma) \end{aligned} \tag{5.31}$$

where $\bar{\lambda}_d^*$ is the optimal reference value, μ' and Σ' are the updated mean and covariance matrix of the distribution, and η is the learning rate controlling how much the mean and covariance should shift toward the optimal action.

Simulation: The simulation, or rollout phase, starts from a state $\bar{g}_{k+i|k}$ and selects random actions $\lambda_{d_{k+i+j|k+i}} \sim \mathcal{N}(\mu, \Sigma)$, $j \in \{0, \ldots, D-1\}$, and compute the new state

thanks to the equation (5.26b). When the rollout process is over, the cumulative cost based on the obtained state/action trajectories

$$\begin{aligned} \bar{\boldsymbol{g}}_{k+i} &= (\bar{g}_{k+i+1|k+i}, \ldots, \bar{g}_{k+i+j|k+i}, \ldots, \bar{g}_{k+i+D|k+i}) \\ \bar{\boldsymbol{\lambda}}_{d_{k+i}} &= (\bar{\lambda}_{d_{k+i+j|k+i}}, \ldots, \bar{\lambda}_{d_{k+i+j|k+i}}, \ldots, \bar{\lambda}_{d_{k+i+D-1|k+i}}) \end{aligned} \tag{5.32}$$

is computed as

$$\begin{aligned} J(\bar{\boldsymbol{g}}_{k+i}, \bar{\boldsymbol{\lambda}}_{d_{k+i}}) = &\sum_{j=0}^{D-1} \gamma^j l(\bar{g}_{k+i+j|k+i}, \bar{\lambda}_{d_{k+i+j|k+i}}) \\ &+ \phi(\bar{g}_{k+i+D|k+i}) \end{aligned} \tag{5.33}$$

where $\gamma^j \in (0, 1]$, $j = 0, \ldots, D-1$, is a discount factor, $l(\cdot, \cdot)$ is the immediate cost defined in (5.25) and $\phi(\bar{g}_D)$ is the final cost.

Backpropagation: The backpropagation phase increases the node visit counts $N(\bar{g})$ and $N(\bar{g}, \bar{\lambda}_d)$ of each node from the leaf node to the root following the path chosen in the selection phase. Also, for each node, the state/action value function is updated by computing (5.28).

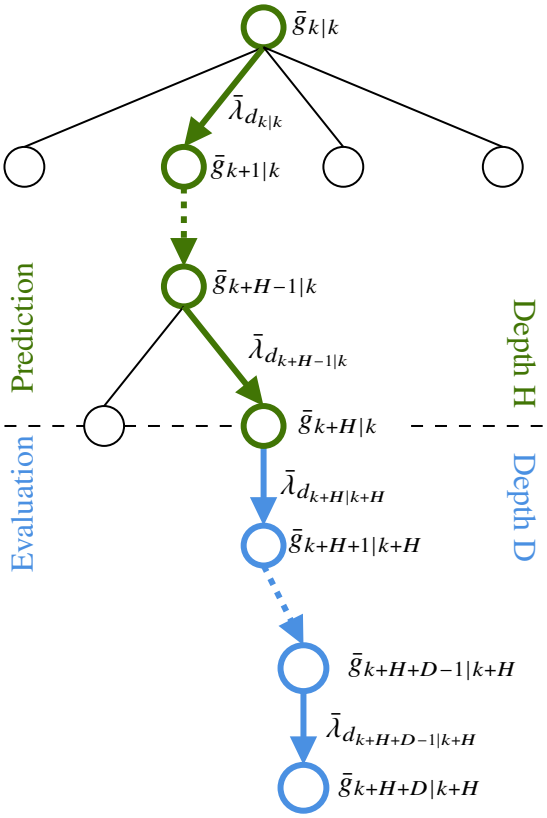


Fig. 5.2: Monte Carlo Tree Search horizons.

Figure 5.2 shows the evolution of the prediction and evaluation horizons of MCTS during the optimization process. The green path represents the prediction horizon with depth H, while the blue path shows the evaluation horizon that is composed of

random actions/state evolutions with depth D. The figure shows a snapshot of the MCTS algorithm where UCT iteratively selects the green path and the simulation phase starts from the node with state $\bar{g}_{k+H|k}$.

5.6 Simulation results

In this section, we present some simulations for the low-level and the MDP controllers. In both cases, the state is estimated by the second-order geometric filter presented in Section 5.3. The B and B_2 matrices related to the model error defined in (5.9) are

$$B = \begin{bmatrix} 0_{7\times 6} \\ B_2 \end{bmatrix}, \quad B_2 = \text{diag}(20, 20, 20, 1, 1, 1),$$

while the matrix D concerning to the measurement error in (5.11) is given by

$$D = \text{diag}(15, 15, 15, 0.1, 0.1, 0.1, 0.1, \\ 1, 1, 1, 0.1, 0.1, 0.1).$$

Finally, the matrices R and Q related to the cost functions (5.12)-(5.13) are set to

$$R = I_{6,6}, \quad Q = \text{diag}(100, 100, 100, 0.1, 0.1, 0.1, 0.1, \\ 0.1, 0.1, 0.1, 0.1, 0.1, 0.1).$$

The coefficients that appear in the dynamics are the following

$$m = 9300\text{Kg}, \quad g_r = 9.81\text{N},$$

$$c_1 = -0.77, \qquad c_2 = 0.0276, \qquad c_5 = 0.9604,$$
$$c_6 = 0.0176 \qquad c_8 = -0.7336,$$

while the parameters C_x, C_y, C_z depend on the angle of attack α and on the sideslip angle β, and their values are listed in specific lookup tables in [68].

5.6.1 Low level controller

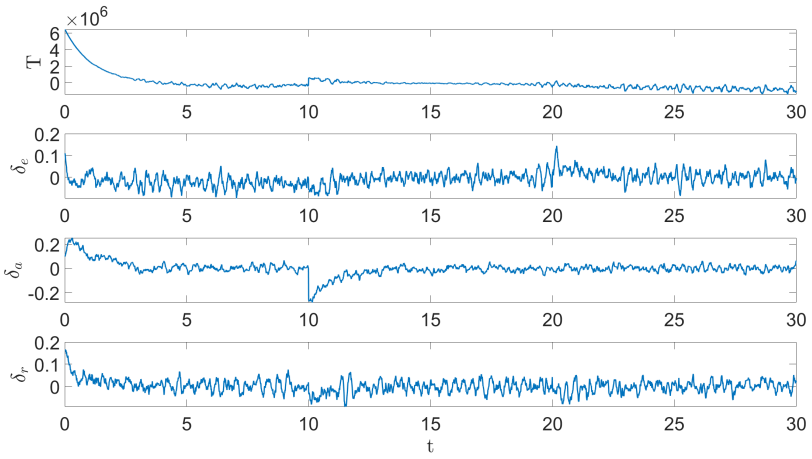
For the simulations aimed at proving the low-level control effectiveness, we set the following initial conditions

$$p(0) = \begin{bmatrix} 0 \\ 0 \\ 15000 \end{bmatrix}, q(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, v(0) = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix}, \omega(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

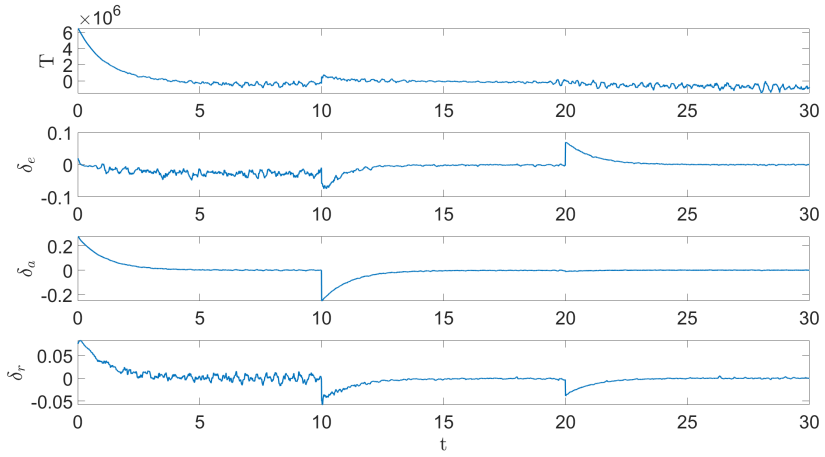
We consider a total time horizon of 30 seconds and desired velocities that change according to Table 5.1. Two different control laws are studied as introduced in Section 5.4.

	$V_T[m/s]$	$\omega_1[rad/s]$	$\omega_2[rad/s]$	$\omega_3[rad/s]$
0 - 10 [s]	450	0	0	0
10 - 20 [s]	500	0.02	-0.05	0.05
20 - 30 [s]	510	0.01	0.03	0.01

Table 5.1: Reference velocities.



(a) First controller with the error ($\widetilde{\lambda}_1$) defined on the Lie group.

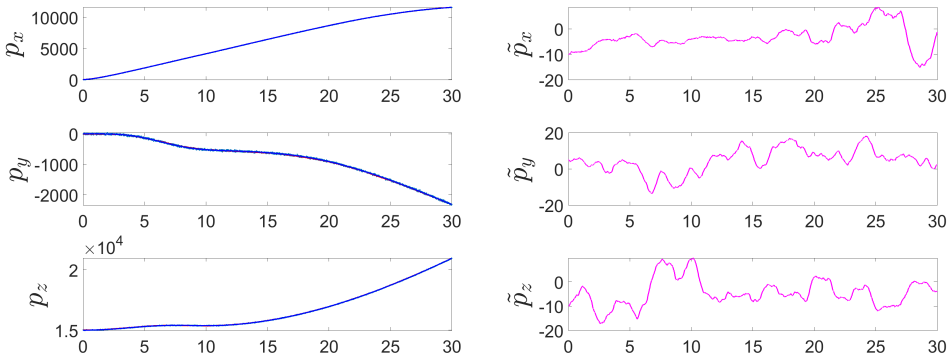


(b) First controller with the error ($\widetilde{\lambda}_2$) defined on the trivialized velocities.

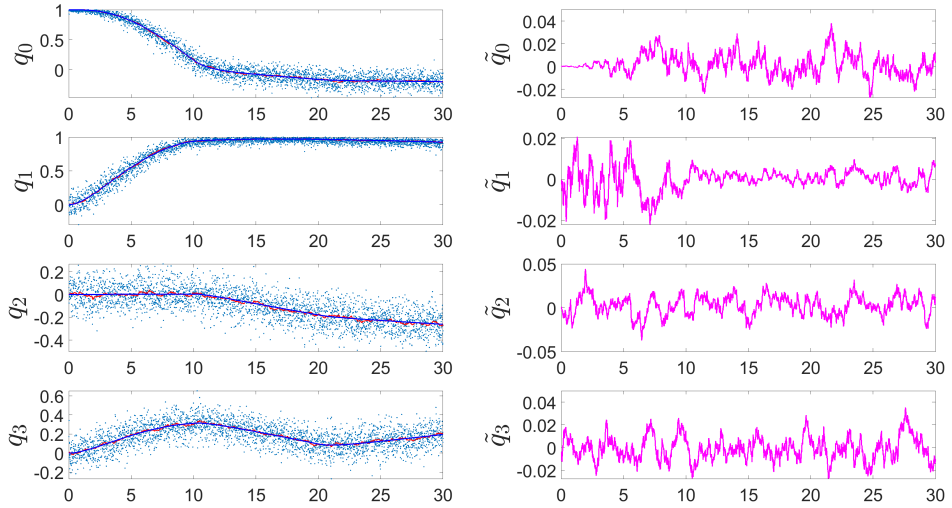
Fig. 5.3: Command inputs cases.

Figure 5.3a and Figure 5.3b show the controls u considering the estimated velocities provided by the choices of $\widetilde{\lambda}_1$ and $\widetilde{\lambda}_2$, in (5.16) and (5.17) respectively. As can be noticed from the time series, even if the reference velocities are the same, the commands that arise exploiting the trivialized velocities on the Lie algebra present a smoother behavior than using velocities on the tangent space. This fact is relevant from a control perspective, and, for this reason, we will prefer the control law with the estimated velocities $\widetilde{\lambda}_2$.

Figure 5.4 shows on the left the real, the measured, and the estimated 3D spatial position, quaternion, linear velocity, and angular velocity trajectories, while on the



(a) 3D spatial position.



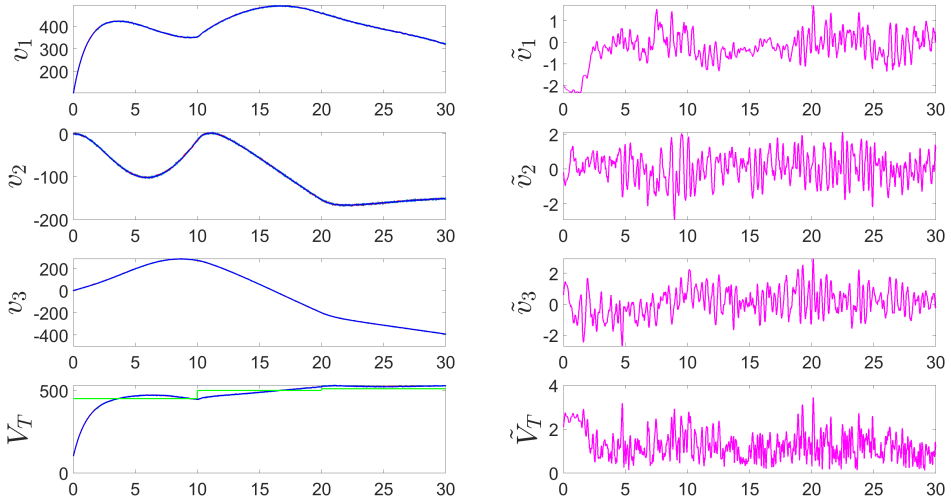
(b) Quaternions.

Fig. 5.4: Real (black), measured (blue), and estimated (red) trajectories (on the left), and their errors (on the right).

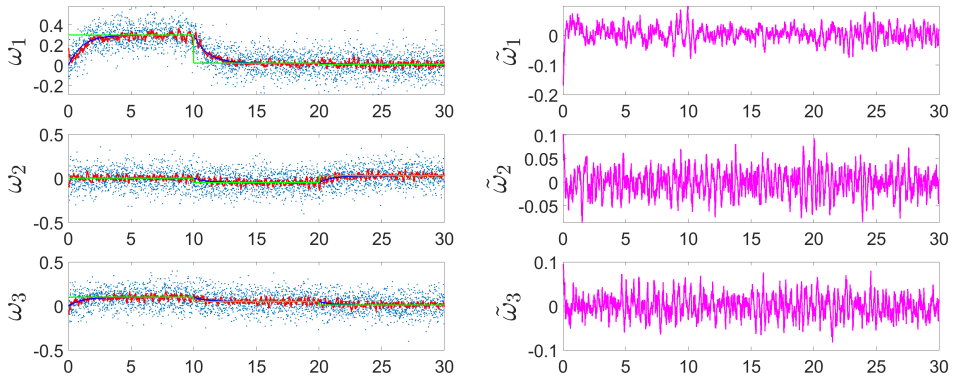
right, their corresponding errors Δ. The trajectories computed by the second-order geometric filter provide good estimations of the state vector. Even if the measurement errors and the velocities are considerable, the feedback system is able to control the aircraft following the given references. In the following subsection, we consider the total control law architecture also incorporating the MDP controller presented in Section 5.5.

5.6.2 MDP controller

We simulate two different scenarios where the aircraft has to reach a target with a defined orientation and avoid the no-fly zones. In the following results, only the trajectories on x and y are shown since the altitude of the aircraft does not have to change. Also, the estimated state and the low-level controls are not reported since they would not be useful in understanding the planner's behavior. Both simulations are executed with a control frequency of $100Hz$. Table 5.2 reports the parameters used in the simulations.



(a) Linear velocities.



(b) Angular velocities.

Fig. 5.5: Real (black), measured (blue), and estimated (red) trajectories (on the left), and their errors (on the right).

	Parameters
H	3
D	3
C	2
γ	0.1
μ	$\text{diag}(500, 0.0, 0.0, 0.0)$
Σ	$\text{diag}(100, 0.5, 0.5, 0.5)$

Table 5.2: Parameters used in the simulations

Figure 5.6 shows the path executed by the aircraft in the first scenario. The initial conditions used for these simulations are

$$p(0) = \begin{bmatrix} 0 \\ 0 \\ 15000 \end{bmatrix}, q(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, v(0) = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix}, \omega(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

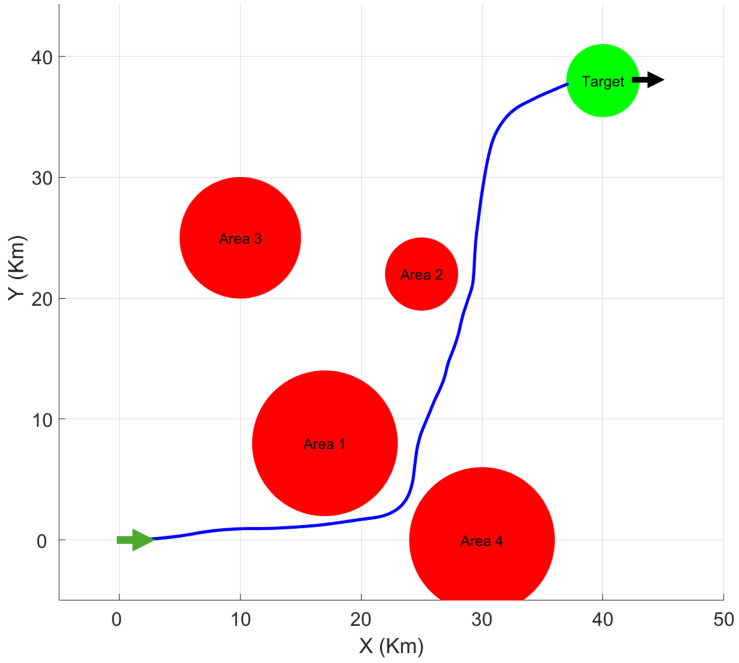


Fig. 5.6: First trajectory of the MDP planner. Green and black arrows are the initial and target orientations. Red circles are the no-fly zones. The green circle is the target.

The target position is $(40, 38)$ km, and the target orientation is $q(T) = [1, 0, 0, 0]^T$. Near "Area 4" it is possible to see how the internal simulation of MCTS allows the aircraft to avoid the no-fly zone. In this case, thanks to rollout simulations, the reference values that move the aircraft forward received a negative cost, thus bringing the aircraft to turn very fast to avoid "Area 4". Finally, it is interesting how the aircraft approached the target. Since the cost function minimizes the position and orientation errors, the first part of the trajectory is devoted to minimizing the position displacement (due to the high impact on the cost), while near the target, the refinement of the attitude is done.

In the second scenario, the initial conditions are

$$p(0) = \begin{bmatrix} 0 \\ 0 \\ 15000 \end{bmatrix}, \quad q(0) = \begin{bmatrix} \frac{\sqrt{2+\sqrt{2}}}{2} \\ 0 \\ 0 \\ \frac{\sqrt{2-\sqrt{2}}}{2} \end{bmatrix},$$
$$v(0) = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix}, \quad \omega(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Figure 5.7 shows the second scenario where the target position is $(38, 35)$ km, and the orientation is $q(T) = [\frac{\sqrt{2}}{2}, 0, 0, \frac{\sqrt{2}}{2}]$. In this scenario, the no-fly zones are smaller than the first one, so the planner computes the shorter path to reach the target, passing very close to the no-fly zones (see "Area 1") and reaching the target with the desired attitude.

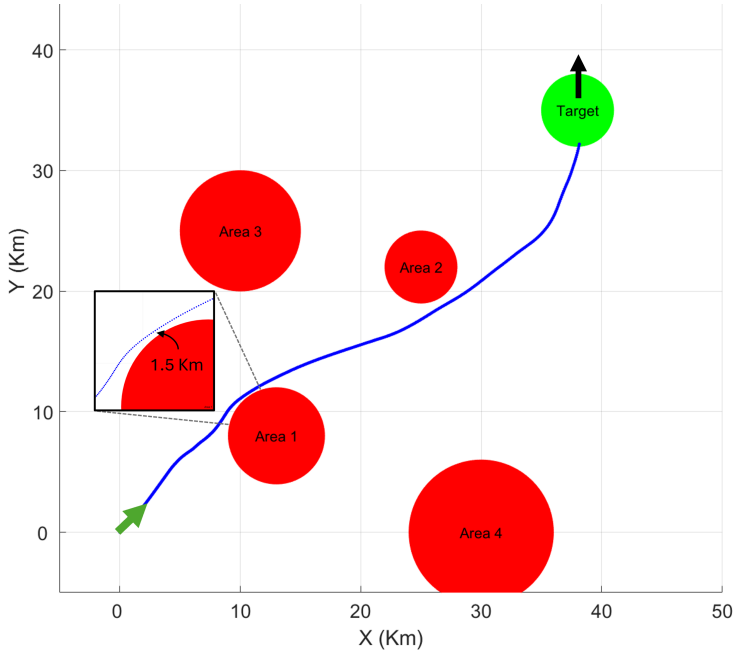


Fig. 5.7: Second trajectory of the MDP planner. Green and black arrows are the initial and target orientations. Red circles are the no-fly zones. The green circle is the target.

5.7 Discussion

This chapter presented a geometric approach to trajectory planning and control for aircraft dynamics by leveraging the structure of Lie groups, specifically the tangent bundle of the Special Euclidean group in $\mathbb{R}^3$, denoted as TSE(3). We retained the two-layer architecture introduced in the previous chapter, but fundamentally restructured the system's formulation and estimation strategy. The low-level controller was designed to track left-trivialized velocities on the Lie algebra, while the high-level MDP planner computed reference trajectories based on this representation using Monte Carlo Tree Search (MCTS). Unlike the previous chapter's use of a POMDP to explicitly model and manage measurement uncertainty, here state uncertainty was addressed through a minimum-energy observer, and the planner operated under the assumption of perfect state knowledge.

In contrast to the POMDP-based approach, which embedded uncertainty directly into the decision-making process, this chapter decouples estimation and planning. By estimating the state via a deterministic geometric observer and planning in a fully observable MDP, we reduce complexity while still capturing the nonlinear structure of the system dynamics. This approach capitalizes on the advantages of geometric mechanics for more accurate and efficient state tracking on Lie groups.

The key takeaway of this chapter is that exploiting the underlying geometric structure of the system—through Lie group formalism and velocity tracking on the Lie algebra—can lead to elegant and efficient solutions for trajectory planning and control. When combined with robust filtering techniques, this method provides an alternative to uncertainty-aware planning by separating estimation and control while preserving system fidelity.

Chapter 6

DECENTRALIZED UAV FORMATION PATH-PLANNING

"Not everything that can be counted counts, and not everything that counts can be counted."

Albert Einstein

Abstract[1] In this chapter, we extend the application of Monte Carlo Tree Search (MCTS) to the control of decentralized multi-agent systems. Specifically, we address the problem of autonomously guiding a fleet of Unmanned Aerial Vehicles (UAVs) toward target positions while avoiding no-fly zones and maintaining formation constraints. Building on the formal guarantees of MCTS for optimal and sub-optimal decision-making under uncertainty, we develop a decentralized planning framework for multi-agent coordination.

Each UAV is equipped with an online high-level path planner formulated as a Markov Decision Process (MDP), which computes optimal local reference trajectories. These references are then passed to a low-level controller based on the UAV's nonlinear dynamics. The MDP incorporates both environmental constraints (e.g., obstacle avoidance and target reachability) and formation requirements (e.g., maintaining relative positioning within the fleet).

To avoid collisions, UAVs exchange their predicted trajectories over a future time horizon. These predicted poses are treated probabilistically, using a time-weighted uncertainty model. Specifically, we apply an exponential probability distribution that assigns higher confidence to near-future states and reduces trust in long-term predictions—reflecting the fact that only the next immediate action will be executed, while future predictions remain speculative. The MDPs are solved online using MCTS, enabling each UAV to adapt its path in real time.

[1] The content of this chapter is based on:
Trotti, Francesco, Alessandro Farinelli, and Riccardo Muradore. "A Markov Decision Process Approach for Decentralized UAV Formation Path Planning." *2024 European Control Conference (ECC).* IEEE, 2024.

The main contributions of this chapter are:

- The development of a decentralized MDP-based path planning framework that leverages each UAV's nonlinear dynamics and accounts for both environmental and formation constraints.
- A distributed coordination strategy that relies on shared predicted trajectories, incorporating uncertainty modeling to ensure robust collision avoidance.

The chapter is organized as follows: in Section 6.1, we state the problem. In Section 6.2, we explain how the system operates, while in Section 6.3, we present the simulation results.

6.1 Problem statement

The fleet is composed of n UAVs that have to keep a desired formation geometry. The fleet has to reach the target area, avoiding some no-fly zones in a 3D scenario and maintaining the desired formation geometry. A nonlinear dynamic model is used to describe the behavior of the UAVs, and each aircraft has an inverse dynamics controller to track the provided reference values. For each UAV, an MDP is formalized to guarantee that

- the UAV maintains the formation geometry
- the UAV avoids the no-fly zones
- the UAV does not collide with other UAVs.

Given these constraints, the MDP has to provide the best reference values (v_d, h_d, ψ_d) for the UAV low-level controller that minimize the cost function, as shown in [48]. Some assumptions commonly used in the literature are needed.

Assumption 6.1 *The UAVs in the fleet have the same dynamic model.*

Assumption 6.2 *All UAVs know the geometry of the formation, and it is expressed with respect to the frame of the lead UAV.*

6.1.1 MDP Formalization

Our approach is based on discrete actions and states; therefore, we discretize the state space $\mathcal{S}$ and the action space $\mathcal{U}$. Since our approach is decentralized, we first present a generic MDP formalization, which is applied to each UAV.

States: The MDP state represents how the system evolves over time. It consists of the UAV state vector and future predictions of the other UAVs

$$s = [\boldsymbol{x}, \hat{\boldsymbol{h}}] \tag{6.1}$$

where $\boldsymbol{x}$ represents the UAV state vector (6.4), and $\hat{\boldsymbol{h}}$ contains the predictions of the future state vectors of the other UAVs; $\hat{\boldsymbol{h}}$ is defined as

$$\hat{\boldsymbol{h}} = \begin{bmatrix} \boldsymbol{x}^i_{t+1} \dots \boldsymbol{x}^i_{t+N} \\ \vdots \quad\quad \vdots \\ \boldsymbol{x}^n_{t+1} \dots \boldsymbol{x}^n_{t+N} \end{bmatrix} \tag{6.2}$$

where i ranges from 0 to n (number of UAVs in the fleet), $t = NT_s$ with T_s the sampling time, and N is a parameter that defines the length of the horizon.

Actions: The actions are the reference values for the UAV low-level controller (6.6)-(6.8)

$$a = [v_d, h_d, \psi_d]^T \tag{6.3}$$

where v_d, h_d and ψ_d are the desired velocity, altitude and yaw angle. Based on these reference values, the controller will handle thrust and the tack and bank angles. The action set is defined as a percentage increment or decrement of the reference value.

Transition model: The transition model defines the next state given the current state and an action. We use the aircraft dynamic model with the control loop explained below. The non-linear dynamic UAV model used in our formulation is taken from [91] [92], and [68]. The aircraft state vector is

$$\boldsymbol{x} = [x \;\; y \;\; h \;\; v \;\; \psi \;\;]^T. \tag{6.4}$$

where x, y, h are the position and altitude of the UAV, v is the UAV airspeed and ψ is the UAV yaw angle. The dynamic model is

$$\begin{aligned} \dot{x} &= v\cos(\psi)\cos(\gamma) \\ \dot{y} &= v\sin(\psi)\cos(\gamma) \\ \dot{h} &= v\sin(\gamma) \\ \dot{v} &= \frac{T-D}{m} - g\sin(\gamma) \\ \dot{\psi} &= \frac{g\tan(\phi)}{v\cos(\gamma)} \end{aligned} \tag{6.5}$$

where T is the thrust, D is the drag force and g is the gravity. The UAV is controlled using an inverse dynamics controller [80], [81] to control the thrust T and the roll and flight path angles ϕ and γ of the UAV in order to track the desired airspeed value v_d, the desired altitude h_d and the desired yaw angle ψ_d. Therefore, the thrust is

$$T = m(K_v(v_d - v) + g\sin\gamma) + D \tag{6.6}$$

where K_v is the positive speed control gain. The flight path angle is controlled to reach the desired altitude

$$\gamma = \arcsin\left(\frac{K_h(h_d - h)}{v}\right) \tag{6.7}$$

where K_h is a positive gain, while the roll angle is controlled to reach the desired yaw angle considering the flight path angle,

$$\phi = \arctan\left(\frac{K_\psi(\psi_d - \psi)v\cos(\gamma)}{g}\right) \tag{6.8}$$

where K_ψ is the positive gain and g is the gravity. The controllers modeled in the MDP are the discrete-time version of the continuous-time control laws (6.6)-(6.8).

By leveraging the UAV's dynamics and its low-level controller, we can simulate the system's evolution under different candidate reference inputs (actions) from the

current state, evaluating their outcomes using a cost function. To account for the uncertainty in the predicted trajectories of other UAVs during planning, we assign a time-dependent probability to each predicted pose using an exponential probability distribution applied to the prediction matrix $\hat{\boldsymbol{h}}$. This commonly adopted approach models the likelihood that a UAV will occupy a certain pose at a given time. The probability density function (PDF) of this exponential distribution is:

$$\mathbb{P}(\boldsymbol{x}^i|t_k) = \lambda e^{-\lambda t_k} \tag{6.9}$$

where $\boldsymbol{x}^i$ is the i-th predicted pose in the matrix $\hat{\boldsymbol{h}}$, and $t_k = k\Delta t$ is the discrete time associated with that prediction. The decay rate λ controls how quickly the probability decreases over time.

This probabilistic modeling reflects the inherent uncertainty in multi-step predictions: although the MCTS-based MDP only executes the immediate next action, it generates a full trajectory over the planning horizon. Since this trajectory may change at the next decision step, future predicted poses become increasingly speculative. The exponential distribution thus emphasizes confidence in near-term predictions while down-weighting those further in time, ensuring more reliable coordination among UAVs.

Cost function: The cost function evaluates the goodness of an action. In our approach, the cost function aims to minimize a combination of metrics, including reaching the target position, maintaining the correct position in the formation, and avoiding no-fly zones.

The component of the cost function responsible for reaching the target position is defined as the Euclidean distance between the UAV $X_{\text{UAV}} = (x, y, h)$ and the target position $X_T = (x_T, y_T, h_T)$

$$r_D = \|X_T - X_{\text{UAV}}\| . \tag{6.10}$$

The cost component that keeps the UAV in the right position within the formation is determined by the displacement between the desired pose (position and orientation) in the formation and the current UAV pose. The desired pose is expressed in the UAV frame placed in the point of symmetry of the formations (Σ_M), and T_M is the homogeneous matrix of the frame Σ_M. Therefore, a transformation matrix is needed to express the desired pose (Σ_d) in the current UAV frame (Σ_U). This transformation matrix is obtained by combining the position vector error and the yaw angle error ($\psi_e = \psi_M - \psi_U$) between the two poses. The transformation matrix is defined as

$$T_U^M = \left[\begin{array}{ccc|c} \cos(\psi_e) & -\sin(\psi_e) & 0 & (x_M - x_U) \\ \sin(\psi_e) & \cos(\psi_e) & 0 & (y_M - y_U) \\ 0 & 0 & 1 & (h_M - h_U) \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6.11}$$

where the upper left 3×3 matrix is the rotational matrix, and the vector 4×1 contains distance between the origins of Σ_M in Σ_U.

The desired formation pose expressed in the current UAV frame (Σ_U) is defined as $T_d = T_M T_U^M$, where T_d is the homogeneous matrix of the desired frame Σ_d. Consequently, the cost value is

$$r_F = (R_U - R_d) + \sum_{i=1}^{3} P_d^i \tag{6.12}$$

where P_d^i represents the i-th position value of the T_d matrix, while R_U and R_d are respectively the current UAV rotation and the desired rotation.

The function to avoid no-fly zones is modeled as a repulsive force [83]

$$r_{Z_i} = \begin{cases} \nu \left(\left(\frac{1}{d_{Z_i}} - \frac{1}{d_0} \right)^2 \right), & \text{if } d_{Z_i} < d_0 \\ 0, & \text{otherwise} \end{cases} \tag{6.13}$$

where, d_{Z_i} is the distance between the UAV and the i-th no-fly zone (Z is the no-fly zones set), d_0 is the repulsive threshold, and $\nu > 0$ is the repulsive coefficient.

Additionally, a repulsive force generated by the position of the other UAVs in the predicted poses matrix ($\hat{\boldsymbol{h}}$) is added to avoid collisions between the aircraft. The repulsive force is scaled by the probability of the state in the predicted poses matrix. In this way, the states with lower probability (states more in the future) have less impact on the total force compared to the states closer to the current time (higher probability). This case could happen when the other constraints (e.g., no-fly zones) force the aircraft to break the formation, and then they could collide with each other. Therefore, the UAV repulsive force is

$$r_{U_i} = \begin{cases} \nu^\sigma \left(\left(\frac{1}{d_{U_i}} - \frac{1}{d_0} \right)^2 \right), & \text{if } d_{U_i} < d_0 \\ 0, & \text{otherwise} \end{cases} \tag{6.14}$$

where d_{U_i} is the distance vector between the current UAV horizon and the i-th UAV in the predicted poses matrix, d_0 is calculated as the distance between the i-th UAV in the predicted poses matrix and the current formation position for the current UAV; while ν^σ is the repulsive coefficient scaled considering the probability (according to (6.9)) along the i-th state vector of the predicted matrix. Therefore, σ is a sample of the probability distributed function (PDF) given the state in the predicted poses matrix and the time. The reward value concerning the no-fly zones and the UAVs is

$$r_R = \sum_{i=1}^{Z} r_{Z_i} + \sum_{i=1}^{n} r_{U_i} \tag{6.15}$$

The overall cost value is the sum of all these components:

$$r = -r_D - r_F - r_R \tag{6.16}$$

6.2 Formation planning

In this section, we present the evolution of the approach over time. Specifically, the MDP computes the best reference values for the UAV low-level controllers to minimize the cost function considering various constraints. At each time step, each MDP provides the best action to the low-level controller and, after, shares its predicted poses of N steps to the other UAVs.

6.2.1 Monte Carlo Tree Search

The MCTS algorithm follows the formalization used in the previous Chapters. Therefore, we briefly recall the four phases, highlighting only the contributions of this chapter.

Selection: The selection is made using the Upper Confidence Bounds applied to Trees (UCT), which is

$$a^* = \text{argmax}_{a \in \mathcal{U}(s)} \left(Q(s,a) + C \sqrt{\frac{\log N(s)}{N(s,a)}} \right) \tag{6.17}$$

where $\mathcal{U}(s)$ is the action set, $Q(s,a)$ is the value obtained by executing action a in state s, C is the exploration-exploitation constant, $N(s)$ is the number of visits to the node with state s, and $N(s,a)$ is the number of visits to the child node of the node with state s when applying action a.

Expansion phase: From the leaf node, several new nodes (states) equal to the size of the discretized action space $\mathcal{U}$ are created. In this case, the action space is not represented via a multivariate distribution but as a uniform distribution of discretized actions.

Simulation phase: The simulation phase is based on the use of the transition model (6.4)-(6.9). From the selected leaf node, the rollout method starts trying different random actions from the current state until a specified maximum depth (D) is reached. The cumulative reward value for the rollout is computed utilizing the cost function and discount factor

$$R = \sum_{i}^{H} \gamma^i r_i \tag{6.18}$$

where r_i represents the partial reward value obtained at each step during the rollout, and D stands for the rollout horizon (maximum depth) and γ is the discount factor.

Back-propagation phase: The back-propagation phase updates the parameters of the nodes as explained in 3.

6.2.2 System evolution

These four phases of the MCTS are executed in a loop until the time constraints are met. When this occurs, the action with the best $Q(s,a)$ at time t_{k+1} is chosen and then it is applied to the low-level UAV controller. Following, the execution of this action, the new state vector of the UAV becomes the new root for the next iteration of the MCTS algorithm. To guarantee the correct length of the vector of predicted poses, the number of the MCTS simulations has to be chosen in order to create a tree with a depth greater than or equal to N. In particular, the vector of predicted poses is formed by applying the UCT strategy to the MCTS tree N times to extract the best estimated states from the current state. It's important to note that only the t_{k+1} action is used in the UAV low-level controller, and the predicted poses are used solely to share a likely trajectory of the other UAVs.

By leveraging this concept, the system remains resilient to communication issues between UAVs when disturbances last for a duration shorter than the vector of

predicted pose length. This is because each UAV knows the actions that other UAVs will take in the next N steps.

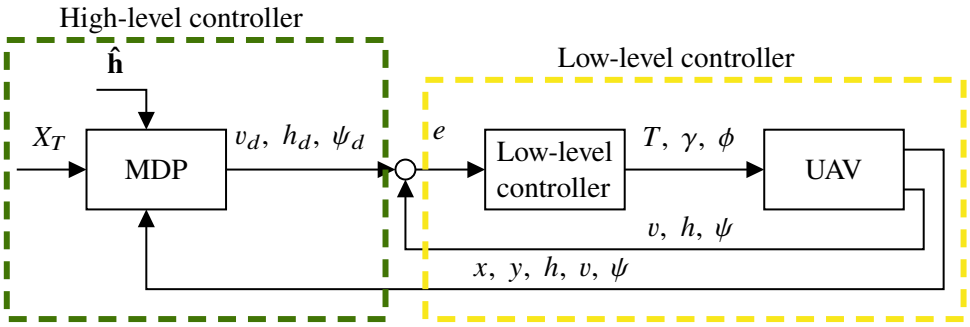


Fig. 6.1: UAV control schema

Each UAV independently solves its own MCTS algorithm, generating the best local reference values for its low-level controller while considering the other UAVs. Figure 6.1 illustrates the control architecture for each UAV ($\hat{\boldsymbol{h}}^i$ represents the vector of predicted poses of the i-th UAV). Consequently, the fleet evolves in a decentralized manner over time, and after the MCTS iterations, each UAV shares its vector of predicted poses with the other UAVs.

6.3 Simulations

The proposed technique has been tested in two different simulated scenarios. In both cases, the fleet must reach a desired area while avoiding no-fly zones and attempting to maintain formation whenever possible. In the first scenario, various no-fly zones are taken into consideration, whereas in the second scenario, a no-fly zone barrier with a breach is considered.

	Scenario 1	Scenario 2
# Sim	600	1000
H	2	4
ν	100	100
k	3	3
n UAV	3	3

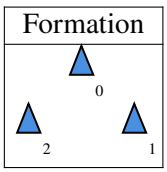


Table 6.1: Parameters and formation

Table 6.1 lists the most important hyper-parameters used during the simulations and also shows the formation geometry. The symmetry point of the formation is located at UAV0. As a result, all the other formation points are expressed relative to the reference frame of the UAV0. In both scenarios, the initial conditions for UAV0 are set to $x_0 = [50, 50, 1.5, 200, 0.78]$, while for the other UAVs, the initial conditions are the same, with their poses adjusted to fit the correct formation position. In the experiments, the altitude is not shown since the target altitude is equal to the

initial condition, and the UAVs maintain the altitude constant during the simulation. The proposed results are tested on the fleet of three UAVs, but the approach is easily extendable to a fleet with more agents since it is local for each UAV, and the complexity is not directly related to the number of agents.

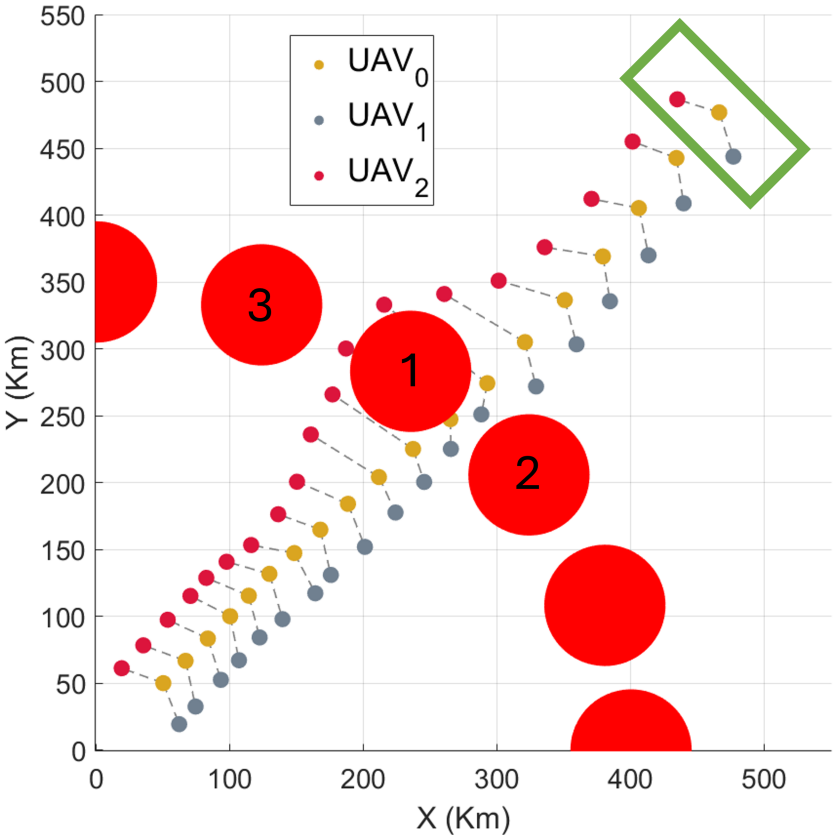
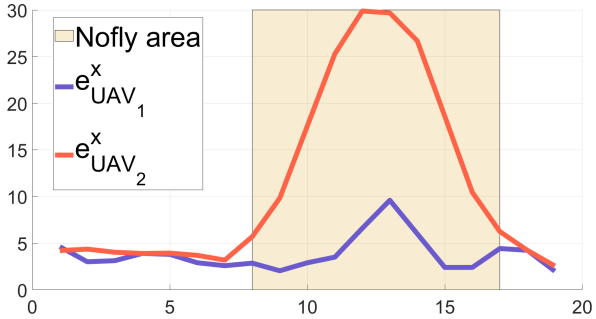
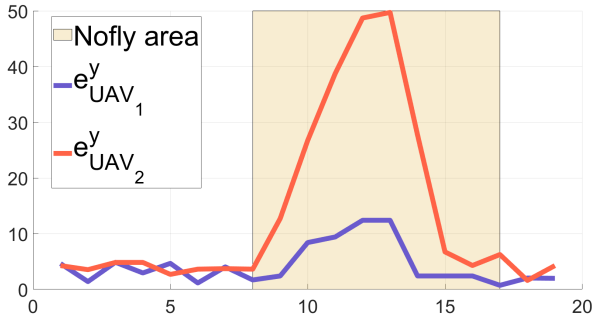


Fig. 6.2: Fleet paths. The green rectangle is the target area, and the red circles are the no-fly zones

In the first scenario, the fleet has to reach the target area, avoiding some no-fly zones. Figure 6.2 shows the paths generated by the proposed methodology. It is possible to see that the UAVs maintain the formation until the no-fly zones are reached. The UAV0 avoids the no-fly 1 zone by passing to the right. Since UAV0 defines the formation geometry, the desired formation position for UAV1 also shifts to the right. However, on the right side of UAV1, another no-fly zone is present (number 2). Consequently, the optimal local action for UAV1 is to move closer to UAV0. This adjustment occurs because, in the cost function of UAV1, the repulsive no-fly zone component becomes more significant than the formation constraint. As a result, UAV1 moves closer to UAV0. On the other hand, in the presence of the no-fly zone 1, the UAV2 changes completely the path. In particular, if the UAV2 had passed to the right side of the no-fly zone 1, a penalty regarding the wrong position in the formation would have been taken into account, and also, a highly repulsive component due to the closeness to the UAV0 and UAV1 would have been added. While passing on the left side of the no-fly zone 1, the UAV2 obtains only a penalty



(a) UAV1 and UAV2 absolute error along x axis.



(b) UAV1 and UAV2 absolute error along y axis.

Fig. 6.3: Error along x and y axis compared to the desired position in the formation

for breaking the formation and avoids possible collision with the other UAVs. These considerations are also highlighted in Figure 6.3, where the absolute error along the x (Figure 6.3a) and y (Figure 6.3b) axes are shown for each UAV. It is possible to see the error increases near the no-fly zone (yellow area in the plots), while in the other parts, the error is small.

6.3.1 Second scenario: Breach Crossing

In the second case, the fleet has to cross different breaches and maintain the formation. Figure 6.4 shows the trajectories executed by the UAVs with a zoom on the most important part of the path. In this case, all UAVs maintain the formation throughout the path. By exploiting the greater number of simulations and the rollout depth, future evolutions of each UAV are analyzed. Since the no-fly zones are modeled as a barrier, the simulations of all UAVs are quite similar except for the displaced formation position. Figure 6.5 shows the absolute error along the x (Figure 6.5a)

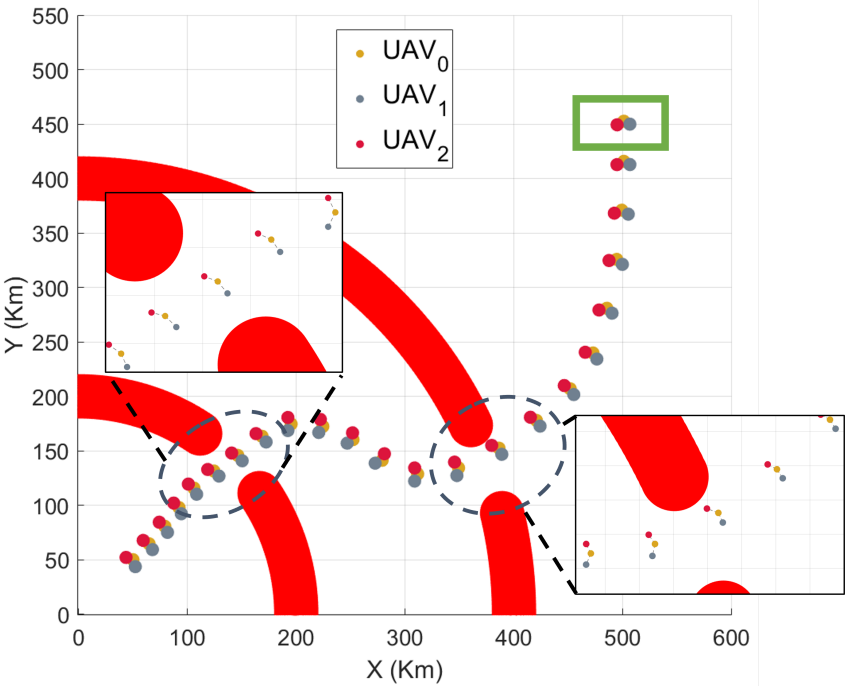
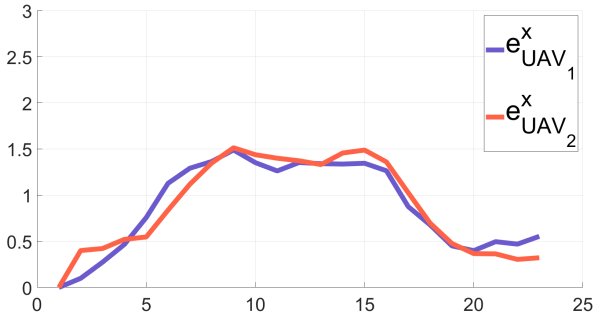
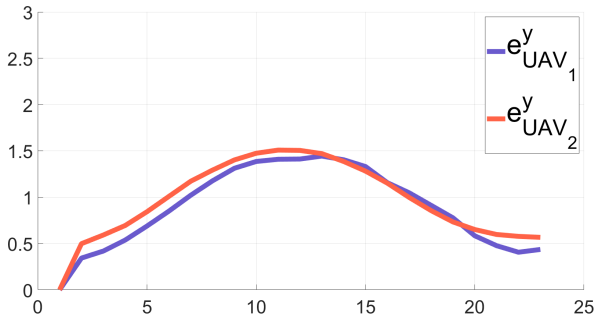


Fig. 6.4: Fleet paths. The green rectangle is the target area, and the no-fly zones are in red



(a) UAV1 and UAV2 absolute error along x axis.



(b) UAV1 and UAV2 absolute error along y axis.

Fig. 6.5: Error along x and y axis compared to the desired position in the formation

and y (Figure 6.5b) axes for each UAV; the error is small but increases a little in the area between the two no-fly zones. This behavior is due to the target attractive component, leading to a negative reward as the UAVs move away from the target.

However, the formation of attractive components and the high rollout depth (the future is analyzed more in-depth) allow us to compensate for the target attractive penalty by finding the best trajectories.

6.4 Discussion

This chapter presented a decentralized trajectory planning and control strategy for a fleet of UAVs using Monte Carlo Tree Search (MCTS) within a Markov Decision Process (MDP) framework. Each UAV autonomously computes its path by solving an MDP online, accounting for nonlinear dynamics, environmental obstacles, and formation flight requirements. A key aspect of the approach is the distributed coordination mechanism based on shared predicted trajectories, modeled probabilistically to address the uncertainty inherent in multi-agent prediction.

Compared to centralized methods or simpler coordination strategies, this framework offers greater scalability and adaptability, particularly in dynamic environments where decentralized, real-time decisions are essential. The integration of time-weighted uncertainty modeling further enhances the system's robustness by ensuring safe coordination despite imperfect trajectory forecasts.

Chapter 7

HIGH-LEVEL PLANNING AND RE-PLANNING

"I do not fear computers. I fear the lack of them."

Isaac Asimov

Abstract[1] In this chapter, we explore the use of Monte Carlo Tree Search (MCTS) for high-level decision-making in mobile robot navigation within a warehouse environment. Building on prior work in planning and control for aircraft systems, we focus here on abstract planning, assuming a reliable and known low-level controller. We formulate the re-planning problem as a Markov Decision Process (MDP), incorporating a stochastic model for obstacle lifespan directly into the transition function. The goal is to locally re-plan the path of only the affected robot, minimizing disruption to the rest of the fleet and avoiding the need for global or partial re-planning. Starting from a path computed by a Multi-Agent Path Finding (MAPF) algorithm—specifically Conflict-Based Search (CBS)—we introduce two MCTS-based re-planning strategies: 1) An MDP approach that iteratively chooses the next action based on minimizing the distance to the target while sampling from the obstacle lifespan distribution. 2) A bandit-style MDP that selects only the initial move and then switches to Space-Time A* from the new position, also accounting for sampled obstacle lifespans.

For comparison, we also implement a baseline MAPF re-planning strategy that re-invokes CBS at each collision. Through simulations using a standard warehouse scenario and real-world experiments, we show that incorporating the stochastic model of obstacle persistence leads to more informed local decisions and improves overall solution quality, reducing both travel time and makespan.

[1] This Chapter is based on Trotti, Francesco, Alessandro Farinelli, and Riccardo Muradore. Path Re-Planning with Stochastic Obstacle Modeling: A Monte Carlo Tree Search Approach, *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024)*

The main outcomes of this chapter are:

- A novel formalization of local re-planning as an MDP using a stochastic obstacle lifespan model.
- The development of two MCTS-based strategies for efficient, localized path re-planning.
- A comprehensive evaluation comparing our methods against traditional MAPF re-planning, using both quantitative and qualitative results.

The chapter is organized as follows: in Section 7.1, we state the problem, and in Section 7.2, we describe the proposed algorithms. In Section 7.3, we compare the proposed algorithms on a standard warehouse scenario [93] and in a real scenario.

7.1 Problem statement

Our local re-planning approach is tailored to scenarios like warehouses, where we define various obstacles: static (e.g., walls or stocked shelves), dynamic, and temporal. Dynamic obstacles include time information regarding their path (e.g., other UGVs following a known trajectory), while temporal obstacles are static obstacles with stochastic lifespans (e.g., a human operator performing a task on a machine). Specifically, given a scenario (a grid or a graph), a set of agents (M), and an optimal collision-free plan for all agents (π), we introduce temporal obstacles at crucial points on the map to prompt the re-planning of specific robot paths. We utilize a Conflict-Based search (CBS) algorithm to find an initial optimal plan for all robots, given a set of initial and goal positions and a random assignment. The re-planning strategy is formalized as a Markov Decision Process (MDP) that considers the stochastic lifespan of encountered obstacles in the transition function. Therefore, the computation of the new path depends on the paths of other robots (considered dynamic obstacles with defined trajectories) and the probability distribution of the lifespan of temporal obstacles. The optimal new path might involve waiting until the obstacle disappears, moving away to clear passage for other robots, or calculating a new collision-free path. The following assumptions commonly used in the literature are needed:

Assumption 7.1 *When a robot re-plans, it shares its path with other robots*

Assumption 7.2 *The obstacle lifespan is independent of observation time*

Assumption 7.3 *Time is fully discretized, and the space is discretized as a grid*

7.1.1 Stochastic Lifespan Obstacle Model

We model the lifespan of any obstacle using the probability density function (PDF) of the gamma distribution

$$\mathbb{P}(X) = f(X, \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} X^{\alpha-1} e^{-\beta X} \tag{7.1}$$

where X represents the believed obstacle lifespan, α is the shape parameter of the gamma distribution, β is the rate parameter of the gamma distribution, and

$\Gamma(\alpha) = (n-1)!$. It is important to note that the gamma distribution is defined for $X \geq 0$, which makes sense as we are modeling the lifespan of an obstacle. The relations between the parameters α and β and the mean (μ) and variance (σ^2) are

$$\alpha = (\frac{\mu}{\sigma})^2, \quad \mu = \frac{\alpha}{\beta}$$
$$\beta = \frac{\mu}{\sigma^2}, \quad \sigma^2 = \frac{\alpha}{\beta^2}.$$

Update rules for the gamma distribution

By leveraging Bayesian inference, it is possible to update the prior distribution $\mathbb{P}(X)$, which follows a gamma distribution with parameters α and β, using the likelihood of observed data to obtain a posterior distribution. We propose two approaches to update the gamma distribution (deterministic and stochastic).

Deterministic update. The deterministic update is used when the robot can directly observe the obstacle at time t. In this case, the update is straightforward and is based on updating the α and β values considering the time t of the observation. Therefore, the new values are

$$\begin{cases} \alpha' = \alpha \\ \beta' = \beta + \frac{t}{X}. \end{cases} \tag{7.2}$$

The shape of the gamma distribution (α) remains the same as before, while the rate parameter (β) is increased by the ratio between the observation time t and the believed obstacle lifespan X.

Stochastic update. The stochastic update is based on Bayes' theorem: given a prior distribution and a probability distribution to observe an event, their combination generates a posterior distribution. We use the gamma function as the prior distribution, and we model the probability that the obstacle is still present at time t, even if the robot does not directly observe it, as an exponential distribution

$$\mathbb{P}(t|X) = \frac{1}{X} e^{-\frac{1}{X}t} \tag{7.3}$$

where t is the observed time. In this way, a shorter obstacle lifespan (X) results in higher event rates, while longer obstacle lifetimes lead to lower event rates. Applying Bayes theorem, we end up with

$$\begin{aligned} \mathbb{P}(X|t) &\propto \mathbb{P}(X)\mathbb{P}(t|X) \\ &\propto \left(\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}\right)(\lambda e^{-\lambda t}) \\ &\propto \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{1}{X} x^{\alpha-1} e^{-\beta x - \frac{t}{X}} \end{aligned} \tag{7.4}$$

To obtain the updated values of α and β, we normalize by integrating over the entire range. This allows us to reshape the gamma distribution based on the previous observations

$$\begin{cases} \alpha' = \alpha + 1 \\ \beta' = \beta + \frac{t}{X}. \end{cases} \tag{7.5}$$

The shape parameter (α) is incremented by one to take into account the newly observed event, while the rate parameter (β) is incremented by the ratio between the observation time t and the believed obstacle lifespan X.

7.2 Re-Planning algorithms

In this section, we describe the three proposed algorithms. Firstly, we describe our benchmarking baseline, which is a re-planning algorithm based on CBS, where the obstacle lifespan is sampled from the gamma distribution each time an obstacle is encountered. Secondly, the two MDP-based algorithms are formalized. The first provides an action to the robot at each iteration, while the second generates a collision-free path.

7.2.1 CBS Re-plan

The CBS Re-plan algorithm is a popular re-planning approach. This planner invokes the CBS algorithm each time an obstacle is encountered, estimating its lifespan by sampling the gamma distribution. However, it becomes computationally inefficient with many agents, requiring recalculation of the paths of all robots for each encountered obstacle. Additionally, if the estimated obstacle lifespan matches the real lifespan, the algorithm provides an optimal solution, and the results are complete as CBS. Therefore, the accuracy of this algorithm is closely related to the obstacle lifespan estimation. The algorithm updates the gamma distribution using the deterministic method each time a robot encounters an obstacle.

7.2.2 MCTS planner

This algorithm is based on the formalization of the problem as an MDP, which is solved using MCTS. The policy generated by the MDP provides an action to move the robot to a new collision-free position. The MDP is formalized as follows.

States: The state consists of the robot position (p), the obstacle position (o), and the plans of other robots (π)

$$s = [p, o, \pi]. \tag{7.6}$$

Actions: The actions are the possible movements of the robot in a Cartesian-like grid

$$a = [\text{up},\ \text{down},\ \text{left},\ \text{right}]. \tag{7.7}$$

Transition model: The transition model is deterministic for the robot movements and stochastic for the obstacle lifespan. Given an action and a current state, the robot is certain to move in the next state, while the state of the obstacle is sampled from the gamma distribution illustrated in Subsection 7.1.1.

Cost function: The cost function is designed to minimize the distance to the goal while penalizing incorrect movements that result in collisions with the obstacles. The reward function is defined as

$$\begin{aligned} R &= ||p - p_g|| + k \\ k &= \begin{cases} 0, & \text{if } p \neq o \\ -\infty, & \text{if } p = o \end{cases} \end{aligned} \tag{7.8}$$

where p and p_g represent the robot actual position and the goal position, respectively, and o represents the position of static, dynamic, or temporal obstacles.

Sampling from the gamma distribution at each MCTS simulation allows the algorithm to evaluate different actions, considering specific obstacle lifespans in each simulation. This strategy simulates and evaluates numerous obstacle lifespan configurations to determine the best local action. After the robot movement in the environment, the gamma distribution is updated in a stochastic manner since the robot is not always close to the obstacle. Therefore, we consider the probability that the obstacle at time t is in place to update the gamma distribution. This process continues by calling MCTS again until the robot reaches the goal.

7.2.3 MCTS heuristic

This algorithm is based on the Bandit MDP, where the simulation phase is limited to one iteration. In this case, the policy generated by the MDP is a completely new collision-free path. The MDP is formalized as follows.

States and **actions** are defined as in Section 7.2.2.

Transition model: The transition model is deterministic for the robot movements and stochastic for the obstacle lifespan. Given an action and a current state, the robot is certain to move to the next state, and from that, it invokes Space-time A^* to find a path. The state of the obstacle is sampled from the gamma distribution to define the believed lifespan used in Space-time A^* for the planning.

Cost function: The reward function is designed to minimize the travel time

$$R = \begin{cases} \sum_{i=1}^{n-1} w_i, & \text{if } n \neq 0 \\ -\infty, & \text{if } n = 0 \end{cases} \tag{7.9}$$

where w_i represents the i-th waypoint, w is the list of waypoints, and n is the total number of waypoints. Also, in this case, the algorithm samples an obstacle lifespan during each simulation and executes the simulation with this fixed value. Differently from the previous algorithm, the policy is the entire optimal path, not just the actual action to move the robot. This approach evaluates the paths generated by Space-time A^* from different starting points and considers different possible obstacle lifespans. If the robot reencounters the obstacle during the path, a deterministic update of the gamma distribution is made, and the algorithm restarts.

7.2.4 Computational Cost

The proposed strategies operate locally, and Assumption 1 is necessary to ensure the correctness and the computation of a new collision-free path. However, this assumption can be easily relaxed in the context of a warehouse or, more broadly, in an Industry 4.0 production line, where network and information sharing between machines is a common feature. Considering a standard publisher/subscriber network infrastructure, the network traffic is minimal, given the capabilities of state-of-the-art networks. In practical terms, assuming a simple message composed of the robot ID (4 bytes) and the new paths ($4n$ bytes, with n the length of the path), the maximum traffic on the network caused by one re-planning robot is determined by the product of the bytes of the package and the number of agents. It is important to note that the broadcasting of the message occurs only when a robot re-plans its

path, resulting in very low frequency under normal conditions. Assumption 2 is also relaxed in our context; the algorithms operate on fully discretized time and space since additional information is unnecessary for planning. A local planner with a low-level controller [48] manages the robot motion through the waypoints generated by the proposed algorithms.

7.3 Simulations

The proposed approach has been tested and validated in two experiments: 1) in a simulated warehouse scenario (a 14×14 grid with 4 agents) [93], and 2) in a research facility with an Industry 4.0 production line using two RB-Kairos mobile robots. The metrics used to evaluate our approach are the total travel time and the total makespan, which are standard metrics for evaluating planning algorithms. The *total travel time* of an agent is the number of time steps it takes for the agent to move from its initial position to its goal position. For the i-th agent is equal to

$$T_i = \sum_{t=1}^{T} \mathbf{1}_{\{p_i(t) \neq g_i\}} \tag{7.10}$$

where

- $p_i(t)$ is the position of agent i at time t,
- g_i is the goal position of agent i,
- $\mathbf{1}_{\{p_i(t) \neq g_i\}}$ is an indicator function that equals 1 if $p_i(t) \neq g_i$ and 0 otherwise,
- T is the time horizon or the time step when all agents reach their goals.

The total travel time for all N agents is

$$\text{Total Travel Time} = \sum_{i=1}^{N} T_i = \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbf{1}_{\{p_i(t) \neq g_i\}}. \tag{7.11}$$

The *makespan* is the total time taken for all agents to reach their respective goal positions. It is defined as the maximum travel time across all agents

$$\text{Makespan} = \max_{i=1,\ldots,N} T_i = \max_{i=1,\ldots,N} \sum_{t=1}^{T} \mathbf{1}_{\{p_i(t) \neq g_i\}}. \tag{7.12}$$

7.3.1 Scenario 14×14 with 4 agents

Obstacles lifespan are computed with a constant shape ($\alpha = 5$ is a reasonable value for the experiments) and with different scale values (β), as shown in Figure 7.1. Each simulated experiment is executed 100 times on a 14×14 grid scenario with 4 agents, and the MCTS simulations are fixed at 300 iterations. The real obstacle lifespan remained constant for all runs, while the believed lifespan was sampled from the gamma distribution at each run. The results are obtained using a computer equipped

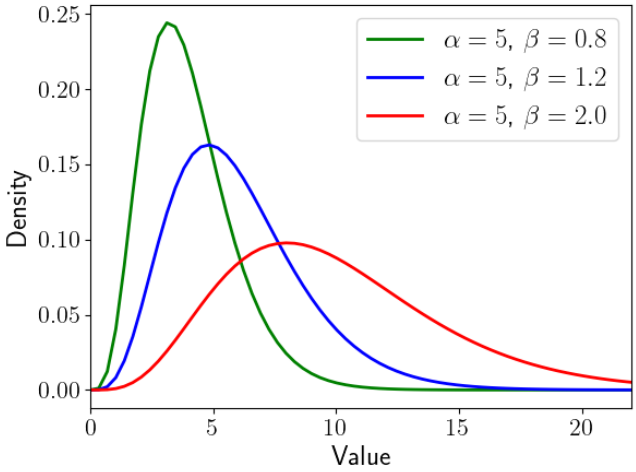


Fig. 7.1: Gamma distribution for α and β values

with standard hardware[2]. In this scenario, an obstacle is placed to block a robot path that consequently obstructs the paths of the other robots.

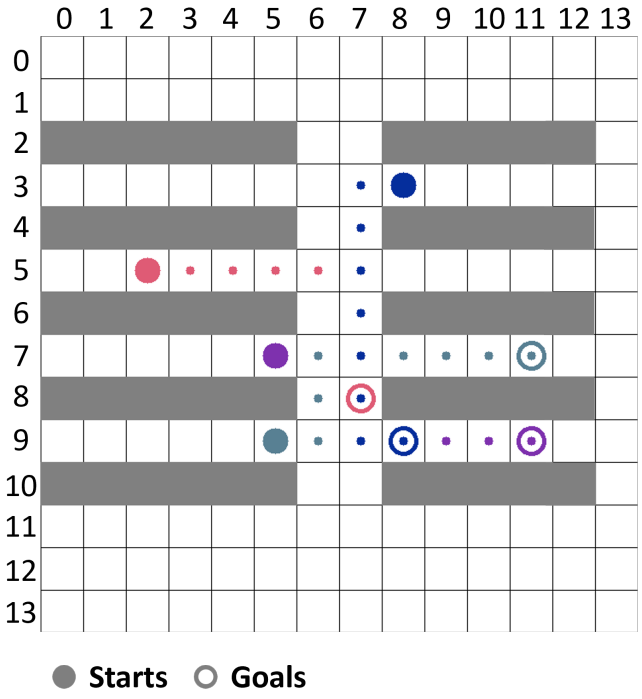
Figure 7.2a shows the scenario without obstacles and the paths generated by the first CBS run. In this case, the total travel time is 36, and the makespan is 9. A snapshot of the scenario when the gray agent encounters the obstacle (at time 5) is shown in Figure 7.2b. The obstacle obstructs the gray agent path, which consequently blocks the path of the blue and red agents.

	CBS Re-plan
●	[(6, 7)(7, 7)(8, 7)(9, 7)(9, 8)]
●	[(7, 7)(7, 6)(6, 6)(5, 6)(5, 7)(5, 8)(5, 9)(5, 10) (5, 11)(5, 12)(5, 13)(6, 13)(7, 13)(7, 12)(7, 11)]
●	[(5, 6)(5, 7)(6, 7)(7, 7)(8, 7)]
●	[(9, 7)(9, 8)(9, 9)(9, 10)(9, 11)]
	MCTS Planner
●	[(6, 7)(7, 7)(8, 7)(9, 7)(9, 8)]
●	[(7, 7)(7, 6), (7, 7), (7, 8), (7, 9), (7, 10), (7, 11)]
●	[(5, 6)(5, 7)(6, 7)(7, 7)(8, 7)]
●	[(9, 7)(9, 8)(9, 9)(9, 10)(9, 11)]
	MCTS Heuristic
●	[(6, 7)(7, 7)(8, 7)(9, 7)(9, 8)]
●	[(7, 7)(7, 6)(7, 7)(7, 8)(7, 9)(7, 10)(7, 11)]
●	[(5, 6)(5, 7)(6, 7)(7, 7)(8, 7)]
●	[(9, 7)(9, 8)(9, 9)(9, 10)(9, 11)]

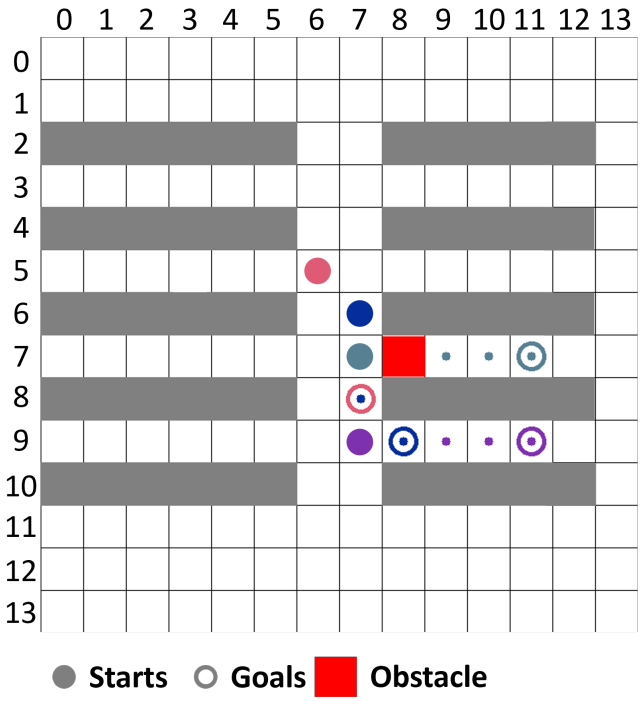
Table 7.1: Re-planned paths in the 14×14 scenario (the pair (x, y) are the coordinate in the grid)

Table 7.1 lists the re-planned paths using the three different algorithms discussed in Section 7.2. In this case, we use $\alpha = 5$ and $\beta = 1.2$ (realistic parameters), with the real obstacle lifespan set to 2 (the obstacle disappears at time 2, so the cell becomes

[2] Intel Core i7-9750H processor and 32GB of RAM.



(a) Scenario without obstacle with the CBS paths.



(b) Scenario with the obstacle at time 5.

Fig. 7.2: 14 × 14 scenario with 4 agents and 1 obstacle

free at time 3). The paths are shown starting from the snapshot at time 5 (since the previous parts of the paths are identical for all algorithms). The CBS Re-plan algorithm completely re-plans the agent paths to avoid the obstacle but increases the total travel time (44) and the makespan (17). The other two algorithms (MCTS planner and MCTS heuristic), based on the MDP and incorporating the stochastic lifespan model, estimate the optimal re-planning strategy more effectively. In this case, only the path of the light blue agent is re-planned because the MDP is designed to replan only the agent that encounters the obstacle. These two algorithms provide the same paths. Precisely, the light blue agent moves back to allow the passage of the other robots and then returns to its previous position towards the goal. This behavior is generated by leveraging multiple simulations with different samples from the gamma distribution, enabling the algorithms to provide the best new paths. These two algorithms plan paths with a total travel time of 38 and a makespan of 11. This case serves as an example to emphasize the differences between the algorithms; in this case, CBS makes the worst choice as its paths are too long compared to the other algorithms, likely due to a large value sampled from the gamma distribution.

	CBS Re-plan			MCTS Planner			MCTS Heuristic		
	$\alpha = 5$ $\beta = 0.8$	$\alpha = 5$ $\beta = 1.2$	$\alpha = 5$ $\beta = 2.0$	$\alpha = 5$ $\beta = 0.8$	$\alpha = 5$ $\beta = 1.2$	$\alpha = 5$ $\beta = 2.0$	$\alpha = 5$ $\beta = 0.8$	$\alpha = 5$ $\beta = 1.2$	$\alpha = 5$ $\beta = 2.0$
Avg. Total Travel T.	44	42	45	35	35	35	36	36	37
Std. Total Travel T.	3.42	5.33	6.19	2.81	3.22	3.00	1.02	1.23	1.34
Avg. Makespan	16	15	17	11	12	12	10	10	11
Std. Makespan	3.74	3.70	3.77	1.02	1.16	1.12	2.81	3.22	3.00
Avg. Exec. time	6.42s	6.28s	7.01s	0.36s	0.47s	0.56s	1.55s	1.57s	1.59s

Table 7.2: Results obtained after 100 runs in the 14×14 scenario with 4 agents using different gamma distribution configurations.

Table 7.2 displays the quantitative results of the three algorithms evaluated on three configurations of the gamma function across 100 runs. Each set of results aligns with the trend observed in the previous experiment: the two MDP-based algorithms exhibit similar performance, leveraging the probabilistic model for the obstacle lifespan. The CBS Re-plan algorithm demonstrates higher mean makespan and mean total travel time compared to the other algorithms. This is because to achieve an optimal makespan or travel time, the gamma distribution's sampled value must be aligned with the real obstacle lifespan. In runs where $\beta = 1.2$, the algorithm shows slightly better performance, likely due to the probability being more centered on the real obstacle lifespan.

The MCTS Planner consistently delivers superior performance across all scenarios, thanks to its stochastic updating of the gamma distribution. In each simulation, the algorithm samples and simulates different values from the gamma distribution, updating the distribution after moving the robot to a new position through a stochastic procedure.

The MCTS Heuristic provides solutions of comparable quality by leveraging internal simulations and testing various configurations of the believed obstacle lifes-

pan. However, it works slightly worse than the MCTS Planner, as the update of the belief of the obstacle lifespan occurs only when the robot perceives the obstacle.

Across all three gamma distributions, results for both MDP-based algorithms (MCTS Planner and MCTS Heuristic) remain consistent. This can be attributed to the explicit utilization of the probabilistic model in the re-planning procedure, enabling the approaches to consider the expected value of the lifespan (as in the CBS re-planning approach) and the distribution shape.

7.3.2 Validation on real robotic platforms

In the second experiment, we deploy the algorithms in a realistic environment involving two mobile robots (RB-Kairos 7.3). Each robot knows the topological map and the position of the other robot.

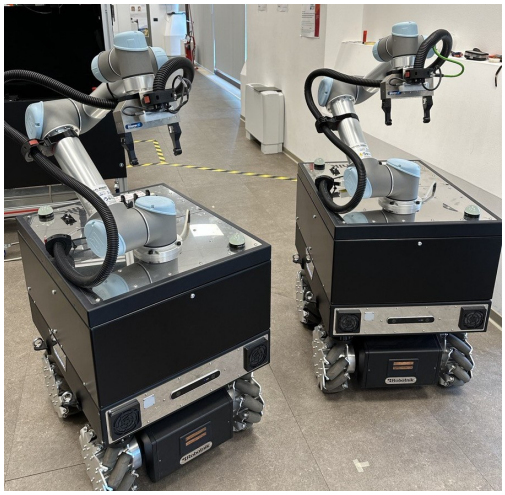


Fig. 7.3: RB-Kairos5

Figure 7.4 illustrates the topological map at the moment of collision: the green robot starts its path from node 0, the blue robot in node 1, and the snapshot is captured at time step 3.

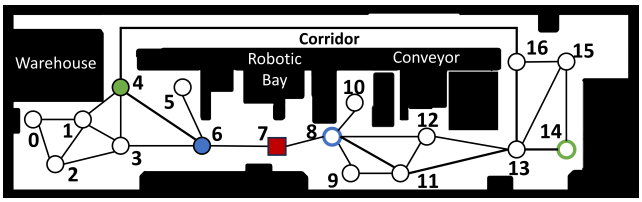


Fig. 7.4: Topological map of our laboratory at the collision time. The filled circles represent the robot positions, the contoured circles the goal positions, and the red square the obstacle

For this experiment, we introduce a real obstacle situated at node 7 with a lifespan of 2. We set $\alpha = 5$ and $\beta = 1.2$; a collision is detected if the subsequent node is obstructed.

	Original path	Total Travel T.	Makespan
●	[1, 3, 6, 7, 8]	11	6
●	[0, 1, 4, 16, 13, 14]		

	CBS Re-plan MCTS Planner MCTS Heuristic	Total Travel T.	Makespan
●	[1, 3, 6, 4, 16, 13, 11, 8]	14	8
●	[0, 1, 4, 16, 13, 14]		

Table 7.3: Original and re-planned paths starting from the initial node on the real scenario (in red the nodes at the collision time)

In Table 7.3, the paths generated by the three algorithms are reported with the total travel time and makespan. All three algorithms yield identical results due to the confined environment. The blue robot (which must estimate the obstacle lifespan) has to calculate a new path, and in this case, the total travel time of the recalculated path is slightly worse than the original one. Achieving a better total travel time is only possible when the believed lifespan matches the real lifespan, but this occurrence has a low probability. Therefore, all the proposed algorithms provide an equal solution.

7.4 Discussion

In conclusion, this chapter demonstrates how Monte Carlo Tree Search (MCTS) can effectively support local decision-making in multi-robot navigation by formalizing the re-planning task as a Markov Decision Process (MDP). By integrating a stochastic model of obstacle lifespan into the transition dynamics, we enable each robot to make informed, localized decisions without requiring global coordination or re-planning. The two MCTS-based strategies—one based on iterative action selection and the other on a bandit-style heuristic—offer flexible solutions to dynamic path obstructions. Experimental results confirm that this localized approach outperforms conventional MAPF re-planning in both efficiency and solution quality. The key takeaway is that modeling obstacle uncertainty explicitly allows for smarter, more adaptive robot behaviors in dynamic environments, paving the way for scalable and robust warehouse automation systems.

Chapter 8

CONCLUSION

"Above all, don't fear difficult moments. The best comes from them."

Rita Levi-Montalcini

This thesis presented the design and implementation of hierarchical control architectures for the autonomous guidance of nonlinear systems—particularly aircraft—under uncertainty. The central objective was to plan and control trajectories optimally or sub-optimally while accounting for real-world stochasticity, including model inaccuracies and measurement noise.

To address uncertainties in system dynamics, we employed the Markov Decision Process (MDP) framework at the high level, which enables the modeling of stochastic transitions and physical constraints. Monte Carlo Tree Search (MCTS) was selected as the primary solver for the MDP due to its favorable theoretical guarantees and scalability in complex decision spaces. For uncertainties arising from noisy measurements, we adopted the Partially Observable Markov Decision Process (POMDP) formalism, embedding belief-state dynamics directly into the planning process.

8.1 Summary of Results

The thesis provides a comprehensive study of how Monte Carlo Tree Search (MCTS), embedded in a hierarchical control architecture, can be used to solve constrained stochastic optimal control problems for autonomous systems. The work is structured around three core contributions

MCTS for Constrained Stochastic Optimal Control

At the heart of the thesis is the use of Markov Decision Processes (MDPs) as high-level planners for nonlinear systems subject to uncertainty. MCTS was used as the primary solver for these MDPs due to its favorable convergence properties, both asymptotic and under finite-time rollouts. We formalized the planning problem as an MDP incorporating system dynamics, input constraints, and cost functions, and we demonstrated how MCTS can compute control sequences that are optimal or provably near-optimal.

This contribution establishes a theoretical and practical foundation for using MCTS to solve stochastic constrained control problems in real-time, even in the presence of nonlinearity and uncertainty.

Robust State Estimation and Uncertainty Handling

Two strategies were developed to handle measurement and model uncertainty in aircraft systems

- **POMDP-Based Approach:** This approach combines a POMDP planner (solved via Partially Observable Monte Carlo Planning, POMCP) with an inverse dynamics controller. It is particularly useful in settings where sensor noise significantly impairs state estimation. The planner operates on a belief space, allowing for robust decision-making in uncertain environments.
- **Lie Group-Based Approach:** This method uses an optimal minimum-energy observer designed on the Lie group structure and a controller defined on the Lie algebra. The planner operates directly on the estimated state, leveraging the geometric structure of the system (e.g., SE(3) for aircraft). This approach is well-suited when reliable state estimates can be obtained and computational efficiency is critical.

Both methods were tested in simulated aircraft navigation tasks that involved reaching target positions while avoiding no-fly zones. The choice between the two approaches depends on the reliability of state estimation: when measurement noise dominates and estimation is uncertain, the POMDP-based approach is preferable. In contrast, when accurate state estimates can be provided by an observer, the Lie group-based strategy offers a more efficient and structure-aware solution.

Decentralized Multi-Agent Path Planning

The hierarchical framework was extended to multi-agent systems, specifically fleets of Unmanned Aerial Vehicles (UAVs), operating in a decentralized fashion. Each UAV ran its own MDP-based planner using MCTS to compute local trajectories while accounting for dynamic constraints, formation requirements, and no-fly zones. To avoid collisions, UAVs shared predicted trajectories with one another and modeled their uncertainty using a time-weighted exponential distribution over future states.

This approach allowed agents to coordinate without a central controller and adapt their actions in real-time. Simulations demonstrated that this decentralized strategy can maintain formation integrity, avoid conflicts, and reduce total mission cost, even in the presence of limited prediction accuracy.

High-Level Re-Planning in Warehouse Robotics

Finally, we applied MCTS to high-level decision-making in mobile robot navigation within a warehouse environment. Here, the focus was on local re-planning in response to unexpected obstacles. The re-planning problem was formulated as an MDP with a stochastic transition function based on probabilistic obstacle lifespans.

Two re-planning algorithms were developed: 1) A full MCTS-based planner that sampled obstacle lifespans and chose actions minimizing expected travel time. 2) A

bandit-style MDP that selected only the initial action and deferred further planning to Space-Time A*.

Both methods were compared against a baseline approach that re-invoked the Conflict-Based Search (CBS) algorithm upon each conflict. Simulations and real-world tests confirmed that incorporating a stochastic obstacle model allows for faster, more efficient local decisions, improving system performance in terms of makespan and total travel time.

8.2 Discussion and Future Work

This thesis has demonstrated how the Monte Carlo Tree Search (MCTS) algorithm, when integrated within the Markov Decision Process (MDP) and Partially Observable MDP (POMDP) frameworks, can be effectively used to address stochastic constrained optimal control problems, including in decentralized multi-agent settings and dynamic environments. However, while the hierarchical control architecture showed strong potential, several limitations and open challenges remain.

One key limitation of the MDP and POMDP formulations lies in the discretization of continuous state, action, and observation spaces. For physical systems—such as aircraft or mobile robots operating in continuous environments—discretization can lead to significant approximation errors, computational bottlenecks, or loss of fidelity. The need for high-resolution discretization to capture dynamic behavior precisely also results in increased computational load, particularly during MCTS-based simulations. This limitation is especially evident when solving real-time planning problems or coordinating a large number of agents, where scalability is crucial.

Another practical challenge is the definition of an effective and compact action space representation. The quality of the MCTS planning process is highly sensitive to how actions are sampled and expanded during the search. Without a suitable action heuristic or domain-informed sampling strategy, the algorithm may waste computational resources exploring suboptimal or irrelevant parts of the search tree. This issue becomes even more pronounced in high-dimensional or underactuated systems.

From an architectural perspective, a comparison between the POMDP-based and Lie group-based approaches for handling state uncertainty suggests trade-offs. The POMDP framework offers a principled way to embed measurement uncertainty into the planner itself, making it well-suited for environments with high sensor noise and requiring robust long-horizon planning. In contrast, the Lie group-based formulation leverages the geometric structure of the system, offering a more compact and elegant representation—particularly useful when high-fidelity state estimation is available through optimal observers. Future work should more systematically investigate the conditions under which one approach outperforms the other, including metrics such as computational cost, tracking accuracy, and robustness to disturbances.

In the context of multi-agent coordination and collision avoidance, this thesis proposed a decentralized planning approach in which UAVs exchange predicted trajectories to avoid collisions while maintaining formation. While this technique proved effective, it also introduced challenges related to the consistency and reliability of shared predictions. The time-weighted uncertainty model introduced in this

work represents an initial step in accounting for these factors. Further improvements could explore belief-space representations or incorporate learning-based models to better predict the intentions and dynamics of other agents in the fleet.

Finally, the warehouse re-planning scenario emphasized the benefits of embedding stochastic models (e.g., obstacle lifespan) directly into the transition dynamics of the MDP. This local re-planning approach significantly improved responsiveness and efficiency while minimizing fleet-wide disruptions. However, the integration of learned priors or adaptive models for obstacle behavior could further enhance performance, especially in more dynamic or unstructured environments.

To address these limitations, several future research directions are proposed:

- **Informed Action Sampling via Learning:** One key area of research is the use of learning-based heuristics to guide MCTS expansion. For instance, reinforcement learning or supervised learning could be used to train policies that estimate action value or likelihood, significantly improving convergence speed by focusing exploration on promising regions of the state-action space.
- **Scenario-Based Stochastic Model Predictive Control (MPC):** A promising direction is to use MCTS as a scenario-based stochastic MPC solver. In this formulation, MCTS acts as an online sampling-based optimizer over uncertain future scenarios, balancing optimality and robustness under model uncertainty. Unlike traditional MPC, which often relies on linearization or convexity, MCTS can handle nonlinear dynamics and nonconvex constraints, making it particularly suitable for real-world, high-fidelity systems.
- **Multi-Agent Coordination:** Future work will deepen the application of MDPs for multi-agent control, focusing on dynamic network representations of agent interactions embedded within the MDP transition model. This would allow scalable, adaptive coordination strategies in uncertain and dynamic environments.
- **Graph-based or topological representations for agent coordination:** Modeling the coordination of multi-agent systems as dynamic graphs (e.g., formation graphs, communication networks) may provide more scalable abstractions for use within MDPs. Integrating such representations into the planner could lead to better coordination policies and allow for dynamic reconfiguration of formations or communication topologies.

In summary, this thesis provides a unified framework that bridges optimal planning, uncertainty modeling, and control for complex autonomous systems. The combination of theoretical insights and practical implementations lays the groundwork for further development of robust and scalable decision-making architectures in autonomous robotics.

REFERENCES

1. A. Serrani, "Lecture notes on nonlinear and adaptive control of advanced aerospace systems," personal communication.

2. F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer, 2019, vol. 49.

3. S. Kobayashi and K. Nomizu, *Foundations of Differential Geometry, Volume 2*. John Wiley & Sons, 1996, vol. 61.

4. V. S. Varadarajan, *Lie groups, Lie algebras, and their representations*. Springer Science & Business Media, 2013, vol. 102.

5. S. M. Sotoudeh and B. HomChaudhuri, "A robust mpc-based hierarchical control strategy for energy management of hybrid electric vehicles in presence of uncertainty," in *2020 American Control Conference (ACC)*. Ieee, 2020, pp. 3065–3070.

6. J. Lehoczky, S. P. Sethi, H. M. Soner, and M. I. Taksar, "An asymptotic analysis of hierarchical control of manufacturing systems under uncertainty," *Mathematics of operations research*, vol. 16, no. 3, pp. 596–608, 1991.

7. D. Q. Mayne, J. B. Rawlings, and C. V. Rao, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

8. J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Book on Control and Decision, 2009.

9. T. Stastny and R. Siegwart, "Nonlinear model predictive guidance for fixed-wing uavs using identified control augmented dynamics," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 432–442.

10. R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A distributed multi-robot coordination algorithm for navigation in tight environments," *arXiv preprint arXiv:2006.11492*, 2020.

11. Y. Kuriki and T. Namerikawa, "Formation control with collision avoidance for a multi-uav system using decentralized mpc and consensus-based control," *SICE Journal of Control, Measurement, and System Integration*, vol. 8, no. 4, pp. 285–294, 2015.

12. A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 2007, pp. 207–226.

13. G. C. Calafiore and L. Fagiano, "Robust model predictive control via scenario optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 219–224, 2012.

14. W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

15. M. Cannon, J. Buerger, B. Kouvaritakis, and S. Rakovic, "Robust tubes in nonlinear model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1942–1947, 2011.

16. A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

17. D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1468–1480, 2011.

18. J. Yan and R. R. Bitmead, "Incorporating state estimation into model predictive control and its application to network traffic control," *Automatica*, vol. 41, no. 4, pp. 595–604, 2005.

19. L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *2006 American Control Conference*. IEEE, 2006, pp. 7–pp.

20. D. Bertsekas and S. E. Shreve, *Stochastic optimal control: the discrete-time case*. Athena Scientific, 1996, vol. 5.

21. L. G. Crespo and J.-Q. Sun, "Stochastic optimal control via bellman's principle," *Automatica*, vol. 39, no. 12, pp. 2109–2114, 2003.

22. L. Buşoniu, T. De Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

23. F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

24. E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

25. D. Corsi, L. Marzari, A. Pore, A. Farinelli, A. Casals, P. Fiorini, and D. Dall'Alba, "Constrained reinforcement learning and formal verification for safe colonoscopy navigation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 289–10 294.

26. L. Marzari, A. Pore, D. Dall'Alba, G. Aragon-Camarasa, A. Farinelli, and P. Fiorini, "Towards hierarchical task decomposition using deep reinforcement learning for pick and place subtasks," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 640–645.

27. L. Marzari, D. Corsi, F. Cicalese, and A. Farinelli, "The #DNN-Verification Problem: Counting Unsafe Inputs for Deep Neural Networks," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023, pp. 217–224.

28. G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 2017, pp. 97–117.

29. T. Wei, L. Marzari, K. S. Yun, H. Hu, P. Niu, X. Luo, and C. Liu, "Modelverification. jl: a comprehensive toolbox for formally verifying deep neural networks," *arXiv preprint arXiv:2407.01639*, 2024.

30. M. Zanon, S. Gros, and M. Palladino, "Stability-constrained markov decision processes using mpc," *Automatica*, vol. 143, p. 110399, 2022.

31. A. B. Kordabad, M. Zanon, and S. Gros, "Equivalence of optimality criteria for markov decision process and model predictive control," *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 1149–1156, 2023.

32. M. Ahmadi, N. Jansen, B. Wu, and U. Topcu, "Control theory meets pomdps: A hybrid systems approach," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5191–5204, 2020.

33. L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

34. C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

35. S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative rnns and monte carlo tree search," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 263–10 269.

36. J. Ousingsawat and M. E. Campbell, "On-line estimation and path planning for multiple vehicles in an uncertain environment," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 14, no. 8, pp. 741–766, 2004.

37. A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 3. IEEE, 2002, pp. 1936–1941.

38. T. Schouwenaars, A. Richards, E. Feron, and J. How, "Plume avoidance maneuver planning using mixed integer linear programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001, p. 4091.

39. R. E. Mortensen, "Maximum-likelihood recursive nonlinear filtering," *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 386–394, 1968.

40. A. Saccon, J. Trumpf, R. Mahony, and A. P. Aguiar, "Second-order-optimal minimum-energy filters on lie groups," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2906–2919, 2015.

41. D. Rigo, C. Segala, N. Sansonetto, and R. Muradore, "Second-order-optimal filter on lie groups for planar rigid bodies," *IEEE Transactions on Automatic Control*, vol. 67, no. 9, pp. 4971–4977, 2022.

42. D. Rigo, N. Sansonetto, and R. Muradore, "Second-order-optimal filtering on se (2)× r2 for the chaplygin sleigh," *Systems & Control Letters*, vol. 178, p. 105568, 2023.

43. ——, "Geometric optimal filtering for an articulated n-trailer vehicle with unknown parameters," *International Journal of Robust and Nonlinear Control*, 2024.

44. D. Rigo, A. Saccon, M. Alirezaei, E. Lefeber, N. Sansonetto, and R. Muradore, "State estimation for a tractor semi-trailer system using a minimum-energy filter," in *2024 European Control Conference (ECC)*. IEEE, 2024, pp. 1–6.

45. D. Rigo, N. Sansonetto, and R. Muradore, "A comparison between the extended kalman filter and a minimum-energy filter in the tse (2) case," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6175–6180.

46. F. Vesentini, D. Rigo, N. Sansonetto, L. Di Persio, and R. Muradore, "Minimum-energy switching geometric filter on lie groups for differential-drive wheeled mobile robots," *European Journal of Control*, vol. 80, p. 101101, 2024.

47. B. R. Floriano, G. A. Borges, H. C. Ferreira, and J. Y. Ishihara, "Hybrid dec-pomdp/pid guidance system for formation flight of multiple uavs," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–20, 2021.

48. F. Trotti, A. Farinelli, and R. Muradore, "An online path planner based on pomdp for uavs," in *2023 European Control Conference (ECC)*. IEEE, 2023, pp. 1–6.

49. S. Ragi and E. K. Chong, "Uav path planning in a dynamic environment via partially observable markov decision process," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, pp. 2397–2412, 2013.

50. S. , Ragi and E. K. Chong, "Dynamic uav path planning for multitarget tracking," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 3845–3850.

51. S. Ragi and E. K. P. Chong, *UAV Guidance Algorithms via Partially Observable Markov Decision Processes*. Dordrecht: Springer Netherlands, 2015, pp. 1775–1810.

52. R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on control systems technology*, vol. 9, no. 6, pp. 777–790, 2001.

53. H. Yamaguchi, "A cooperative hunting behavior by mobile robot troops," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 4. IEEE, 1998, pp. 3204–3209.

54. W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 1859–1864.

55. Q. Wang, H. Gao, F. Alsaadi, and T. Hayat, "An overview of consensus problems in constrained multi-agent coordination," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 275–284, 2014.

56. Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

57. Y. Kuriki and T. Namerikawa, "Consensus-based cooperative formation control with collision avoidance for a multi-uav system," in *2014 American Control Conference*. IEee, 2014, pp. 2077–2082.

58. J. Zhang, J. Yan, P. Zhang, and X. Kong, "Collision avoidance in fixed-wing uav formation flight based on a consensus control algorithm," *IEEE Access*, vol. 6, pp. 43 672–43 682, 2018.

59. Z. Zhen, G. Tao, Y. Xu, and G. Song, "Multivariable adaptive control based consensus flight control system for uavs formation," *Aerospace Science and Technology*, vol. 93, p. 105336, 2019.

60. D. Silver and J. Veness, "Monte-carlo planning in large pomdps," *Advances in neural information processing systems*, vol. 23, 2010.

61. M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

62. R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

63. P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.

64. M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.

65. R. Stern, "Multi-agent path finding–an overview," *Artificial Intelligence: 5th RAAI Summer School, Dolgoprudny, Russia, July 4–7, 2019, Tutorial Lectures*, pp. 96–115, 2019.

66. G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.

67. F. R. Garza and E. A. Morelli, "A collection of nonlinear aircraft simulations in matlab," Tech. Rep., 2003.

68. B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

69. J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

70. Z. Feldman and C. Domshlak, "Monte-carlo planning: Theoretically fast convergence meets practical efficiency," *arXiv preprint arXiv:1309.6828*, 2013.

71. R. Durrett, *Probability: theory and examples*. Cambridge university press, 2019, vol. 49.

72. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

73. R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

74. D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

75. P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

76. C. I. Byrnes and A. Isidori, "On the attitude stabilization of rigid spacecraft," *Automatica*, vol. 27, no. 1, pp. 87–95, 1991.

77. B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

78. D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

79. T. Söderström, *Discrete-time stochastic systems: estimation and control*. Springer Science & Business Media, 2012.

80. A. Isidori, *Nonlinear control systems II*. Springer, 2013.

81. H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

82. L. T. Nguyen, *Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability*. National Aeronautics and Space Administration, 1979, vol. 12854.

83. C. W. Warren, "Global path planning using artificial potential fields," in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 316–317.

84. B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer Publishing Company, Incorporated, 2008.

85. J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

86. E. Zupan and D. Zupan, "On higher order integration of angular velocities using quaternions," *Mechanics Research Communications*, vol. 55, pp. 77–85, 2014.

87. K. Nomizu, "Invariant affine connections on homogeneous spaces," *American Journal of Mathematics*, vol. 76, no. 1, pp. 33–65, 1954.

88. J. Q. Gallier and J. Quaintance, *Differential geometry and Lie groups*. Springer, 2020, vol. 12.

89. A. Cogliati and P. Mastrolia, "Cartan, Schouten and the search for connection," *Historia Mathematica*, vol. 45, no. 1, pp. 39–74, 2018.

90. M. M. Postnikov, *Geometry VI: Riemannian Geometry*. Springer Science & Business Media, 2013, vol. 91.

91. J. Sun, J. M. Hoekstra, and J. Ellerbroek, "Openap: An open-source aircraft performance model for air transportation studies and simulations," *Aerospace*, vol. 7, no. 8, p. 104, 2020.

92. R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

93. R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Bartak, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Symposium on Combinatorial Search (SoCS)*, pp. 151–158, 2019.