


UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)

Scuola di Dottorato

Ingegneria dei Sistemi, Informatica, Matematica e Ricerca Operativa (ISIMR)

Indirizzo

Ingegneria dei Sistemi e Informatica

La presente tesi è cofinanziata con il sostegno della Commissione Europea, Fondo Sociale Europeo e della Regione Calabria. L'autore è il solo responsabile di questa tesi e la Commissione Europea e la Regione Calabria declinano ogni responsabilità sull'uso che potrà essere fatto delle informazioni in essa contenute

CICLO

XXVII

MODEL PREDICTIVE CONTROL STRATEGIES FOR UNMANNED VEHICLES

Settore Scientifico Disciplinare ING-INF 04

Direttore: Ch.mo Prof. Sergio Greco

Firma _____

Supervisori: Ch.mo Prof. Giuseppe Franzè

Firma _____

Ch.mo Prof. Pietro Muraca

Firma _____

Dottorando: Dott. Walter Lucia

Firma _____

Abstract

This dissertation analyzes the obstacle avoidance motion planning problem for ground and aerial vehicles operating in uncertain environments. By resorting to set-theoretic and predictive schemes based ideas, receding horizon control algorithms are proposed as an effective solution for the obstacle avoidance problem.

Two obstacle scenarios with different levels of uncertainty are first considered for ground vehicles and different vehicle model descriptions are taken into account. In both cases, control architectures exploiting adequate inner ellipsoidal approximations of the exact one-step controllable sets are proposed in order to derive adequate receding horizon control strategies. The main merit of these set-theoretic approaches relies on the fact that constraints fulfilment and uniformly ultimate boundlessness can be proved regardless of any obstacle scenario occurrences.

Then, an hybrid control strategy based on the Command Governor (CG) framework is proposed for aerial vehicles. In particular, the CG scheme is extended in order to take into account non-convex and time-varying constraints typically arising in path planning obstacle avoidance problems. Experimental results on the quadrotor Qball-X4 show applicability and effectiveness of the proposed approach.

Finally, all the proposed control architecture share as common and relevant denominator the fact that the needed computations are move into the off-line phase so rendering the on-line burden modest and therefore affordable in real-time contests.

Abstract (Italiano)

Questa tesi affronta il problema del controllo e della pianificazione del moto di veicoli autonomi terrestri e aerei operanti in ambienti incerti. In particolare vengono presentate diverse strategie di controllo ad orizzonte recedente basate sui principi delle tecniche di controllo predittive e “set-theoretic”.

Nella prima parte è affrontato il problema del controllo dei veicoli terrestri dove sono considerati due diversi livelli di incertezza dello spazio di lavoro. Per entrambi i casi vengono sviluppate delle opportune strategie di controllo ad orizzonte recedente sfruttando delle approssimazioni ellissoidali degli insiemi di controllabilità ad un

passo. Il principale merito di questi approcci è dato dal fatto che il soddisfacimento dei vincoli e l'"uniformly ultimate boundlessnes" possono essere garantiti *a-priori* qualsiasi sia l'evoluzione dello scenario di ostacolo.

Nella seconda parte è affrontato il problema del controllo dei veicoli in volo utilizzando una strategia di controllo ibrida basata sul noto schema del gestore del riferimento (Command Governor (CG)). In particolare l'architettura del CG è estesa in modo che essa sia in grado di tenere in considerazione i tipici vincoli non convessi e tempo varianti dei problemi di controllo in ambienti con ostacoli. I risultati sperimentatali ottenuti con il quadrotore Qball-X4 mostrano l'applicabilità e l'efficacia dell'approccio proposto.

Infine, tutte le strategie presentate hanno come comune denominatore il fatto che la maggior parte delle computazioni richieste si svolgono nella fase fuori linea così da rendere la complessità del calcolo in linea modesto e sostenibile per applicazioni in tempo reale.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Control of unmanned vehicles

In recent years, autonomous vehicles (UVs) have become increasingly important tools in various civil and military operations: border interdiction prevention [50, 80], search and rescue mission [103], wild fire surveillance [24], monitoring over nuclear reactors [106], power plants inspection [20], agricultural services, mapping and photographing [56], battle damage assessment [51] and so on. A wide variety of robotic vehicles is currently in use or being developed, ranging from unmanned fixed-wing aircraft, helicopters, blimps and hovercraft to ground and planetary rovers, earth-orbiting spacecraft and deep-space probes. Although the level of autonomy differs among the types of vehicles and the missions they are used for, many such systems require no or minor human control from a base or ground station. The primary reason for deploying autonomous vehicles is often a reduction in cost or elimination of human risk associated with a particular mission: unmanned systems do not require operator safety and life support systems, and can therefore be made smaller and cheaper than their manned counterparts. Furthermore, autonomous vehicles allow operations in remote or harsh environments and often possess the capability to operate continuously or to complete missions lasting longer w.r.t the timespan capability of a manned system counterpart.

The problem of motion planning and control of UVs deals with finding appropriate strategies such that the vehicle motion satisfies the requirements of a specified task [70]. Avoidance of collisions with obstacles is a key component of the a navigation where the primary objective is to reach a target through the possibly time-varying and unknown obstacle-free part of the environment. To cope with this issue, two aspects must be taken into account: the generation of a *safe* trajectory capable in principle to reach the target by avoiding obstacle occurrences along the path and finding a control action whose the aim is to drive "as close as possible" the UV to the planned path under the satisfaction of the constraints arising from the specific vehicle dynamics [84].

Despite extensive research, this problem still represents a relevant challenge because of unavoidable uncertainties in the operating scenario, inherent deficiencies in

perception abilities and computational capabilities of the robot and restrictions on the vehicle mobility due to nonholonomic kinematic constraints, limited control ranges and under-actuations and so on, see e.g. [60]. In particular, the nonholonomic kinematic constraints arising during the motion of wheeled vehicles lead to quite challenging control problem [30], [35]. Moreover, the non-existence of pure-state feedbacks for the asymptotic stabilization of fixed configurations, proved in [16], has had the effect of increasing the research on the following topics: i) fixed point asymptotic stabilization [100]; ii) asymptotic stabilization of feasible trajectories [52].

1.2 Literature overview

1.2.1 Mere path planning methods

First attempts proposed in literature share as a common denominator a partial (not complete) management of the path following obstacle avoidance problem. The control unit design is either left completely out (e.g. [110], [111]) or it is obtained by considering elementary vehicle model descriptions (see [96]).

Most of these early methods uses simplified kinematic vehicle models, which may lead to conservative results. As an example, the full dynamic capabilities may not be exploited, or a safety margin may have to be included that accounts for uncertainties in the actual motion when the kinematic reference trajectory is tracked by a lower level control law [59], [69].

In the last decades, ground and aerial autonomous vehicles have been strongly taken into consideration and different approaches have been proposed for dealing with the path planning problem in uncertain environments.

Path planning for unmanned ground vehicles (UGVs)

Much attention has been devoted to the motion planning problem for UGVs: in particular the possibility of extending road-map and potential functions methods to the case of dynamic obstacle scenarios [86], [92], [96], [110], [111]. In [86] Probabilistic Roadmap Methods (PRM) are considered with the aim to overcome the assumption of static environments properly of this class of strategies. A straightforward and high-demanding solution consists in updating the roadmap after an obstacle has changed its position. Therefore, the proposed algorithm creates a robust roadmap in the pre-processing phase by using the observation that the behaviour of the moving obstacles is often not unconstrained but restricted to pre-specified areas. In [92], the so-called Partial Motion Planner (PMP) mechanism is designed so that uncertainties arising from planning within dynamic environments can be handled. The main idea is to pre-calculate admissible state trajectories by using the Inevitable Collision States (ICS) framework that, though it is capable to generate safe paths, is subject to high computational burdens which could lead to violate the real-time constraints under which the robot must take a decision. By considering dynamic objects characterized

by piecewise constant velocities, an explicit kinematic model of the robot is considered in [96]: the family of feasible trajectories and their corresponding steering controls are derived in a closed form. In [110], the path planning problem under nonholonomic constraints is addressed by using the so-called *Follow the Gap Method* (FGM). There, by computing a gap array around the robot, the appropriate gap is selected, the best heading vector through the gap derived and the final angle to the target point computed. Along similar lines is the contribution in [111] where a hybrid approach using *a-priori* knowledge of the environment guarantees that the autonomous vehicle cannot be trapped in deadlocks.

Path planning for unmanned aerial vehicles (UAVs)

Motion planning for UAVs have extensively studied by the control community. Based on these efforts, many strategies developed for UGVs have been extended to cover three-dimensional (3-D) environments. It is well known that their computational loads are often prohibitively large and/or the existing algorithms do not scale up well to problems in the 3-D space [13, 71].

From a practical perspective specific characteristics of UAVs pose different challenges to the trajectory planning. For example, a ground vehicle has the ability to stop and go backwards, whereas an aircraft must maintain a minimum velocity. Some rovers and helicopters can make quick turns on the spot, but have a slower turn rate when moving forward.

Current UAV obstacle avoidance techniques can be categorized into two broad classes: Planning and Reactive. Planning paradigms use a world map and plan a trajectory for the vehicle to follow. Such approaches must find trajectories that not only geometrically avoid obstacles but also ensure that the dynamics of the vehicle are capable of following the paths. This can be prohibitively expensive to compute if the trajectory has to be continuously revised. In contrast, reactive methods overcome the real-time problem by using simple formulas to react to obstacles as they appear, but they cannot guarantee an appropriate solution to every possible situation. Finally, vision-based reactive methods have been popular because payload weight is a serious limitation for UAVs, see e.g. [82], [46], [19].

1.2.2 Model predictive strategies for avoidance motion planning problems

Model predictive control (MPC) or receding horizon control (RHC) originated in the process industry, and has received wide attention in the broader field of control theory and other applications. The benefits of such an approach is that it allows to naturally handle multivariable systems, to systematically take actuator limitations into account and to operate closer to its constraints: therefore better performance with respect to traditional techniques can be achieved [79]. For detailed discussions and tutorials, one can refer to the following survey papers [47], [85], [81], [99].

Because of its natural flexibility, the receding horizon control philosophy has been recently used to deal with trajectory planning problems. Note that most of contributions consider obstacle-free environments [53] or are limited to tracking predetermined trajectories around obstacles [114]. On the other and, it is interesting

to underline that the contribution [7] based on the potential function approach proposed a scheme to manage obstacle environments, while in [33] a splines is exploited to produce collision-free trajectories.
Moreover, in [109] a novel receding horizon approach based on mixed-integer linear programming (MILP) was introduced. MILP is a powerful optimization framework that allows inclusion of integer variables and discrete logic in a continuous linear optimization problem. These variables can be used to model logical constraints such as obstacle and collision avoidance rules, while the dynamic and kinematic properties of the vehicle are formulated as continuous constraints, see [37]. Although independently developed, the MILP-based trajectory planning approach of [109] is a special case of the broader class of control for mixed logical dynamical (MLD) systems firstly presented in [8] and then exploited in [9], [5], [12].

Although limited to a single static obstacle configuration, of interest for the dissertation developments are the results outlined in [98]. There, *ad hoc* algorithms for the computation of the set of states that can be robustly steered in a finite number of steps via state feedback control to a given target set while avoiding pre-specified zones or obstacles are achieved by resorting to polyhedral algebra concepts. Moreover, it is shown that such regions can be important to adequately deal with obstacle avoidance problems even if the computational burdens may become prohibitive in specific situations, e.g. the given plant model is subject to bounded disturbances.
Unfortunately, although the above methods provide elegant solutions, they cannot straightforwardly modified to comply with dynamic environments because the huge computational complexity arising during the on-line operations.

Hence based on this reasoning, fast MPC algorithms (see e.g. [88], [125]) have been used to achieve appropriate strategies for UVs operating within time-varying obstacle scenarios, see to cite a few the following recent contributions [112], [87], [126], [58], [32] and [22]. In [112] the authors combines an MPC-based controller with local obstacle map building using on-board laser scanners. In [87], the dynamic window approach (DWA) navigation scheme is recast within a continuous time nonlinear model control predictive (MPC) framework by resorting to the ideas developed in [94]. In particular the algorithm is based on the jointly use of a model-based optimization scheme and a convergence-oriented potential field method. Although interesting, the approach suffers of unavoidable high computational burdens mainly due to the MPC phase that it is slightly mitigated by splitting the dissipative controls into two subsets with piecewise controls. The authors of [126] consider the collision avoidance as well as navigation toward the destination using a MPC approach. The related on-line optimization is solved by means of nonlinear programming, while a local path planning generator makes use of the so-called *distance* and parallax methods. The main handicap of this method relies on the heavy computational loads pertaining to the computation of the predictive controller action. In this contribution, this difficulty is attacked by considering a very large sensing range that allows to have available sufficient time for solving the underline constrained optimization problem. Along similar lines is also [58] where the authors address the set-point regulation problem of a wheeled mobile robot in a known dynamic environment populated with static and moving obstacles subject to robot kinematic and dynamic constraints is ad-

dressed by using the nonlinear model predictive control in polar coordinates. In [32], a novel procedure to approximately solve the path planning problem in dynamic uncertain environments is proposed. There the authors develop a closed-loop receding horizon control algorithm whose solution integrates prediction, estimation, and planning by taking care of the so-called chance constraints arising from the uncertainty on robot and obstacle locations. The drawback of such an approach relies on the need to make available the agent prediction paths whose computation could be high demanding. The authors of [22] present a two-mode MPC scheme which ensures feasibility retention despite of time-varying state constraints: the standard operative mode is based on the classical Receding Horizon Control (RHC) philosophy, while the safe mode, activated under time-varying constraints scenarios, allows to confine the state trajectory within an opportune robustly positively invariant set. An important aspect leaves out concerns with the definition of a procedure capable to determine new feasible paths when the standard mode cannot be recovered.

1.3 Proposed predictive schemes for unnamed vehicles

The obstacle avoidance motion planning for ground and aerial vehicle is tracked by developing novel MPC strategies: set-theoretic RHC schemes for unmanned ground vehicles, hybrid command governor (CG) based architectures for unmanned aerial vehicles.

1.3.1 Set-theoretic receding horizon control for UGVs

In the sequel, two different uncertain obstacle configurations are considered:

> **Structured Environments:** *The UGV operates in a working environment where the admissible obstacle locations are known, but at each time instant it is unpredictable which is the current obstacle configuration.*
>
> **Unstructured Environments:** *Both static and moving (agents) objects are considered and two configurations result:* ***static obstacle scenario*** *and* ***moving obstacle scenario****. No a-priori information are available on the agent behaviours.*

According to these scenarios, *ad-hoc* control strategies will be developed: ***OA-MPC***, ***SOS-OA-MPC*** and ***DOA-MPC*** schemes.

Structured Environments. The consequence of such a set-up is that the resulting working environment gives rise to a certain degree of uncertainty that if not properly treated can lead to collisions during the vehicle navigation. To deal with this problem, a discrete-time RHC strategy based on set-theoretic ideas is developed so that the prescribed saturation and geometric constraints are always fulfilled despite of any obstacle scenario occurrence. The key motivation supporting such an approach relies on the capabilities of the RHC philosophy combined both with set-invariance concepts and the ellipsoidal calculus to guarantee control performance and computational load savings under constraints satisfaction and disturbances effect attenuation

requirements, see [2].
Then, the main ingredients of the proposed strategy can be summarized as follows:

- A stabilizing state feedback law and a robust positively invariant ellipsoidal set centred at the goal location;
- A set of initial states that, according to the obstacle scenario configurations, can be steered to the target in a finite number of steps;
- An on-line receding horizon strategy obtained by deriving the smallest ellipsoidal set complying with the current obstacle configuration. The control move is computed by minimizing a performance index such that the one-step ahead state prediction belongs to the successor set.

A relevant feature of this scheme is the capability to move off-line most of computations and to ensure that there exists at each time instant a feasible solution complying with the time-varying obstacle configuration prescriptions. Then, a second important merit relies on the needed computational resources that are significantly modest because the command input computation prescribes at most the solution of a Quadratic Programming (QP) problem under linear constraints. Finally, a further significant feature is that the proposed strategy can straightforwardly adapted to deal with heterogeneous plant descriptions: linear, uncertain and polynomial models.

Unstructured Environments. Differently from the **Structured** set-up, this configuration gives rise to an unbounded number of possible obstacle-scenarios that make the ***OA-MPC*** strategy unaffordable. As a consequence, a quite different approach capable to mitigate such a drawback must be developed. Here, this is achieved by jointly exploiting controllability and robust positively invariance concepts. Then, the key ingredients of the proposed ***DOA-MPC*** scheme can be summarized as follows:

- An off-line derived one-step controllable set family that allow to characterize admissible paths between initial and final planar locations according to the **static obstacle scenario** configurations;
- An on-line procedure based on two independent phases and aimed to work around obstacles along the current path: from an obstacle side (starting point) a sequence of robust positively invariant sets is derived, while on the other side (target) a new controllable set sequence is computed;
- Simple semi-definite programming (SDP) problems for the on-line computation of appropriate control actions compatible with the moving agents.

1.3.2 Hybrid Command governor scheme for UAVs

An hybrid command governor controller based on the ideas developed in [6], [4], [23], [118] is designed to give fully autonomous capability to UAVs to perform flight missions within cluttered environment where the obstacle avoidance is a fundamental task. The proposed control architecture consists of three main modules: a **reference manager (CG)** whose aim is to generate at each time instant a feasible set-point to be tracked during the on-line operations; a **planner** unit that determines a finite

sequence of locations in order to allow obstacle avoidance during the vehicle navigation and a **control module (SCG-OA)** that is in charge to manage switching events when time-varying constraint paradigms are considered.

In principle a modification of the constraint set structure should prescribe a new CG design during the on-line operations, but this could be computationally very demanding and, therefore, not usable in practical situations. To overcome this drawback, the key idea here is to avoid the *CG re-design* by exploiting the following arguments:

- time-varying constraints arise due to a shift with respect the current equilibrium;
- different constraints sets are overlapped and their shapes are invariant w.r.t. any equilibrium shift.

Finally, it is worth to remark that the resulting hybrid strategy has similar low computational loads as those required by the basic CG unit.

1.4 Contributions

This thesis presents a set-theoretic control framework based on model predictive ideas for dealing with obstacle avoidance problems arising when moving unmanned vehicles operate in dynamic environments. The proposed approach jointly generates safe paths within the obstacle-free working region and computes adequate control actions in such a way that the vehicle is capable to take care of such trajectories by satisfying prescribed constraints. One of its main merits is the capability to keep the on-line computational burdens affordable for real-time implementations. This is a significant contribution especially if compared to the existing literature where either of a lack of an explicit control module (focused only to path planning tasks) or remarkable computational loads are serious limitations, e.g. tracking capabilities [69, 83].
This dissertation is in large parts based on the following contributions:

Journals

I) *Systems & Control Letters*: "The obstacle avoidance motion planning problem for autonomous vehicles: a low-demanding receding horizon control scheme" (*Provisionally Accepted*) [41];

Conferences

II) *American Control Conference (ACC), 2013*: "An obstacle avoidance receding horizon control scheme for autonomous vehicles" [44];

III) *Control Automation (MED), 2013*: "An obstacle avoidance model predictive control scheme: A sum-of-squares approach" [76];
IV) *Decision and Control (CDC), 2013*: "A model predictive control scheme for mobile robotic vehicles in dynamic environments" [40];
V) *Decision and Control (CDC), 2014*: "An obstacle avoidance and motion planning Command Governor based scheme: the Qball-X4 Quadrotor case of study" [77].

Submitted manuscripts

VI) "A receding horizon control strategy for autonomous vehicles in dynamic environments" (*Under review*) [43];
VII) "An obstacle avoidance model predictive control scheme for mobile robots subject to nonholonomic constraints: a sum-of-squares approach" (*Under review*) [42].
VIII) "An Hybrid Command Governor scheme for unmanned aerial vehicles: obstacle avoidance and motion planning" (*Under review*) [78].

1.5 Dissertation outline

The results of this thesis are collected in two main chapters:

1. **Control of unmanned ground vehicle**;
2. **Control of unmanned aerial vehicle**.

The first part concerns with the development of a set-theoretic based MPC scheme capable to accomplish motion planning, obstacle avoidance and command input computation tasks for unnamed ground vehicles (Chapters $3-4$). Then, Chapter 5 is devoted to address similar problems for unmanned aerial vehicle (UAV) by properly adapting the high level Command Governor (CG) strategy.
A detailed description of each chapter is below reported:

Chapter 2 provides definitions and preliminary results on the following topics:
- Set-theoretic control approach;
- Polynomial systems and Sum-of-Squares relaxations;
- Command Governor design.

Chapter 3 develops a set-theoretic receding horizon control framework for dealing with the obstacle avoidance problem for UGVs operating in highly structured environments. Simulations on different model descriptions of the autonomous point mobile vehicle are provided and commented;

Chapter 4 derives a MPC scheme capable to address the obstacle avoidance motion planning problem for unknown time-varying obstacle scenarios. Then, simulation campaigns on a point mobile robot model show the effectiveness of the proposed control architectures;

Chapter 5 first defines the path planning obstacle avoidance problem for UAVs. Then, an adequate CG characterization is developed in order to deal with time-varying constraints. The validity of the proposed strategy is testified by real-time laboratory experiments on the Quadrotor Qball-X4;

Chapter 6 concludes the dissertation by summarizing the proposed results and outlining future research directions.

2

Background material

In this chapter the theoretic background on the basis of this dissertation is presented. The first section collects standard definitions on control and computer science community, the remaining sections collect background materials on: ellipsoidal receding horizon control, polynomial systems, Sums of Squares based optimization technicalities and command governor control strategy.

2.1 Definitions

Consider the discrete-time nonlinear system

$$x(t+1) = f(x(t), u(t), d(t)) \tag{2.1}$$

subject to the following set-membership state and input constraints:

$$u(t) \in \mathcal{U}, \quad \forall t \geq 0 \tag{2.2}$$
$$x(t) \in \mathcal{X}, \quad \forall t \geq 0 \tag{2.3}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^n$, $t \in \mathbb{Z}_+ := \{0, 1, ...\}$, $x(t) \in \mathbb{R}^n$ denotes the plant state, $u(t) \in \mathbb{R}^m$ the control input, $\mathcal{U}$, $\mathcal{X}$ compact subsets of $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively and $d(t)$ is an exogenous disturbance belonging to the following compact and convex set

$$d(t) \in \mathcal{D} \subset \mathbb{R}^d, \ \ \forall t \in \mathbb{Z}_+ \tag{2.4}$$

Definition 2.1. *A set $\Xi \subseteq \mathcal{X}$ is Robustly Positively Invariant (RPI) for (2.1) if there exists a control law $u(t) \in \mathcal{U}$ such that once the closed-loop solution $x_{CL}(t)$ enters inside that set at any given time t_0, it remains in it for all future instants, i.e. $x_{CL}(t_0) \in \Xi \to x_{CL}(t) \in \Xi, \forall d(t) \in \mathcal{D}, \forall t \geq t_0$.* □

Definition 2.2. *Let S be a neighborhood of the origin. The closed-loop trajectory of (2.1) is said to be Uniformly Ultimate Bounded in S if for all $\mu > 0$ there exist $T(\mu) > 0$ and $u(t) \in \mathcal{U}$ such that, for every $\|x(0)\| \leq \mu$, $x_{CL}(t) \in S$ for all $t \geq T(\mu)$.* □

Definition 2.3. *Given a set $\Xi \subseteq \mathcal{X}$ (target set), the predecessor set $Pre(\Xi)$, is the set of states for which there exists a causal control $u(t) \in \mathcal{U}$ such that $\forall d \in \mathcal{D}$ the one-step state evolution is in Ξ, i.e.,*

$$Pre(\Xi) := \{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D} : f(x,u,d) \in \Xi\} \tag{2.5}$$

□

As a consequence, by induction, it is possible to determine the sets of states k-steps controllable to Ξ via the following recursion (see [10]):

$$\begin{aligned} &\Xi_0 := \Xi \\ &\Xi_n := Pre(\Xi_{n-1}), \quad n = 1, \ldots, k \end{aligned} \tag{2.6}$$

Definition 2.4. *Given two sets $\mathcal{A}$ and $\mathcal{B}$, the subtraction*

$$\mathcal{A} \sim \mathcal{B} := \{a \in \mathcal{A} : a + b \in \mathcal{A}, \forall b \in \mathcal{B}\} \tag{2.7}$$

is said P-difference and $\sim$ the P-difference operator. □

Definition 2.5. *Given a square symmetric matrix $P = P^T \geq 0$ (shaping matrix) and a vector $x_0 \in \mathbb{R}^n$, then the set*

$$\mathcal{T}(P, x_0) := \left\{x \in \mathbb{R}^z : (x - x_0)^T P^{-1} (x - x_0) \leq 1\right\}$$

represents a x_0-centered ellipsoid and

$$\mathcal{T}(P) := \left\{x \in \mathbb{R}^n : x^T P^{-1} x \leq 1\right\}$$

a 0-centered ellipsoid. □

Definition 2.6. *Given an ellipsoidal set $\mathcal{T}(P, x_0)$ and its support function*

$$\max_{x \in \mathcal{T}(P)} c^T x = l^T x_0 + \sqrt{c^T P c}$$

then

$$\mathcal{T}(P, x_0) := \left\{x \in \mathbb{R}^n : l^T x \leq l^T x_0 + \sqrt{c^T P c} \quad \forall c \in \mathbb{R}^n\right\} \tag{2.8}$$

is called dual-parametrization of $\mathcal{T}(P, x_0)$. □

Definition 2.7. *Given a set $W \subset \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$, the distance is defined as:*

$$dist(x, W) = \inf_{w \in W} \|x - w\|_*,$$

where $*$ is any relevant norm. □

Definition 2.8. *Given two sets W, $R \subset \mathbb{R}^n$, the distance between them is defined as:*

$$dist(W, R) = \inf \{\|w - r\|_* \,|\, w \in W,\ r \in R\}$$

□

Definition 2.9. *An oriented graph is an ordered pair $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ such that*

- $\mathbf{V}$ *is the vertex set;*
- $\mathbf{E}$ *is a subset of ordered pairs of $\mathbf{V}$ known as the edge set, i.e.*

$$\mathbf{E} \subset \{\{u, v\} \,|\, u, v \in \mathbf{V}\}$$

□

Definition 2.10. *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ an oriented graph, the reachable vertex set $\mathbf{V}_r$ is defined as*

$$\mathbf{V}_r := \{v \in \mathbf{V} | \exists u \in \mathbf{V} : \{u, v\} \in \mathbf{E}\}$$

□

Definition 2.11. *Let $f(x)$ and $g(x)$ be two differentiable vector fields. The Lie bracket of $f(x)$ and $g(x)$ is another vector field defined as*

$$[f, g](x) := \frac{\partial g}{\partial x}(x) f(x) - \frac{\partial f}{\partial x}(x) g(x)$$

□

2.2 Dual mode receding horizon predictive control approach

The objective of this section is to provide a background on the Dual-Mode Model Predictive Control (***DUAL-MPC***) strategy, based on ellipsoidal calculus and viability theory, developed in [2]. This approach represents a starting point for the set-theoretic strategies developed in Chapters 3 and 4.

2.2.1 Problem formulation and dual mode approach

Consider plants described by linear discrete-time invariant (LTI) models subject to an exogenous bounded disturbance $d(t)$

$$x(t+1) = \Phi x(t) + Gu(t) + G_d d(t) \tag{2.9}$$

and suppose that the control objective is to design a control strategy able to stabilize (2.9) and track a set point (i.e. 0_x) under the hard constraints (2.2)-(2.3).

A convenient way of addressing such a problem in an MPC contest is that of resorting to a dual-mode scheme. First the largest domain of attraction (DoA) of (2.9) complying with (2.2)-(2.3) is *off-line* computed, i.e. all the initial conditions $x(0)$

for which there exists an admissible path to 0_x; then the DoA is *on-line* exploited to compute feasible control actions able to steer the state configuration to the desired equilibrium point, e.g. 0_x.

Next two sections will be devoted to describe the *off-line* and the *on-line* stages of the ***DUAL-MPC*** control architecture.

2.2.2 Off-line phase: RPI region and One-step ahead Controllable sets

Let K be a stabilizing feedback gain and Ξ an RPI region:

$$(\Phi + GK)\Xi + G_d\mathcal{D} \in \Xi$$

complying with (2.2)-(2.3), then starting from the definitions (2.5)-(2.6), the set of states of (2.9) controllable in k steps to Ξ, regardless of any disturbance $d(t) \in \mathcal{D}$ can be computed via the following recursion:

$$\begin{array}{l} \Xi_0 := \Xi \\ \Xi_k := \{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D} : \Phi x + Gu + G_d d \in \Xi_{k-1}\} \end{array} \tag{2.10}$$

Unfortunately the shapes of Ξ_k grows in complexity as k increase and may become computationally intractable very quickly. A possible way to solve such a drawback is to exploit ellipsoidal inner approximations of Ξ_k as

$$\begin{array}{l} \mathcal{T}_0 := In[\Xi] \\ \mathcal{T}_k := In[\{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D} : \Phi x + Gu + G_d d \in \mathcal{T}_{k-1}\}] \end{array} \tag{2.11}$$

where $\mathcal{T}_k$, if non-empty, satisfies $\mathcal{T}_k \subset \Xi_k$ and $In[\cdot]$ denotes the inner ellipsoidal approximation operator according to some optimality criterion, e.g. maximum volume, trace, etc. Moreover each ellipsoidal set enjoys the property that for any $x \in \mathcal{T}_k \setminus \mathcal{T}_{k-1}$ there exists $u \in \mathcal{U}$ so that, for any $d \in \mathcal{D}$, $\Phi x + Gu + G_d d \in \mathcal{T}_{k-1}$

The family $\mathcal{T}_k$ always exists whenever the uncertain systems are quadratically stabilizable by means of a linear feedback, provided that $\mathcal{D}$ is sufficiently small (or alternatively, provided $\mathcal{U}$ and $\mathcal{X}$ sufficiently large). The exact quantification of the relative sizes of $\mathcal{U}$, $\mathcal{X}$ and $\mathcal{D}$ for the existence of the family $\{\mathcal{T}_k\}$ cannot be absolute but it depends on the system at hand, see [57] and [104]for more details.

Finally, for sake of completeness, a computational algorithm for (2.11) is detailed in Appendix A.

2.2.3 On-line phase: ellipsoidal MPC algorithm

The above developments allow to synthesize very easily a Receding Horizon control strategy capable to steer the state trajectory to the equilibrium point 0_x. On-line, at each time instant t, the smallest index k such that $x \in \mathcal{T}_k$ is first determined. Then, a command $u(t)$ is computed by minimizing a convenient running cost under the

condition $\hat{x}(t+1/t) := \Phi x(t) + Gu(t) \in (\mathcal{T}_{k-1} \sim G_d D) = \tilde{\mathcal{T}}_{k-1}$, which is used to ensure contraction and viability to the closed-loop trajectories irrespective of any admissible disturbance occurrence.

Remark 2.12. In this respect, it is worth to pointing out that, unlike [123] and [31], the determination of the ellipsoidal set $\mathcal{T}_k$ does not depend on the running cost to be used during the on-line operations. This allows more flexibility in the choice of the cost which could eventually even changed on-line along the system trajectories.

Let $\{\mathcal{T}_k\}$ a sequence of non-empty ellipsoidal sets off-line computed by means of (2.11), then the resulting ellipsoidal MPC algorithm is:

---

*Ellipsoidal-MPC **(E-MPC))***

1: $k(t) := \min\{k : x(t) \in \mathcal{T}_k\}$
2: ***if*** $k(t) == 0$ ***then***
3: $\quad u(t) = Kx(t)$
4: ***else***

$$u(t) = \arg\min J_{k(t)}(x(t), u(t)) \quad s.t \tag{2.12}$$

$$\Phi x + Gu \in \tilde{\mathcal{T}}_{k-1}, \quad u \in \mathcal{U} \tag{2.13}$$

5: ***end if***
6: *apply* $u(t)$; $t \leftarrow t+1$*; goto Step 1*

Observe that the cost $J_{k(t)}(x(t), u(t))$ may depend on $k(t)$ and be defined on an infinite horizon if e.g. implicit dual-mode MPC schemes are of interest. Otherwise, typical choices include $J_{k(t)}(x(t), u) = ||\Phi x + Gu||^2_{\tilde{P}_k^{-1}(k(t)-1)}$, where $\tilde{\mathcal{T}}_k = \left\{x : x^T \tilde{P}_k^{-1} x \leq 1\right\}$, or $J_{k(t)}(x(t), u) = ||u||^2_\Psi$, $\Psi = \Psi^T \geq 0$ if one is interested in approximating the one-step minimum-time or, respectively, the one-step minimum-energy algorithms. Since the optimization problem in Step 4 is always feasible, the following stability result holds true:

Proposition 2.13. *Let the sequence of ellipsoids $\{\mathcal{T}_k\}$ be non-empty and $x(0) \in \bigcup_k \mathcal{T}_k$ then the **E-MPC** control algorithm always satisfies the constraints (2.2)-(2.3) and ensures robust stability. In particular there exists a finite time $\bar{t}$ such that $x(t) \in \mathcal{T}_0, \forall t \geq \bar{t}$*

Proof. The proof follows trivially from the above discussion. Convergence in finite time derives from the choice of a limited number $\bar{k}$ of ellipsoid $\mathcal{T}_k$, $k = 1, \ldots, \bar{k}$. □

2.3 Semidefinite programming relaxations for semialgebraic problems and polynomial nonlinear systems

Recent improvement on semi-definite programming solvers and development of efficient polynomial optimization algorithms have contributed to enhance the interest towards the so-called *Sums-of-Squares* (SOS) based techniques for control design purpose.

Therefore the objective of this section is to provide background materials on polynomials, polynomial nonlinear models, semialgebraic sets and SOS based convex relaxations [91], [63]. These concepts are instrumental to understand Section 3.4.2 where the one-step ahead controllable set recursion (2.6) for non linear model vehicles is approximately computed by means of a SOS based optimization problem.

2.3.1 Polynomial definitions

Definition 2.14 (Monomials). *A **Monomial** m_α in n variables is a function defined as $m_\alpha := x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$ for $\alpha \in \mathbb{Z}_+^n$. The degree of a monomials is defined as $deg(m_\alpha) := \sum_{i=1}^n \alpha_i$.* □

Definition 2.15 (Polynomials). *A **Polynomials** f in n variables is a finite liner combination of monomials,*

$$f := \sum_\alpha c_\alpha m_\alpha = \sum_\alpha c_\alpha x^\alpha$$

with $c_\alpha \in \mathbb{R}$. Define $\mathcal{R}_n$ to be the set of all polynomials in n variables also denoted as $\mathcal{R}[x_1, \dots, x_n]$ to emphasize the independent variables. The degree of f is defined as $deg(f) := \max_\alpha deg(m_\alpha)$ (provided that the associated c_α is non-zero) □

Definition 2.16 (Positive Semidefinite Polynomials). *A positive semidefinite polynomial is a polynomial $p(x) \in \mathcal{R}_n$ such that $p(x) \geq 0 \ \forall x \in \mathbb{R}^n$. Define $\mathcal{P}_n$ to be the set of all positive polynomials in n variables*

$$\mathcal{P} := \{p \in \mathcal{R}_n | p(x) \geq 0, \forall x \in \mathbb{R}^n\}$$

also defined as $\mathcal{P}_{n,d} := \mathcal{P}_n \cap \mathcal{R}_{n,d}$ where the subscripts n, d indicates polynomial in n variables and maximum degree d, respectively. □

Definition 2.17 (Sums-of-Squares (SOS) Polynomials). *A SOS polynomial is a polynomial that can be represented as sums of squares of other polynomials. The set of all SOS polynomials in n variables Σ_n is defined as*

$$\Sigma_n := \left\{ s \in \mathcal{R}_n | \exists M < \infty, \exists \{p_i\}_{i=1}^M \subset \mathcal{R}_n \text{ such that } s = \sum_{i=1}^M p_i^2 \right\}$$

Additionally, define $\Sigma_{n,d} = \Sigma_n \cap \mathcal{R}_{n,d}$ □

Definition 2.18 (Multiplicative Monoid). *Given $\{g1, \ldots, g_t\} \in \mathcal{R}_n$, the Multiplicative Monoid generated by g_j, $j = 1, \ldots t$ is the set of all finite products of g_j's, including 1 (i.e. the empty product). It is denoted as $\mathcal{M}(g1, \ldots, g_t)$. For completeness define $\mathcal{M}(\emptyset) := 1$.* □

An example: $\mathcal{M}(g_1, g_2) = \left\{g_1^k g_2^k | k1, k2 \in \mathbb{Z}_+\right\}$

Definition 2.19 (Cone). *Given $\{f1, \ldots, f_r\} \in \mathcal{R}_n$, the Cone generated by f_i, $i = 1, \ldots r$ is*

$$\mathcal{P}(f_1, \ldots, f_r) := \left\{ s0 + \sum_{i=1}^{l} s_i b_i | l \in \mathbb{Z}_+, s_i \in \Sigma_n, b_i \in \mathcal{M}(f1, \ldots, f_r) \right\}$$

For completeness note that $\mathcal{P}(\emptyset) := \Sigma_n$ □

Note that if $s \in \Sigma_n$ and $f \in \mathcal{R}_n$, then $f^2 s \in \Sigma_n$ as well. This allows to express a cone of $\{f_1, \ldots, f_r\}$ as a finite sum of 2^r terms.

An example: $\mathcal{P}(f_1, f_2) = \{s_0 + s_1 f_1 + s_2 f_2 + s_3 f_1 f_2 | s_0, \ldots, s_3 \in \Sigma_n\}$.

Definition 2.20. ***(Ideal).*** *Given $\{h1, \ldots, h_u\} \in \mathcal{R}_n$, the Ideal generated by h_k, $k = 1, \ldots, u$ is*

$$\mathcal{I} := \left\{ \sum_{k=1}^{u} h_k p_k | p_k \in \mathcal{R}_n \right\}$$

For completeness note that $\mathcal{I}(\emptyset) := 0$ □

2.3.2 SOS polynomials properties and computational aspects

Since every $s \in \Sigma_n$ is a sum of squared polynomials, it is clear that $s(x) \geq 0$, $\forall x \in \mathbb{R}^n$, which implies that $\Sigma_n \subseteq \mathcal{P}_n$. An interesting question is whether the set of SOS polynomials is equal to or strictly contained in the set of positive semidefinite polynomials. It has been proved (see [102] for an overview) that there are only three combinations of number of variables and degree such that the set of SOS polynomials is equivalent to the set of positive semidefinite ones:

1. Polynomials in one variable, $n = 1$;
2. Quadratic polynomials, $d = 2$;
3. Quartics in two variables, $n = 2, d = 4$.

This result dates to Hilbert and is related to his 17th problem.

In general working with polynomials in $\mathcal{P}_n$ can be difficult since there is no full parametrization of such a set nor an efficient tests to check if a given polynomial is inside it. However, given the number of variables, n, and degree of polynomials, d, it is possible to form a full parametrization of $\Sigma_{n,d}$ which directly leads to an efficient semidefinite programming test to check if a polynomial is SOS (see proof

of Theorem 1 in [101]). Noticing that all SOS polynomials must always be of even degree, as a consequence the parameterization of the set $\Sigma_{n,2d}$ for some $n, d \in \mathbb{Z}_+$ can be considered in the following statement:

Statement 2.21. *If $s \in \Sigma_{n,2d}$, then there exists $p_i \in \mathcal{R}_{n,d}$, $i = 1, \ldots, M$, for some finite M such that*

$$s = \sum_{i=1}^{M} p_i^2$$

Using the above result it is possible pose a full parametrization of $\Sigma_{n,2d}$, often referred ad the "Gram matrix" approach. In [26] is showed that $p \in \Sigma_{n,2d}$ iff $\exists Q \geq 0$ such that $p(x) = z_{n,d}(x)^T Q z_{n,d}(x)$, with $z_{n,d}(x)$ a vector of all the monomials of degree less than or equal to d ordered in an appropriate manner. As an example, with $n = 2, d = 2$

$$z_{2,2}(x)^T = [\,1,\, x_1,\, x_2,\, x_1^2,\, x_1x_2,\, x_2^2\,]$$

In [91] has been showed that the algorithm to check if any $Q \geq 0$ exists for a given $p \in \mathcal{R}_n$ is an LMI, and proved the following extension

Proposition 2.22. *Given a finite set $\{p_i\}_{i=0}^m \in \mathcal{R}_n$, the existence of $\{a_i\}_{i=1}^m \in \mathbb{R}$ such that*

$$p_0 + \sum_{i=1}^{m} a_i p_i \in \Sigma_n$$

is an LMI feasible problem.

This theorem is useful since it allows one to answer questions like the following SOS programming example

Example 2.23. Given $p_0, p_1 \in \mathcal{R}_n$, does there exists a $k \in \mathcal{R}_n$ such that

$$p_0 + kp_1 \in \Sigma_n \tag{2.14}$$

To answer this question, write k as a linear combination of its monomials $\{m_j\}$, $k = \sum_{j=1}^{s} a_j m_j$. Rewrite (2.14) using this decomposition

$$p_0 + kp_1 = p_0 + \sum_{j=1}^{s} a_j (m_j p1)$$

which, since $(m_j p1) \in \mathcal{R}_n$, can be checked by Proposition 2.22.

Many software packages, for instance SOSTOOL [93], YALMIP [75] and GloptiPoly [54], have been developed aimed to solve the LMIs that result from Proposition 2.22. This packages set up the LMIs from the polynomial problem, do some smart preprocessing to reduce problem size and use semidefinite programming solvers,e.g. [115], to solve the LMIs.

2.3.3 The positivstellensatz

In this section a central theorem from real algebraic geometry, the Positivstellensatz (P-satz), is stated. This is a powerful theorem which generalizes many well known results such as the S-Procedure. In particular the latter is very important in control theory because it enables to cast some control problems, e.g. stability, controllability and disturbance analysis, for systems with polynomial vector field into tractable convex optimization problems.

Proposition 2.24 (Positivstellensatz). *Given polynomials* $\{f_1, \ldots, f_r\}, \{g_1, \ldots, g_t\}$ *and* $\{h_1, \ldots, h_u\}$ *in* $\mathcal{R}_n$*, the following are equivalent [11]:*

- *The set*
$$\left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{c} f_1(x \geq 0, \ldots, f_r(x) \geq 0) \\ g_1(x) \neq 0, \ldots, g_t(c) \neq 0 \\ h_1(x) = 0, \ldots, h_u(x) = 0 \end{array} \right\} \quad \textit{is empty}$$
- *There exist polynomials* $f \in \mathcal{P}(f_1, \ldots, f_r)$*,* $g \in \mathcal{M}(g_1, \ldots, g_t)$*, and* $h \in \mathcal{I}(h_1, \ldots, h_u)$ *such that*
$$f + g^2 + h = 0$$

□

Remark 2.25. Note that when there are only inequalities constraints, and they describes a compact region, this theorem can be improved to reduce the number of free parameters [108], and with slightly stronger assumptions [95]

The LMI based test for SOS polynomials from Proposition 2.22 can be used to prove that the set emptiness condition from the P-satz holds by finding specific f, g and h such that $f+g^2+h = 0$. These f, g and h are known as P-satz certificates since they certify that the equality is satisfied. The following proposition states precisely how semidefinite programming can be used to search for certificates [91].

Proposition 2.26 (P-satz Certificates). *Given polynomials* $\{f_1, \ldots, f_r\}, \{g_1, \ldots, g_t\}$ *and* $\{h_1, \ldots, h_u\}$ *in* $\mathcal{R}_n$*, if the set*

$$\{x \in \mathbb{R}^n \,|\, f_i(x) \geq 0, g_i(x) \neq 0, h_k(x) = 0, i = 1, \ldots, r, j = 1, \ldots, t, k = 1, \ldots, u\}$$

is empty then the search for bounded degree P-satz refutations can be done using semidefinite programming. If the degree bound is chosen large enough the semidefinite program will be feasible and give the refutation certificates.

To enforce the usefulness of the P-satz the next two subsections will provide two examples that become convex and thus tractable when the P-satz is combined with the result of Proposition 2.22.

An LMI test for $\mathcal{P}_n$

Using the P-satz, it is possible test if a polynomial $p \in \mathcal{R}_n$ is in $\mathcal{P}_n$. If $p \in \mathcal{P}_n$, then $\forall x \in \mathbb{R}^n, p(x) \geq 0$. Equivalently $\{x \in \mathbb{R}^n : p(x) < 0\}$ is empty, or in the P-satz format

$$\{x \in \mathbb{R}^n \mid -p(x) \geq 0, p(x) \neq 0\} \quad \text{is empty}$$

This condition hold iff $\exists f \in \mathcal{P}(-p)$ and $g \in \mathcal{M}(p)$ such that $f + g^2 = 0$. Using the definition of the cone and the monoid, $p \in \mathcal{P}_n$ iff $\exists s_0, s_1 \in \Sigma_n$ and $k \in \mathbb{Z}_+$ such that

$$s_0 - ps_1 + p^{2^k} = 0$$

If we fix k and the degree of s_1 to be d, the above condition can be rewritten as $p \in \mathcal{P}_n$ iff $\exists s_1$ such that

$$s_1 \in \Sigma_{n,d} \tag{2.15}$$

$$ps_1 - p^{2k} \in \Sigma_{n,\hat{d}} \tag{2.16}$$

with $\hat{d} = \max(2kdeg(p), d + deg(p))$. Making use of this manipulation the conditions (2.15-2.16) can be checked with LMIs (via Proposition 2.22) that give a sufficient conditions for a polynomial to be positive semidefinite.

The S-procedure

Given symmetric $n \times n$ matrices $\{A_k\}_{k=0}^m$ the S-procedure states: if there exists nonnegative scalars $\{\lambda_k\}_{k=1}^m$ such that $A_0 - \sum_{k=1}^m \lambda_k A_k \geq 0$, then

$$\bigcap_{k=1}^{m} \left\{x \in \mathbb{R}^n : x^T A_k x \geq 0\right\} \subseteq \left\{x \in \mathbb{R}^n : x^T A_0 x \geq 0\right\}$$

This can be rewritten according to the P-satz form as

$$\mathcal{W} = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} x^T A_1 x \geq 0, \ldots, x^T A_m x \geq 0 \\ -x^T A_0 x \geq 0,\ x^T A x \neq 0 \end{array} \right\} \quad \text{is empty}$$

If the λ_k exist, define $Q := A_0 - \sum_{k=1}^{m} \lambda_k A_k$. By assumption $Q \geq 0$ and thus $x^T Q x \in \Sigma_n$. Define $g(x) := x^T A_0 x \in \mathcal{M}(x^T A_0 x)$ as well as

$$f(x) := (x^T Q x)(-x^T A_0 x) + \sum_{k=1}^{m} \lambda_k (-x^T A_0 X)(x^T A_k x)$$

By their non-negativity each $\lambda_i \in \Sigma_n$ and because $x^T Q x \in \Sigma_n$ the function $f(x)$ is in the come $\mathcal{P}(x^T A_1 x, \ldots, x^T A_m x, -x^T A_0 x)$. An easy rearrangement gives $f + g^2 = 0$, which illustrates that f and g are Positivstellensatz certificates that prove that $\mathcal{W}$ is empty.

Remark 2.27. The S-procedure given above can be straightforwardly generalized to deal with non-quadratic function and non-scalar weights, see [63] for details.

2.3.4 Polynomial recasting of nonlinear systems

It has been shown in [107] that any system with non-polynomial nonlinearities can be converted through a simple series of steps to a polynomial system with a larger state dimension, but with a series of equality constraints restricting the states to a manifold of the original state dimension. In some cases the recasting is "exact", in the sense that the transformed system has a polynomial vector field with the same dimension as the original system

Example 2.28. Consider for example the non linear differential equation

$$\dot{x} = ce^{-\alpha x(t)}$$

by setting $p(t) = ce^{-\alpha x(t)}$ it can be rewritten as

$$\dot{p} = -\alpha p^2(t)$$

In other cases, recasting increases the state dimension but equality constraints that arise from the recasting restrict the system to the original manifold. In particular, the constraints that arise can be either polynomial, or include non-polynomial terms and therefore a suitable recasting algorithm is required. The here presented procedure is a modified version of the algorithm proposed in [107] appearing in [90]. This is applicable to a very large class of non-polynomial systems, namely those whose vector field is composed of sums and products of elementary functions, or nested elementary functions of elementary functions. For elementary functions here are considered the functions with explicit symbolic derivatives such as exponential (e^x), logarithm ($\ln x$), power (x^a), trigonometric ($\sin x$, $\cos x$, etc.), and hyperbolic functions ($\sinh x$, $\cosh x$, etc.).

Recasting procedure

Suppose that the original continuous time system is given in the form

$$\dot{z}_i = \sum_j \alpha_i \prod_k F_{ijk}(z) \tag{2.17}$$

where $i = 1, \ldots, s$, $\alpha_j' s$ are real numbers and $z = [z_1, \ldots, z_s]^T$ is the state vector. In the above equation, $F_{ijk}(z)$ are assumed to be elementary functions, or nested elementary functions of elementary functions. For the above system, the recasting algorithm is:

Polynomial Recasting Algorithm ***(PRA)***

1: Let $x_i = z_i, \forall i = 1, \ldots, s$;

2: For each $F_{ijk}(z)$ that is not of the form $F_{ijk}(z) = z_l^a$, where a is some integer and $1 \leq l \leq s$, introduce a new variable x_r. Define $x_r = F_{ijk}(z)$;

3: Compute the differential equation describing the time evolution of x_r using the chain rule of differentiation;
4: Replace all appearances of such $F_{ijk}(z)$ in the system equations by x_r;
5: Repeat Steps 2-4, until system equations with rational forms are obtained;

The above recasting process generally produces a recasted system whose dimension is higher than the dimension of the original system. To describe the original model faithfully, constraints of the form

$$x_r = F(x_1, \ldots, x_s),\ \forall r > s \tag{2.18}$$

should be taken into account. These constraints define an $s-$dimensional manifold on which the solutions to the original differential equations lie. In general such constraints cannot be converted into polynomial forms, even though sometimes there exist polynomial constraints that are induced by the recasting process. For example:

- Two variables introduced for trigonometric functions such as $x_2 = \sin x_1$, $x_3 = \cos x_1$ are constrained via $x_2^2 + x_3^2 = 1$.
- Introducing a variable to replace a power function such as $x_2 = \sqrt{x1}$ induces the constraints $x_2^2 - x_1 = 0, x_2 \geq 0$.
- Introducing a variable to replace an exponential function such as $x_2 = \exp(x_1)$ induces the constraint $x_2 \geq 0$.

Therefore the non-polynomial system (2.17) can generally rewritten as

$$\begin{cases} \dot{x}_a = f_a(x_a, x_b) \\ \dot{x}_b = f_b(x_a, x_b) \end{cases} \tag{2.19}$$

where $x_a = [x_1, \ldots, x_s]^T$ are the state variables of the original system (2.17), $x_b = [x_{s+1}, \ldots, x_n]^T$ the additional state variables and $f_a(x_a, x_b),\ f_b(x_a, x_b)$ polynomials forms. Moreover, the following further equality/inequality requirements

$$G_a(x_a, x_b) = 0 \tag{2.20}$$

$$G_b(x_a, x_b) \geq 0 \tag{2.21}$$

are necessary so that $f_a(x_a, x_b)$ and $f_b(x_a, x_b)$ faithful represent the original nonlinear system (2.17) .Note that F, G_a and G_b are column vectors of functions with appropriate dimensions.

Finally, to better clarify the ***PRA*** the interested reader can find two recasting examples in Appendix B where two non-liner ground vehicle models are taken into consideration .

2.4 Basic command governor (CG) design

The aim of this section is to present the basic properties and capabilities of the well known Command Governor (CG) control architecture, see [6], [4], [23] and [48] for exhaustive dissertations. This strategy represents a starting point for the hybrid control architecture for unmanned aerial vehicle presented in Chapter 5.

2.4.1 CG Control Architecture and Problem Formulation

Consider the control architecture shown in Fig. 2.1, consisting of the **Plant**, the **Primal Controller** and the **CG** device. Suppose that the **Primal System** can be linearized around a given equilibrium point and discretized as

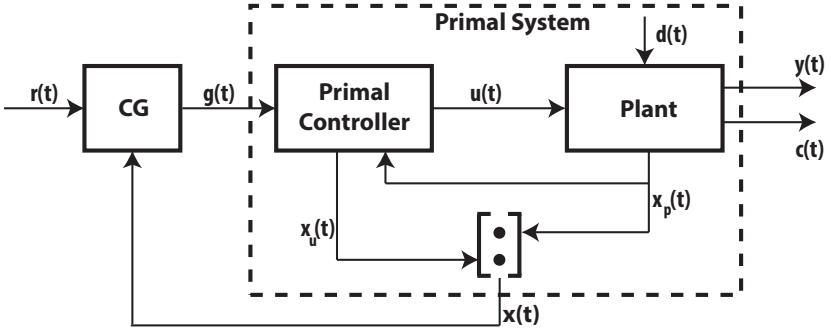

Fig. 2.1. Command Governor Structure

$$\begin{cases} x(t+1) = \Phi x(t) + Gg(t) + G_d d(t) \\ y(t) \quad = H_y x(t) \\ c(t) \quad = H_c x(t) + Lg(t) + L_d d(t) \end{cases} \tag{2.22}$$

where $x(t) \in \mathbb{R}^n$ is the overall state including the plant and primal controller states, x_p and x_c respectively; $g(t) \in \mathbb{R}^m$ is the CG action, i.e. a suitably modified version of the reference signal $r(t) \in \mathbb{R}^m$; $d(t) \in \mathcal{D} \subset \mathbb{R}^{n_d}$, $\forall t \in Z_+$ is an exogenous disturbance, with $\mathcal{D}$ a specified convex and compact set such that $0_{n_d} \in \mathcal{D}$; $y(t) \in \mathbb{R}^m$ is the plant output which is required to track $r(t)$; $c(t) \in \mathbb{R}^{n_c}$ the constrained output vector

$$c(t) \in \mathcal{C}, \ \ \forall t \in \mathbb{Z}_+ \tag{2.23}$$

with $\mathcal{C}$ a specified convex and compact set. It is also assumed that:

Assumption 2.29. *The primal controller has been designed so that*

a) Φ is a Schur matrix
*b) The **Primal System** (2.22) is offset-free, i.e. $H_y(I_n - \Phi)^{-1}G = I_m$*

Observe that the typical structure of a CG-equipped control system consists of two nested loops. The internal loop is designed via a generic linear control method, without taking into account the prescribed constraints, and allows the designer to specify relevant system properties for small-signal regimes, e.g. stability, disturbance rejection. The outer loop consists of the CG unit which, whenever necessary, is in charge to modify the reference to be applied to the closed-loop system so as to avoid constraint violation. The basic idea is that of maintaining the closed-loop system within

its nominal linear regime, where the stability and all other closed-loop properties are preserved.

Statement 2.30. *The CG design problem is that of generating, at each time instant t, the set-point $g(t)$ as a function of the current state $x(t)$ and reference $r(t)$*

$$g(t) := \overline{g}(x(t), r(t)) \tag{2.24}$$

in such a way that, regardless of disturbances, constraints (2.23) are always fulfilled along the system trajectories generated by the application of the modified set-points $g(t)$ and possibly $y(t) \approx r(t)$. Moreover, it is required that: $g(t) \to \hat{r}$ whenever $r(t) \to \hat{r}$, where $\hat{r}$ is either r or its best feasible approximation; the CG has a finite settling time, viz. $g(t) = \hat{r}$ for a possibly large but finite t whenever the reference stays constant after a finite time.

2.4.2 CG solution and properties

By linearity, it is possible to separate the effects of the initial conditions and input from those of disturbances, i.e. for each generic system variable $n(t) : n(t) = \overline{n}(t) + \tilde{n}(t)$, where $\overline{n}(t)$ is the disturbance-free component and $\tilde{n}(t)$ depends only on disturbances. Therefore, the disturbance-free solutions of (2.22) to a constant command $g(t) = \omega$ are:

$$\begin{aligned} \overline{x}_\omega &:= (I_n - \Phi)^{-1} G\omega \\ \overline{y}_\omega &:= H_y (I_n - \Phi)^{-1} G\omega \\ \overline{c}_\omega &:= H_c (I_n - \Phi)^{-1} G\omega + L\omega \end{aligned} \tag{2.25}$$

Consider next the following set recursions:

$$\begin{gathered} \mathcal{C}_0 := \mathcal{C} \sim L_d \mathcal{D}, \\ \mathcal{C}_k := \mathcal{C}_{k-1} \sim H_c \Phi^{k-1} G_d \mathcal{D}, \ldots, \mathcal{C}_\infty := \bigcap_{k=0}^{\infty} \mathcal{C}_k, \end{gathered} \tag{2.26}$$

It can be shown that the sets $\mathcal{C}_k$ are nonconservative restrictions of $\mathcal{C}$ such that $\overline{c}(t) \in \mathcal{C}_\infty$, $\forall t \in \mathbb{Z}_+$, implies that $c(t) \in \mathcal{C}$, $\forall t \in \mathbb{Z}_+$. Thus, one can consider only disturbance-free evolutions of the system and adopt a "worst-case" approach. For reasons which will appear clear soon, it is convenient to introduce the following sets for a given $\delta > 0$

$$\mathcal{C}^\delta := \mathcal{C}_\infty \sim \mathcal{B}_\delta \tag{2.27}$$

$$\mathcal{W}^\delta := \left\{ \omega \in \mathcal{R}^m : \ \overline{c}_\omega \in \mathcal{C}^\delta \right\} \tag{2.28}$$

where $\mathcal{B}_\delta$ is a ball of radius δ centered at the origin. We shall assume that there exists a possibly vanishing $\delta > 0$ such that $\mathcal{W}^\delta$ is non-empty. In particular, $\mathcal{W}^\delta$ is the set of all commands whose corresponding steady-state solutions satisfy the constraints with a tolerance margin δ. From the foregoing definitions and assumptions, it follows

that $\mathcal{W}^\delta$ is closed and convex.
The main idea is to choose at each time step a constant virtual command $g(\cdot) \equiv \omega$, with $\omega \in \mathcal{W}^\delta$, such that the corresponding evolution fulfils the constraints over a semi-infinite horizon and its "distance" from the constant reference is minimal. Such a command is applied, a new state is measured and the procedure is repeated. In this respect we define the set $\mathcal{V}(x)$ as

$$\mathcal{V}(x) = \left\{\omega \in \mathcal{W}^\delta : \ \bar{c}(k, x, \omega) \in \mathcal{C}_k, \ \forall k \in \mathcal{Z}_+\right\} \tag{2.29}$$

where $\bar{c}(k, x, \omega) = H_c \left(\Phi^k x + \sum_{i=0}^{k-1} \Phi^{k-i-1} G\omega \right) + L\omega$ is the constraint disturbance-free virtual evolution at time k from the initial condition x under the constant command $g(\cdot) \equiv \omega$. As a consequence $\mathcal{V}(x) \subset \mathcal{W}^\delta$, and, if non-empty, it represents the set of all constant virtual sequences in $\mathcal{W}^\delta$ whose evolutions starting from x satisfies the constraints also during transients. As a consequence, we have the following definition:

Definition 2.31. *The state $x \in \mathbb{R}^n$ is defined $\mathcal{C}^\delta-$admissible if there exist $\omega \in W^\delta$ such that $c(k, x, \omega) \in \mathcal{C}, \forall k \in \mathbb{Z}_+$. The pair (x, ω) is said $\mathcal{C}^\delta-$executable.* □

Thus the CG output is chosen according to the solution of the following constrained optimization problem

$$g(t) = \arg \min_{\omega \in \mathcal{V}(x(t))} \|\omega - r(t)\|_\Psi \tag{2.30}$$

where $\Psi = \Psi^T > 0_m$ and $\|\omega - r(t)\|_\Psi := (\omega - r(t))^T \Psi (\omega - r(t))$.
Relevant contributions on this approach can be found in [4] [23], [48], [49], [118], while the main CG properties are below summarized:

Property 2.32. *Consider system (2.22) along with the CG selection rule (2.30). Let Assumption (2.29) be fulfilled and $\mathcal{V}(x(0))$ be non-empty. Then the following properties hold true:*

a) The minimizer in (2.30) uniquely exists at each $t \in \mathbb{Z}_+$ and is obtained by solving a convex constrained optimization problem, viz. $\mathcal{V}(x(0))$ non-empty implies $\mathcal{V}(x(t))$ non-empty along the trajectories generated by the CG command (2.30);
b) The set $\mathcal{V}(x(t))$, $\forall x(t) \in \mathbb{R}^n$, is finitely determined, viz. there exists an integer k_0 such that if $\bar{c}(k, x(t), \omega) \in \mathcal{C}_k$, $k \in \{0, 1, \ldots, k_0\}$, then $\bar{c}(k, x(t), \omega) \in \mathcal{C}_k \ \forall k \in \mathbb{Z}_+$;
c) The constraints are satisfied for all $t \in \mathbb{Z}_+$;
d) The overall system is asymptotically stable. Specifically, whenever $r(t) \equiv r$, $g(t)$ converges in finite time either to r or to its best steady-state admissible approximation $\hat{r}$, with

$$\hat{g} := \hat{r} := \arg \min_{\omega \in \mathcal{W}^\delta} \|\omega - r\|_\Psi^2 \tag{2.31}$$

□

Finally, computational details on the CG design and the optimization problem (2.30) are reported in the Appedix C

Part I

Control of unmanned ground vehicle

3

A set-theoretic receding horizon control approach for structured environments

This chapter formally addresses the obstacle avoidance motion planning problem for ground autonomous vehicles operating in uncertain structured environments. By resorting to set-theoretic ideas, a receding horizon control algorithm is derived for robots described by means of both linear and nonlinear models.
In order to improve as much as possible the overall comprehension and clarity the proposed framework is first developed for constrained linear time-invariant systems and it is then extended in order to deal with polytopic and polynomial state space descriptions.

3.1 Problem Statement

This section first collects the main definitions and assumptions used in this chapter and the problem to be solved is stated at the end.

Definition 3.1. *Let Ob_j^i be an obstacle. Then, then obstacle scenario $\mathcal{O}^i$ at the time instant t is defined as*

$$\mathcal{O}^i(t) := \{Ob_1^i, \ldots, Ob_{n_i}^i\} \tag{3.1}$$

where n_i denotes the number of the involved obstacles. □

Definition 3.2. *Let $\mathcal{O}^i(t)$ be an obstacle scenario. Then the non-convex obstacle-free region pertaining to $\mathcal{O}^i$ is identified as follows*

$$\mathcal{O}_{free}^i(t) := \{x \in \mathbb{R}^n : \; h_i(x) > 0_{n_f}\} \tag{3.2}$$

where $h_i : \mathbb{R}^n \to \mathbb{R}^{n_f}$ and n_f is the number of component-wise inequalities. □

Assumption 3.3. *Assume without loss of generality that each Ob_j^i has a polyhedral convex structure described as the intersection of l_j half-spaces:*

$$Ob_j^i : \begin{bmatrix} (H_j^i)_1^T \\ \vdots \\ (H_j^i)_{l_j}^T \end{bmatrix} p \leq \begin{bmatrix} (g_j^i)_1 \\ \vdots \\ (g_j^i)_{l_j} \end{bmatrix} \tag{3.3}$$

where $p := B\,x \in \mathbb{R}^2$ are the planar components of the state space $x \in \mathbb{R}^n$ and $B \in \mathbb{R}^{2\times n}$ a projection matrix.

Remark 3.4. Notice that Assumption 3.3 is not restrictive because if an obstacle has not a polygonal description, it is possible to recast it into a convex polyhedral from by resorting to adequate outer approximations, see e.g. the numerical procedure outlined in [119]. □

Assumption 3.5. *The autonomous vehicles operate within dynamic environments where possibly moving obstacles (agents) rely and give rise to uncertain obstacle scenarios (configurations) according to the following behavior rules:*

- *each agent can occupy a finite set of pre-specified positions ;*
- *time instants at which an obstacle scenario changes its location are unknown.*

Remark 3.6. Note that Assumption 3.5 mainly restrict the admissible dynamic environments pointing out the attention on a particular class of *partially known* working areas where a finite and known set of obstacle scenarios $\mathcal{O}^i$ may occur, e.g. production/assembly lines [17], restricted areas (buildings or rooms) where surveillance or housekeeping are required tasks [38] and so on.

3.1.1 Obstacle Avoidance Motion Planning (*OAMP*) Problem

Given an autonomous vehicle described by 2.9 and a set of l obstacle scenarios $\mathcal{O}^i$, $i = 1, \ldots, l$, determine a state-feedback control policy

$$u(t) = g(x(t)) \tag{3.4}$$

compatible with (2.2)-(2.3) and (3.2), such that starting from an initial condition $x(0)$ the robot trajectory $x(t)$ is driven to a target position x_f regardless of any admissible obstacle scenario occurrence (3.1). □

In the sequel, the problem will be addressed by means of a receding horizon control approach based on set-theoretic concepts presented in Section 2.2. In order to exploit such approach for the ***OAMP*** problem, the following key questions must be analyzed in depth:

- How can the nonconvex obstacle constraints be recast into computationally tractable conditions?
- How can one define a sequence of one-step controllable sets such that there exists at least a feasible path ($x(0) \rightarrow x_f$) complying with obstacle avoidance purposes?

A solution to both these technical issues is provided in the next section.

3.2 A set-theoretic approach

The aim of this section is to provide a set-theoretic based solution to the ***OAMP*** problem.

Notice that the *basic* ellipsoidal MPC schema presented in Section 2.2.1 can be extended to the proposed framework if the following key aspects are concerned:

(a) Given an obstacle scenario $\mathcal{O}^i$, determine a sequence of one-step controllable sets $\{\mathcal{T}_k^i\}_{k=0}^{N_i}$ such that there exists at least a feasible path to the goal x_f;
(b) Given the sequences $\{\mathcal{T}_k^i\}_{k=0}^{N_i},\ i = 1,\dots,l$, ensure viability properties under time-varying scenarios.

The next developments are devoted to translate these ideas into tractable conditions and computational algorithms.

3.2.1 One step controllable sets

Point **(a)** relies on the estimation of the domain of attraction (DoA), i.e. all the initial conditions $x(0)$ for which there exists an admissible path to x_f. Note that in fact the *basic* set construction (2.11) of $\left\{\mathcal{T}_k^i\right\}_{k=0}^{N_i}$ may give rise to "small" DoA estimates, i.e. $\bigcup_{i=1}^{l} \mathcal{T}_{N_i}^i$, because in presence of the constraints (2.2),(2.3) and (3.2) a saturation effect may occur on the one-step controllable sets growth. To overcome such a drawback, here the following algorithm is proposed:

---

*One-step Controllable Set Procedure (**OCSP**)*

1: Given the goal x_f and chosen the initial condition $x(0)$ as follows

$$x(0) := \arg\max_{x \in \mathcal{X}} \|x - x_f\|_2$$

design the pair $(K_0, \mathcal{T}_0)$, with $\mathcal{T}_0$ a RPI region centered in x_f where K_0 is the stabilizing state-feedback gain complying with the constraints (2.2), (2.3), (3.2). Store the subscript $m = 0$ into an index vector hereafter named IR^i. Let $\mathcal{T}_0^i = \mathcal{T}_0$ and $x_{eq}^0 = x_f$ be the initial terminal region and equilibrium point, respectively;

2: Derive the sequence $\{\mathcal{T}_k^i\}_{k=1}^{N_m}$ by using recursions (2.11) under the additional state constrain (3.2). The integer N_m is the saturation level for the region growth;

3: Store the index N_m into an index vector denoted as LR^i;

*4: **if** $x(0) \notin \mathcal{T}_{N_m}^i$, **then***

*5: **if** there exists a candidate equilibrium x_{eq}^{m+1} such that:*

a) $x_{eq}^{m+1} \neq x_{eq}^{s}$, $s = 0, 1, \ldots, m$;
b) $x_{eq}^{m+1} := \arg \min_{x_{eq} \in \mathcal{T}_{N_m}^i} \|x_{eq} - x(0)\|_2$

then
Design a new pair $(K_{m+1}, \mathcal{T}_{N_m+1}^i)$, *with* $\mathcal{T}_{N_m+1}^i$ *centered in* x_{eq}^{m+1} *and* K_{m+1} *satisfying (2.2), (2.3), (3.2). Store the corresponding index* $N_m + 1$ *into* IR^i;
Add $\{\mathcal{T}_k^i\}_{k=1}^{N_m}$ *to the previous computed sequence;*
$m \leftarrow m + 1$, $\mathcal{T}_0^i := \mathcal{T}_{N_m+1}^i$ *and goto Step 2;*
else
Stop;
end if
else
Stop;
end if

A graphical description of the ***OCSP*** *modus operandi* is given in Fig. 3.1.

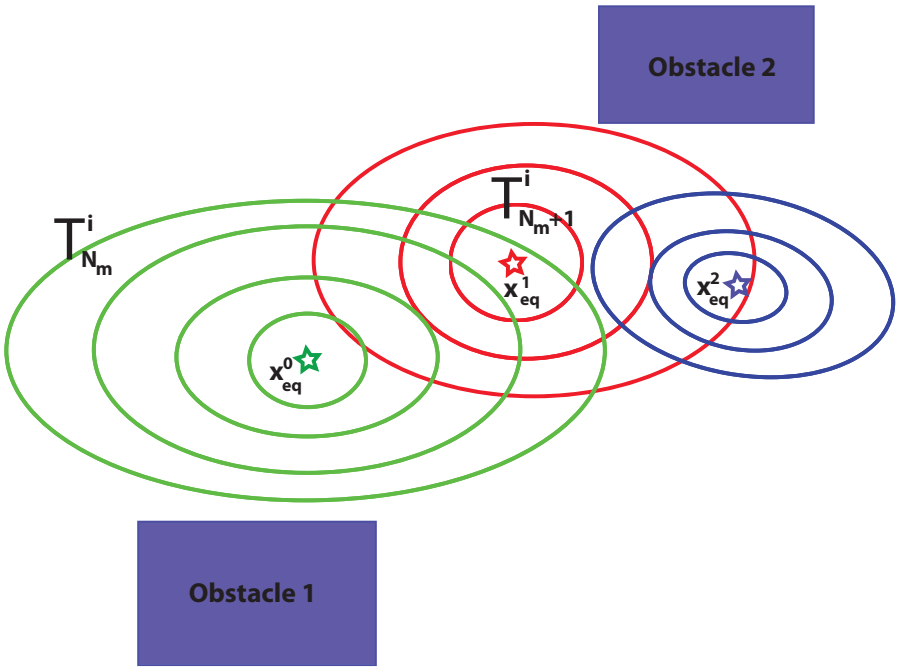

Fig. 3.1. Controllable set sequences construction

Remark 3.7. Notice that

- the vector IR^i keeps track of all the robust positively invariant regions obtained by ***OCSP*** and its relevance will appear clear in Section 3.3 where the receding horizon scheme is detailed;
- the index vector LR^i stores all the "saturation regions" for each subsequence of $\{\mathcal{T}_k^i\}$ whose aim is to ensure collision avoidance (see the discussion pertaining to the formulas (3.13)-(3.17));
- the pairs $(K_m, \mathcal{T}_m^i),\ m \in IR^i$ are computed via standard LMI techniques, see [64] for computational details. □

3.2.2 Obstacle constraints convexification

Here, the non-convex constraints describing the admissibility regions (3.2) are translated into computationally tractable requirements. This is achieved via the following arguments:

- by exploiting the polyhedral structure (3.3) the obstacle-free region (3.2) can be described as

$$\mathcal{O}_{free}^i := \bigcap_{j=1}^{n_i} \left\{ x \in \mathbb{R}^n : \bigcup_{s=1}^{l_j} \left\{ (H_j^i)_s^T Bx > (g_j^i)_s \right\} \right\}, \tag{3.5}$$

- by denoting as

$$(c_j^i)^a := \left\{ x \in \mathbb{R}^n : \bigcup_{s=1}^{l_j} \left\{ (H_j^i)_s^T Bx > (g_j^i)_s \right\} \right\} \tag{3.6}$$

the *active* constraint region pertaining to each $j-th$ obstacle of $\mathcal{O}^i$ and by using the fact that the construction of controllable set families is based on the equilibrium x_{eq}, (see Fig. 3.2) the following statement holds true:

Statement 3.8. *Let $\mathcal{T}_k^i$ be a one-step controllable set, the computation of the predecessor set $\mathcal{T}_{k+1}^i$ via (2.11) requires that at most two amongst all the constraints $(H_j^i)_s^T Bx > (g_j^i)_s$ are imposed for each $j-th$ obstacle in order to define $(c_j^i)^a$.*

Then, the region (3.5) can be convexified by means of the following algorithm:

---

*Convexification procedure (**CP**)*

---

*0. Let x_{eq}, $Ob_j^i\ j = 1 \dots n_i$, and $\mathcal{B}_\delta$ be the equilibrium point used by the **OCSP** procedure, the obstacles of the $i-th$ scenario and a ball of radius $\delta > 0$, respectively;*

1. *For each Ob_j^i, compute*

$$(p_j^i)^{min} = \arg\min_{p \in Ob_j^i} \{dist(p, Bx_{eq})\}$$

and the hyperplane $(h_j^i)^1 := \left\{(H_j^i)_{s_1}^T Bx = (g_j^i)_{s_1}\right\}$ such that $(p_j^i)^{min} \in (h_j^i)^1$;

2. *For each Ob_j^i, determine the closest hyperplane*

$$(h_j^i)^2 := \left\{(H_i^j)_{s_2}^T Bx = (g_j^i)_{s_2}\right\},\ (h_j^i)^2 \neq (h_j^i)^1,$$

to the point $(p_j^i)^{min}$;

3. *For each Ob_j^i, define the "external" half-spaces as*

$$(sp_j^i)^1 := \left\{(H_j^i)_{s_1}^T Bx > (g_j^i)_{s_1}\right\}$$

$$(sp_j^i)^2 := \left\{(H_j^i)_{s_2}^T Bx > (g_j^i)_{s_2}\right\}$$

4. *For each Ob_j^i*
   a) *if $Bx_{eq} \in ((sp_j^i)^1 \cap (sp_j^i)^2 \sim B_\delta)$, then $(c_j^i)^a = (sp_j^i)^1 \cap (sp_j^i)^2$;*
   b) *else if $Bx_{eq} \in ((sp_j^i)^1 \cap (sp_j^i)^2)$ and $d(Bx_{eq}, (sp_j^i)^1) < \delta$, then $(c_j^i)^a = (sp_j^i)^2$;*
   c) *else if $Bx_{eq} \in ((sp_j^i)^1 \cap (sp_j^i)^2)$ and $d(Bx_{eq}, (sp_j^i)^2) < \delta$, then $(c_j^i)^a = (sp_j^i)^1$;*
   d) *else $(c_j^i)^a = (sp_j^i)^1$;*

5.

$$(c^i)^a = (c_1^i)^a \cap (c_2^i)^a \cap \ldots \cap (c_{n_i}^i)^a = \bigcap_{j=1}^{n_i} \left\{x \in \mathbb{R}^n \mid (H_j^i)^a Bx > (g_j^i)^a\right\} \quad (3.7)$$

Remark 3.9. Note that the *Step 4* allows to achieve a convex relaxation of the union set operator in (3.5) whose rationale directly comes out from Statement 3.8 arguments, while the *Step 5* computes the convexification of the whole region (3.5). Therefore, for each $\mathcal{O}^i$ the following additional geometric constraints must be imposed

$$(H_j^i)^a Bx > (g_j^i)^a,\ j = 1, \ldots, n_i. \quad (3.8)$$

Moreover, the vanishing tolerance level δ is instrumental to univocally identify the half-spaces to which the equilibrium x_{eq} belongs and, as a consequence, the constraints to be imposed at the *Step 2* of the ***OCSP*** in place of (3.2). □

Finally, Fig. 3.2 provides a graphical illustration of the conditions outlined in *Step 4*. There by referring to the obstacle labelled with Ob_1, the point Bx_{eq} belongs to the intersection of the two half-spaces characterized by $(h_1^i)^1$ and $(h_1^i)^2$ under the restriction imposed by the tolerance level δ. This matches the prescriptions of the case $4.a$), therefore the constraints to impose are

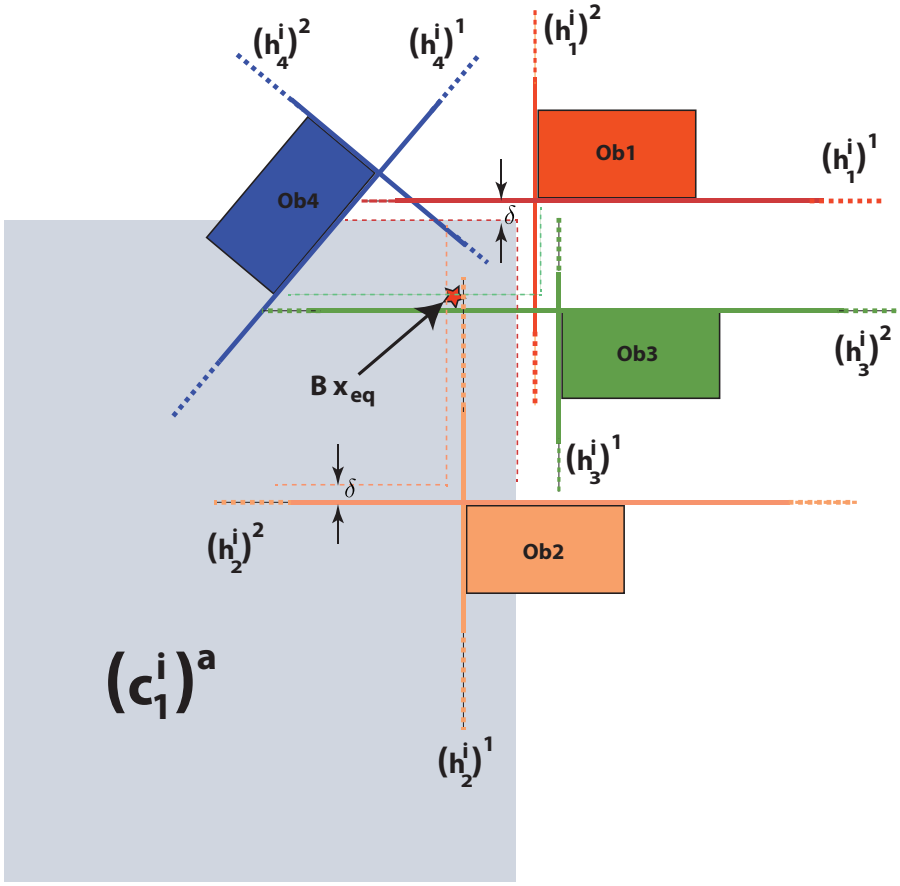

Fig. 3.2. Convexification procedure description

$$(H_1^i)_{s_1} Bx > (g_1^i)_{s_1}$$
$$(H_1^i)_{s_2} Bx > (g_1^i)_{s_2}$$

and the grey zone in Fig. 3.2 describes the convexified region $(c_1^i)^a$. A similar reasoning applies to the other obstacles.

3.2.3 Time-varying obstacle scenario occurrences

The point **(b)** focuses on the difficulties arising when unpredictable obstacle scenario changes occur. Since the l controllable set sequences $\{\mathcal{T}_k^i\}$, $i = 1, \ldots, l$, are computed under the hypothesis that a single obstacle scenario $\mathcal{O}^i$ takes place, the viability retention cannot be ensured when $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$ because a switching to a different set sequence must be imposed.

Here, the idea is to design two further families of one-step controllable sets, hereafter named *Obstacle* and *Scenario Switching* sequences, whose combined use allows to *safely* switch to $\{\mathcal{T}_k^{i'}\}$.

Obstacle controllable sequences

The *Obstacle* sequences $\{\mathcal{T}_k^{Ob_j^{i'}}\}$, $j = 1, \ldots, n_{i'}$, have the aim to encircle the corresponding obstacles $Ob_j^{i'}$, , $j = 1, \ldots, n_{i'}$ and they are introduced because of the following argument:

Statement 3.10. *Let $\mathcal{O}^i$ be the current scenario, if at time $\bar{t}$ the obstacle scenario change $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$ occurs and the current state $x(\bar{t}) \notin \mathcal{T}_k^{i'}$, for some k, then collisions could happen because the sequence $\{\mathcal{T}_k^i\}$ is no longer admissible.*

In order to avoid such an undesired event, the *Obstacle* sets are designed such that when $x(\hat{t}) \in \{\mathcal{T}_k^{Ob_j^{i'}}\}, \hat{t} \geq \bar{t}$, the trajectory $x(t)$, $\forall t \geq \hat{t}$ remains confined into them until the switching to the correct sequence $\{\mathcal{T}_k^{i'}\}$ is made admissible.

The computation of the sequences $\{\mathcal{T}_k^{Ob_j^{i'}}\}$, $j = 1, \ldots, n_{i'}$, is performed by using the ***OCSP*** procedure where both $x(0)$ and x_f are substituted with the equilibria $x_{eq}^{Ob_j^{i'}}$, $j = 1, \ldots, n_{i'}$. These equilibrium points are selected as those at the maximum distance from the obstacles $Ob_j^{i'}$, $j = 1, \ldots, n_{i'}$, and satisfying the following condition:

$$dist(Bx_{eq}^{Ob_j^{i'}}, Ob_j^{i'}) < dist(\mathcal{T}_{max_j}^i, Ob_j^{i'}),\ j = 1, \ldots, n_{i'}, \tag{3.9}$$

where $\mathcal{T}_{max_j}^i$ are the sets corresponding to the greatest indices s of $\{\mathcal{T}_s^i\}$ such that

$$\mathcal{T}_{max_j}^i \cap Ob_j^{i'} = \emptyset,\ j = 1, \ldots, n_{i'} \tag{3.10}$$

For the sake of clarity, an example of the above reasoning is given in Fig. 3.3.

Controllable sequences in a Scenario switching

Although the *Obstacle* sequences have the important merit to avoid collisions when the scenario $\mathcal{O}^{i'}$ occurs, their use leads to the following drawback:

Statement 3.11. *Let $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$ be a generic scenario change occurrence, once the trajectory $x(\cdot)$ enters inside the region defined by $\{\mathcal{T}_s^{Ob_j^{i'}}\}$, it is by construction driven to the terminal region $\mathcal{T}_0^{Ob_j^{i'}}$ centered at the equilibrium $x_{eq}^{Ob_j^{i'}}$ and it will be there confined.*

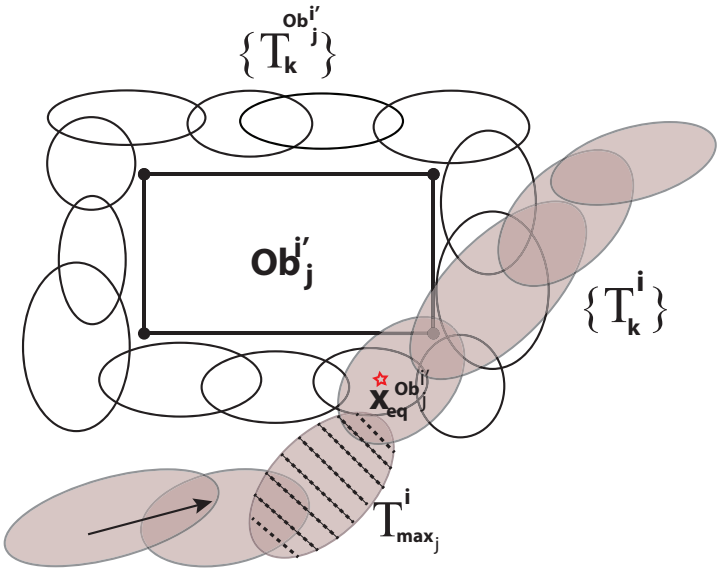

Fig. 3.3. Obstacle sequence construction. The dashed ellipsoid is $\mathcal{T}^i_{max_j}$. The arrow denotes the increasing direction of the subscript k.

This situation clearly affects any vehicle chance to reach the target x_f and for this reason the trajectory $x(\cdot)$ must be driven towards the controllable sequence $\{\mathcal{T}_k^{i'}\}$. To this aim, the *Scenario Switching* sequences $\{\mathcal{T}_k^{SW_j^{i'}}\}$, $j = 1, \ldots, n_i$, are introduced and designed as follows:

1. *for any admissible scenario change* $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$, *choose an equilibrium* $x_{eq}^{SW_j^{i'}}$ *belonging to some* $\mathcal{T}_k^{i'}$;
2. *apply the* ***OCSP*** *with* $x_{eq}^{Ob_j^{i'}}$ *in place of* $x(0)$ *and* $x_{eq}^{SW_j^{i'}}$ *in place of* x_f.

Fig. 3.4 provides a graphical interpretation of the generation of *Scenario Switching* sequences. Then, the switching $\{\mathcal{T}_k^i\} \rightarrow \{\mathcal{T}_k^{i'}\}$ is accomplished as follows:

- *Let* $x(\hat{t}) \in \{\mathcal{T}_k^{Ob_j^{i'}}\}$ *be the current state. As soon as*

$$x(t) \in \mathcal{T}_k^{SW_j^{i'}}, \text{ for some } k \text{ and for some } t \geq \hat{t}, \tag{3.11}$$

  the set-membership to $\{\mathcal{T}_k^{SW_j^{i'}}\}$ *is considered;*
- *As soon as*

$$x(\bar{t}) \in \mathcal{T}_k^{i'}, \text{ for some } k \text{ and for some } \bar{t} \geq \hat{t}. \tag{3.12}$$

  the set-membership to $\{\mathcal{T}_k^{i'}\}$ *can be used.*

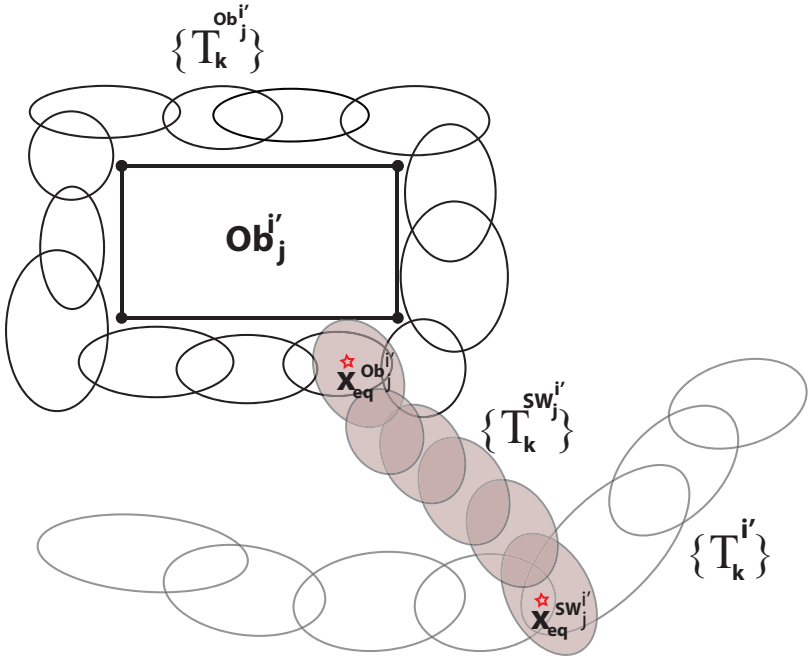

Fig. 3.4. Scenario Switching sequence construction

A relevant aspect left out in the above developments is to guarantee that under a scenario change the robot does not bump on any obstacle during the switching phase. To this end, the sequences $\{\mathcal{T}_k^i\}$ and $\{\mathcal{T}_k^{SW_j^i}\}$ are derived under the following further condition:

$$\|B(x^+ - x)\| \leq \epsilon, \tag{3.13}$$

where $x^+ = \Phi x + Gu$ is the disturbance-free one-step evolution. The inequality (3.13) limits the maximum displacement of the one-step state evolution so that obstacle constraints are satisfied with a tolerance level $\epsilon > 0$. The scalar ϵ can be determined by exploiting the *Obstacle* sequences properties as below detailed:

- *for each scenario $\mathcal{O}^i$ and for each $j-th$ obstacle, consider the sequence $\{\mathcal{T}_k^{Ob_j^i}\}$ and determine the sequence $\{\mathcal{T}_m^{IN_j^i}\}$ as follows:*

$$\begin{cases} \mathcal{T}_m^{IN_j^i} := & \mathcal{T}_{LR_j^i(m)}^{Ob_j^i} \bigcap \mathcal{T}_{LR_j^i(m+1)}^{Ob_j^i}, \ m = 1, \ldots, \dim(LR_j^i) - 1 \\ \mathcal{T}_{\dim(LR_j^i)}^{IN_j^i} := \mathcal{T}_{LR_j^i(1)}^{Ob_j^i} \bigcap & \mathcal{T}_{LR_j^i(\dim(LR_j^i))}^{Ob_j^i} \end{cases} \tag{3.14}$$

  where the index vectors LR_j^i store the saturation regions for each sequence $\{\mathcal{T}_k^{Ob_j^i}\}$;

- *compute the minimum d_j^i and maximum D_j^i euclidean distances between the $j-th$ obstacle (polyhedron) and the indexed set sequence $\{\mathcal{T}_m^{IN_j^i}\}$;*
- *compute*

$$d_{min} := \max_{\substack{i=1,\dots,l,\\ j=1,\dots,p_i}} d_j^i \tag{3.15}$$

$$D_{max} := \min_{\substack{i=1,\dots,l,\\ j=1,\dots,p_i}} D_j^i \tag{3.16}$$

$$\epsilon := D_{max} - d_{min} \tag{3.17}$$

An illustration of formulas (3.15)-(3.16) is provided in Fig. 3.5.

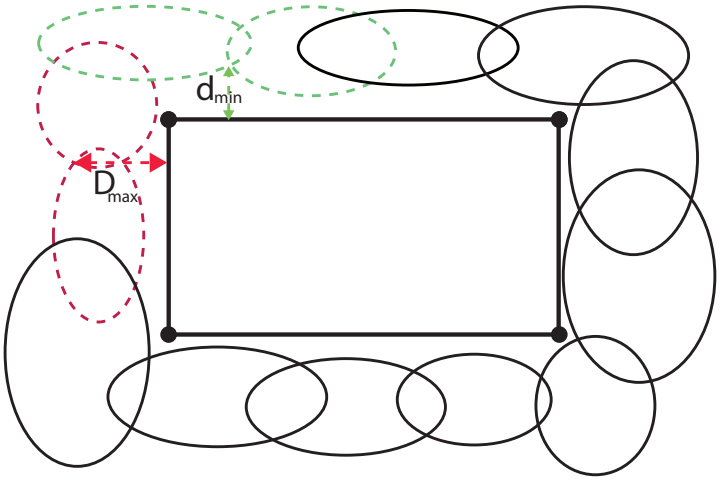

Fig. 3.5. Tolerance level ϵ computation

Remark 3.12. The shortest distance d_j^i between two convex sets can be obtained by using the algorithm proposed in [29], whereas an efficient procedure to estimate the maximum distance D_j^i is given in [74] where Hausdorff distance arguments are exploited. □

One-step controllable families computation

The consequence of all the above developments is that the set sequences must be computed by using the following scheme:

*Families Construction Procedure (**FCP**)*

0. *Let* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *be the oriented graph with* $\mathbf{V}$ *the* l *obstacle scenarios and* $\mathbf{E}$ *the set of ordered pairs* $\{i, i'\}, i, i' \in \mathbf{V}$, *such that the switching* $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$ *is admissible;*
1. *Generate the sequences* $\{\mathcal{T}_k^i\}$, $i = 1, \ldots, l$;
2. *Generate the Obstacle families* $\{\mathcal{T}_k^{Ob_j^i}\}$, $\forall i \in \mathbf{V}_r$, $j = 1, \ldots, n_i$, *according to (3.9)-(3.10);*
3. *Estimate the tolerance level* ϵ *by (3.15)-(3.17);*
4. *Update the sequences* $\{\mathcal{T}_k^i\}$, $i = 1, \ldots, l$ *under the additional constraint (3.13);*
5. *Generate the sequences* $\{\mathcal{T}_k^{SW_j^i}\}$, $\forall i \in \mathbf{V}_r$, $j = 1, \ldots, n_i$ *under the additional constraint (3.13).*

Finally, Fig. 3.6 depicts a sketch of the switching *modus operandi* under an obstacle scenario change.

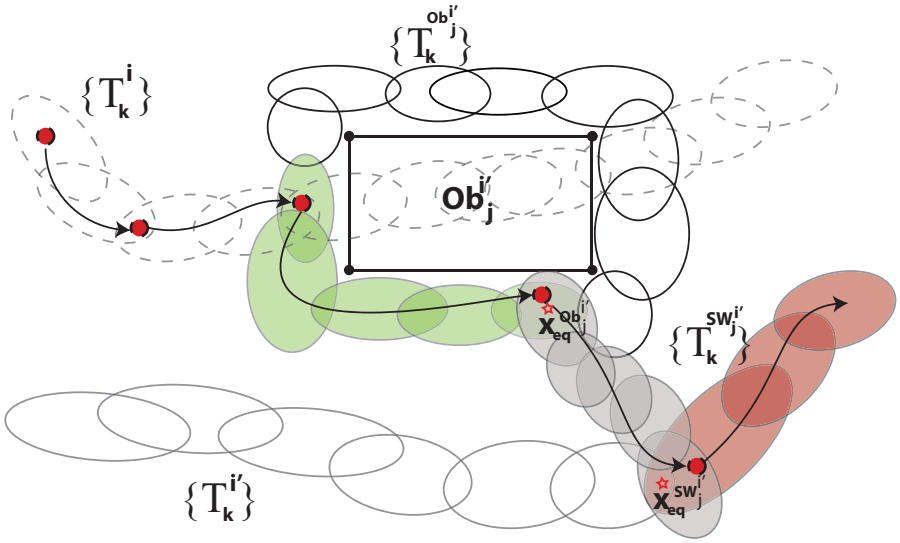

Fig. 3.6. Obstacle scenario change: $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$: $\mathcal{O}^i$ sequence (white), Obstacle sequence (green), Scenario Switching sequence (grey), $\mathcal{O}^{i'}$ sequence (red)

3.3 RHC algorithm

In this section a Receding Horizon Control strategy is outlined by collecting all the above developments. In the sequel the following further assumptions are made:

Assumption 3.13. *At each time instant t the robot is informed about the current obstacle scenario, i.e. $sc(t)$.* □

Assumption 3.14. *At each time instant t, an obstacle scenario change $\mathcal{O}^i \rightarrow \mathcal{O}^{i'}$ can occur if the actual robot position $x_p(t) = B\bar{x}(t)$ is such that:*

$$\bar{x}(t) \notin \bigcup_{j=1}^{n_{i'}} \{x \in \mathbb{R}^n \mid H_j^{i'} Bx \leq g_j^{i'} + d_{min}\} \tag{3.18}$$

□

Remark 3.15. The rationale behind Assumption 3.14 is to require that an obstacle configuration change can occur only when the current robot position does not lie in the region pertaining to the new obstacle scenario $\mathcal{O}^{i'}$.

---

*Obstacle Avoidance MPC (**OA-MPC**) Algorithm*

Off-line:

1 Given the obstacle scenarios $\mathcal{O}^i$, $i = 1, \ldots, l$, the initial condition $x(0)$ and the goal x_f, compute the non-empty robust invariant ellipsoidal region $\mathcal{T}_0$ and the stabilizing feedback gain K_0 complying with the constraints (2.2), (2.3), (3.8);

*2 Apply the **FCP** scheme in order to generate the sequences $\{\mathcal{T}_k^i\}$, $i = 1, \ldots, l$, $\{\mathcal{T}_k^{Ob_j^i}\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$, and $\{\mathcal{T}_k^{SW_j^i}\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$ such that*

$$x(0) \in \left(\bigcup_{\substack{i=1,\ldots,l \\ k}} \mathcal{T}_k^i \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\ldots,n_i, \\ k}} \mathcal{T}_k^{Ob_j^i} \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\ldots,n_i, \\ k}} \mathcal{T}_k^{SW_j^i} \right)$$

3 Store the ellipsoidal sequences.

On-line:

*1: **if***

$$x(t) \in \left(\bigcup_{k} \mathcal{T}_k^{sc(t)} \right) \cup \left(\bigcup_{\substack{j=1,\ldots,n_i, \\ k}} \mathcal{T}_k^{Ob_j^{sc(t)}} \right) \cup \left(\bigcup_{\substack{j=1,\ldots,n_i, \\ k}} \mathcal{T}_k^{SW_j^{sc(t)}} \right)$$

then $curr := sc(t)$

*2: **else*** $curr := prec$;

*3: **end if***

*4: **if*** $x(t) \in \bigcup_k \mathcal{T}_k^{curr}$ ***then***

5: *goto Step 10 by considering $\mathcal{T}_k^{curr}$ as the candidate sequence, i.e. $\mathcal{T}_k^{candidate}$.*

6: ***end if***

7: ***if*** $x(t) \in \bigcup_{\substack{j=1,\dots,n_i,\\ k}} \mathcal{T}_k^{SW_j^{curr}}$ ***then*** $\mathcal{T}_k^{candidate} := \mathcal{T}_k^{SW_j^{curr}}$

8: ***else*** $\mathcal{T}_k^{candidate} := \mathcal{T}_k^{Ob_j^{curr}}$

9: ***end if***

10: *Find* $k(t) := \min_k \{x(t) \in \mathcal{T}_k^{candidate}\}$

11: ***if*** $k(t) \in IR^{candidate}$ ***then*** $u(t) = K_{k(t)}x(t)$

12: ***else***

13: ***if*** *curr==sc(t)* ***then***

$$u(t) = \arg\min J_{k(t)}(x(t), u) \quad s.t. \tag{3.19}$$

$$\Phi x(t) + Gu \in \tilde{\mathcal{T}}_{k(t)-1}^{candidate},\ u \in \mathcal{U} \tag{3.20}$$

14: ***else***

$$u(t) = \arg\min J_{k(t)}(x(t), u) \quad s.t. \tag{3.21}$$

$$\|B(\Phi x(t) + Gu - x(t))\|_2^2 \leq \epsilon^2 \tag{3.22}$$

$$\Phi x(t) + Gu \in \tilde{\mathcal{T}}_{k(t)-1}^{candidate},\ u \in \mathcal{U} \tag{3.23}$$

15: ***end if***

16: ***end if***

17: *Apply* $u(t)$; *prec:=curr;* $t := t + 1$*; goto Step 1*

Notice that as in (2.12)-(2.13) the optimization problems (3.19)-(3.20) and (3.21)-(3.23) are independent of $J_{k(t)}(x, u)$. A typical choice for autonomous vehicle is

$$J_{k(t)}(x, u) = \|\Phi x(t) + Gu\|^2_{\left(\tilde{P}_{k(t)-1}^{candidate}\right)^{-1}}$$

where $\|\Phi x(t) + Gu\|^2_{\left(\tilde{P}_{k(t)-1}^{candidate}\right)^{-1}}$ characterizes the one-step ahead state prediction with $\tilde{P}_{k(t)-1}^{candidate}$ the shaping matrix of the ellipsoidal region

$$\tilde{\mathcal{T}}_{k(t)-1}^{candidate} := \left\{ x \in \mathbb{R}^n : (x - x_{eq}^{k(t)-1})^T \left(P_{k(t)-1}^{candidate}\right)^{-1} (x - x_{eq}^{k(t)-1}) \leq 1 \right\}$$

and $x_{eq}^{k(t)-1}$ the center of $\tilde{\mathcal{T}}_{k(t)-1}^{candidate}$. This choice is used to impose a decrease of the membership index $k(t)$ as fast as possible and to consequently minimize the time necessary to reach the goal location.

Remark 3.16. It is worth to underline that in Step 13 the computation of the current command input $u(t)$ is achieved without imposing the one-step constraint (3.13) as done in Step 14. In fact, note that if the robot state measurement belongs to the current obstacle scenario $sc(t)$ collisions do not occur, therefore the requirement (3.13) becomes useless. As a consequence, the overall control performance are improved. □

The next proposition shows that the proposed ***OA-MPC*** algorithm enjoys the feasibility retention and closed-loop stability properties.

Proposition 3.17. *Let the sequences of sets* $\left\{\mathcal{T}_k^i\right\}$, $\left\{\mathcal{T}_k^{SW_j^i}\right\}$ *and* $\left\{\mathcal{T}_k^{Ob_j^i}\right\}$ *be non-empty and*

$$x(0) \in \left(\bigcup_{\substack{i=1,\dots,l \\ k}} \mathcal{T}_k^i \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\dots,n_i, \\ k}} \mathcal{T}_k^{Ob_j^i} \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\dots,n_i, \\ k}} \mathcal{T}_k^{SW_j^i} \right)$$

Then, the ***OA-MPC*** *algorithm always satisfies the constraints and ensures Uniformly Ultimate Boundedness for all time-varying occurrences of* $\mathcal{O}^i$, $i = 1, \dots, l$.

Proof. Note that existence of solutions at time t implies existence of solutions at time $t+1$, because the optimization problems in Steps 13 and 14 are always feasible. In fact, by construction there exists an input vector u satisfying the constraints (2.2), (2.3), (3.8) and (3.22) such that the set-membership requirement in (3.20) holds true. Then thanks to the ***FCP*** procedure and under the additional constraint constraints (3.13), at the next time instant $t+1$ the existence of a solution $u(t+1)$ for the Step 13 or 14 is ensured.

Finally, Uniformly Ultimate Boundedness of the strategy follows by noting that the trajectory is in the worst case confined within to the following union of sets:

$$\left(\bigcup_{\substack{i=1,\dots,l \\ k}} \mathcal{T}_k^i \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\dots,n_i, \\ k}} \mathcal{T}_k^{Ob_j^i} \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\dots,n_i, \\ k}} \mathcal{T}_k^{SW_j^i} \right)$$

□

3.4 Extension to nonlinear vehicle models

Since the autonomous ground vehicles exhibit highly nonlinear behaviours, a linear kinematic description may result too simple for both analysis and control aims. Most of the literature contributions makes use of the well-known feedback linearization approach in order to design trajectory tracking controllers (Section 8.5.2 in [113]):

one of the main advantages is that a linear model for UGVs can be used. Nonetheless, *direct* nonlinear control schemes are by no means useful when the involved nonlinearities are not mild or the model uncertainty prevents the exploitation of feedback linearization technicalities, see [].

In view of this reasoning, the aim of this section is to derive proper characterizations of the set-theoretic MPC scheme (the ***OA-MPC*** algorithm) for both uncertain polytopic and polynomial models by preserving all the properties (Proposition 3.17) pertaining to the LTI case.

3.4.1 Robust polytopic uncertain models

Consider autonomous vehicles described by the following polytopic family of discrete-time linear systems [14]

$$x(t+1) = \Phi(\mathbf{p}(t))x(t) + G(\mathbf{p}(t))u(t) + G_d(\mathbf{p})wd(t) \tag{3.24}$$

where $\mathbf{p}(t) \in \mathbb{R}^{n_\mathbf{p}}$ is an unknown set-membership, possibly time-varying, parameter vector. In (3.24) the system matrices $\Phi(\mathbf{p}), G(\mathbf{p})$ and $G_w(\mathbf{p})$ belong to the polytopic matrix family

$$\Omega(\mathcal{P}) := \left\{ (\Phi(\mathbf{p}), G(\mathbf{p}), G_d(\mathbf{p})) = \sum_{\gamma=1}^{n_\mathbf{p}} p_\gamma (\Phi_\gamma, G_\gamma, G_{d_\gamma}) \right\}$$

where the triple $(\Phi_\gamma, G_\gamma, G_{d_\gamma})$ denotes the vertices of the polytope $\Omega(\mathcal{P})$, viz. $(\Phi_\gamma, G_\gamma, G_{d_\gamma}) \in \text{vert}\{\Omega(\mathcal{P})\}$, $\forall \gamma = 1, \ldots, n_\mathbf{p}$. The vector parameter $\mathbf{p} = [p_1, \ldots, p_{n_\mathbf{p}}]$ is assumed to belong to the unit simplex

$$\mathcal{P} := \left\{ \mathbf{p} \in \mathbb{R}^{n_\mathbf{p}} : \sum_{\gamma=1}^{n_\mathbf{p}} p_\gamma = 1,\, p_\gamma \geq 0 \right\}$$

Remark 3.18. By considering the UGV nonlinear model (2.1), a multi-model state space description can be achieved by resorting to the following ideas proposed in [64]:

1. *Operating points approximation:* suppose that for (2.1) input/output data sets at different operating points are available. From each data set, a number of linear models (involving the same state vector x) are derived. Then it is reasonable to assume that any analysis and design methods for the polytopic system (3.24) with vertices given by the linear models will apply to the real system;
2. *Polytopic embedding:* suppose that the Jacobian $[\partial f/\partial x \; \partial f/\partial u]$ of (2.1) is known to lie in the polytope $\Omega(\mathcal{P})$. Then it can be shown that every trajectory (x, u) of the original nonlinear system is also a trajectory of (2.1) for some linear time varying (LTV) system in $\Omega(\mathcal{P})$ [73]. Thus the original nonlinear system can be approximated (possibly conservatively) by a polytopic uncertain LTV system.

□

In order to adapt the ***OA-MPC*** algorithm to the description (3.24), the following two technical aspects have to be revisited:

(a) *One-step ahead controllable sets approximation.* A new ellipsoidal approximation of each element of the one step controllable family $\{\Xi_k\}$ is required to comply with polytopic models.
(b) *On-line **OA-MPC** algorithm.* On the basis of the new one-step controllable set construction, the optimization problem in Steps 13 and 14 must be accordingly modified.

(a) One-step ahead controllable set approximation

The ellipsoidal set computation of Section 2.2.1 is extended to take care of the multi-model state space description (3.24).
Consider the following set manipulations:

$$\begin{aligned}
&\{\ x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D},\, \forall \gamma = 1, \dots n_p,\ \Phi_\gamma x + G_\gamma u + G_d d \in \mathcal{T}_{k-1}\} \\
&\quad \supset \{x \in \mathcal{X} :, \exists\, u \in \mathcal{U} : \forall \gamma = 1, \dots n_p,\ \Phi_\gamma x + G_\gamma u + G_d d \in In[\mathcal{T}_{k-1} \sim G_d \mathcal{D}]\} \\
&\quad = Proj_x\{[x,u] : u \in \mathcal{U}, x \in \mathcal{X} \text{ and } \forall \gamma = 1, \dots n_p,\ [x,u] \in \tilde{\mathcal{T}}_{k-1}^{\gamma}\}
\end{aligned} \tag{3.25}$$

where the ellipsoidal set $\tilde{\mathcal{T}}_{k-1}^{\gamma}$ is defined as follows in the extended space x, u :

$$\tilde{\mathcal{T}}_{k-1}^{\gamma} := \{[x,u] : \forall \gamma = 1, \dots, n_p,\ \Phi_\gamma x + G_\gamma u \in In[\mathcal{T}_{k-1} \sim G_d \mathcal{D}]\}.$$

Then, the resulting ellipsoidal one-step ahead set approximation is

$$\mathcal{T}_k := Proj_x \left[In \left[\bigcap_{\gamma=1}^{n_p} \tilde{\mathcal{T}}_{k-1}^{\gamma} \bigcap \left(\bigcap \mathcal{T}_h^{\mathcal{X}} \times \bigcap \mathcal{T}_k^{\mathcal{U}} \right) \right] \right] \tag{3.26}$$

where the projection along x can be computed by solving the LMI optimization problem (A.3).

(b) On-line *OA-MPC* algorithm

The control action computation in Steps 13 and 14 has to be modified according to the polytopic plant structure. In particular, Step 13 becomes

$$u(t) = \arg\min_u \max_\gamma \|\Phi_\gamma x(t) + G_\gamma u\|^2_{\left(\tilde{P}^{candidate}_{k(t)-1}\right)^{-1}} \quad s.t. \tag{3.27}$$

$$\Phi_\gamma x(t) + G_\gamma u \in \tilde{\mathcal{T}}^{candidate}_{k(t)-1},\ u \in \mathcal{U},\ \gamma = 1, \dots n_p \tag{3.28}$$

whereas Step 14 modifies as

$$u(t) = \arg\min_u \max_\gamma \|\Phi_\gamma x(t) + G_\gamma u\|^2_{\left(\tilde{P}^{candidate}_{k(t)-1}\right)^{-1}} \quad s.t. \tag{3.29}$$

$$\|B(\Phi_\gamma x(t) + G_\gamma u - x(t))\|_2^2 \leq \epsilon^2 \tag{3.30}$$

$$\Phi_\gamma x(t) + G_\gamma u \in \tilde{\mathcal{T}}_{k(t)-1}^{candidate},\ u \in \mathcal{U},\ \gamma = 1, \dots n_p \tag{3.31}$$

Notice that the minimum-time cost function assumes the following form:

$$J_{k(t)}(x,u) = \max_\gamma \|\Phi_\gamma x(t) + G_\gamma u\|^2_{\left(\tilde{P}_{k(t)-1}^{candidate}\right)^{-1}},\ \gamma = 1, \dots, n_p$$

As a consequence, a non-convex min-max optimization problem comes out. To overcome such a difficulty, it is possible to exploit the fact that the proposed control framework is irrespective of the chosen cost index: for instance $J_{k(t)}(x(t),u)$ can be modified by referring to any model inside the polytope $\Omega(\mathbf{p})$

$$J_{k(t)}(x,u) = \|\Phi(\mathbf{p})x(t) + G(\mathbf{p})u\|^2_{\left(\tilde{P}_{k(t)-1}^{candidate}\right)^{-1}}$$

Under this choice, the main consequence is that optimizations (3.27)-(3.28) and (3.29)-(3.31) can be recast into semi-definite programming problems (SDPs).

3.4.2 Polynomial models

Polynomial models are a quite general class of non-linear systems able to describe non-linear plants. In [107] it has been proved that any non-linear model can be recast as a polynomial description via a suitable recasting procedure (see e.g. the ***PRA*** procedure described in Section (2.3.4). As it is well-known, the dynamics of autonomous ground vehicles gives rise to non-polynomial models and nonlinear requirements (e.g. nonholonomic constraints) that can be manipulated and recast into a polynomial form, see Appendix B. In this section, it will be shown that the jointly use of set-theoretical arguments and SOS optimization technicalities allow to develop a low-demanding polynomial scheme for solving the proposed ***OAMP*** problem.

Consider the following autonomous vehicles described by polynomial discrete-time systems

$$x(t+1) = f_p(x(t), u(t)) + G_d d(t) \tag{3.32}$$

where $f_p : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^d \rightarrow \mathbb{R}^n$ is an array of multivariate polynomials, $x(t) = [x_a^q(t)^T\ x_a^v(t)^T\ x_b(t)^T]^T \in \mathcal{X} \subset \mathbb{R}^n$ the state with $x_a^q \in \mathbb{R}^q$ and $x_a^v \in \mathbb{R}^v$ accounting for the robot pose and its derivative respectively,

$$d(t) \in \mathcal{D} \subset \mathbb{R}^d,\ \ \mathcal{D} := \{d \in \mathbb{R}^d \,|\, c_d(d) \leq 1\} \tag{3.33}$$

with $c_d(d) \in \mathcal{R}[d]$ a convex and compact polynomial function representing unmodeled dynamics,

$$x(t) \in \mathcal{X},\ \forall t \geq 0,\ \ \mathcal{X} := \{x \in \mathbb{R}^n \,|\, c_x(x) \leq 1\}, \tag{3.34}$$

$$u(t) \in \mathcal{U},\ \forall t \geq 0,\ \ \mathcal{U} := \{u \in \mathbb{R}^m \,|\, c_u(u) \leq 1\}, \tag{3.35}$$

where $c_x(x)$ and $c_u(u)$ are convex and compact polynomial scalar functions belonging to $\mathcal{R}[x]$ and $\mathcal{R}[u]$, respectively.

As already stated the aim of this section is to present an MPC strategy extension for the class of nonlinear systems (3.32) subject to nonholonomic constraints. In particular, the set-theoretic based scheme for LTI models (Sections 3.2-3.3) will be extended to comply with the polynomial plant description (3.32). To this end, the following points deserve an adequate consideration within the polynomial framework:

(a) Computation of controllable set sequences $\{\Xi_k\}$;
(b) Relaxation of the on-line optimization:

$$\begin{array}{l} u(t) = \arg\min J_{k(t)}(x(t), u) \\ \text{subject to} \end{array} \tag{3.36}$$

$$f_p(x(t), u) \in \tilde{\Xi}_{k(t)-1}^{candidate},\ u \in \mathcal{U} \tag{3.37}$$

(a) One-step controllable families: SOS characterization

The set of states $k-$steps controllable to Ξ can be, in principle, computed via the following recursion

$$\begin{array}{l} \Xi_0 := \Xi \\ \Xi_k := \{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D} : f_p(x,u) + G_d d \in \Xi_{k-1}\} \\ \quad = \{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : f_p(x,u) \in \Xi_{k-1} \sim G_d\mathcal{D}\} \\ \quad = \left\{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : f_p(x,u) \in \tilde{\Xi}_{k-1}\right\} \end{array} \tag{3.38}$$

An alternative description of Ξ_k can be given in terms of appropriate polynomial level surface function $V_k(x,u) :\ \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^+$ in the extended space (x,u) [39]

$$\Xi_k := Proj_x \{x \in \mathcal{X},\ u \in \mathcal{U} : V_k(x,u) \leq 1\} \tag{3.39}$$

with $V_k \in \mathcal{P}[x,u]$. As a consequence, the set

$$\{x \in \mathcal{X},\ u \in \mathcal{U} : V_k(x,u) \leq 1\}$$

has the structure of a semi-algebraic subset of the real $n \times m$-dimensional space [91]. An equivalent description of $\tilde{\Xi}_k$ and $\tilde{\Xi}_k$ is

$$\Xi_k := \{x \in \mathbb{R}^n : \Psi_k(x) \leq 1\}$$

$$\tilde{\Xi}_k := \left\{x \in \mathbb{R}^n : \tilde{\Psi}_k(x) \leq 1\right\} = \Xi_k \sim G_d\mathcal{D}$$

with $\Psi_k(\cdot)$, and $\tilde{\Psi}_k(\cdot) \in \mathcal{P}[x]$.
With these premises, assume that the vehicle is subject to nonholonomic constraints of the following form:

$$a_p^T(x_a^q(t))x_a^v(t) = 0,\ p = 1,\dots,n_a,\ \forall t \in \mathbb{Z}_+, \tag{3.40}$$

with n_a the number of nonholonomic constraints and $a_p : \mathbb{R}^q \to \mathbb{R}^v$, $p = 1,\dots,n_a$, non linear functions of the vehicle pose, see [30]. Then, by defining a collection of polynomial functions $\{V_k(x,u)\}$, the sets of states $k-$steps controllable to Ξ_0, complying with constraints (3.35)-(3.34) and (3.40), can be characterized as follows:

$$\Xi_k = \text{Proj}_x\{x \in \mathcal{X} \subseteq \mathbb{R}^n, u \in \mathcal{U} \subseteq \mathbb{R}^m |\, V_k(x,u) \le 1\} \tag{3.41}$$

where $V_k(x,u)$ satisfies the condition

$$\begin{gathered} V_k(x,u) > 0,\ \ \forall u \in \mathbb{R}^m \backslash \{0\},\ \forall x \in \mathbb{R}^n \backslash \{0\} \\ V_k(0,0) = 0 \end{gathered} \tag{3.42}$$

and the following set containment

$$\begin{aligned} &\{x \in \mathbb{R}^n,\, u \in \mathbb{R}^m :\, V_k(x,u) \le 1\} \\ &\bigcap_{n_a} \{x \in \mathbb{R}^n,\, u \in \mathbb{R}^m :\, c_x(x) \le 1\} \bigcap \{x \in \mathbb{R}^n, u \in \mathbb{R}^m :\, c_u(u) \le 1\} \\ &\bigcap_{p=1} \left\{x \in \mathbb{R}^n,\, u \in \mathbb{R}^m : a_p^T(x_q(t))x_v(t) = 0\right\} \\ &\subseteq \left\{x \in \mathbb{R}^n,\, u \in \mathbb{R}^m :\, \tilde{\Psi}_{k-1}(f_p(x,u)) \le 1\right\} \end{aligned} \tag{3.43}$$

Note that condition (3.43) rephrases recursion (3.38) in the extended space (x,u) and (3.42)-(3.43) define a contractive family $\{V_k(x,u)\}$ as required for candidate non-increasing Lyapunov functions [10]. Furthermore (3.43) defines a semi-algebraic sets that can be rewritten according to the P-satz formalism as

$$\left\{ \begin{array}{l|ll} & 1 - V_k(x,u,d) \ge 0 & \\ & 1 - c_x(x) \ge 0 & \\ & 1 - c_u(u) \ge 0 & \\ x \in \mathbb{R}^n & a_1^T(x_q(t))x_v(t) = 0 & \\ u \in \mathbb{R}^m & \quad\vdots & \text{is empty} \\ & a_{n_a}^T(x_q(t))x_v(t) = 0 & \\ & \tilde{\Psi}_{k-1}(f_p(x,u)) - 1 \ge 0 & \\ & \tilde{\Psi}_{k-1}(f_p(x,u)) - 1 \ne 0 & \end{array} \right. \tag{3.44}$$

Condition(3.44) can be tested making use of the next bounded degree P-satz refutation certificate

$$\left\{ \begin{array}{l} s_0 + (1 - V_k(x,u))s_1 + (1 - c_u(u))s_2 + (1 - c_x(x))s_3 + \\ +(\tilde{\Psi}_{k-1}(f_p(x,u) - 1)s_4 + (1 - V_k(x,u))(1 - c_u(u))s_5 + \\ +(1 - V_k(x,u))(1 - c_x(x))s_6 + (1 - V_k(x,u))(\tilde{\Psi}_{k-1}(f_p(x,u)) - 1)s_7 + \\ +(1 - c_u(u))(1 - c_x(x))s_8 + (1 - c_u(u))(\tilde{\Psi}_{k-1}(f_p(x,u)) - 1)s_9 + \\ +(1 - c_x(x))(\tilde{\Psi}_{k-1}(f_p(x,u)) - 1)s_{10} + \sum_{p=1}^{n_a} (a_p^T(x_q(t))x_v(t))z_p) \\ +(1-c_u(u))(1-c_x(x))(\tilde{\Psi}_{k-1}(f_p(x,u))-1)s_{11} + (\tilde{\Psi}_{k-1}(f_p(x,u))-1)^{2n_1} = 0 \end{array} \right. \tag{3.45}$$

where $n_1 \in \mathbb{Z}_+$, $s_0, \ldots, s_{10} \in \sum_{n+m+d}$ and $z_1, \ldots, z_{n_a} \in \mathcal{R}_{n+m+d}$. Starting from (3.45) a simplified and sufficient SOS optimization problem can be derived by noticing that

- s_1 is positive semidefinite and, as a consequence, it can be removed from (3.45) introducing an inequality ($\leq$);
- n_1 is a free parameter, then it can be fixed ($n_1 = 1$);

Remark 3.19. Notice that further simplifications of 3.45 are possible. For further examples refers to the ideas and technicalities developed in [116].

Collecting all the above developments the following proposition holds true:

Proposition 3.20 (One-step controllable set P-satz certificate). *Conditions (3.42)-(3.43) are satisfied if there exists a candidate function $V_k(x,u) \in \Sigma_{n+m}$, polynomial multipliers $\hat{s}_p \in \Sigma_{n+m}$, $p = 1 \ldots 6$, $\hat{z}_p \in \mathcal{R}_{n+m}$, $p = 1, \ldots n_a$ and a radially unbounded positive definite function l_1 such that:*

$$V_k(x,u) - l_1(x,u) \in \Sigma_{n+m+d} \tag{3.46}$$

$$\begin{cases} -[\,(1 - V_k(x,u))\hat{s}_1 + (1 - c_u(u))\hat{s}_2 + (1 - c_x(x))\hat{s}_3(\tilde{\Psi}_{k-1}(f(x,u) - 1)\hat{s}_4 + \\ +(1 - V_k(x,u,d))(1 - c_u(u))\hat{s}_5 + (1 - V_k(x,u))(1 - c_x(x))\hat{s}_6 + \\ +\sum_{p=1}^{n_a} (a_p^T(x_q(t))x_v(t))\hat{z}_p)] \in \Sigma_{n+m} \end{cases} \tag{3.47}$$

Remark 3.21. An inner approximation of the one-step controllable region in the extended space (x,u) can achieved by means of the following BMI optimization problem (see [54] and [122] for examples of fitting solvers)

$$\begin{aligned} \left\{\hat{V}_k(x,u),\, \{\hat{s}_p\}_{p=1}^{6},\, \{\hat{z}_p\}_{p=1}^{n_a}\right\} := \arg \min_{\substack{V_k(x,u,d)\in\sum_{n+m} \\ s_p\in\Sigma_{n+m}, i=1,\ldots,6, \\ z_p\in\mathbb{R}^{n+m}, p=1,\ldots,n_a, \\ l_1\in\Sigma_{n+m}}} & \text{Trace } (Q_k^{-1}) \\ s.t.\ (3.46) - (3.47) & \end{aligned} \tag{3.48}$$

where Q_k is the Gram matrix of the decomposition

$$V_k(x,u) = \varphi^T(x,u) Q_k^{-1} \varphi(x,u)$$

and Q_k can be viewed as a shaping matrix of an ellipsoid in the extended space (x,u). □

Remark 3.22. Once the set $\{x \in \mathcal{X},\, u \in \mathcal{U} : V_k(x,u) \leq 1\}$ is obtained, the projected set Ξ_k along x can be easily computed by exploiting the quantifier elimination procedure detailed in [18]. □

Remark 3.23. Finally, notice that the additional constraint

$$\|B(f(x(t),u)-x(t))\|_2^2 \leq \epsilon^2$$

straightforwardly translates into a SOS requirement and, therefore, can be used within the polynomial version of the ***FCP*** procedure. □

On-line optimization: relaxed SDP problem

Here, a convenient formulation of the optimization (3.36)-(3.37) is provided so that it can be rephrased into a SDP problem. As detailed in Section 3.3 , the on-line phase of the RHC algorithm (namely ***OA-MPC***) prescribes the solution of one of the following optimization problems:

$$u(t) = \arg\min_u \; J_{k(t)}(x(t),u) \quad s.t. \tag{3.49}$$

$$f_p(x(t),u) \in \tilde{\Xi}_{k(t)-1}, \; u \in \mathcal{U} \tag{3.50}$$

or

$$u(t) = \arg\min_u \; J_{k(t)}(x(t),u) \quad s.t. \tag{3.51}$$

$$\|B(f_p(x(t),u)-x(t))\|_2^2 \leq \epsilon^2 \tag{3.52}$$

$$f_p(x(t),u) \in \tilde{\Xi}_{k(t)-1}, \; u \in \mathcal{U} \tag{3.53}$$

As it is evident the optimization (3.49)-(3.50) and (3.51)-(3.53) are both computationally high demanding for a real time resolution and, therefore, a relaxation procedure have to be considered.
To this end, note first that (3.51)-(3.53) can be used in place of (3.49)-(3.50) whenever the latter is required. This allows to simply define an affordable on-line optimization problem without alter the ***OA-MPC*** simplicity.
Then the sets of states and inputs satisfying (3.52)-(3.53) belongs to

$$\tilde{\mathcal{S}}_{k(t)-1} := \left\{ x \in \mathbb{R}^n, \, u \in \mathcal{U} : \begin{vmatrix} \tilde{\Psi}_{k(t)-1}(f_p(x,u)) \leq 1 \\ \|B(f_p(x(t),u)-x(t))\|_2^2 \leq \epsilon^2 \end{vmatrix} \right\} \tag{3.54}$$

with $\tilde{\Psi}_{k(t)-1}(f_p(x,u)$ and $\|B(f_p(x(t),u)-x(t))\|_2^2$ SOS polynomials. Then, the following inner ellipsoidal approximation of $\mathcal{S}_{k(t)-1}$ is considered in the extended space $(x,\, u)$:

$$\tilde{\mathcal{T}}(\Xi_{k(t)-1}) := \left\{ [x^T,\, u^T]^T \in \mathbb{R}^{n+m} : [x^T,\, u^T] P_{k(t)-1}^{-1} [x^T,\, u^T]^T \leq 1 \right\} \subseteq \tilde{\mathcal{S}}_{k(t)-1} \tag{3.55}$$

Using (3.55) in place of $\tilde{\mathcal{S}}_{k(t)-1}$ does not ensure feasibility retention of the ***OA-MPC*** algorithm: such a property is simply recovered by computing the entire sequence $\{\Xi_k\}$ under the satisfaction of the relaxation (3.55), i.e.

*Polynomial Ellipsoidal Set (**PES**) Procedure*

1. Given the set Ξ_{k-1}, compute the inner ellipsoidal approximation $\tilde{\mathcal{T}}(\Xi_{k-1})$ of $\tilde{\mathcal{S}}_{k-1}$ by solving an Eigenvalue LMI optimization problem (EVP) [25];
2. Solve the optimization problem (3.48) with
$$[x^T,\, u^T]P_{k(t)-1}^{-1}[x^T,\, u^T]^T \leq 1$$
in place of $\tilde{\Psi}_{k-1}(f_p(x,u)) \leq 1$;
3. Compute the projection (3.39).

Finally, the optimization (3.51)-(3.53) can be recast into a simple LMI problem.

Proposition 3.24. *The optimization (3.51)-(3.53) is recast into the following semi-definite programming problem:*

$$\min_{u} \gamma \tag{3.56}$$

$$\text{s.t.} \quad \begin{bmatrix} \gamma & [x(t)^T\, u^T] \\ * & P_{k(t)-1} \end{bmatrix} \geq 0 \tag{3.57}$$

Proof. The aim of the optimization problem (3.51)-(3.53) is to compute a control action u such that $f_p(x(t),u) \in \tilde{\Xi}_{k(t)-1}$ while minimizing a given cost function $J_{k(t)}(x(t),u)$. Then, by using the ellipsoidal approximation $\tilde{\mathcal{T}}(\Xi_{k(t)-1})$, a reasonable cost function is given by

$$\tilde{\mathcal{T}}(\Xi_{k(t)-1},\gamma) := \left\{ [x^T,\, u^T]^T \in \mathbb{R}^{n+m} \colon [x^T,\, u^T]P_{k(t)-1}^{-1}[x^T,\, u^T]^T \leq \gamma, \gamma \in \mathbb{R}_+ \right\}$$

with γ a slack variable. With the latter the optimization (3.51)-(3.53) can be rewritten as

$$\begin{aligned} &\min_{u} \gamma \quad s.t. \\ &\left[x(t)^T, u^T\right] \in \tilde{\mathcal{T}}(\Xi_{k(t)-1},\gamma), \quad u \in \mathcal{U} \end{aligned} \tag{3.58}$$

Finally, making use of Schur complements, optimization (3.58) simply translates into (3.56)-(3.57). □

SOS RHC algorithm

For sake of completeness the modified version of the ***OA-MPC*** algorithm, complying with the non-linear polynomial framework, is here reported:

*Sum-of-Squares Obstacle Avoidance MPC (**SOS-OA-MPC**) Algorithm*

Off-line:

1 Given the obstacle scenarios $\mathcal{O}^i$, $i = 1, \ldots, l$, the initial condition $x(0)$ and the goal x_f, compute a non-empty robust invariant ellipsoidal region $\mathcal{T}_0 \subset \mathbb{R}^n$ and a stabilizing feedback gain K_0 complying with the constraints (3.33)-(3.35), (3.40) and (3.7);

*2 Jointly apply the **FCP** and **PES** procedures to generate the sequences $\{\mathcal{T}_k^i\}$, $i = 1, \ldots, l$, $\{\mathcal{T}_k^{Ob_j^i}\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$, and $\{\mathcal{T}_k^{SW_j^i}\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$, and the following inner ellipsoidal approximations*

- $\{\mathcal{T}_k^i\} \rightarrow \left\{\tilde{\mathcal{T}}\left(\Xi_k^i\right)\right\}$, $i = 1, \ldots, l$;
- $\left\{\mathcal{T}_k^{Ob_j^i}\right\} \rightarrow \left\{\tilde{\mathcal{T}}\left(\Xi_k^{Ob_j^i}\right)\right\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$;
- $\left\{\mathcal{T}_k^{SW_j^i}\right\} \rightarrow \left\{\tilde{\mathcal{T}}\left(\Xi_k^{SW_j^i}\right)\right\}$, $\forall i \in \mathbf{V_r}$, $j = 1, \ldots, n_i$;

such that

$$x(0) \in \left(\bigcup_{\substack{i=1,\ldots,l \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^i\right) \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\ldots,n_i, \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^{Ob_j^i}\right) \right) \cup \left(\bigcup_{\substack{\forall i \in \mathbf{V_r} \\ j=1,\ldots,n_i, \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^{SW_j^i}\right) \right)$$

3 Store the ellipsoidal sequences.

On-line:

1: ***if***

$$x(t) \in \left(\bigcup_{k} \tilde{\mathcal{T}}\left(\Xi_k^{sc(t)}\right) \right) \cup \left(\bigcup_{\substack{j=1,\ldots,n_i, \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^{Ob_j^{sc(t)}}\right) \right) \cup \left(\bigcup_{\substack{j=1,\ldots,n_i, \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^{SW_j^{sc(t)}}\right) \right)$$

then $curr := sc(t)$

2: ***else*** $curr := prec$;

3: ***end if***

4: ***if*** $x(t) \in \bigcup_{k} \tilde{\mathcal{T}}\left(\Xi_k^{curr}\right)$ ***then***

5: *goto Step 10 by considering $\tilde{\mathcal{T}}\left(\Xi_k^{curr}\right)$ as the candidate sequence, i.e. $\tilde{\mathcal{T}}\left(\Xi_k^{candidate}\right)$;*

6: ***end if***

7: ***if*** $x(t) \in \bigcup_{\substack{j=1,\ldots,n_i, \\ k}} \tilde{\mathcal{T}}\left(\Xi_k^{SW_j^{curr}}\right)$ ***then*** $\tilde{\mathcal{T}}\left(\Xi_k^{candidate}\right) := \tilde{\mathcal{T}}\left(\Xi_k^{SW_j^{curr}}\right)$

8: ***else*** $\tilde{\mathcal{T}}\left(\Xi_k^{candidate}\right) := \tilde{\mathcal{T}}\left(\Xi_k^{Ob_j^{curr}}\right)$

9: ***end if***
10: *Find* $k(t) := \min_k \{x(t) \in \tilde{\mathcal{T}}\left(\Xi_k^{candidate}\right)\}$
11: ***if*** $k(t) \in IR^{candidate}$ ***then*** $u(t) = K_{k(t)}x(t)$
12: ***else***
13: *solve (3.56)-(3.57)*
14: ***end if***
15: *Apply* $u(t)$; *prec:=curr;* $t := t + 1$*; goto Step 1*

Proposition 3.25. *Then, the* ***SOS-OA-MPC*** *algorithm always satisfies the constraints and ensures Uniformly Ultimate Boundedness for all time-varying occurrences of* $\mathcal{O}_i$, $i = 1, \ldots, l$.

Proof. Using the same arguments exploited in Proposition 3.17.

3.5 Simulations

The aim of this section is to present results on the effectiveness of the proposed obstacle avoidance MPC strategy. The first two numerical examples (**Barrier** and **Cafeteria**) concern with a linear point mobile robot while the final experiment considers a **Non-Linear Differential Drive** vehicle. Numerical comparisons with the well established road-map *Cell decomposition* algorithm [1] are provided both in terms of path planning performance and computational complexity. The latter is justified by the fact that to the best author's knowledge, *complete* competitor schemes (planner + controller) are not present in literature. All the simulations have been implemented within the Matlab R2013 b environment, making use of the Multi-Parametric, Ellipsoidal, and SOS Toolboxes [55], [66], [93] over a laptop PC equipped with a Intel Core(TM) 2 Duo CPU.

3.5.1 Barrier and Cafeteria examples

Consider the point mobile robot model described in [67] whose the state consists of position and velocity components $x = [p_x \ p_y \ v_x \ v_y]^T$ and motions are governed by the following discrete-time LTI model:

$$x(t+1) = \Phi x(t) + Gu(t) + G_d d(t) \tag{3.59}$$

where $u \in \mathbb{R}^2$ is the acceleration vector and

$$\Phi = \begin{bmatrix} I_2 & \Delta t\, I_2 \\ 0_2 & I_2 \end{bmatrix}, \ G = \begin{bmatrix} \frac{(\Delta t)^2\, I_2}{2} \\ \Delta t I_2 \end{bmatrix}, \ G_w = G \tag{3.60}$$

with $\Delta t = 1\ s$ and

$$d(t) \in \mathcal{D} := \{d \in \mathbb{R}^2 : \|d\|_2 \leq 0.01\}, \forall t \geq 0. \tag{3.61}$$

Moreover, the following constraint on the acceleration vector is prescribed

$$\|u(t)\|_2^2 \leq 0.028, \ \forall t \geq 0 \tag{3.62}$$

Barrier configuration

This example refers to a critical case of study because the working environment is represented by a very narrow area and the single polyhedral obstacle Ob_1 moves along a straight line so that it virtually describes a particular obstacle configuration hereafter denoted as *barrier*, see Fig. 3.7. The obstacle positions are below reported:

Obstacle	**width**	**heigth**	**scenario**	**center of gravity**
Ob_1	1	1	1	$[2.5; 0.5]$
Ob_1	1	1	2	$[2.5; 1.5]$
Ob_1	1	1	3	$[2.5; 2.5]$
Ob_1	1	1	4	$[2.5; 3.5]$

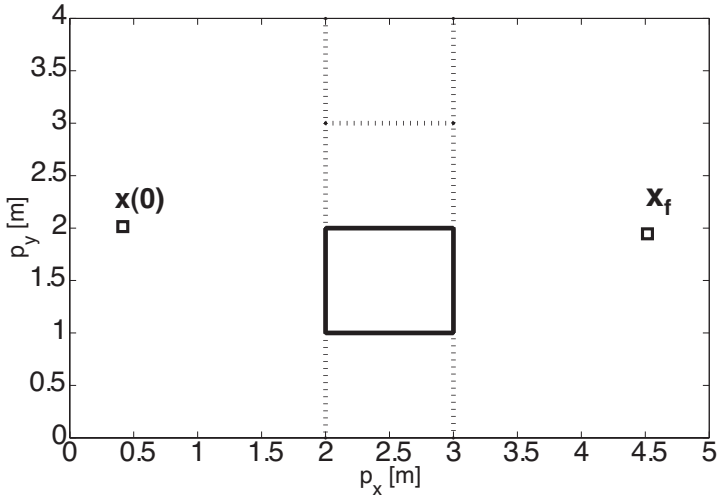

Fig. 3.7. Working planar scenario

In particular, we have considered the following situations:

a) The production line correctly operates and the obstacle circularly moves from the scenario 1 to the scenario 4;
b) At a certain time instant a malfunction is detected, the production line stops its normal behaviour and it is necessary to come back from the phase 3 to the phase 2. Then, it resumes the prescribed operations.

In order to implement the ***OA-MPC***-algorithm, the terminal pair $(K_0, \mathcal{T}_0)$ has been first determined as follows:

$$K_0 = \begin{bmatrix} -0.1447 & -0.1446 & -0.2780 & -0.2678 \\ -0.1446 & -0.1447 & -0.2678 & -0.2780 \end{bmatrix},$$

$$\mathcal{T}_0 = \{x \in \mathbb{R}^n \mid (x - x_f)^T (P_0)^{-1} (x - x_f) \leq 1\}$$

with

$$(P_0)^{-1} = \begin{bmatrix} 0.1925 & 0.1792 & -0.0595 & -0.594 \\ 0.1792 & 0.1925 & -0.0594 & -0.0595 \\ -0.0595 & -0.0594 & 0.1078 & 0.1078 \\ -0.0594 & -0.0595 & 0.1078 & 0.1078 \end{bmatrix}$$

Then, the ellipsoidal families have been off-line derived by using the ***FCP*** procedure:

- $\{\mathcal{T}_s^{Ob_1^1}\}_{s=1}^{190}$, $\{\mathcal{T}_s^{Ob_1^2}\}_{s=1}^{147}$, $\{\mathcal{T}_k^{Ob_1^3}\}_{k=1}^{149}$, $\{\mathcal{T}_k^{Ob_1^4}\}_{k=1}^{200}$
- $\{\mathcal{T}_k^1\}_{k=1}^{72}$, $\{\mathcal{T}_k^2\}_{k=1}^{87}$, $\{\mathcal{T}_k^3\}_{k=1}^{98}$, $\{\mathcal{T}_k^4\}_{k=1}^{72}$
- $\{\mathcal{T}_k^{SW_1^1}\}_{k=1}^{10}$, $\{\mathcal{T}_k^{SW_1^2}\}_{k=1}^{2}$, $\{\mathcal{T}_k^{SW_1^3}\}_{k=1}^{43}$, $\{\mathcal{T}_k^{SW_1^4}\}_{k=1}^{105}$

with the estimated tolerance level $\epsilon = 0.2867$.

The on-line numerical results are collected in the next Figs. 3.8-3.13. Specifically, Figs. 3.8-3.10 refer to **a)** and it is interesting to note how the set-membership signal in Fig. 3.9 copes with the obstacle configurations occurrences, see Fig. 3.8. On the other hand the second group of Figs. 3.11-3.13 reports the results for the scenario **b)** by assuming that a malfunctioning occurs within the time interval $[55\ 80]\, s$. By taking a look at the grey zones of Figs. 3.11-3.12, it is relevant to observe that the ***OA-MPC*** algorithm prescribes that the current state first belongs to the family $\{\mathcal{T}_k^{Ob_1^2}\}$ and then to $\{\mathcal{T}_k^{Ob_1^3}\}$: therefore this undesired phenomenon is efficiently ridden out.

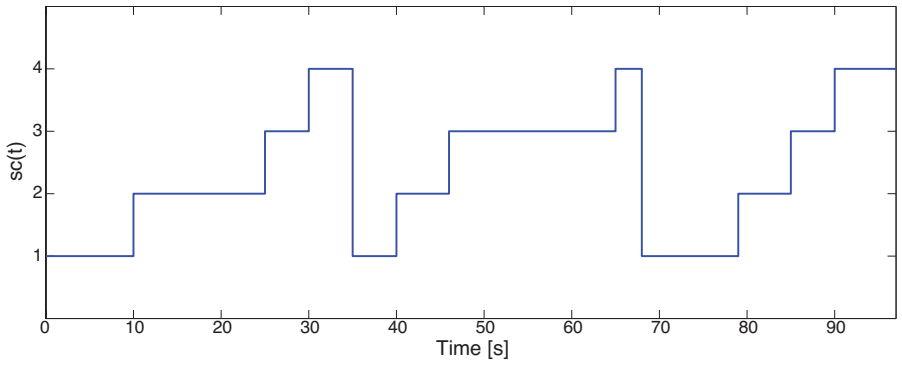

Fig. 3.8. Obstacle scenario switchings

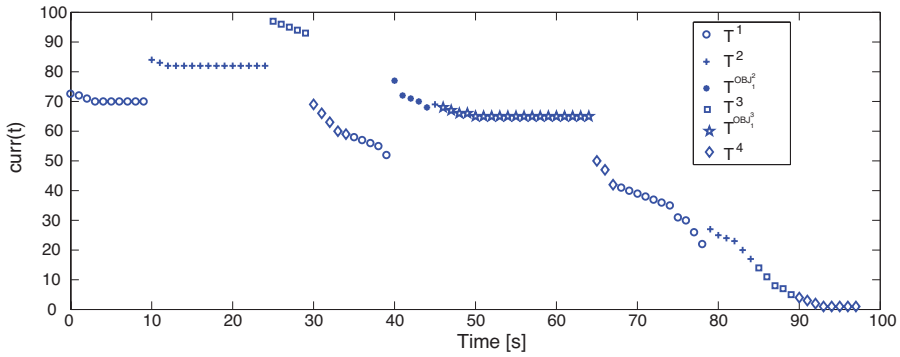

Fig. 3.9. Set-membership signal

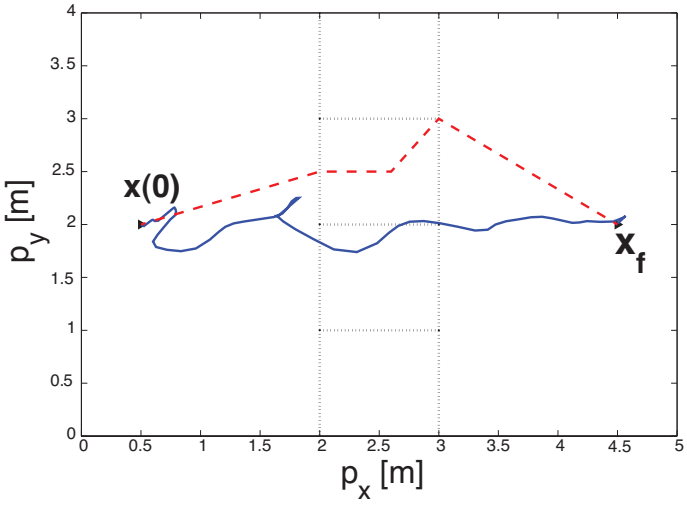

Fig. 3.10. Robot path under time-varying obstacles: ***OA-MPC*** algorithm (blue continuous line), *Cell decomposition* (red dashed line)

Comparisons with the modified *Cell Decomposition* algorithm have been performed by evaluating the path planning length. The results are summarized in Table 3.1 where it is clearly shown that the proposed strategy, although not oriented to the path planning optimization, provides performance similar to those obtained by the road-map algorithm.

Finally, Table 3.2 reports the computational complexity pertaining to both the **Off-line** and **On-line** phases of the ***OA-MPC*** algorithm evaluated by computing the average CPU time (seconds). As expected, most of the overall computational loads have been moved to the **Off-line** phase ($918\ s$) so rendering affordable the use of the proposed strategy from a practical point of view.

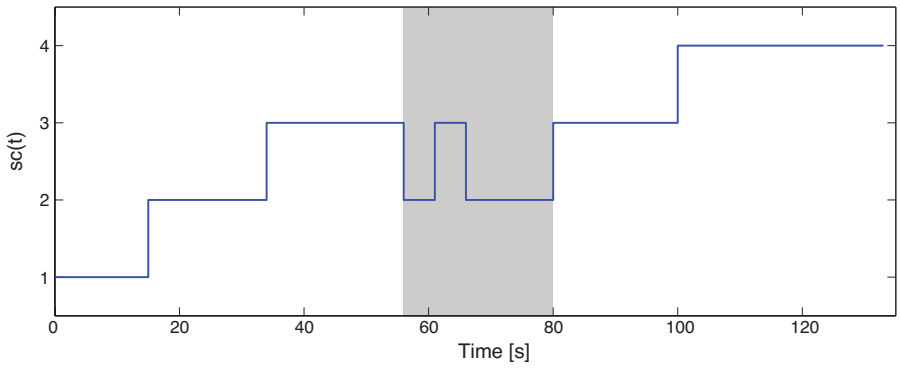

Fig. 3.11. Obstacle scenario switchings

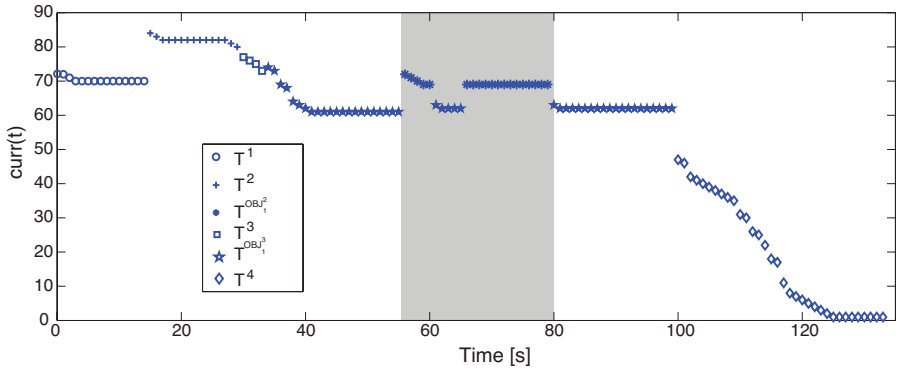

Fig. 3.12. Set-membership signal

	Cell Decomposition	OA-MPC
a)	4.60 m	5.68 m
b)	7.05 m	5.59 m

Table 3.1. Paths length (meters)

Off-line burdens	Basic sequences:	330
	Obstacle sequences:	480
	Scenario switching sequences:	108
		918
On-line burdens		0.05

Table 3.2. Numerical burdens: average CPU time (seconds)

On the other hand, it is relevant to put in evidence that the modified *Cell decomposition* algorithm, whose cost is only related to the path planning (not to the control action computation), presents an on-line computational load of a magnitude order greater than the proposed ***OA-MPC*** scheme, i.e. $0.45\,s$. The latter is mainly due to the fact that the road-map method must on-line recomputed the visibility graph whenever an obstacle scenario change occurs.

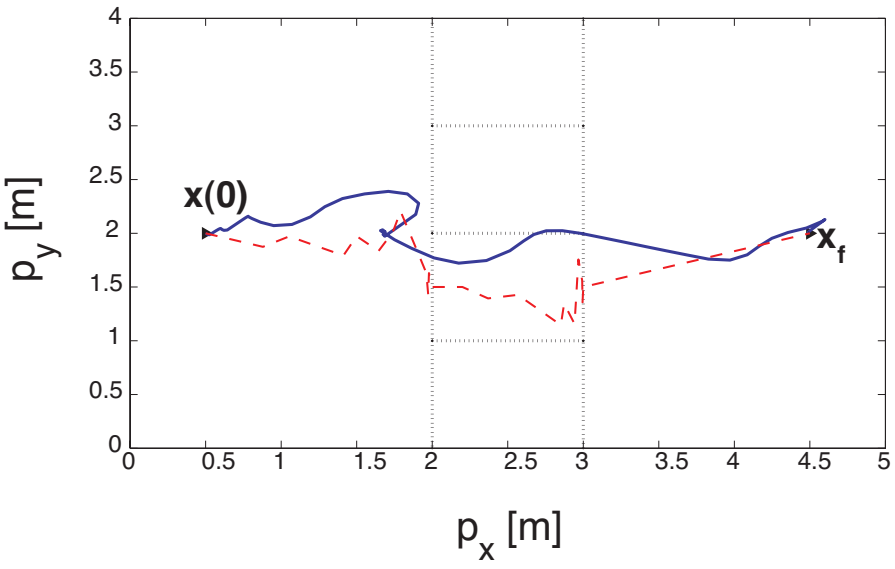

Fig. 3.13. Robot path under time-varying obstacles: ***OA-MPC*** algorithm (blue continuous line), *Cell decomposition* (red dashed line)

Cafeteria

The small cafeteria (size: $21\,m^2$) depicted in Fig. 3.14 consists of three areas: counter, kitchen and tables. Due to the limited free space, few guests can jointly access and they can move along prescribed lines (dashed lines): entrance-to-counter and counter-to-tables. In order to improve the cost efficiency, an autonomous ground vehicle for carrying plates from the kitchen to the counter and *vice-versa* is used. The main drawback of this idea is related to the motion of the people (time-varying obstacles) that may block the robot transit.

In the sequel, the *cafeteria-guest-robot* architecture is recast within the proposed strategy under the following assumptions:

1. almost two guests at a time are allowed along the lines;
2. people move following the directions indicated by the red arrows;
3. the walkable path is partitioned in six zones (dotted polygons in Fig. 3.14).

Assumption 1 defines the number of moving obstacles whereas Assumption 3 fixes the obstacle locations within the environment. Notice that an obstacle scenario change occurs only when a guest trespasses the limits of his current zone.
By considering the combinations with repetition (i, i') of the six polygons, we have that the number of the admissible obstacle scenarios is $l = 21$. Moreover under the Assumption 2, the oriented graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, whose rules are shown in Fig. 3.15, is obtained. Finally, in order to provide a mapping between the l obstacle scenarios and

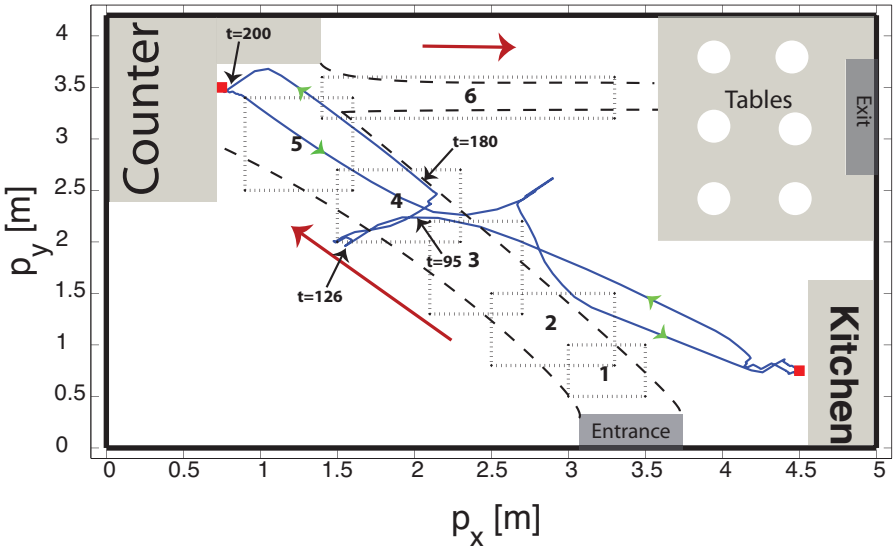

Fig. 3.14. The cafeteria structure

the nodes $\mathbf{V}$ we have that: *At each node* (i, i'), $i = 1, \dots l$, $i' = i, \dots l$, *corresponds the following scenario index:*

$$\textbf{scenario} \leftarrow 6(i-1) - \sum_{k=1}^{i-1} k + i'$$

To appreciate the *modus operandi* of the proposed scheme, let us analyze the robot behaviour from the kitchen to the counter shown in Fig. 3.14. First, at $t = 95\,s$ the obstacle scenario change $\overbrace{(5,6)}^{20} \rightarrow \overbrace{(1,5)}^{5}$ (see Fig. 3.16) hampers the transit of the robot towards the counter because the occurrence of the obstacle zone **5** and the fact that $x(95) \in \mathcal{T}_{55}^{20}$ do not guarantee the obstacle avoidance. Since in principle the sequence $\{\mathcal{T}_k^5\}$ should be used, a switching to $\{\mathcal{T}_k^{Ob_2^5}\}$ is initially performed and, as a consequence, the robot trajectory deviates from the *ideal path*, see Fig. 3.14.

Later, at $t = 126\,s$ the switching $\overbrace{(2,5)}^{10} \rightarrow \overbrace{(2,6)}^{11}$ occurs, the area **5** is walkable and therefore the strategy is capable to select the "optimal" set sequence, i.e. $\{\mathcal{T}_k^{11}\}$. Finally, from $t = 180\,s$ forward the robot proceeds towards the target by exploiting the ellipsoidal sequence $\{\mathcal{T}_k^8\}$ and at $t = 200\,s$ reaches the terminal region $\{\mathcal{T}_0^8\}$ where the counter target is located. The same reasoning holds true for the opposite path counter-to-kitchen.

Finally the computational burdens are detailed in Table 3.3. By comparing these results with those related to the modified *Cell decomposition* algorithm, it appears

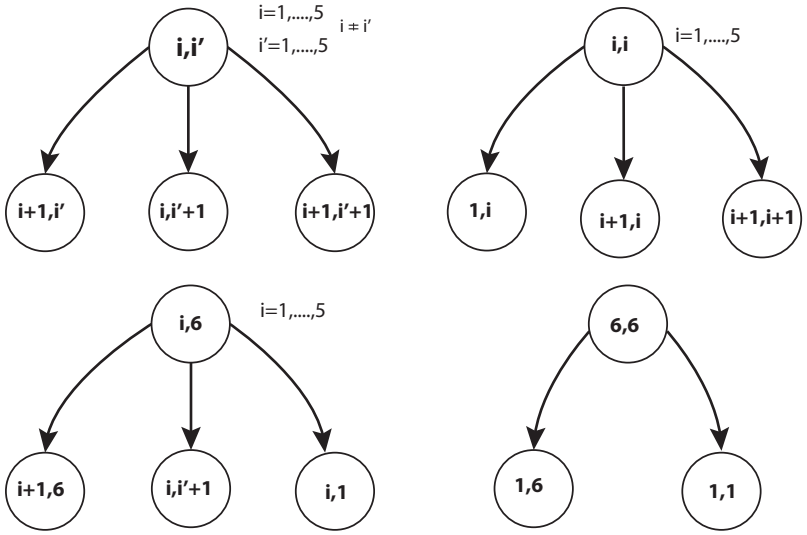

Fig. 3.15. Oriented graph rules

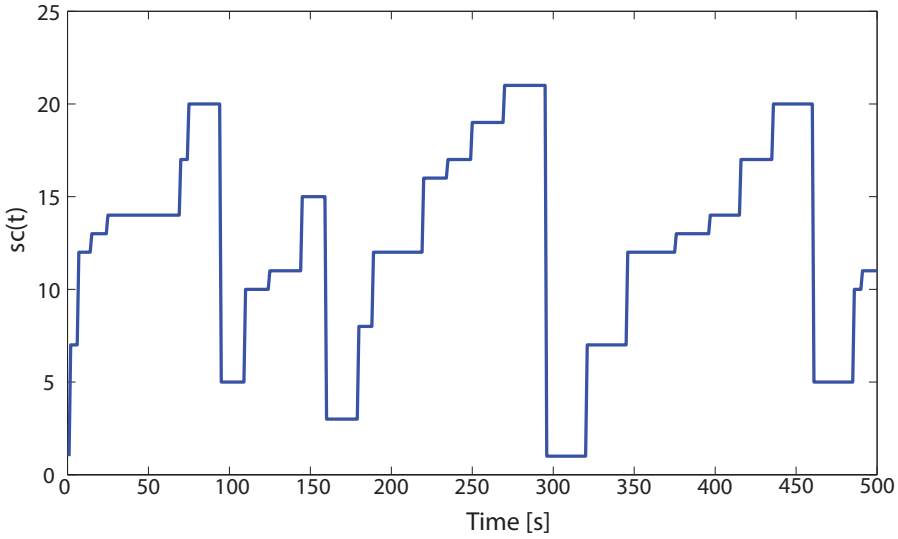

Fig. 3.16. Obstacle scenario switchings

that this road-map method requires $0.94\,s$ per step (very close to the sampling time $T_c = 1\,s$) while the on-line computational load of the proposed scheme is unchanged w.r.t. the *Barrier configuration* example. The main consequence of this analysis is that, even if the ***OA-MPC*** off-line computational burdens grows up with the obstacle scenario number, the on-line phase is always computational affordable. On the other

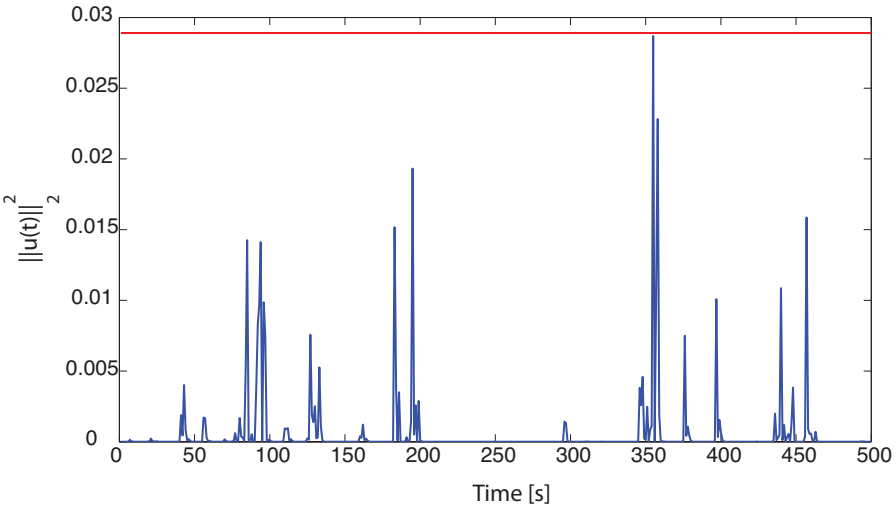

Fig. 3.17. Command input

hand road-map methods present the non-negligible disadvantage of increasing on-line loads when complex dynamic environments are taken into consideration.

Table 3.3. Numerical burdens: average CPU time (seconds)

Off-line burdens	Basic sequences:	1800
	Obstacle sequences:	300
	Scenario switching sequences:	800
		2900
On-line burdens		0.05

3.5.2 Non-Linear differential drive

The aim of this simulation is to illustrate the behavior of the proposed ***SOS-OA-MPC*** strategy when a non-linear vehicle model is considered. In particular the simulation involves the differential drive dynamical model, recast into a polynomial filed, described in Appendix B.2. The vehicle's parameters are collected in Table 3.4

The extended state space description B.9 has been discretized by using Euler forward differences with the sampling time $\Delta t = 0.5\,s$. and the following constraints are prescribed

$$|\tau_1(t)| \leq 0.45\,[Nm],\ \forall t \geq 0, \tag{3.63}$$

Parameter	Value
m	80
I	2 $K_g m^2$
r	0.075 m
R	0.325 m

Table 3.4. Differential drive: Parameters

$$|\tau_2(t)| \leq 0.45\,[Nm],\ \forall t \geq 0, \tag{3.64}$$

$$\|v(t)\|^2 \leq 0.16\,[m/s],\ \forall t \geq 0, \tag{3.65}$$

where

$$\|v(t)\|^2 = x_4^2(t) + x_5^2(t)$$

Assume that the robot navigates within a planar dynamic environment where the following obstacle scenarios are hypothesized: $\mathcal{O}^1$ (continuous lines) and $\mathcal{O}^2$ (dashed lines), see Fig. 3.19. The planar environment is described by

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} 0 \\ 25 \\ 0 \\ 30 \end{bmatrix} \tag{3.66}$$

while the obstacles locations are below reported:

Obstacle	width	heigth	scenario	center of gravity
Ob_1	1	1	1	$[14.5; 13.5]$
Ob_1	1	1	2	$[14.5; 14.5]$
Ob_2	1	1	{1,2}	$[5.5; 18.5]$

The aim of the simulation is to drive the robot from the initial planar position $p_0 = [20\ 5]^T$ to the target $p_f = [5\ 22]^T$ for any obstacle scenario change (from $\mathcal{O}^1$ to $\mathcal{O}^2$ and *viceversa*), see Fig. 3.18.
Then, inner ellipsoidal approximations of one-step controllable set sequences have been off-line derived:

- $\left\{\tilde{\mathcal{T}}\left(\Xi_k^1\right)\right\}_{k=1}^{167}$ and $\left\{\tilde{\mathcal{T}}\left(\Xi_k^2\right)\right\}_{k=1}^{150}$
- $\left\{\tilde{\mathcal{T}}\left(\Xi_k^{Ob_1^1}\right)\right\}_{k=1}^{114}, \left\{\tilde{\mathcal{T}}\left(\Xi_k^{Ob_2^1}\right)\right\}_{k=1}^{138}, \left\{\tilde{\mathcal{T}}\left(\Xi_k^{Ob_1^2}\right)\right\}_{k=1}^{137}$ and $\left\{\tilde{\mathcal{T}}\left(\Xi_k^{Ob_2^2}\right)\right\}_{k=1}^{138}$

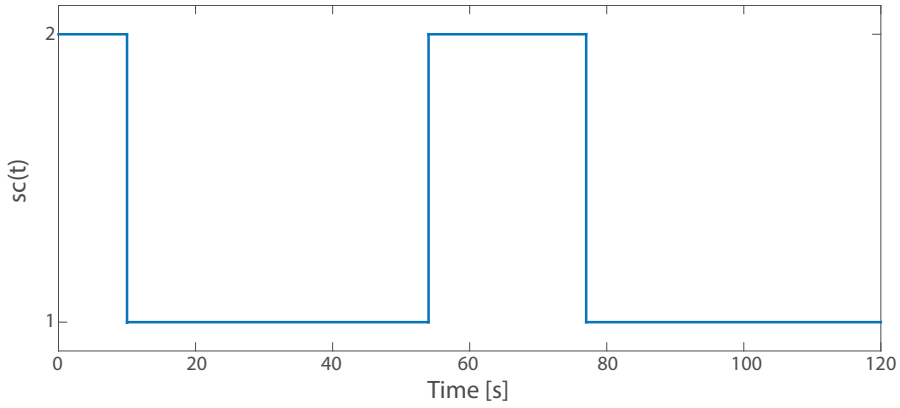

Fig. 3.18. Obstacle scenario switchings

- $\left\{\tilde{\mathcal{T}}\left(\Xi_k^{SW_1^1}\right)\right\}_{k=1}^{47}, \left\{\tilde{\mathcal{T}}\left(\Xi_k^{SW_2^1}\right)\right\}_{k=1}^{22}, \left\{\tilde{\mathcal{T}}\left(\Xi_k^{SW_1^2}\right)\right\}_{k=1}^{36}$ and $\left\{\tilde{\mathcal{T}}\left(\Xi_k^{SW_2^2}\right)\right\}_{k=1}^{30}$

In order to implement the ***SOS-OA-MPC*** algorithm, the terminal pair $(K_0, \mathcal{T}_0)$ has been first determined as follows:

$$K_0 = \begin{bmatrix} -0.127 & -0.127 & -0.250 & -0.239 \\ -0.127 & -0.127 & -0.2392 & -0.250 \end{bmatrix}, \ \mathcal{T}_0 = \{x \in \mathbb{R}^n \mid x^T (P_0)^{-1} x \leq 1\}$$

with

$$P_0 = \begin{bmatrix} 0.211 & 0.195 & -0.061 & -0.061 & 0.021 & 0.032 \\ 0.195 & 0.211 & -0.061 & -0.061 & 0.015 & -0.013 \\ -0.061 & -0.061 & 0.102 & 0.102 & 0.140 & 0.120 \\ -0.061 & -0.061 & 0.102 & 0.102 & 0.050 & -0.032 \\ 0.210 & 0.015 & 0.140 & 0.050 & 0.210 & 0.040 \\ 0.032 & -0.013 & 0.120 & -0.032 & 0.040 & 0.001 \end{bmatrix}$$

All the relevant results are summarized in Figs. 3.19-3.22. In Fig. 3.19 the robot trajectory under time-varying obstacles is depicted. It can be noted that the ***SOS-OA-MPC*** strategy is capable to achieve good control performance regardless of any obstacle change. Moreover as it results from Figs. 3.20-3.21, the prescribed constraints are always fulfilled.

To appreciate the *modus operandi* of the proposed scheme, it is important to analyze the signal shown in Fig. 3.22, because it provides the smaller ellipsoid of the pre-computed families containing the current state $x(t)$. Note that the markers (□ and ■) and (○ and •) have been used to denote the set-membership to the same sequences $\{\tilde{\mathcal{T}}\left(\Xi_k^1\right)\}$ and $\{\tilde{\mathcal{T}}\left(\Xi_k^2\right)\}$, respectively. The latter is instrumental to put in light that if the current obstacle scenario $sc(t)$ is different from that identified by the ***SOS-OA-MPC*** algorithm (namely Steps 1-9), the robot trajectory proceeds along the sequence of the admissible scenario used at the previous time instant (Step 2: *curr:=prec*). Nonetheless, this does not comprise neither the constraints satisfaction

nor the feasibility retention thanking to the one-step evolution bound (3.52).
Let us consider the time interval $[0,\ 30]$ (the grey zone of Fig. 3.22) where the switching $\mathcal{O}^2 \rightarrow \mathcal{O}^1$ occurs. At $t = 10\,sec$ the obstacle configuration changes while the current state is such that

$$x(10) \notin \left(\bigcup_{k=1}^{167} \tilde{\mathcal{T}}\left(\Xi_k^1\right)\right) \cup \left(\bigcup_{k=1}^{114} \tilde{\mathcal{T}}\left(\Xi_k^{Ob_1^1}\right)\right) \cup \left(\bigcup_{k=1}^{138} \tilde{\mathcal{T}}\left(\Xi_k^{Ob_2^1}\right)\right) \cup \left(\bigcup_{k=1}^{47} \tilde{\mathcal{T}}\left(\Xi_k^{SW_1^1}\right)\right) \cup \left(\bigcup_{k=1}^{22} \tilde{\mathcal{T}}\left(\Xi_k^{SW_2^1}\right)\right)$$

therefore as prescribed the used scenario does not change ($curr\ =\ 2$), see the marker • in Fig. 3.22. Then at $t\ =\ 22\,s.$ the *correct* scenario is recovered when the current state belongs to the obstacle sequence:

$$x(22) \in \tilde{\mathcal{T}}\left(\Xi_{30}^{Ob_1^1}\right)$$

Finally, at $t = 30\,s.$ the switching is accomplished, i.e.

$$x(30) \in \tilde{\mathcal{T}}\left(\Xi_{127}^1\right)$$

The same reasoning applies for the other obstacle scenario occurrences.

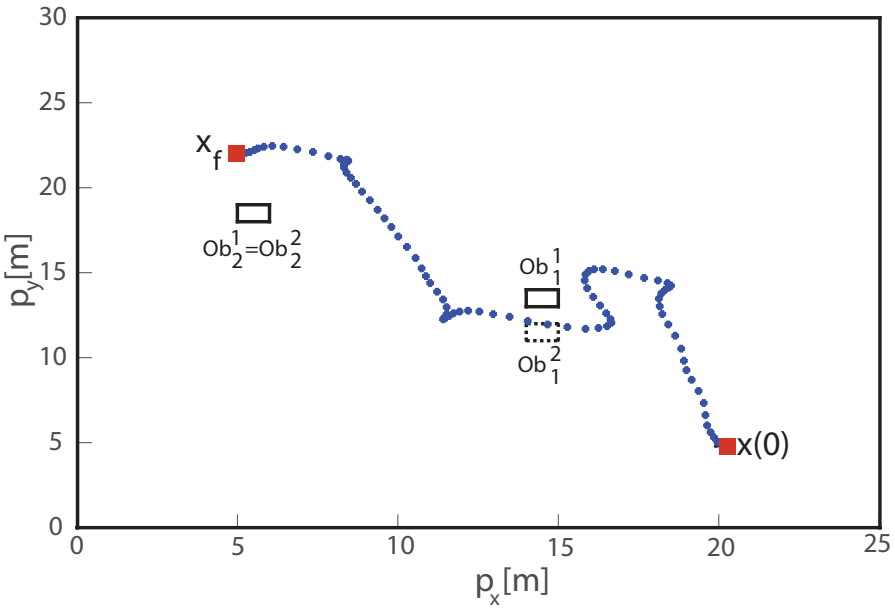

Fig. 3.19. Robot path under time-varying obstacles

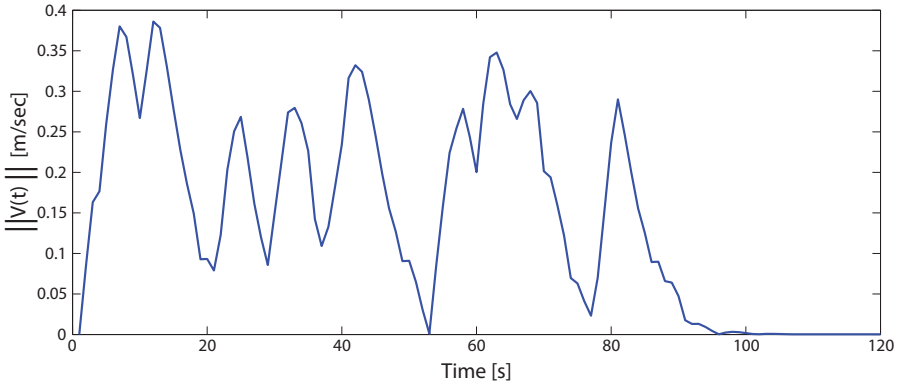

Fig. 3.20. Linear velocity

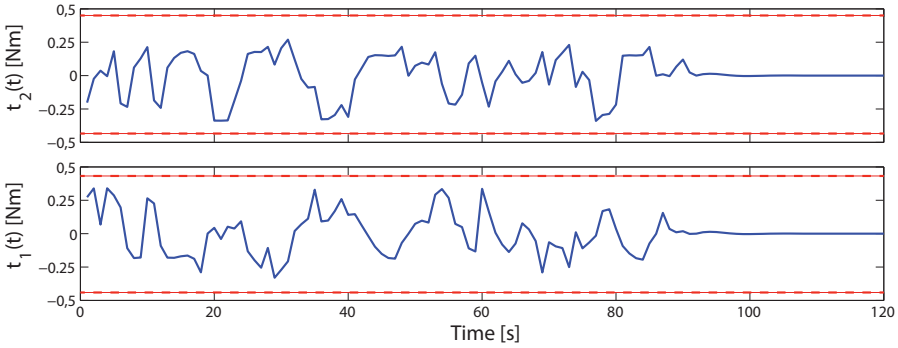

Fig. 3.21. Applied torques

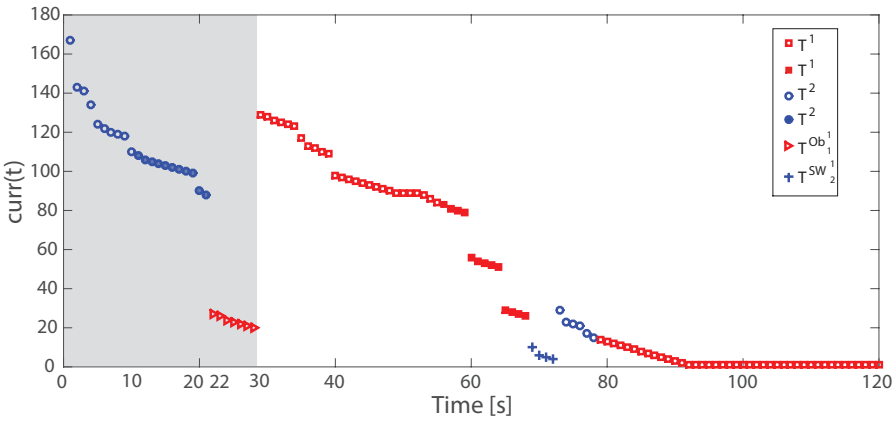

Fig. 3.22. Set-membership signal

3.6 Conclusions

In this chapter, a receding horizon control strategy has been developed for solving the obstacle avoidance motion planning problem for autonomous vehicles operating within structured working environments. Set-theoretic ideas have been used to take care of all admissible time-varying obstacle scenarios. From a methodological point of view, the proposed strategy is able to move in the off-line phase most of computations pertaining to the RHC controller design so that the overall framework becomes appealing in practical applications. Although initially proposed for LTI models, the scheme results to be quite independent from the plant model as testified by the polytopic and polynomial extensions. Finally, numerical comparisons have put in light that the idea to combine into a *single step* the design of the path planning module and of the predictive controller allows to deal with critical obstacle scenarios not easily manageable by dynamic path planning units.

4

A set-theoretic receding horizon control approach for unknown time varying environments

In this Chapter, a receding horizon control scheme for solving the obstacle avoidance motion planning problem of autonomous vehicles operating in uncertain dynamic environments is developed. The proposed scheme represents a remarkable generalization of the ***OA-MPC*** strategy developed in Chapter 3; the main difference relies on the removal of the hypothesis concerning the knowledge of all admissible time-varying obstacle scenarios. As a consequence, at the price of a higher but still affordable on-line computational loads, the control architecture here presented is capable to work around unpredictable obstacle occurrences. Specifically, the scheme is based on a joint and smart exploitation of robust positively invariant regions and one-step ahead controllable sets. Finally, simulation studies on a mobile robot described by a point mass and numerical comparisons with the competitor ***OA-MPC*** algorithm are provided in terms of the overall control performance.

4.1 Problem Statement

Consider robots described by the state space description (2.9) subject to (2.2)-(2.3) and operating within a dynamic environment where fixed and moving objects are arranged together. Take into account scenarios where some obstacles (**static obstacle scenario**) are fixed, i.e their planar coordinates are constant and a-priori known, whereas others (agents) can move under the hypothesis that they are equipped with semi-autonomous capabilities able to recognize the *non-free* locations (**moving obstacle scenario**).

Definition 4.1. *Let $p = Bx \in \mathbb{R}^2$ be the planar components of the state space $x \in \mathbb{R}^n$ and $B \in \mathbb{R}^{2\times n}$ a projection matrix, then the **static obstacle scenario** is defined as the following non-convex region*

$$\mathcal{O}_s := \{p \in \mathbb{R}^2 : \ f_s(p) > 0_{n_s^f}\} \tag{4.1}$$

where $f_s : \mathbb{R}^2 \to \mathbb{R}^{n_s^f}$ and n_s^f is the number of component-wise inequalities □

Definition 4.2. *At each time instant $t \in \mathbb{Z}_+$, the admissibility region for the* **moving obstacle scenario** *is defined as*

$$\mathcal{O}_d(t) := \{p \in \mathbb{R}^2 : \ f_d(p,t) > 0_{n_s^d}\} \tag{4.2}$$

with $f_d : \mathbb{R}^2 \times \mathbb{Z}_+ \to \mathbb{R}^{n_s^d}$ and n_s^d with the same meaning of n_s^f. □

Definition 4.3. *At each t the obstacle-free region is given by*

$$\mathcal{O}(t) := \mathcal{O}_s \cap \mathcal{O}_d(t) \tag{4.3}$$

□

In the sequel, the following assumptions are made:

Assumption 4.4. *Each obstacle/agent is a convex polygon.* □

Assumption 4.5. *Form and dimensions of each obstacle are a-priori known.* □

Assumption 4.6. *The robot and agents are equipped with a vision module (e.g. laser scanners [68]) capable to detect obstacles within a pre-specified radius R. Let R^c_{min} be the minimum curvature radius of the given robot, then the vision module is such that the field of view is 360^o and*

$$R > R^c_{min} \tag{4.4}$$

Moreover, the ball centered in the current robot planar position with radius R^c_{min} is defined as the safety region where the agents cannot enter.

Remark 4.7. *Assumption* 4.5 allows to provide a formal description of the proposed strategy. Nonetheless, it could be removed at the price of a more conservative approach that must consider the use of sophisticated vision units and *ad-hoc* heuristics, see e.g. [128] and references therein. *Assumption* 4.6 is standard (see e.g. [22], [83]) and the *safety* region is instrumental to recognize that a location within the working environment is inaccessible, see Fig. 4.1. □

4.1.1 Dynamical-obstacle avoidance motion planning (*D-OAMP*) problem

Given the static obstacle scenario $\mathcal{O}_s$, determine a state-feedback control policy

$$u(t) = g(x(t)) \tag{4.5}$$

compatible with (2.2)-(2.3) and (4.3), such that starting from an admissible initial condition $x(0)$ the robot trajectory $x(t)$ is driven to a target position x_f regardless of any admissible occurrence of $\mathcal{O}_d(t)$. □

In the sequel, the ***D-OAMP*** problem is tackled on the basis of the same theoretical ideas exploited in Chapter 3 and on a dual-mode control strategy. Anyhow, strictly speaking, the results here proposed represent a remarkable generalization because the

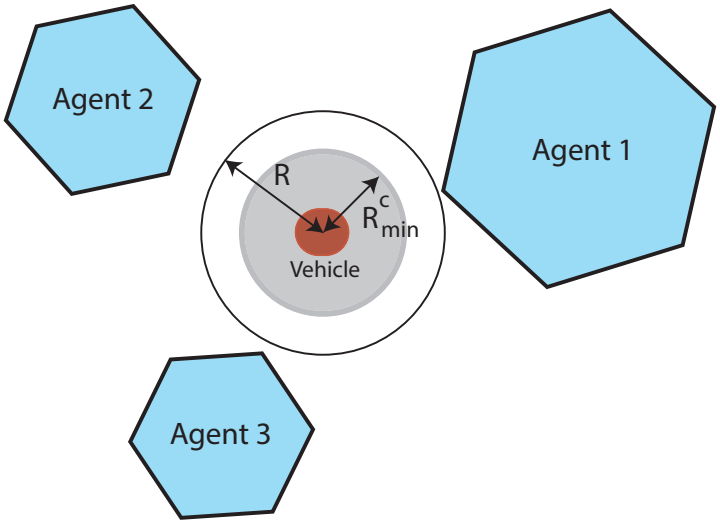

Fig. 4.1. Safety region description (the grey ball)

full knowledge of all admissible obstacle scenarios is not required (see Assumption 3.5 in Section 3.1) and the agents can move outside pre-specified locations. Then, in order to provide this required extension it is mandatory to find a solution to the following challenging questions:

- *How is it possible to work around unexpected obstacle occurrences along the nominal path* $x(0) \to x_f$ *?;*
- *How can admissible paths be generated complying with time-varying obstacle scenarios such that the control strategy feasibility is guaranteed and the computational burdens are low as "much as possible"?*

The next sections will provide a solution to this two problems by considering as a starting point the ideas underlying the ***OA-MPC*** strategy.

4.2 *D-OAMP* Control Framework

In this section, we provide the main ingredients for addressing the ***D-OAMP*** problem. First, Fig. 4.2 depicts the proposed control architecture which consists of the following units:

- **Off-line module -** By using the knowledge on the static obstacle scenario $\mathcal{O}_s$, it computes a sequence of one-step controllable sets, hereafter denoted as $\{\mathcal{T}_k^s\}_{k=0}^{N_s}$, $N_s \in \mathbb{N}$, such that there exists at least a feasible path to the goal x_f;
- **Vision module -** At each time instant t and within the vision radius R (see Fig. 4.1), it provides the information to characterize the region $\mathcal{O}_d(t)$;

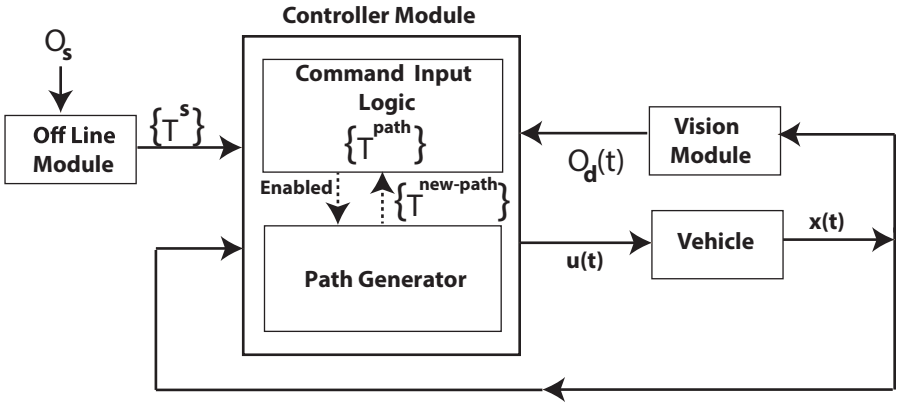

Fig. 4.2. Control architecture

- **Controller module -** By exploiting the off-line sequence $\{\mathcal{T}_k^s\}$, the current vehicle state measurement $x(t)$ and the current **moving obstacle scenario** $\mathcal{O}_d(t)$, the **Command Input Logic** sub-module checks the path $\{\mathcal{T}_k^{path}\}$ admissibility: if necessary, activates the construction of a new path and computes the correct control action. If *Enabled*, the **Path Generator** unit has the aim to determine a new feasible one-step controllable sequence.

Note that the actions of the **Off-line module** are performed by exactly using the ***OCSP*** procedure developed in Section 3.2 where a convexification procedure for nonconvex constraints is also provided. Therefore from now on the following actions undertaken during the on-line phase will be of interest:

- Checking the actual path admissibility with respect to (4.3);
- If the current obstacle configuration $\mathcal{O}_d(t)$ prevents the use of the sequence $\{\mathcal{T}_k^{path}\}$, then a new path must be determined by computing a new one-step controllable set sequence $\{\mathcal{T}_k^{new-path}\}$ under the constraints imposed by (4.3).

4.3 On-line phase

In this section, we describe how the proposed strategy is capable to deal with unpredictable path occlusions due to the agent dynamical behaviors. To this end, we shall provide details on the relevant vehicle actions: current path admissibility and path reconfiguration management.

4.3.1 Path admissibility check

Given the actual **moving obstacle scenario** $\mathcal{O}_d(t)$, the path admissibility check prescribes to infer if the sequence $\{\mathcal{T}_k^{path}\}_{k=0}^{N_{path}}$ is travelable, i.e. the one-step state

evolution $x^+ := \Phi x(t) + Gu + G_d d$ is complying with the constraints imposed by $\mathcal{O}(t)$.
This can be gathered by first determining the current state set-membership

$$k_{min} := \arg\min_k \{x(t) \in \mathcal{T}_k^{path}\} \tag{4.6}$$

and then by evaluating the following set inclusion

$$\mathcal{T}_{k_{min}}^{path} \subseteq \mathcal{O}_d^t \tag{4.7}$$

Therefore, if (4.7) holds true then $x^+ \in \mathcal{T}_{k_{min}-1}^{path} \subseteq \mathcal{T}_{k_{min}}^{path}$ and the current sequence $\{\mathcal{T}_k^{path}\}$ is admissible.

4.3.2 Path reconfiguration management

By assuming that

$$\mathcal{T}_{k_{min}}^{path} \nsubseteq \mathcal{O}_d(t)$$

the current path is no longer admissible, then a new one-step controllable sequence $\{\mathcal{T}_k^{new-path}\}$ has to be generated so that a feasible path complying with $\mathcal{O}(t)$ could be determined.
First notice that, by using *Assumption* 4.5 and the current set sequence $\{\mathcal{T}_k^{path}\}$, (continuous black line regions in Fig. 4.3), it is possible to find the closer equilibrium point (say $x_{eq}^{succ} \in \bigcup_i \mathcal{T}_k^{path}$) to the current vehicle position $x(t)$ which lies beyond the "active" obstacles. The latter can be achieved by identifying the index k_{eq}^{succ} of the RPI region such that

$$k_{min}^{Ob} := \arg\max_k \{\mathcal{T}_k^{path} \nsubseteq \mathcal{O}_d(t) \text{ and } k < k_{min}\} \tag{4.8}$$

$$k_{eq}^{succ} := \arg\max_k \{k \in IR^{path} \mid k \leq k_{min}^{Ob}\} \tag{4.9}$$

where IR^{path} is the ordered vector of the indices of the RPI regions belonging to the sequence $\{\mathcal{T}_k^{path}\}$. As a consequence x_{eq}^{succ} is the equilibrium point from which $\mathcal{T}_{k_{eq}^{succ}}^{path}$ has been generated. Then, the path reconfiguration can be split into two "independent" tasks (see Fig. 4.3):

- **(ON-1)** starting from the current state $x(t)$, generate a sequence of overlapped RPI ellipsoids $\mathcal{E}_k := \left\{x \in \mathbb{R}^n : x^T (Q_k)^{-1} x \leq 1\right\}$ (continuous red line ellipsoids) complying with (4.3) aimed to cover x_{eq}^{succ};
- **(ON-2)** starting from x_{eq}^{succ}, generate a sequence of one-step controllable sets $\{\mathcal{T}_k^{new-path}\}$ (dashed line ellipsoids) by applying the ***OCSP*** procedure with x_{eq}^{succ} in place of x_f, $x(t)$ in place of $x(0)$ and (4.3) in place of (4.1).

The rationale behind this choice concerns with the objective to reduce as much as possible the interval time required to work around an obstacle. In fact if the only **(ON-2)** action were applied, the vehicle would be confined around $x(t)$ (RPI centred in $x(t)$) until the entire sequence $\{\mathcal{T}_k^{new-path}\}$ is computed. This action is doable in virtue of **Assumption** 4.6 because there always exists an obstacle-free region of radius R_{min}^c around the current vehicle position. On the other hand, if during such a computation the agent moves in such a way that the condition (4.7) is no longer valid, then the sequence $\{\mathcal{T}_k^{new-path}\}$ becomes useless. Therefore, the exploitation of **(ON-1)** allows to safely move the autonomous vehicle amongst two overlapping RPI regions without waiting for the complete derivation of $\{\mathcal{T}_k^{new-path}\}$: this has the important merit to avoid most of computations when agents unfavourably modify their actual positions. Moreover it is worth to underline that, because the vehicle moves towards x_{eq}^{succ}, the ***OCSP*** will end when the following condition is satisfied:

$$\mathcal{T}_{k_1}^{new-path} \cap \mathcal{E}_{k_2} \neq \emptyset, \quad \text{for some } k_1, k2 \in \mathbb{Z}_+ \tag{4.10}$$

Remark 4.8. As a proof of the utility of the RPI regions computation (**(ON-1)** task) can be useful make a performance comparison with a "*naive*" path reconfiguration solution that prescribe to only use the task **(ON-2)** until the vehicle current position $x(t)$ is reached. To this end a numerical comparison is provided in Section 4.5. □

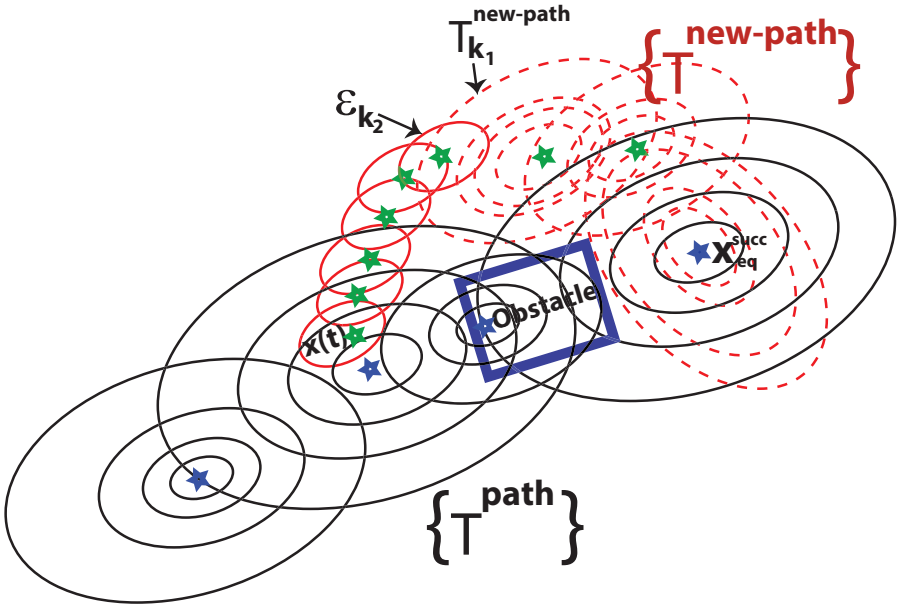

Fig. 4.3. Path generation

The construction of the overlapped sequence $\{\mathcal{E}_k\}$ can be straightforwardly achieved by using the following argument. Given the current state $x(t)$, first the RPI centered on the equilibrium x_{eq}^0 whose planar components are those of $x(t)$ is computed; then at each iteration k (see Fig. 4.4) a new equilibrium $x_{eq}^{k+1} \in \mathcal{E}_k$ is considered and the RPI region $\mathcal{E}_{k+1}$ is computed by ensuring the following overlapping property

$$\mathcal{E}_k \cap \mathcal{E}_{k+1} \neq \emptyset$$

Notice that each RPI ellipsoid is computed via a semidefinite programming problem (SDP) as detailed in [14].

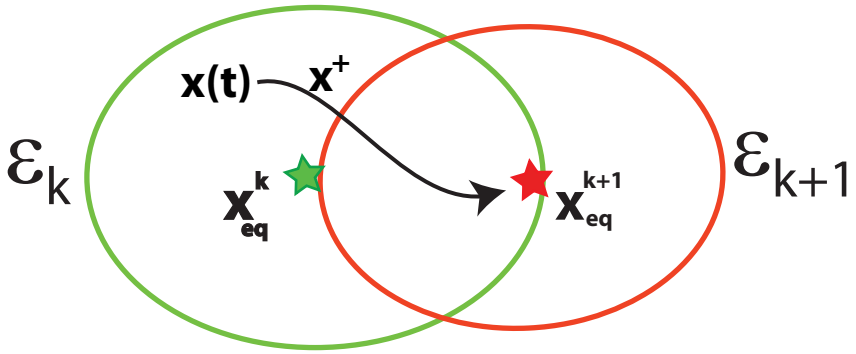

Fig. 4.4. Overlapped RPI regions

Finally, we have to clarify how it is ensured that there always exist a control action capable to move the vehicle amongst two overlapped RPI regions, namely $\mathcal{E}_k$ and $\mathcal{E}_{k+1}$.

For the sake of comprehension consider the scenario depicted in Fig. 4.4: here by construction $x(t)$ and x_{eq}^{k+1} belong to $\{\mathcal{E}_k\}$ and, as a consequence, are feasible initial conditions for steering the state trajectory to the RPI set defined by the equilibrium x_{eq}^k. Therefore, by linearity it is ensured that the constrained tracking problem $x(t) \to x_{eq}^{k+1}$ with $x(t) \in \mathcal{E}_k$ has always a feasible solution and it can be formulated as a tracking one-step MPC problem.
Let $\tilde{x} := x - x_{eq}^{k+1}$ and $\tilde{u} := u - u_{eq}^{k+1}$ be the shifted state and input variables w.r.t. the equilibrium $(x_{eq}^{k+1},\, u_{eq}^{k+1})$ to be tracked, then:

$$\tilde{u}(t) = \arg\min \|\Phi\tilde{x}(t) + G\tilde{u}\|^2_{(Q_{k+1})^{-1}} \quad \text{s.t.} \tag{4.11}$$

$$\Phi\tilde{x}(t) + G\tilde{u} \in \tilde{\mathcal{E}}_k, u \in \mathcal{U}, \tag{4.12}$$

where Q_{k+1} is the shaping positive definite matrix of $\mathcal{E}_{k+1}$ and $\tilde{\mathcal{E}} := \mathcal{E} \sim G_d\mathcal{D}$.

Proposition 4.9. *The optimization problem (4.11)-(4.12) has a solution at each time instant t when implemented in a receding horizon fashion.*

Proof. By collecting the above discussion. □

For the sake of clarity, Figure 4.5 summarizes the on-line **Path generator** module actions by means of a detailed activity diagram.

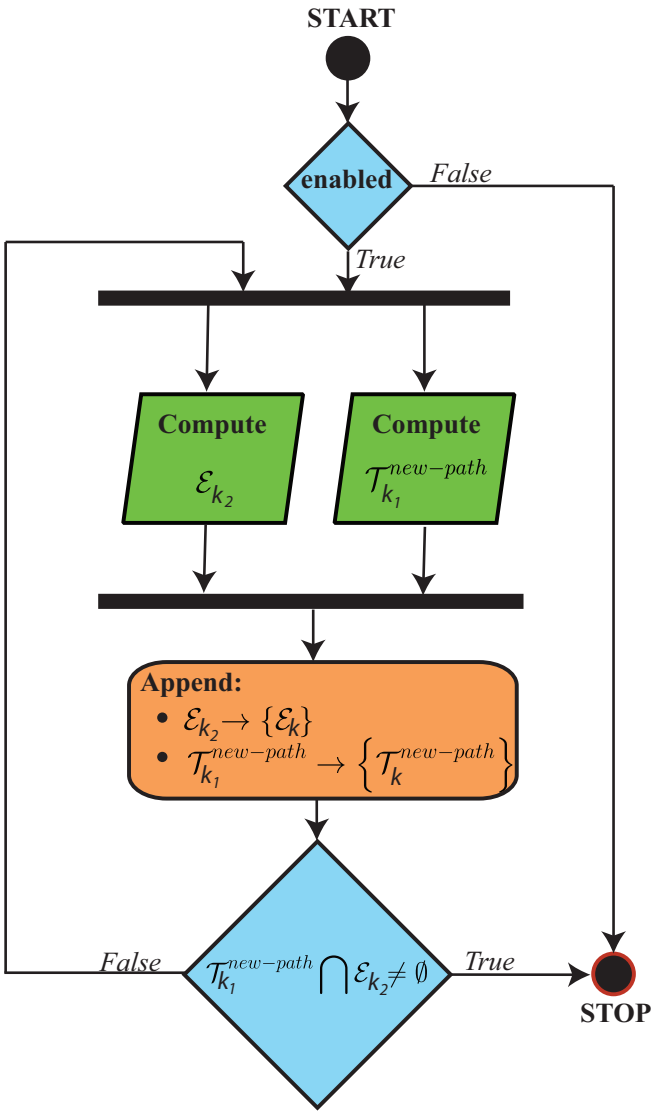

Fig. 4.5. Path generator module: Activity diagram

Remark 4.10. Since **(ON-1)** and **(ON-2)** are preemptable and independent tasks, the execution can be preempted by the highest priority task, i.e the **Command Input**

Logic task u(t), and executed in any order within the sampling time. Note that it is required a scheduling rule capable to guarantee that both **(ON-1)** and **(ON-2)** go on. The latter can be achieved by "fairly" splitting the remaining available sampling time after the execution of the main task $u(t)$. In order to clarify such a concept Fig. 4.6 provides a scheduling policy example: if one of the two tasks **(ON-1)** and **(ON-2)** cannot be completed, then at the next time instant it will assume the relative high-priority. Moreover, when $\{\mathcal{E}_k\}$ is empty an exception rule must be considered: **(ON-1)** becomes the highest priority task because at least a RPI region is necessary in order to compute a feasible command $u(t)$ to be applied to the vehicle. Finally, a scheduling improvement could result if parallel architectures are available for the execution of **(ON-1)** and **(ON-2)**. □

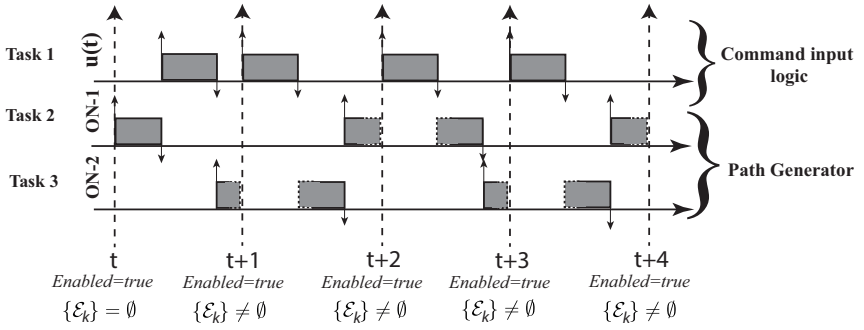

Fig. 4.6. Sheduling policy: Path Generator enabled

4.4 RHC algorithm

In this section, all the above development are summarized in the following computational scheme split in an off-line and on-line phase. The first must be executed just once and it can interpreted like a "setup" phase where all the heavier computational tasks are executed; the latter must be performed at each sampling time instant t according to the scheduling policy outlined in Remark 4.10 and under the following further assumption

Assumption 4.11. *At each time instant t the vehicle is informed about the* **moving obstacles scenario** $\mathcal{O}_d(t)$. □

*Moving Obstacle Avoidance MPC (**MOA-MPC**)- Algorithm*

Off-line:

1: Given the **static obstacle scenario** $\mathcal{O}_s$*, the initial state condition* $x(0)$ *and the goal* x_f*, compute the non-empty RPI ellipsoidal region* $\mathcal{T}_0$ *and the stabilizing feedback gain* K_0 *complying with (2.2), (2.3), (4.1);*

2: Apply the ***OCSP*** *procedure to generate* $\{\mathcal{T}_k^s\}_{k=0}^{N_s}$ *such that* $x(0) \in \bigcup_{k=0}^{N_s} \mathcal{T}_k^s$;

3: Store the ellipsoidal sets $\mathcal{T}_k^{path} := \mathcal{T}_k^s$, $k = 1, \ldots, N_s$. *Set* $\{\mathcal{T}_k^{new-path}\} := \emptyset$ *and* $\{\mathcal{E}_k\} = \emptyset$.

On-line:

1: ***if*** $x(t) \in \bigcup_{k=0}^{N_{path}}$ ***then***

2: *compute* k_{min} *by (4.6)*

3: ***if*** $\mathcal{T}_{k_{min}}^{path} \subseteq \mathcal{O}_d^t$ ***then***

4: ***if*** $k_{min} \in IR^{path}$ ***then*** $u = K_{k_{min}} x(t)$

5: ***else***

6:
$$u(t) = \arg\min \|\Phi x(t) + Gu\|^2_{\left(P^{path}_{k_{min}-1}\right)^{-1}} \tag{4.13}$$

$$\text{s.t. } \Phi x(t) + Gu \in \tilde{\mathcal{T}}^{path}_{k_{min}-1},\ u \in \mathcal{U} \tag{4.14}$$

7: ***end if***

8: ***else*** *goto Step 11*

9: ***end if***

10: ***else***

11: ***if*** $\mathcal{T}_{k_1}^{new-path} \cap \mathcal{E}_{k_2} \neq \emptyset$ ***then*** *Enabled:=false*

12: ***else*** *Enabled:=true*

13: ***end if***

14: ***if*** $x(t) \in \bigcup_{k=0}^{new-path} \mathcal{T}_k^{new-path}$ ***then***

15: $\{\mathcal{T}_k^{path}\} \leftarrow \{\mathcal{T}_k^{new-path}\}$, $\{\mathcal{T}_k^{new-path}\} := \emptyset$, $\{\mathcal{E}_k\} := \emptyset$, *goto Step 2*

16: ***else***

17:
$$k_{min} := \arg\max_k \{x(t) \in \mathcal{E}_k\}$$
$$u(t) = \arg\min \|\Phi x(t) + Gu\|^2_{\left(Q_{k_{min}+1}\right)^{-1}} \tag{4.15}$$

$$\text{s.t. } \Phi x(t) + Gu \in \tilde{\mathcal{E}}_{k_{min}},\ u \in \mathcal{U} \tag{4.16}$$

18: ***end if***

19: *Apply* $u(t)$; $t := t + 1$; *goto Step 1*

20: ***end if***

Note that the running costs $\|\Phi x(t)+Gu\|^2_{\left(P^{path}_{k_{min}-1}\right)^{-1}}$ and $\|\Phi x(t)+Gu\|^2_{\left(Q^{path}_{k_{min}-1}\right)^{-1}}$ characterizes the one-step ahead state prediction with $P^{path}_{k_{min}-1}$ and $Q^{path}_{k_{min}-1}$ the shaping matrices of the ellipsoidal region $\mathcal{T}^{path}_{k_{min}-1}$ and of the RPI region $\mathcal{E}_{k_{min}-1}$, respectively. Moreover, Fig. 4.7 provides a graphical description of the main actions pertaining to the ***MOA-MPC*** on-line phase.

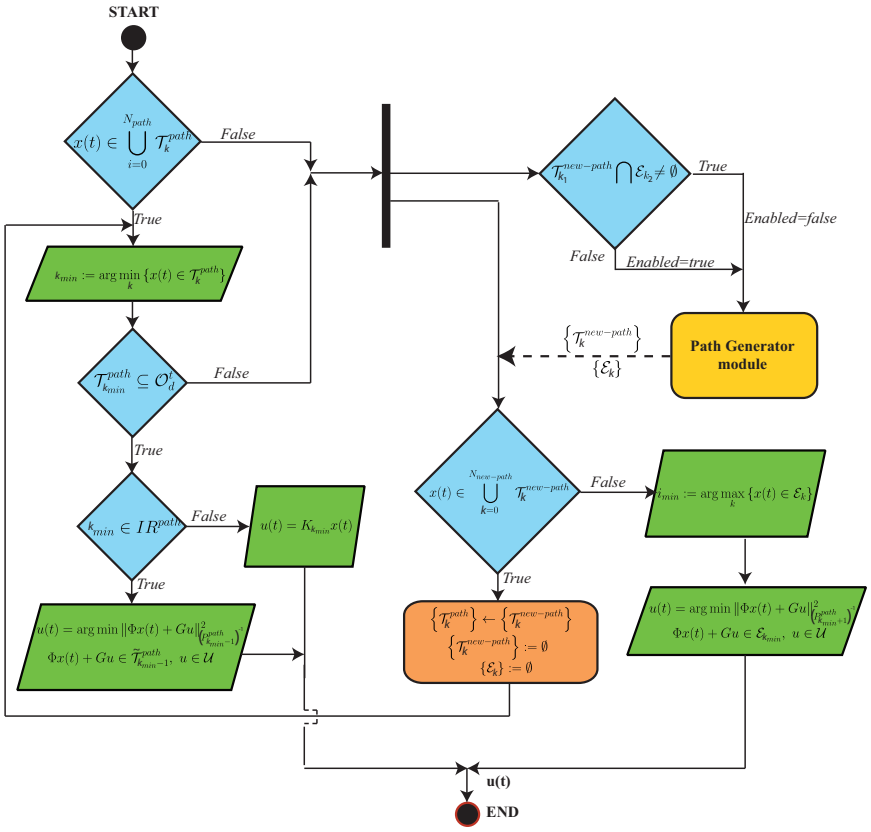

Fig. 4.7. The ***MOA-MPC*** on-line phase: activity diagram

The next proposition shows that the proposed ***MOA-MPC*** algorithm enjoys feasibility retention and closed-loop stability.

Proposition 4.12. *Let the sequence of sets* $\mathcal{T}_k^s$ *be non-empty and* $x(0) \in \bigcup_{k=0}^{N_s} \mathcal{T}_k^s$. *Then, the* ***MOA-MPC*** *algorithm always satisfies the constraints and ensures Uniformly Ultimate Boundedness for admissible occurrences of* $\mathcal{O}_d(t),\ \forall t \geq 0$.

Proof - Note that existence of solutions at time t implies existence of solutions at time $t+1$, because the optimization problems in Steps 6 and 17 are feasible: (4.13)-(4.14) has always doable (there exists an input vector u satisfying the constraints (2.2), (2.3), (4.3) such that the set-membership requirement in (4.14) holds true); the optimization (4.15)-(4.16) has always a solution in virtue of **Proposition** 4.9. Then, the feasibility retention is ensured thanking to the properties arising from the construction of the off-line one-step controllable set sequence and by using the arguments discussed in Section 4.3 that allow to derive a new path compatible with any admissible **moving obstacle scenario** $\mathcal{O}_d(t)$. Finally, Uniformly Ultimate Boundedness of the strategy follows by noting that the trajectory is in the worst case confined to

$$\bigcup_{k=0}^{N_s} \mathcal{T}_k^s \cup \left(\bigcup_{t\geq 0} \bigcup_{k} \mathcal{T}_k^{new-path(t)} \right)$$

□

4.5 Illustrative example

The aim of this example is to present results on the effectiveness of the here proposed obstacle avoidance ***MOA-MPC*** strategy. To this end, comparisons in terms of control performance with its "*naive*" version , hereafter denoted as ***Naive-MOA-MPC***, and with set-theoretic based scheme ***OA-MPC*** developed in the previous chapter are also provided. In the sequel refer to ***OA-MPC*** and to ***MOA-MPC*** as the *static* and the *dynamic* approach, respectively.

All computations have been carried out on a PC Intel Quad Core with the Matlab, LMI, Optimization and ET [66] Toolboxes and considering the same point mobile robot (3.59)-(3.60), disturbance set (3.61) and constraint (3.62) used in Section 3.5.1.

In addition, as required by the control architecture, the following numerical value for the vision an the minimum curvature radius are considered:

$$R^c_{min} = 1.49\,m, \quad R = 2\,m$$

In this example, the planar environment is described by

$$M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} 0 \\ 13 \\ 0 \\ 19 \end{bmatrix}, \; M = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}$$

and the following obstacle configuration, taken from [97] and depicted in Fig. 4.8, is considered:

- the **static obstacle scenario** $\mathcal{O}_s$:

$$Ob^s_1 : M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} -1 \\ 2 \\ -4.5 \\ 5.5 \end{bmatrix} \qquad Ob^s_2 : M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} -2 \\ 4 \\ -1 \\ 3 \end{bmatrix} \tag{4.17}$$

$$Ob^s_3 : M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} -10 \\ 13 \\ -5.75 \\ 8.25 \end{bmatrix} \qquad Ob^s_4 : M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} -16 \\ 18 \\ -2 \\ 4 \end{bmatrix} \tag{4.18}$$

$$Ob^s_5 : M \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \begin{bmatrix} -17 \\ 18 \\ -9 \\ 10 \end{bmatrix} \tag{4.19}$$

- the **moving obstacle scenarios** $\mathcal{O}_d(t)$ arise when the agents (dashed polygons) move along the paths indicated by the arrows shown in Fig. 4.8.

The aim of this simulation is to solve the ***D-OAMP*** problem with $x(0) = [6,\ 2,\ 0,\ 0]^T$ and $x_f = [17.5,\ 10.8,\ 0,\ 0]^T$ under the satisfaction of (3.62) and (4.17)-(4.19) and regardless of any $d(t) \in \mathcal{D}$ and of any admissible $\mathcal{O}_d(t)$ occurrence.

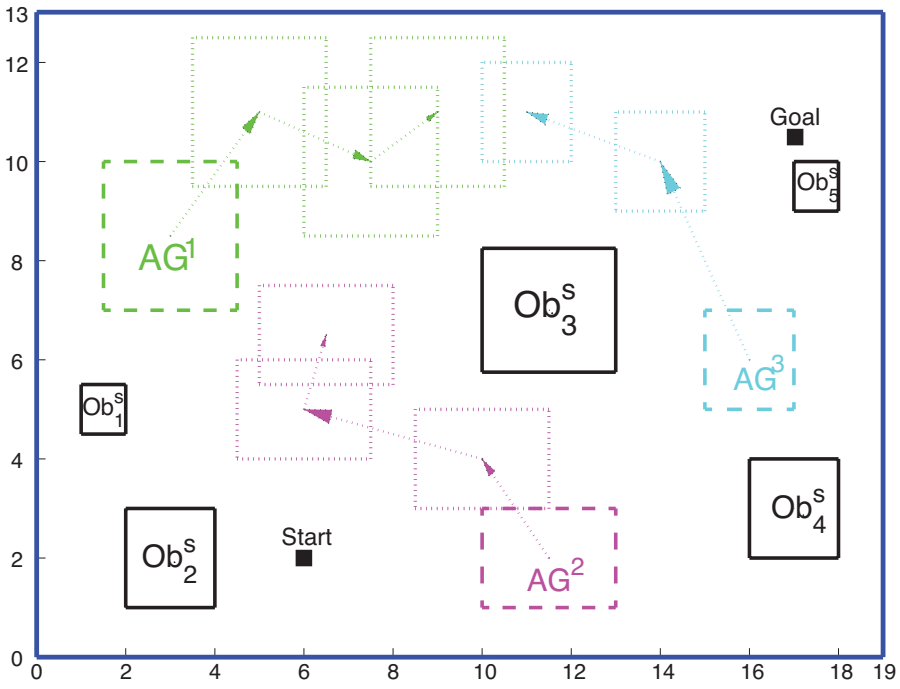

Fig. 4.8. Obstacle scenario: obstacles (continuous line) and agents (dashed line)

In order to implement the ***MOA-MPC*** algorithm, the terminal pair $(K_0, \mathcal{T}_0)$ has been first determined:

$$K_0 = \begin{bmatrix} -0.4486 & 0.0141 & -1.0523 & 0.0238 \\ 0.0141 & -0.4486 & 0.0238 & -1.0523 \end{bmatrix}$$

$$\mathcal{T}_0 = \{x \in \mathbb{R}^n \mid x^T (P_0)^{-1} x \leq 1\}$$

with

$$P_0 = \begin{bmatrix} 209.11 & -26.22 & 104.47 & -18.09 \\ -26.22 & 209.11 & -18.09 & 104.47 \\ 104.47 & -18.09 & 227.25 & -27.34 \\ -18.09 & 104.47 & -27.34 & 227.25 \end{bmatrix}$$

then, the family of ellipsoids $\{\mathcal{T}_k^s\}_{k=0}^{242}$ pertaining to the **static obstacle scenario** $\mathcal{O}_s$ has been off-line derived by using the ***OCSP*** procedure, see Fig. 4.9.
The numerical results are summarized in Figs. 4.10-4.13. First notice that, under the two **moving obstacle scenario** $\mathcal{O}_d(10)$ and $\mathcal{O}_d(170)$ occurrences (see Fig. 4.11), the prescribed constraints are always fulfilled, see Figs. 4.12 and 4.13.

To appreciate the *modus operandi* of the proposed scheme, analyze the robot behaviour shown in Figs. 4.11. During the first $10\,s$, the vehicle moves along the path

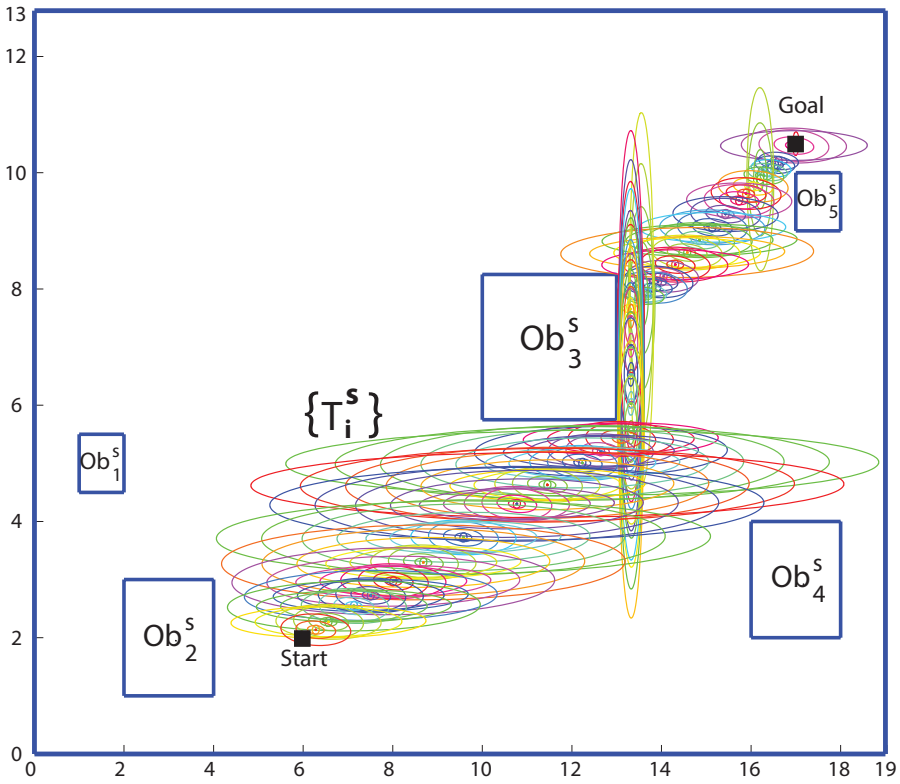

Fig. 4.9. Obstacle scenario and off-line ellipsoids sequence $\{\mathcal{T}_i^s\}$

defined by the static ellipsoidal sequence, i.e $\left\{\mathcal{T}_k^{path}\right\} \equiv \{\mathcal{T}_k^s\}$, as testified by the set-membership signal shown in Fig. 4.10, where the expected set contraction results (optimization (4.13)-(4.14)). At $t = 10\,s$ the **Vision module** detects $\mathcal{O}_d(10)$ (see the dashed polygon in Fig. 4.11 (top-left side)), the **Path generator** module is activated (*Enabled:=True*) because the feasibility check (4.7) fails and the computation of a new set sequence complying with $\mathcal{O}_d(10)$ starts (**ON1** and **ON2** actions). From $t = 11\,s$ onward the input is obtained by solving the optimization (4.15)-(4.16): until $t = 50\,s$ the computed RPI regions $\mathcal{E}_k$ (**ON1** task) are included in $\bigcup_k \mathcal{T}_k^{path}$, while within the time window $[51\ 142]\,s$ one has that $\mathcal{E}_k \notin \bigcup_k \mathcal{T}_k^{path}$ (see the grey zone in Fig. 4.10). Such a behaviour puts in light that the ***MOA-MPC*** scheme has began the computation of a new path capable to work around the agent AG^2. Then at $t = 142\,s$ $x(142) \in \mathcal{T}_{173}^{new-path}$, see the set-membership signal in Fig. 4.10 and the upper-right graph of Fig. 4.11, and as a consequence $\left\{\mathcal{T}_k^{new-path}\right\}$ is used in

place of $\left\{\mathcal{T}_k^{path}\right\}$. The same reasoning holds true for the **moving obstacle scenario** occurrence $\mathcal{O}_d(163)$: in particular one has that $x(249) \in \mathcal{T}_{28}^{new-path}$ and from $t = 250\,s$ onward the vehicle is driven to the target x_f, see the lower-right graph of Fig. 4.11. Moreover, the on-line computational burden pertaining to the on-line tasks are reported in Table 4.1, where the need of using the scheduling policy described in Remark 4.10 comes out.

Table 4.1. Average CPU times (seconds per step)

Task	Average CPU time [s]
u(t): Optimizations (4.13)-(4.14), (4.15)-(4.16)	0.35
ON-1	0.36
ON-2	0.55

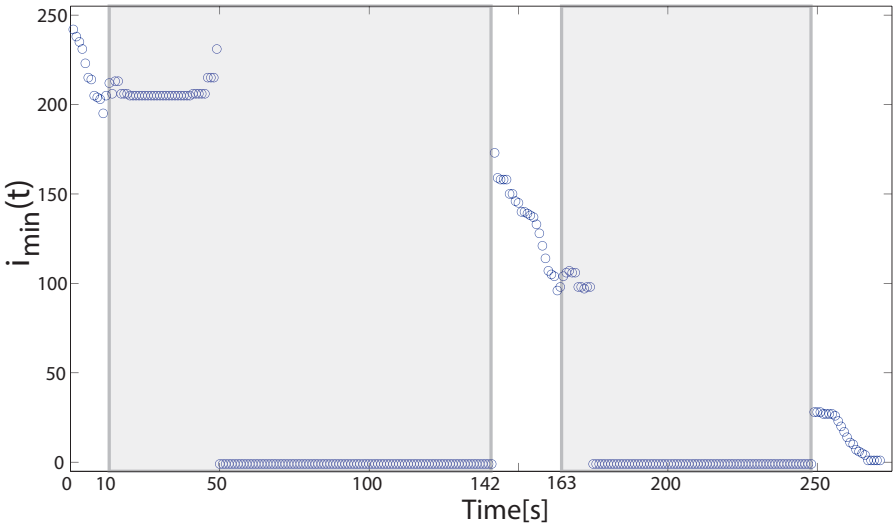

Fig. 4.10. Set-membership signal

The numerical comparisons between ***MOA-MPC***, "*naive*" ***MOA-MPC*** and ***OA-MPC*** in Figs. 4.12-4.13 are instrumental to show benefits and improvements of the here proposed strategy. First by considering the command input behaviours in Fig. 4.12, it is worth to note that the ***MOA-MPC*** significantly outperforms its "naive" version while is slight worse than ***MOA-MPC***. The rationale of these results can be explained via the following arguments:

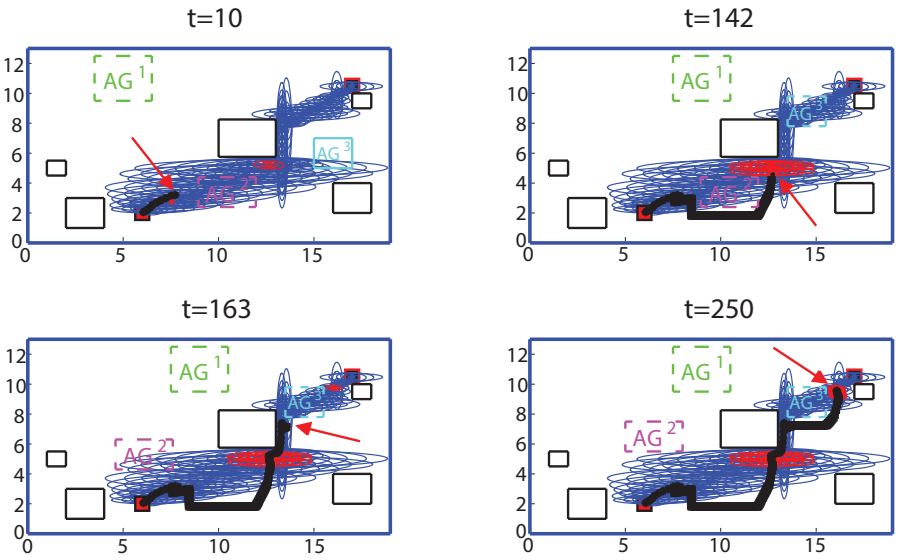

Fig. 4.11. Screen shots

- the ***OA-MPC*** algorithm off-line computes and stores the one-step controllable set sequences pertaining to all the admissible obstacle scenario occurrences. Therefore on one side the on-line phase is quickly performed (no new paths must be on-line determined) while on the other hand it is oriented to highly structured environments with the need of substantial memory resources;
- the high memory requirements of ***OA-MPC*** are mitigated by exploiting the *dynamic* strategy of ***MOA-MPC*** which is based on the on-line computation of new admissible paths. Nonetheless, in the "*naive*" ***MOA-MPC*** this is achieved by using only the **(ON-2)** task that forces the vehicle to wait (zero norm of the input signal) for the whole construction of a new feasible path, see the middle graph of Fig. 4.12 during the time intervals $[10\ 141]$ and $[204\ 300]$;
- the proposed path reconfiguration module is capable to significantly reduce the dead-lock phase of its "*naive*" version as explicitly shown in the lower graph of Fig. 4.12 where it is possible to see the decrease on the zero-norm input time intervals. Moreover, both the on-line computational load and the memory requirement are not substantially affected.

Finally for the sake of completeness Fig. 4.13 summarizes the trajectories obtained by the three schemes. As expected, the main differences amongst *static* (dotted line) and *dynamic* (dashed and continuous lines) algorithms arise when the occlusion is such that the current one-step controllable sequence is no longer admissible. In this case, the ***OA-MPC*** algorithm is forced to switch to a precomputed one-step controllable set sequence by travelling a more involved path because any modification during the on-line phase is not allowed, see the grey zone in Fig. 4.13.

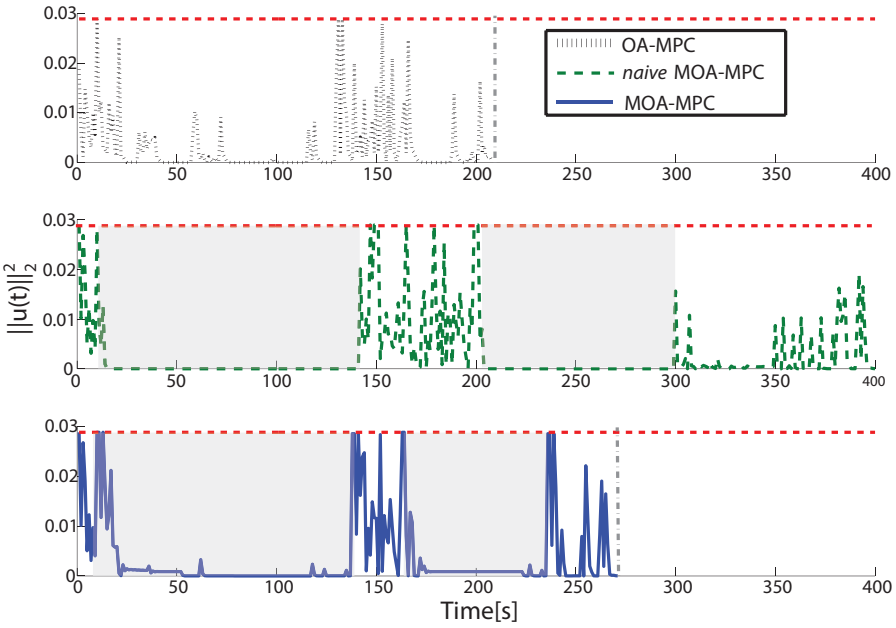

Fig. 4.12. Command inputs. The dashed line represent the prescribed constraint.

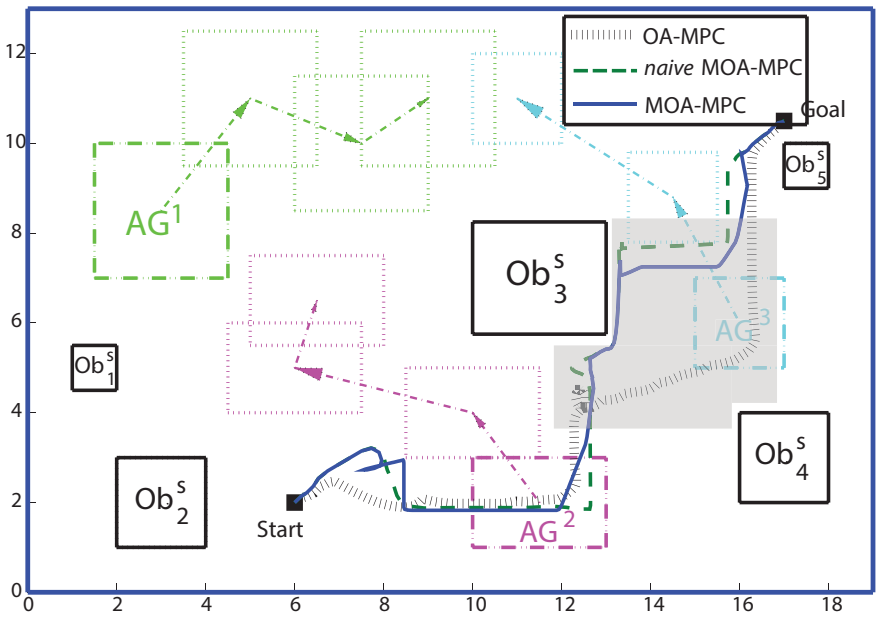

Fig. 4.13. Vehicle trajectories

4.6 Conclusions

In this Chapter a model predictive control strategy has been developed with the aim to comply with the obstacle avoidance and motion planning tasks pertaining to autonomous vehicles when dynamic working environments are considered. Set-theoretic ideas have been extensively used in order to manage unpredictable agent behaviours. The required computational resources (spatial and memory complexities) have been proved to be modest and therefore the strategy could be affordable in practical applications. This represents a significant improvement within the related literature as testified by the results provided in the simulation section.

Part II

Control of unmanned aerial vehicle

5

An Hybrid Command Governor approach

In this Chapter an obstacle avoidance control scheme for autonomous aerial vehicles is presented. The strategy is based on Command Governor (CG) ideas that are here extended in order to take into account non-convex and time-varying constraints typically arising in path planning obstacle avoidance problems.

In particular, a predictive approach different from the set-theoretic framework of Chapters 3-4 is here adopted. The main reasons of this choice relies on the need to use a reliable and fast controller capable to manage high order dynamical systems (e.g. aerial vehicles) for which the computations of recursions (2.5) should be unfeasible because of the huge number of involved variables (states and inputs).

In order to show effectiveness and applicability of the proposed CG based strategy, a real-time environment has been designed. Specifically a quadrotor Qball-X4 jointly with a Vicon Motion Capture System is considered to set-up an experimental real-time campaign. The Quanser's real-time control software has been used to embed the proposed CG-based algorithm on the on-board Gumstix architecture, while the control task has been stated as follows: drive the quadrotor state trajectory to a desired space location by avoiding collisions with three beams.

5.1 Problem Statement

Let consider an unmanned aerial vehicles (UAV) whose dynamics or their linearizion around an equilibrium point is described by the following linear time invariant (LTI) system

$$\begin{cases} x_v(t+1) = Ax_v(t) + Bu(t) + B_d d(t) \\ \qquad p(t) = H_v x_v(t) \end{cases} \tag{5.1}$$

where $p(t) \in \mathbb{R}^3$ denotes the vehicle position in the 3D space, $t \in \mathbb{Z}_+ := \{0, 1, ...\}$, $x_v(t) \in \mathbb{R}^{n_v}$ the plant state, $u(t) \in \mathbb{R}^{m_v}$ the control input and $d(t) \in \mathcal{D} \subset \mathbb{R}^{n_d}$ an exogenous disturbance, with $\mathcal{D}$ a convex and compact set such that $0_{n_d} \in \mathcal{D}$. Moreover, (5.1) is subject to the set-membership state and input constraints $u(t) \in \mathcal{U} \in \mathbb{R}^{m_v}$ (2.2) and $x_v(t) \in \mathcal{X} \in \mathbb{R}^{n_v}$ (2.3).

Throughout this Chapter, the following definitions will be used.

Definition 5.1. *Let Ob_j be a convex obstacle. Then an obstacle scenario $\mathcal{O}$ is defined as*

$$\mathcal{O} := \{Ob_1, \ldots, Ob_{n_o}\} \tag{5.2}$$

where n_o denotes the number of objects involved. □

Definition 5.2. *Let $\mathcal{O}$ be an obstacle scenario. Then, the non-convex obstacle-free region pertaining to $\mathcal{O}$ is defined as follows*

$$\mathcal{O}_{free} := \{x_p \in \mathbb{R}^{n_p} : \; h_i(x_p) > 0\} \tag{5.3}$$

where $h_i : \mathbb{R}^{n_p} \to \mathbb{R}^{n_f}$ and n_f the number of component-wise inequalities. □

By considering the dynamical evolution of UAVs when they are executing predefined missions within obstacle cluttered environments, the following assumptions are taken into account:

Assumption 5.3. *(Uncertain working scenario)*

- *The obstacle configuration is static: the objects have fixed positions;*
- *UAVs detect objects within the vision radius (active region): a-priori information about the obstacle scenario are not on-board available.*

This set-up concerns with *partially known* working areas where a specific set of obstacle scenarios may occur, see e.g. autonomous road-following missions [21], monitoring and/or surveillance tasks [28], [24] and so on.

5.1.1 UAV obstacle avoidance motion planning (*UAV-OAMP*) Problem

Given an obstacle scenario $\mathcal{O}$ and a target p_{goal}, find a state-feedback control policy

$$u(t) = \eta(x_v(t), p_{goal}) \tag{5.4}$$

compatible with (5.3), (2.2) and (2.3), such that starting from an initial condition $x_v(0)$ the robot state trajectory $x_v(t)$ is asymptotically driven to a target state x_v^{goal} such that $p_{goal} = H_v x_v^{goal}$. □

In the sequel, the problem will be addressed by developing an adequate extension of the command governor strategy presented in Section 2.4, which formally takes care of possible obstacle occurrences. In particular, in order to design a fully autonomous flight control law complying with the ***UAV-OAMP*** problem, the control architecture shown in Fig 5.1 is proposed. It consists of three main modules: a **reference manager (CG)** whose aim is to generate at each time instant a feasible set-point to be tracked during the on-line operations; a **planner** unit that determines a finite sequence of locations in order to allow obstacle avoidance during the vehicle navigation and a **control module (*SCG-OA*)** that is in charge to manage switching events when time-varying constraint paradigms are considered.

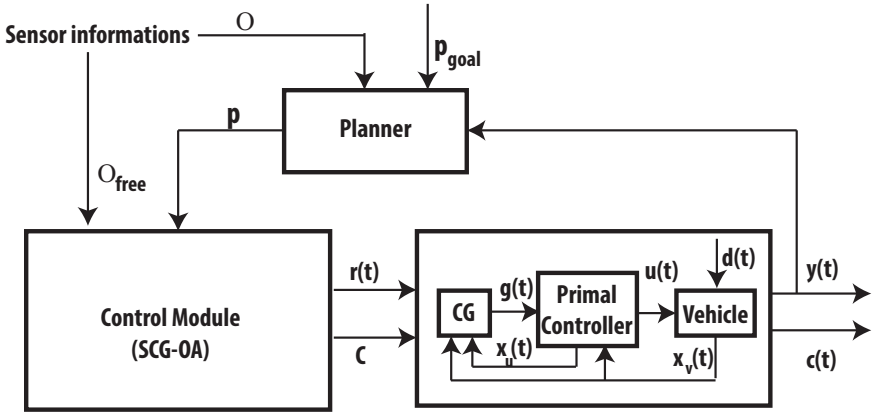

Fig. 5.1. Control Architecture

In principle, as better clarified in the next section, a modification of the constraint set structure should prescribe a new CG design during the on-line operations, but this could be computationally very demanding and, therefore, not usable in practical situations. To overcome this drawback, the key idea here is to avoid the *CG re-design*. Next section will be devoted to find solution to such a problem by exploiting the following arguments:

- time-varying constraints arise due to a shift with respect the current equilibrium;
- different constraints sets are overlapped and their shapes are invariant w.r.t. any equilibrium shift.

5.2 An obstacle avoidance command governor strategy

The basic CG framework (Section 2.4) requires that the constrained set $\mathcal{C}$ satisfy the following requirements:

- compact and convex - to ensure validity of Property 2.32(a);
- time-invariance - to guarantee viability retention (Properties2.32(b)-(d)).

The consequence of these restrictions is that the CG strategy in its classical version cannot be used when the ***UAV-OAMP*** problem is considered: the obstacles give rise to non-convex geometrical constraints that could change in principle at each time instant. Then, the goal of this section is to retain the CG feasibility by adequately managing time-varying constraints.
To this end, the key point is to convexify at each time instant an active region around the current vehicle position by means of a suitable obstacle constraints updating procedure. In order to ensure feasibility retention a critical aspect to be taken into account is to define a scheme which characterizes the admissible switching amongst

different convexified regions. The next sections will be devoted to formally develop this idea.

5.2.1 Viability retention under time-varying constraints

Consider the command governor structure (2.22) and two compact and convex constraint sets $\mathcal{C}_i$ and $\mathcal{C}_j$ such that

$$\mathcal{C}_i \neq \mathcal{C}_j \text{ and } \mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset \tag{5.5}$$

Then, as shown in e.g. [3], the following property holds true

$$\mathcal{W}_i^\delta \cap W_j^\delta \neq \emptyset \tag{5.6}$$

where

$$\mathcal{W}_s^\delta = \left\{\omega \in \mathbb{R}^m : c_w \in \mathcal{C}_s^\delta\right\} \text{ and } \mathcal{C}_s^\delta = \mathcal{C}_s \sim B^\delta,\ s = i, j.$$

As a consequence, the constraint scenario switching admissibility can be proved.

Proposition 5.4. *Given the system (2.22) and the constraint scenarios $\mathcal{C}_i^\delta$ and $\mathcal{C}_j^\delta$ satisfying (5.5). Then, there always exists a finite concatenation of virtual constant commands $\omega \in \mathcal{W}_i^\delta \cup \mathcal{W}_j^\delta$ such that, starting from an arbitrary $\mathcal{C}_i^\delta$ or $\mathcal{C}_j^\delta-$admissible initial state $x(0)$, the state trajectory of (2.22) is driven to $\bar{x}_\omega$ with $\omega \in \mathcal{W}_i^\delta \cup \mathcal{W}_j^\delta$. Moreover under a constraint scenario change there exists a finite switching time $t_{sw} < \infty$ and an arbitrary small scalar $\Delta_x > 0$ such that $\|x(t_{sw}) - \bar{x}_{\bar{\omega}}\| < \Delta_x$, where $\bar{\omega} \in \mathcal{W}_i^\delta \cap \mathcal{W}_j^\delta$. and $x(t_{sw})$ is $\mathcal{C}_i^\delta$ and $\mathcal{C}_j^\delta-$admissible (see Fig. 5.2) for a graphical illustration).*

Proof. W.l.o.g. assume that the initial condition $x(0)$ is $\mathcal{C}_i^\delta-$admissible. By resorting to viability arguments [6], any $\bar{x}_{\bar{\omega}}$, with $\bar{\omega} \in \mathcal{W}_i^\delta$, can be reached via a finite sequence of virtual constant commands. As a consequence, the state trajectory $x(\cdot)$ can be driven in finite time within a neighborhood of the equilibrium $\bar{x}_{\bar{\omega}}$, $\bar{\omega} \in \mathcal{W}_i^\delta \cap \mathcal{W}_j^\delta$. By considering a perturbed state

$$x_{\bar{\omega}} = (I - \Phi)^{-1} G\bar{\omega} + \tilde{x}$$

with $\tilde{x}$ an additive perturbation such that $||\tilde{x}|| \leq \rho$, it is possible to apply a new command $\underline{\omega} \in \mathcal{W}_i^\delta \cap \mathcal{W}_j^\delta$ so that the constrained vector becomes

$$c(k, x_{\bar{\omega}}, \underline{\omega}) = H_c \Phi^k x_{\bar{\omega}} + H_c \sum_{l=0}^{k-1} \Phi^l G\underline{\omega} + L\underline{\omega}$$

By using simple algebraic manipulations, we have that

$$c(k, x_{\bar{\omega}}, \underline{\omega}) = c_{\underline{\omega}} + H_c \Phi^k \tilde{x} + H_c (I - \Phi)^{-1} G(\bar{\omega} - \underline{\omega})$$

with $c_{\underline{\omega}} \in \mathcal{C}_j^\delta$. As a consequence also $c(k, x_{\bar{\omega}}, \underline{\omega}) \in \mathcal{C}_j$ if the following condition holds true

$$H_c\Phi^k\tilde{x} + H_c(I-\Phi)^{-1}G(\bar{\omega}-\underline{\omega}) \subset B_\delta$$

This is straightforwardly ensured by requiring that

$$\Delta_x = ||\tilde{x}|| \leq \frac{\delta}{2\sigma(H_c)M} \text{ and } ||\bar{\omega}-\underline{\omega}|| \leq \frac{\delta(1-\lambda)}{2\sigma(H_c)\sigma(G)M^2}$$

where $\lambda \in [0,1)$ is such that $\forall x \in \mathbb{R}^n$ one has that $\Phi^k x||x|| \leq M\lambda^k||x||$, $\forall k \in \mathbb{Z}_+$; $\sigma(H_c)$ and $\sigma(G)$ are the maximum singular values of the matrices H_c and G respectively. □

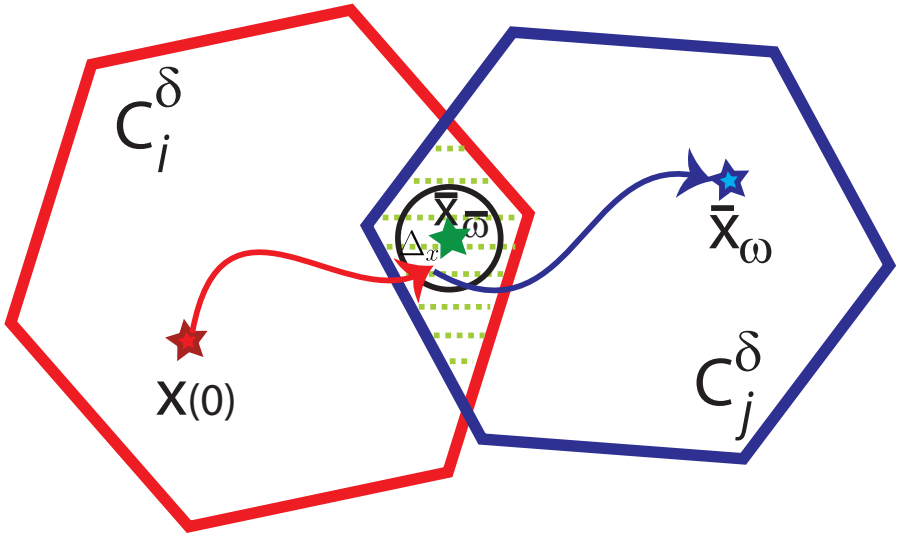

Fig. 5.2. Viability retention under constraints change

5.2.2 Obstacle avoidance: time-varying constraints

Here, the results of Proposition 5.4 are specialized in order to provide a solution to the proposed ***OAMP*** problem. To this end, by exploiting the fact that the vehicle state can be split as geometrical (x_g) and non-geometrical (x_{ng}) components, model (2.22) can be re-written as

$$\begin{cases} x(t+1) = \begin{bmatrix} x_g(t+1) \\ x_{ng}(t+1) \\ x_u(t+1) \end{bmatrix} = \Phi x(t) + Gg(t) + G_d d(t) \\ p(t) = y(t) = H_y x(t) \\ c(t) = \begin{bmatrix} c_g \\ c_{ng} \\ c_u \end{bmatrix} = H_c x(t) + Lg(t) + L_d d(t) \end{cases} \tag{5.7}$$

where $x_g \in \mathbb{R}^3$ are the spatial components, $x_u \in \mathbb{R}^{n_u}$ the primal controller state and $x_{ng} \in \mathbb{R}^{n-3-n_u}$ the non-geometrical state variables. Moreover, c_g, c_{ng} and c_u take care of the constraints prescribed for each state component.

Let $x(0)$ and $\mathcal{C}^\delta$ be the initial condition and the constrained set, respectively. Under the CG action, the vehicle will asymptotically reach the best feasible approximation $\hat{g}$ of the desired state space position p_{goal}. Though in principle it is possible to directly solve the ***UAV-OAMP*** problem, the basic CG strategy usually leads to $\hat{g} \neq p_{goal}$ because obstacles occlude the ideal CG path $x(0) \rightarrow x_v^{goal}$ and the constrained set $\mathcal{C}^\delta$ cannot cover the whole non-convex $\mathcal{O}_{free}$ region.
Then, this difficulty can be overcome by using the following arguments:

- Compute the equilibrium condition corresponding to $\hat{g}$:

$$\begin{array}{l} \bar{x}_{eq} = (I - \Phi)^{-1} G \hat{g} \\ \bar{c}_{eq} = H_c \bar{x}_{eq} + L \hat{g} \end{array} \tag{5.8}$$

- Due to linearity of (2.22), the state space is shifted according to $\bar{x}_{eq}$:

$$\begin{array}{l} \tilde{x} = x - \bar{x}_{eq} \\ \tilde{g} = g - \hat{g} \end{array}$$

$$\begin{cases} \tilde{x}(t+1) = \Phi \tilde{x}(t) + G\tilde{g}(t) + G_d d(t) \\ \tilde{p}(t) = H_y \tilde{x}(t) \\ \tilde{c}(t) = H_c \tilde{x}(t) + L\tilde{x}_g(t) + L_d d(t) \end{cases} \tag{5.9}$$

- A new shifted (w.r.t. $\bar{c}_{eq}$) constrained set results:

$$\tilde{c}(t) \in \tilde{\mathcal{C}}^\delta := shift(\mathcal{C}^\delta, \bar{c}_{eq}) \tag{5.10}$$

Moreover, note that by construction

$$\tilde{\mathcal{C}}^\delta \cap \mathcal{C}^\delta \neq \emptyset$$

because these sets at least share the equilibrium $\bar{c}_{eq}$: the properties outlined in Proposition 5.4 hold true.

As a consequence, the ***UAV-OAMP*** problem can be solved by using a suitable sequence of shifts on the initial constrained set $\mathcal{C}$ until the target position p_{goal} is reached. However, to pursue such an idea it could be necessary to modify shapes or dimensions of $\tilde{\mathcal{C}}^\delta$ in order to guarantee that at each time instant t

$$\tilde{\mathcal{C}}^\delta \subseteq \mathcal{O}_{free} \tag{5.11}$$

This could be a serious drawback because if a complete re-computing of the CG controller is required, the strategy could become on-line high demanding and, in the worst case, inapplicable.
Then, the following developments provide a solution to this key question. Recall that the set $\mathcal{V}(x)$ can be written w.l.o.g. as (see Appendix C for details)

$$TH_c\Phi^k x(t) + TR_k^c\omega \leq b^k, \quad k = 0, \ldots, \bar{k} \tag{5.12}$$

$$T(H_c(I-\Phi)^{-1} + L)\omega \leq b^{k_\epsilon} - (\epsilon + \delta)\left[\sqrt{T_i^T T_i}\right] \tag{5.13}$$

where

$$k_\epsilon = \frac{\log\epsilon + \log(1-\lambda)\left[\bar{\sigma}(H_c)\bar{\sigma}(G_d)Md_{max}\right]}{\log\lambda} \tag{5.14}$$

and b^k is computed using the following recursion

$$\begin{cases} b_i^0 = b_i - \sup\limits_{d\in\mathcal{D}} T_i^T L_d d, \ i = 1, \ldots, z \\ b_i^k = b_i^{k-1} - \sup\limits_{d\in\mathcal{D}} T_i^T H_c \Phi^{k-1} G_d d, \ i = 1, \ldots, z, \ k > 0 \end{cases} \tag{5.15}$$

Then, from (5.12)-(5.15) it clearly follows that a change on the constrained set shape (T) will require a complete CG re-design while a variation on its dimensions (b) does not affect the CG structure.
As a conclusion of the above discussion, the next section will be devoted to develop a CG-based procedure capable to accomplish the ***UAV-OAMP*** task by preserving the requirement (5.11) under constrained set dimension variations.

5.3 Switching command governor obstacle avoidance algorithm (SCG-OA)

Let p_{start}, p_{goal}, $\mathcal{O}$ and $\mathcal{C}$ be the starting and target positions, the obstacle configuration and the constraint set, respectively. In the sequel, the following assumptions are made:

Assumption 5.5. *There exists an admissible path connecting p_{start} and p_{goal};* □

Assumption 5.6. *There exists a path module, hereafter named planner, whose task is to determine a sequence of intermediate target positions to pass trough so that the obstacles can be avoided and the goal location reached.* □

By using as inputs the obstacle configuration, the current vehicle position and the target, this unit returns the next location p to be reached:

$$p = \text{planner}(\mathcal{O}, p(t), p_{goal});$$

Assumption 5.7. *There exists a vision module (cameras or group of sensors) used to measure the obstacle-free region dimensions around the current vehicle position $p(t)$.* □

Then, the following computationally tractable algorithm results:

Naive-SCG-OA *algorithm*

1: Initialization: $\bar{x}_{eq} = 0_x$, $\bar{c}_{eq} = 0_{n_c}$, $\hat{g} = 0_m$ *and* $\tilde{\mathcal{C}} = \mathcal{C}$;

2: Compute:

$$p = \textit{planner}(\mathcal{O}, p(t), p_{goal})$$

$$p_{reach} = \arg \min_{\omega \in W^{\delta}} ||\omega - (p - \hat{g})||^2_{\Psi} \tag{5.16}$$

$$x^p_{reach} = (I - \Phi)^{-1} G p_{reach} + \bar{x}_{eq} \tag{5.17}$$

3: ***if*** $||x(t) - x^p_{reach}||_2 < \Delta_x$ ***then***

4: $\hat{g} \leftarrow p_{reach}$; *compute* $\bar{x}_{eq}$, $\bar{c}_{eq}$ *by (5.8) and update* $\tilde{\mathcal{C}}$ *according to (5.10);*

5: Compute a new reference p

$$p = \textit{planner}(\mathcal{O}, x(t), p_{goal})$$

6: Update x^p_{reach} *according to (5.16)-(5.17)*

7: ***end if***

8: Compute the command governor action $\hat{g}(t)$

$$\tilde{g}(t) = \arg \min_{\omega \in \mathcal{V}(x(t))} ||\omega - (p - \hat{g})||^2_2$$

9: Apply $g(t) = \hat{g} + \tilde{g}(t)$ *;* $t = t + 1$ *and goto Step 3.*

It is worth to underline that the ***Naive-SCG-OA*** algorithm may give rise to a certain level of conservativeness because the Step 3 (due to the viability property conditions outlined in Proposition 5.4 could impose a *stop-and-go* phase until the vehicle reaches the pre-assigned target. A possible way to overcome this drawback is to exploit the knowledge about the future (at the next time instant) candidate equilibrium point, i.e. x^p_{reach} computed by means of (5.17), with $\tilde{C}_{reach}$ the corresponding constraint set (hereafter named *constraint window*). Specifically by denoting with $\mathcal{V}^{\tilde{C}_{reach}}(x)$ the set of feasible commands pertaining to x_{reach}, then Step 3. becomes:

Smart Step 3

3: ***if*** $x(t) \in \mathcal{V}^{\tilde{C}_{reach}}(x(t))$ ***then***

4: $\tilde{C} \leftarrow \tilde{C}_{reach}$ *and* $\hat{g} \leftarrow p_{reach}$;

5: Compute $\bar{x}_{eq}$, $\bar{c}_{eq}$ *by (5.8) and*

$$p = \textit{planner}(\mathcal{O}, x(t), p_{goal})$$

6: Update x^p_{reach} *according to (5.16)-(5.17);*

7: ***end if***

From now on, we will simply denote with ***SCG-OA*** the algorithm that make use of the ***Smart Step 3***.

5.4 Experimental setup

In this section, the overall infrastructure used for experiment purposes is discussed, see Fig. 5.3. It consists of a Quanser Qball-X4 quadrotor equipped with a data acquisition board (HIQ) and a Gumstix computer, a vision Vicon system and an host computer used to manage runtime bi-directional information.

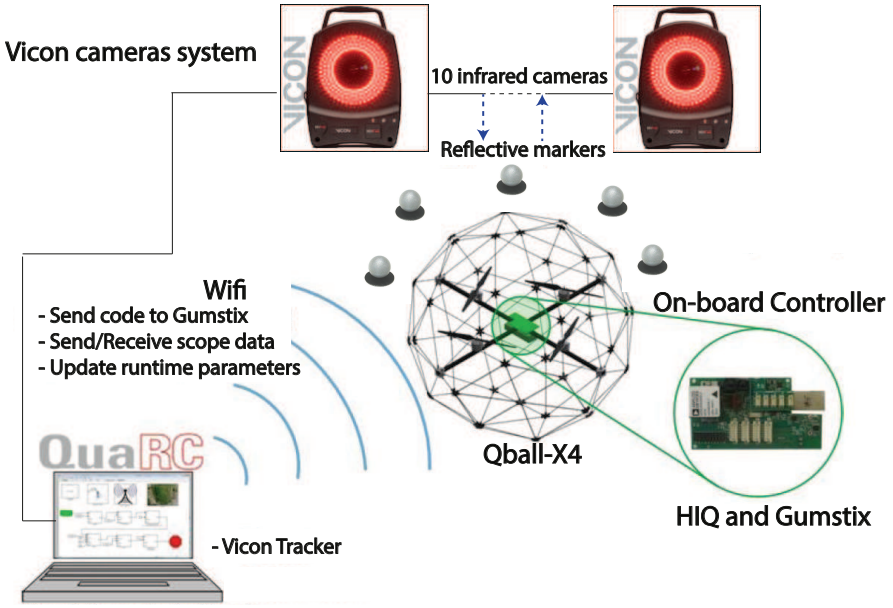

Fig. 5.3. Real-time platform

5.4.1 Description and dynamics of the quadrotor UAV system

A brief working mechanism description of a quadrotor unit and the corresponding non-linear dynamical model are provided [105, 127]. In what follows, upper case letters refer to the global (inertial) workspace axes (X^w, Y^w, Z^w) while lower case letter to the body frame (x^b, y^b, z^b). Notice that such reference systems have the same orientation when the UAV is sitting upright on the ground, see Fig. 5.4.
Quadrotors are equipped with four rotors, coupled w.r.t. the x^b- and y^b- axes, and generate lift forces (F_i) and moments (τ_i). Moreover each rotor pair, i.e. (1)-(2) and (3)-(4), rotates in opposite direction w.r.t. the other pair so that the total produced

moment can be zeroed, see the right side of Fig. 5.5. In order to achieve an effect on the roll angle ϕ along the x^b-axis, one can increase the angular velocity of the rotor (2) and can decrease the angular velocity of rotor (1) while keeping the whole thrust constant. Likewise, the angular velocity of rotor (3) is increased and the angular velocity of rotor (4) is decreased in order to produce a variation on the pitch angle θ along the y^b-axis. Moreover, the yaw motion ψ along the z^b-axis is obtained by jointly increasing the speed of rotors (1)-(2) and decreasing the speed of rotors (3)-(4).

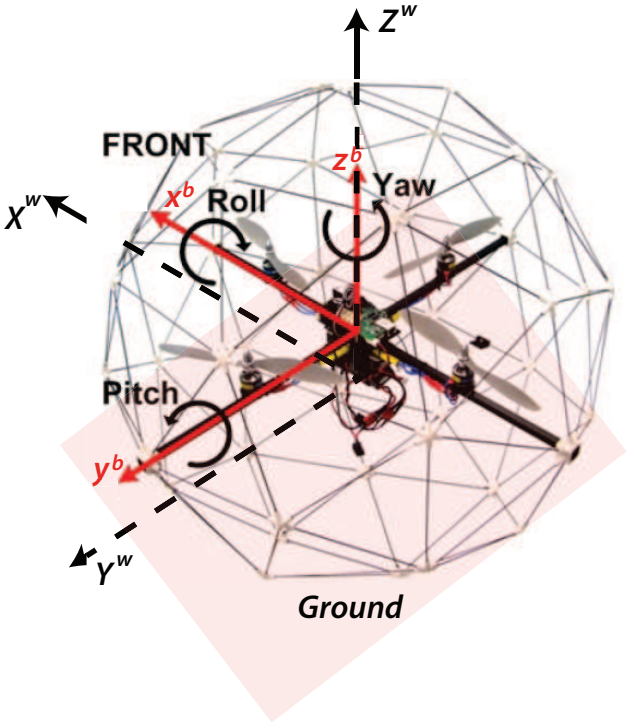

Fig. 5.4. Qball-X4 axis and sign convention

Then by neglecting propeller drags, the resulting quadrotor model is as follows:

$$\begin{cases} \begin{bmatrix} \ddot{X}_c \\ \ddot{Y}_c \\ \ddot{Z}_c \end{bmatrix} = \frac{1}{m}\left(\sum_{i=1}^{4} F_i\right) R\, e_3 + (g_r(z) - g)e_3 \\ \ddot{\phi} = l(F_3 - F_4)/J_x \\ \ddot{\theta} = l(F_1 - F_2)/J_y \\ \ddot{\phi} = \rho(F_1 + F_2 - F_3 - F_4)/J_z \end{cases} \tag{5.18}$$

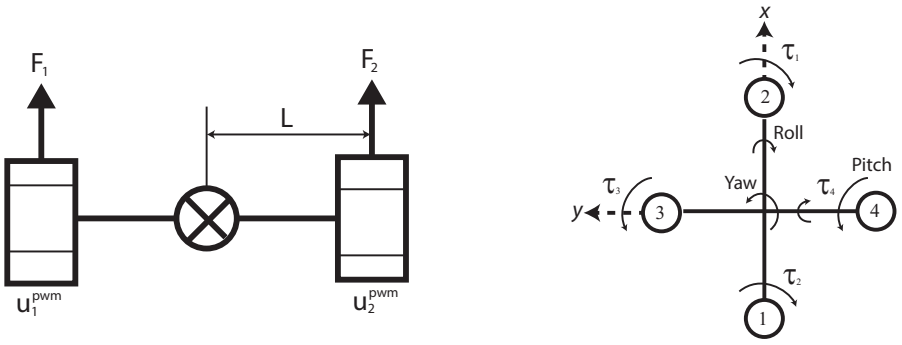

Fig. 5.5. Qball-X4 axis and sign convention

where $e_3 = [0,\, 0,\, 1]^T$, J_x, J_y and J_z are the moments of inertia along x^b, y^b, z^b directions respectively, m the system mass, g the gravity vector, ρ the force-to-moment scaling factor, $[X_c,\, Y_c,\, Z_c]^T$ the vector denoting the quadrotor center of mass in the inertial frame, R the rotation matrix from the body frame (b) to the workspace frame (w) and $[\varphi,\, \theta,\, \phi]^T$ the vector of the roll, pitch and yaw Euler angles, respectively. Moreover $g_r(Z_c)$ describes the ground effect that bothers the quadrotor when it is sufficiently close to the ground

$$g_r(Z_c) = \left\{ \begin{array}{cc} \frac{Gr}{Z_c^2} - \frac{Gr}{Z_0^2}, & \text{if } 0 < Z_c \leq Z_0 \\ 0 & \text{otherwise} \end{array} \right\} \tag{5.19}$$

where G_r is the ground effect and Z_0 the lower quadrotor height for which the ground effect is significant.
By resorting to the definitions of normalized lift force (u_z), roll (u_ϕ), pitch (u_θ) and yaw (u_ψ) moments:

$$\begin{array}{l} u_z = (F_1 + F_2 + F_3 + F_4)/m \\ u_\phi = (F_3 - F_4)/J_x \\ u_\theta = (F_1 - F_2)/J_y \\ u_\psi = \rho(F_1 + F_2 - F_3 - F_4) \end{array} \tag{5.20}$$

equation (5.18) can be straightforwardly re-written as:

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u_z(\cos x_7\ \sin x_9\ \cos x_{11} + \sin x_9\ \sin x_{11}) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u_z(\cos x_7\ \sin x_9\ \sin x_{11} + \sin x_9\ \cos x_{11}) \\ \dot{x}_5 &= x_6 \\ \dot{x}_6 &= u_z(\cos x_7\ \cos x_9) - g + g_r(z) \\ \dot{x}_7 &= x_8 \\ \dot{x}_8 &= u_\phi\, l \\ \dot{x}_9 &= x_{10} \\ \dot{x}_{10} &= u_\theta\, l \\ \dot{x}_{11} &= x_{12} \\ \dot{x}_{12} &= u_\psi \end{cases} \tag{5.21}$$

where

$$x_v = \left[\, X_c\ \dot{X}_c\ Y_c\ \dot{Y}_c\ Z_c\ \dot{Z}_c\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi} \,\right]^T$$

$$u = \left[\, u_z\ \Delta u_\phi\ \Delta u_\theta\ \Delta u_\psi \,\right]^T$$

denote the state space and input vectors.

5.4.2 A simplified model for the Quanser Qball-X4 quadrotor

In this section, the motion equations (5.21) are specialized to the Quanser Qball-X4 quadrotor, see [61]. In particular, a less involved structure of (5.21) is achieved by using standard quadrotor modelling assumptions.

Actuator dynamics

In the Qball-X4 there are four brushless motors (E-flite Park 400), using a 104.7 inch propeller. The thrust generated by each propeller is modeled using first-order systems:

$$F_i = K\frac{\omega}{s+\omega}u_i^{pwm} \tag{5.22}$$

where u_i^{pwm} is the PWM input to the DC-motor actuator, ω is the actuator bandwidth and K is a positive gain. A state variable v will be hereafter used to represent the actuator dynamics, which is defined as follows

$$v_i = \frac{\omega}{s+\omega}u_i^{pwm}$$

Roll/Pitch models

Under the assumption that the rotations w.r.t. the $x-$ and $y-$ axes are decoupled, the roll/pitch dynamical behaviour can be modelled as illustrated in Fig. 5.5. There, the motion/rotation along each axis is due to the action of two propellers, i.e. the pair $(F_1,\, F_2)$, respectively $(\tau_1,\, \tau_2)$, for the $x-$ axis and $(F_3,\, F_4)$, respectively $(\tau_3,\, \tau_4)$,

for the $y-$ axis. Moreover, the roll ϕ and pitch θ angles are subject to the following first order dynamics:

$$J_x\ddot{\phi} = (F3 - F4)\,, \quad J_y\ddot{\theta} = (F1 - F2) \tag{5.23}$$

Therefore by combining eqs. (5.22) and (5.23), the following state space description comes out:

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{2KL}{J_x} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u_\phi \tag{5.24}$$

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{2KL}{J_y} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u_\theta \tag{5.25}$$

with $\Delta u_\phi = (u_3^{pwm} - u_4^{pwm})$ and $\Delta u_\theta = (u_1^{pwm} - u_2^{pwm})$.

Height model

The quadrotor motion along the vertical direction ($Z-$ axis) is affected by all the four propellers. By assuming that $F_z = F_i$, $i = 1, \ldots, 4$, the latter can be modeled as

$$M\ddot{Z}_c = 4F_z \cos\phi \cos\theta - mg \tag{5.26}$$

Finally, by combining eqs. (5.22) and model (5.26) the following state space description results:

$$\begin{bmatrix} \dot{Z}_c \\ \ddot{Z}_c \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{4K\cos\phi\cos\theta}{m} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} Z \\ \dot{Z} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} u_z + \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} \tag{5.27}$$

where $F_z = K\frac{\omega}{s+\omega}u_z$

$X_c - Y_c$ position model

Under the following assumption on the yaw angle:

$$\psi \equiv 0$$

the $X_c - Y_c$ position dynamical model becomes:

$$\begin{aligned} M\ddot{X}_c &= 4F_z \sin\theta \\ M\ddot{Y}_c &= -4F_z \sin\phi \end{aligned} \tag{5.28}$$

Then, eqs. (5.22) and (5.28) allow to derive the following state space description

$$\begin{bmatrix} \dot{X}_c \\ \ddot{X}_c \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{4K\sin\theta}{M} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} u_z \tag{5.29}$$

$$\begin{bmatrix} \dot{Y}_c \\ \ddot{Y}_c \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0\ 1 & 0 \\ 0\ 0 & -\frac{4K\sin\phi}{M} \\ 0\ 0 & -\omega \end{bmatrix} \begin{bmatrix} Y \\ \dot{Y} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} u_z \tag{5.30}$$

Yaw model

By assuming a linear characteristics between the motor torque τ_i and the PWM signal u_i^{pwm} :

$$\tau_i = K_z u_i^{pwm},\ K_z \in \mathbb{R}^+,\ i = 1, \ldots, 4,$$

the motion along the $z-$axis can be modeled as

$$J_z \ddot{\psi} = K_z \Delta\tau_\psi$$

or equivalently as

$$\begin{bmatrix} \dot{\psi}_z \\ \ddot{\psi}_z \end{bmatrix} = \begin{bmatrix} 0\ 1 \\ 0\ 0 \end{bmatrix} \begin{bmatrix} \psi_z \\ \dot{\psi}_z \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_z}{J_z} \end{bmatrix} \Delta u_\psi \tag{5.31}$$

with $\Delta u_\psi = u_1^{pwm} + u_2^{pwm} - u_3^{pwm} - u_4^{pwm}$.

Qball-X4 linear Model, parameters and command inputs

By collecting all the above developments, the dynamical behaviour of the Qball-X4 quadrotor is described by eqs. (5.22)-(5.31).
Note that the effective commands applied to the four propellers are given by the following relationships:

$$\begin{cases} u_1^{pwm} = u_z + \Delta u_\theta + \Delta u_\psi \\ u_2^{pwm} = u_z - \Delta u_\theta + \Delta u_\psi \\ u_3^{pwm} = u_z + \Delta u_\phi - \Delta u_\psi \\ u_4^{pwm} = u_z - \Delta u_\phi - \Delta u_\psi \end{cases}$$

Finally, Table 5.1 collects the Qball-X4 physical parameters.

5.4.3 Sensors and Vicon system

The HiQ provides several high-resolution avionics sensors, which are used to measure and control the stability of aerial vehicles. Moreover in order to measure the spatial Cartesian coordinates of the Qball an indoor high speed Vicon cameras system is used [121]. This infrared cameras system has the capability to identify and track reflective markers and body (a body is a set of marker described as connected) in the working environment with extreme precision. In particular the qball X4 body is modeled using five reflective markers described inside the Vicon system as a rigid body (Fig. 5.6-5.7)

Parameters	Value
K	120 N
ω	15 rad/s
J_x	0.03 $K_g m^2$
J_y	0.03 $K_g m^2$
J_z	0.04 $K_g m^2$
m	1.53 K_g
K_z	4 Nm
L	0.2 m

Table 5.1. Qball-X4 parameters

QBall-X4 data acquisition and Vicon cameras system

A Quanser's on board avionics data acquisition card (DAQ), an HiQ and an embedded Gumstix computer are in charge of acquiring measurements from the on-board sensors and to drive the Qball-X4 motors. Specifically, the HiQ avionics input/output (I/O) includes:

- 10 PWM outputs (servo motor outputs); 3-axis gyroscope, range configurable for $\pm 75^\circ/s$, $\pm 150^\circ/s$, or $\pm 300^\circ/s$, resolution;
- $0.0125^\circ/s/LSB$ at a range setting of $\pm 75^\circ/s$; 3-axis accelerometer, resolution $3.33mg/LSB$;
- 6 analog inputs, $12 - bit$, $+3.3V$; 3-axis magnetometer, 0.5 mGa/LSB;
- 8 channel RF receiver input (optional); 4 Maxbotix sonar inputs;
- 2 pressure sensors, absolute and relative pressure.

Moreover, the Qball-X4 spatial Cartesian coordinates are obtained by means of an indoor high speed Vicon© cameras system [121]. Within the working environment, this infrared cameras system has the capability to quickly identify and track reflective markers with high accuracy. Note that the Qball-X4 body has been modeled by using five reflective markers within the Vicon system, see Fig. (5.6)-(5.7).

On board controller implementation

The Quanser's real-time control software allows to develop and evaluate performance controllers on the Qball-X4 by using the Matlab/Simulink environment. This software is capable to access to the $C-$code, automatically generated by Matlab and Simulink coders from a Simulink model, and to compile it by exploiting the on-board Gumstix computer architecture.
Finally, recall that the main computational task of the ***SCG-OA*** algorithm is the solution of the constrained quadratic optimization problem (2.30). Because the Matlab Code generation does not support the Matlab Optimization toolbox [27], a suitable customization for solving the constrained QP has been used: ADMM (alternating direction method of multipliers) [15].

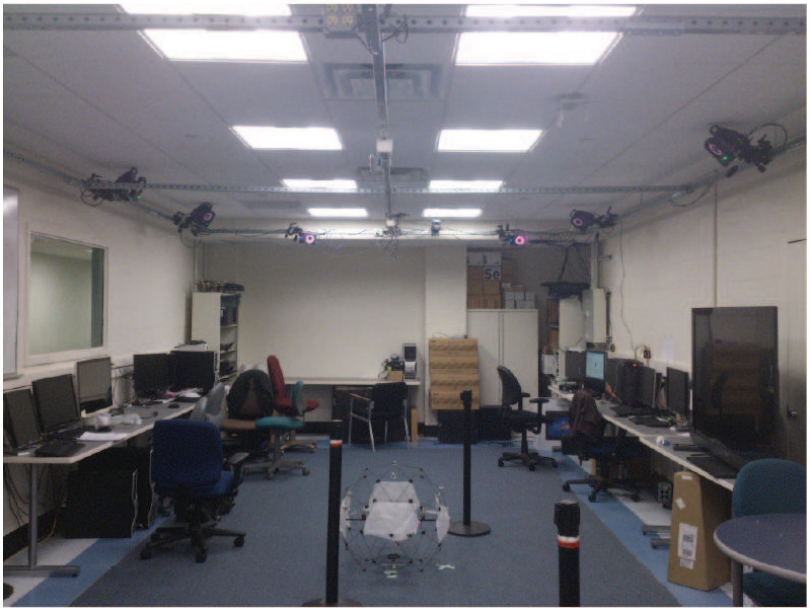

Fig. 5.6. Robust System Lab at Northeastern University: Vicon camera system and Qball-X4

Fig. 5.7. Qball-X4 and Vicon Markers

5.5 Experiments

The aim of this section is to show the effectiveness of the proposed ***SCG-OA*** algorithm. First, a simulation campaign has the aim to both validate the CG controller and to provide comparisons with the default LQ Quanser controller. Then, a real-

time laboratory experimental set-up is considered and the related numerical results presented and discussed.
In what follows the Qball-X4 model (5.22)-(5.31) has been linearized around the following equilibrium point:

$$\bar{x}_{v_{eq}} = \left[0\ 0\ 0\ 0\ 0\ 0.12\ 0\ 0\ 0\ 0\ 0\ 0\right]^T,\ \bar{u}_{eq} = \left[0.031,\ 0,\ 0,\ 0\right]^T \tag{5.32}$$

and then discretized by using the Euler forward differences method with $\Delta t = 0.001\,s$ the sampling time interval.
The set of constraints is below summarized:

- *Working environment (room)* :

$$-1.2 \leq X_c \leq 1.2,\ \ -1 \leq Y_c \leq 3.5,\ \ 0 \leq Z_c \leq 2.5$$

- *Roll and pith angle (validity of the linearized model)*:

$$|\phi| \leq 0.3,\ |\theta| \leq 0.3 \tag{5.33}$$

- *Control command effort (rotor saturation)*:

$$|\Delta u_\phi| \leq 0.025,\ |\Delta u_\theta| \leq 0.025,\ 0.011 \leq |u_z| \leq 0.051 \tag{5.34}$$

The following CG parameters have been selected: $\delta = 10^{-5}$ and $\Psi = I_3$. The constraint horizon $k_0 = 120$ was computed via the numerical procedure proposed in [48]. Notice that, by keeping out the on-board saturation devices, the LQ Quanser controller (standard equipment on the Qball-X4) has been used as primal controller for CG purposes.

5.5.1 CG validation and comparisons

The CG control strategy has been designed by means of a linearizion procedure of the Qball-X4 non linear model. The test bed control task is below described:

Starting from the initial condition $p_{start} = (0, 0, 0.12)m$ the Qball-X4 have to track the following reference signal:

1. $t \in [0,\ 15]\,s. \rightarrow p_{goal} = [0,\ 0,\ 0.5]^T$;
2. $t \in (15,\ 25]\,s. \rightarrow p_{goal} = [1.5,\ 0,\ 0.5]^T$;
3. $t \in (25,\ 40]\,s. \rightarrow p_{goal} = [-1,\ 0,\ 0.5]^T$;
4. $t \in (40,\ 50]\,s. \rightarrow p_{goal} = [-1,\ 0,\ 0.35]^T$;
5. $t \in (50,\ 70]\,s. \rightarrow p_{goal} = [1.5,\ -0.35,\ 0.5]^T$;
6. $t \in (70,\ 85]\,s. \rightarrow p_{goal} = [0,\ 0,\ 0.5]^T$;
7. $t \in (85,\ 90]\,s. \rightarrow p_{goal} = [0,\ 0,\ 0.12]^T$;

All the relevant results are reported in Figs.5.8-5.10. Fig. 5.8 shows the tracking error signals along the X^w and Y^w axes: there it is clearly enlighten that the proposed *SCG-OA* strategy outperforms the LQ Quanser controller. Notice that this behaviour is less evident on the Y^w-axis because shorten displacements are required. Moreover, it is worth to underline that even if all the prescribed constraints are always fulfilled (Figs. 5.9, 5.10), the LQ Quanser must be supported by off-designed saturation devices. This leads to unavoidable drawbacks: nonlinear phenomena and conservative results in linear regimes.

Finally, the main advantage of the CG customization comes out from Fig. 5.10 where the *SCG-OA* command signal is more active than the LQ Quanser competitor: as a consequence this justifies the achieved better tracking performance shown in Fig.5.8.

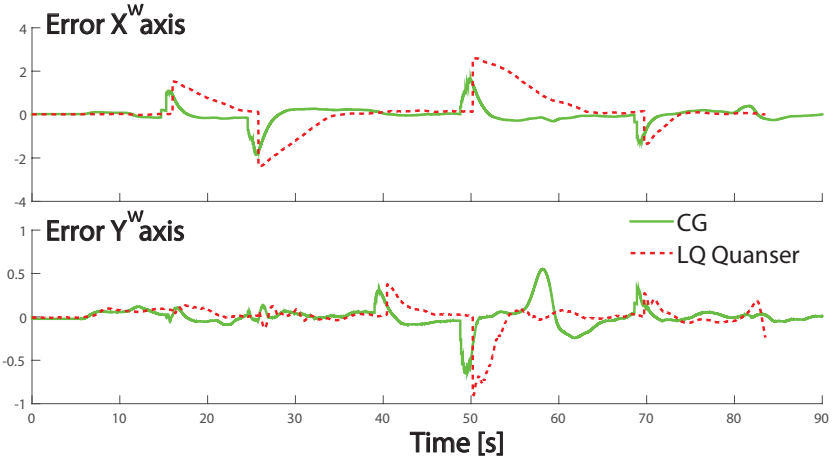

Fig. 5.8. Tracking error signals

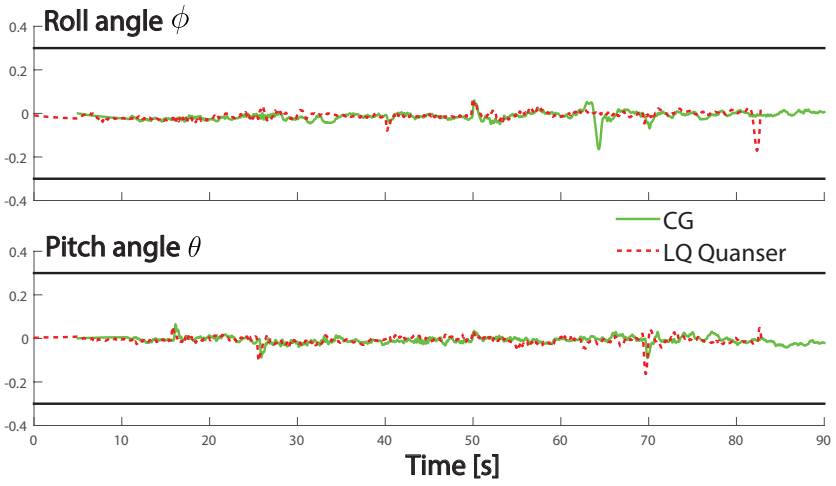

Fig. 5.9. Roll ϕ and Pitch θ angles

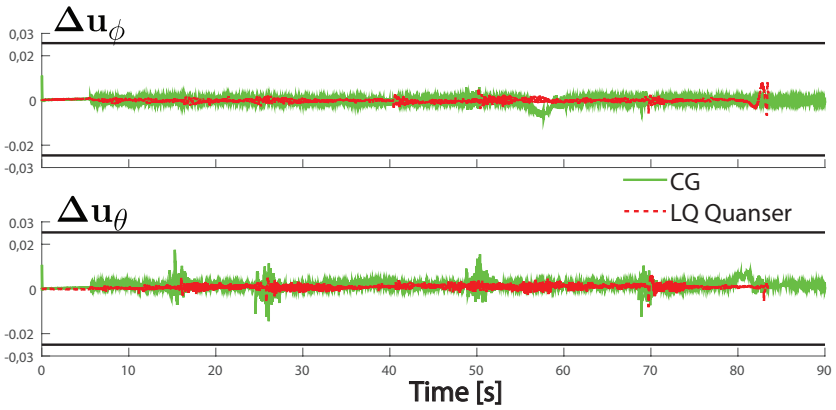

Fig. 5.10. Commands Input: Δu_ϕ and Δu_θ

5.5.2 Real time obstacle avoidance laboratory experiment

Experiments are instrumental to show the benefits of the proposed approach when a critical obstacle scenario is considered. Note that to take care of the Qball-X4 dimensions, the obstacle dimensions have been increased in such a way that the quadrotor can be considered as a point mass, see [83] for details. Moreover, we have built the following obstacle scenario:

- a room: $X \times Y \times Z = 4 \times 1.2 \times 3m^3$;
- Obstacle positions: summarized in Table 5.2 and shown in Fig. 5.11.

Obstacle	width(m)	depth(m)	height(m)	center of gravity(m)
Ob^1	0.7	0.7	0.7	$[0.6; 0.2, 0.35]$
Ob^2	0.7	0.7	0.7	$[1.8; -0.4, 0.35]$
Ob^3	0.7	0.7	0.7	$[3.1; 0.2, 0.35]$

Table 5.2. Obstacle configuration

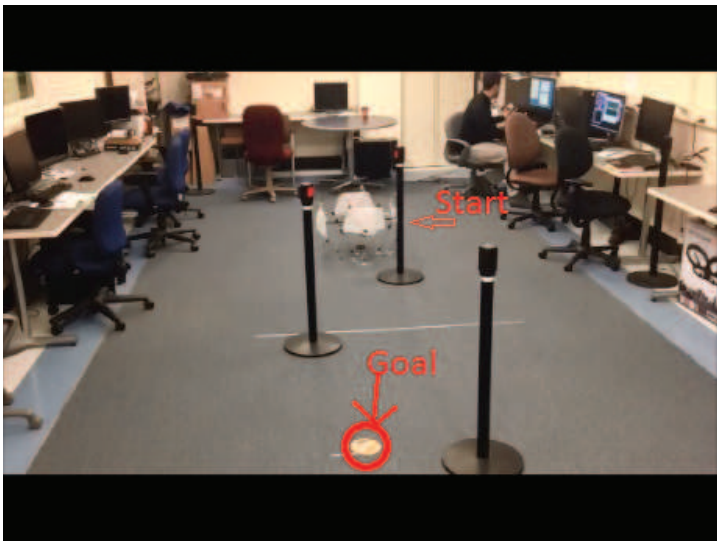

Fig. 5.11. Three Obstacle scenario: Start and Goal locations

Then, the aim of the real-time experiment is:

Starting from the initial condition $p_{start} = (0, 0, 0.12)m$, see Fig. 5.11, it is required that the Qball-X4 quadrotor reaches the target $p_{goal} = (2.9, -0.3, 0.50)m$ and comes back to p_{start} by avoiding the obstacle occurrences along the path.

All the relevant results are reported in Figs. 5.12-5.20.

Figs. 5.12-5.13 depict the Qball-X4 state trajectories under the action of the Naive SCG-OA (dashed line) and ***SCG-OA*** (continuous line) algorithms in the 2D and 3D spaces, respectively, whereas Fig. 5.14 accounts for the dynamical behaviours along the three axes. As expected, the ***SCG-OA*** algorithm shows remarkably lower settling times (about $43\ s$) than its ***Naive SCG-OA*** competitor (about $60s$). Figs. 5.15-5.17 show the behaviours of the planner and command governor modules. There, it is interesting to remark that the **CG** unit is always able to modify the reference provided by the **planner** module in order to ensure constraint satisfaction at each time instant. To better clarify the proposed ***SCG-OA*** strategy, Fig. 5.18 describes the updating phases of the constraint window w.r.t. the current quadrator position. There, we have denoted with $p_{reach}^{(\cdot)}$ the points provided by the **planner** in

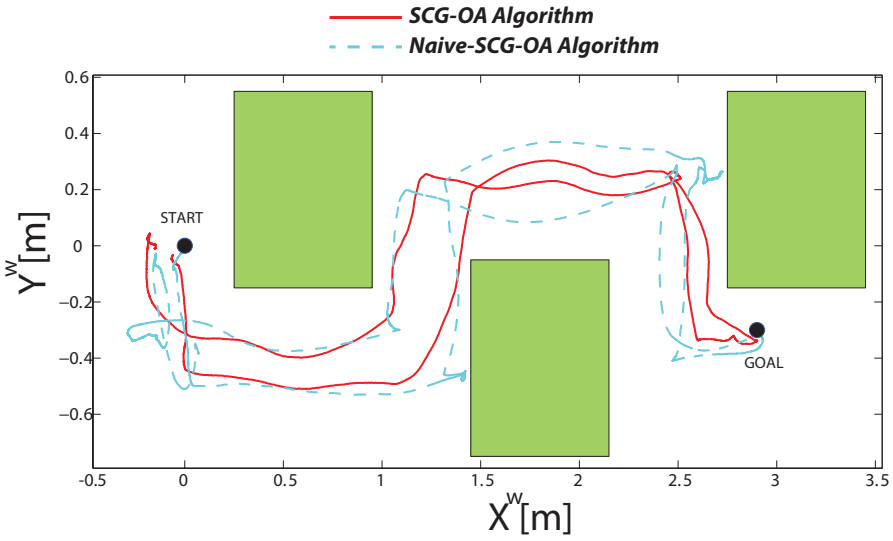

Fig. 5.12. Trajectory: $X^w - Y^W -$ axes projection

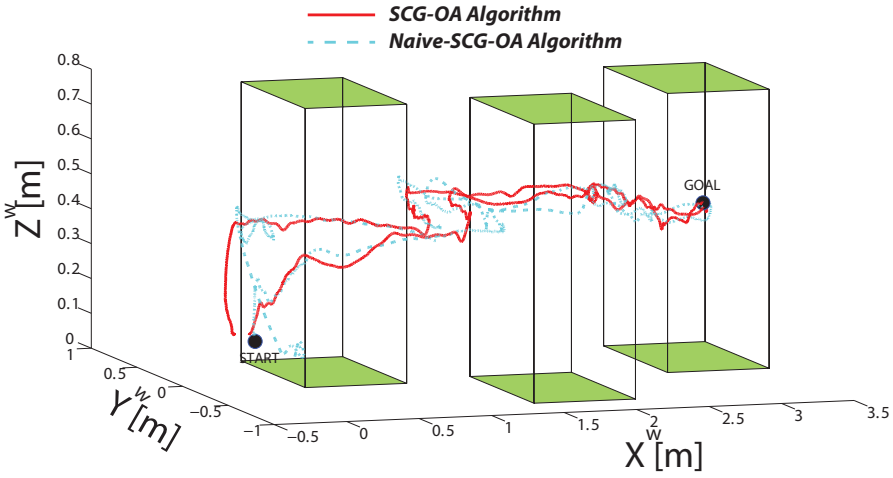

Fig. 5.13. 3D Trajectory

order to avoid obstacle occurrences. Screen shots take care of five events (constraint window switchings) that are necessary to comply with the prescribed goal. As an example, consider the screen shots labelled with **(A)**, **(B)** where the obstacle Ob_1 is circumvented by modifying the constraint window (from the continuous border

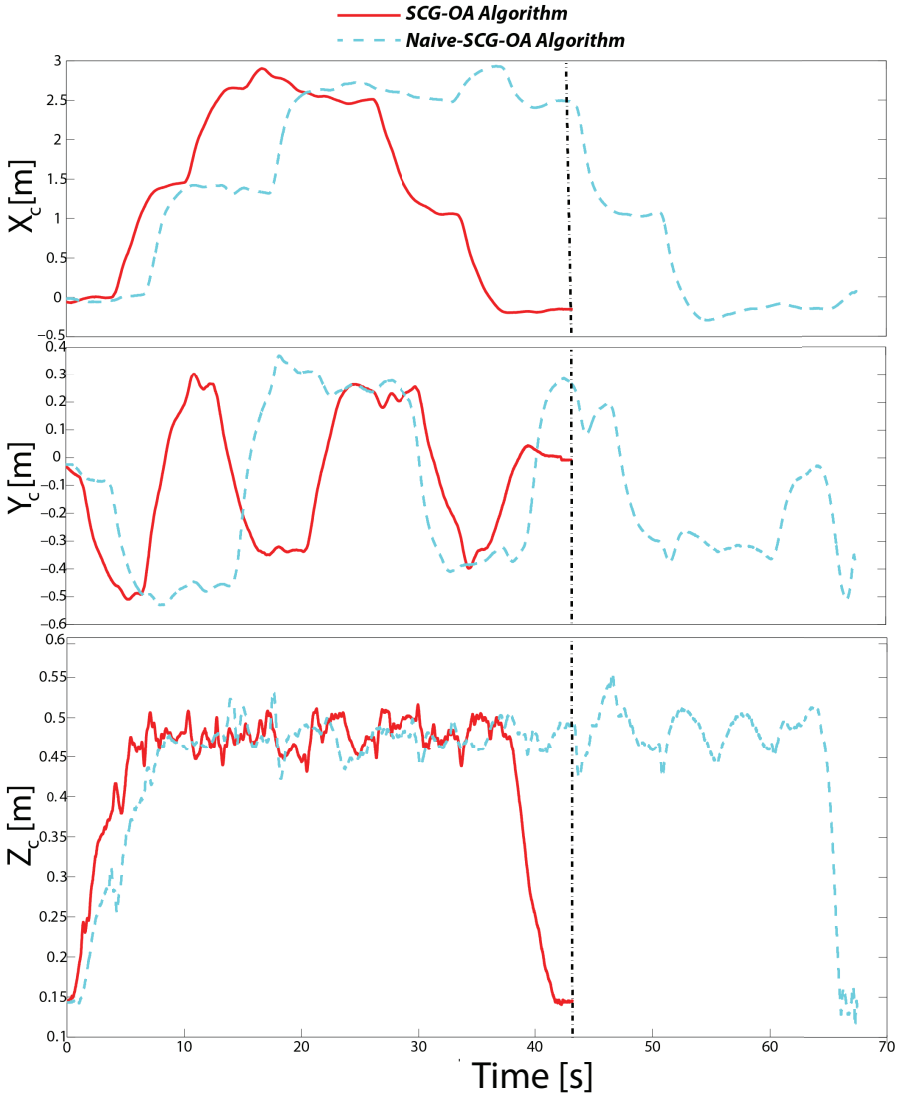

Fig. 5.14. Trajectory along X^w, Y^w, Z^w-axes

area (blue) to the dashed border one (red)) and by updating initial and final points, namely $(start, p^1_{reach}) \rightarrow (p^1_{reach}, p^2_{reach})$. The same reasoning applies to the other two obstacle occurrences.
Moreover, Fig. 5.19 provides roll and pitch angles trends, while Fig. 5.20 reports the applied control inputs, i.e. Δ_r, Δ_p and u_v . As it is clearly highlighted, the prescribed constraints (5.33)-(5.34) are always satisfied.

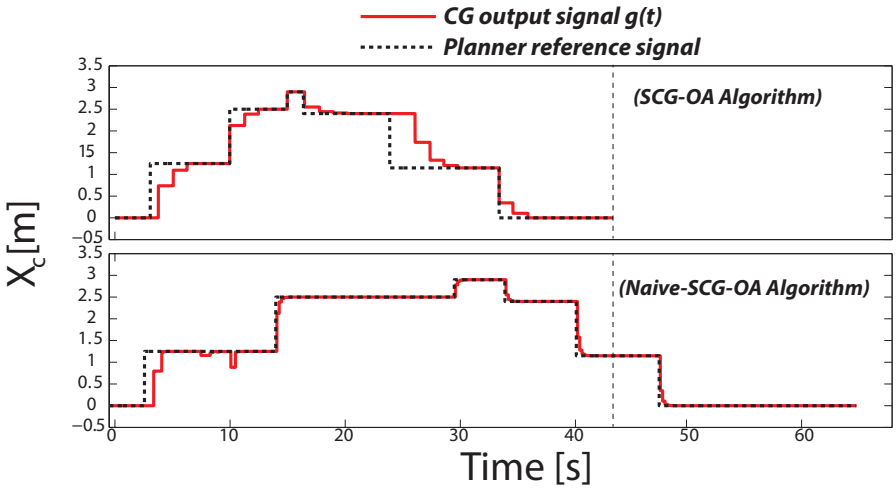

Fig. 5.15. Command Governor and Planner: X axis

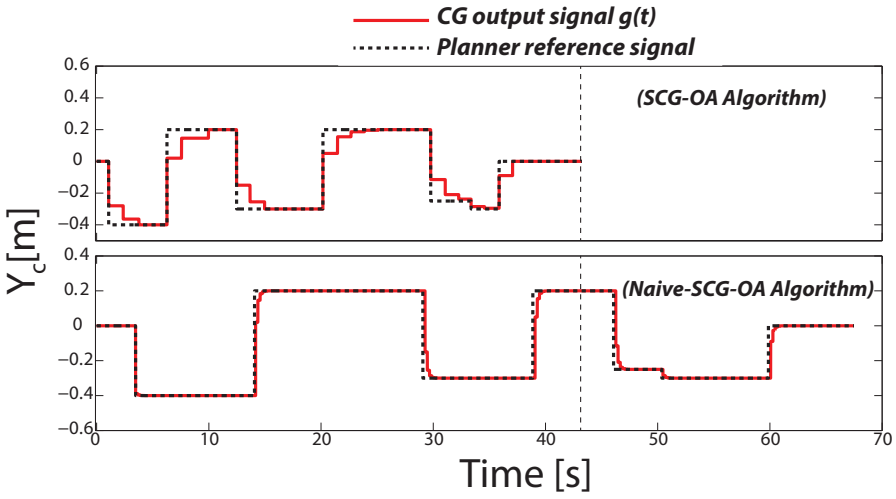

Fig. 5.16. Command Governor and Planner: Y axis

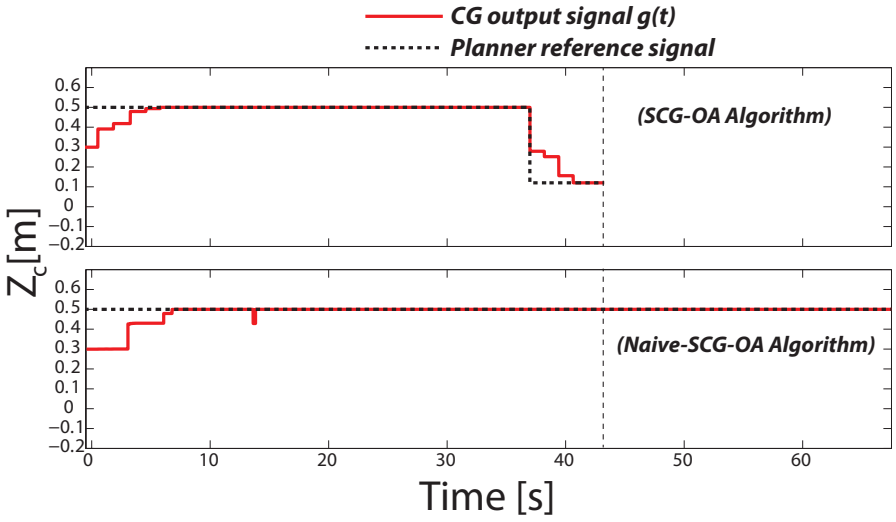

Fig. 5.17. Command Governor and Planner: Z axis

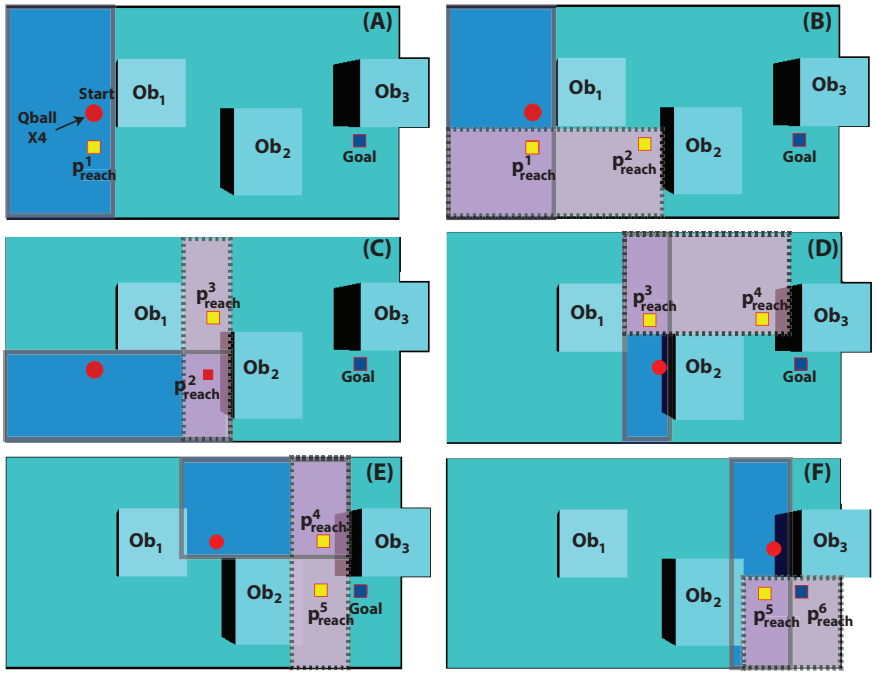

Fig. 5.18. Constraints updating procedure

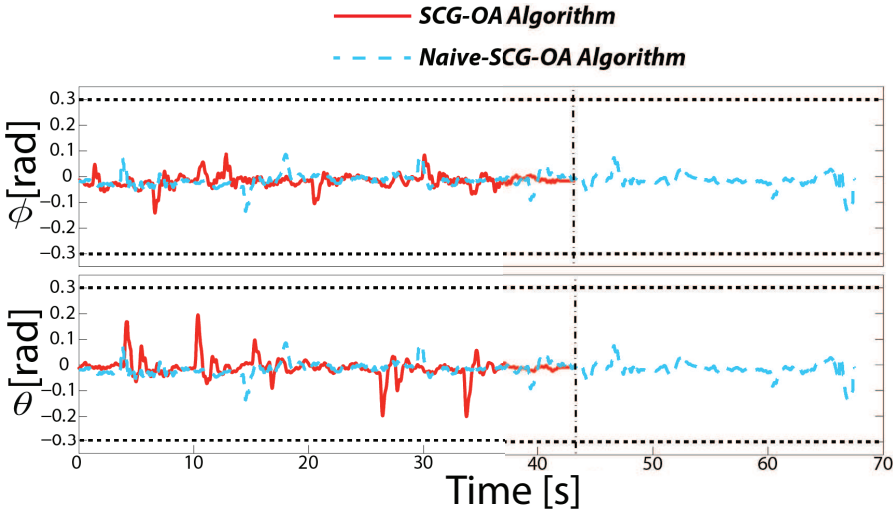

Fig. 5.19. Roll ϕ and Pitch θ angles

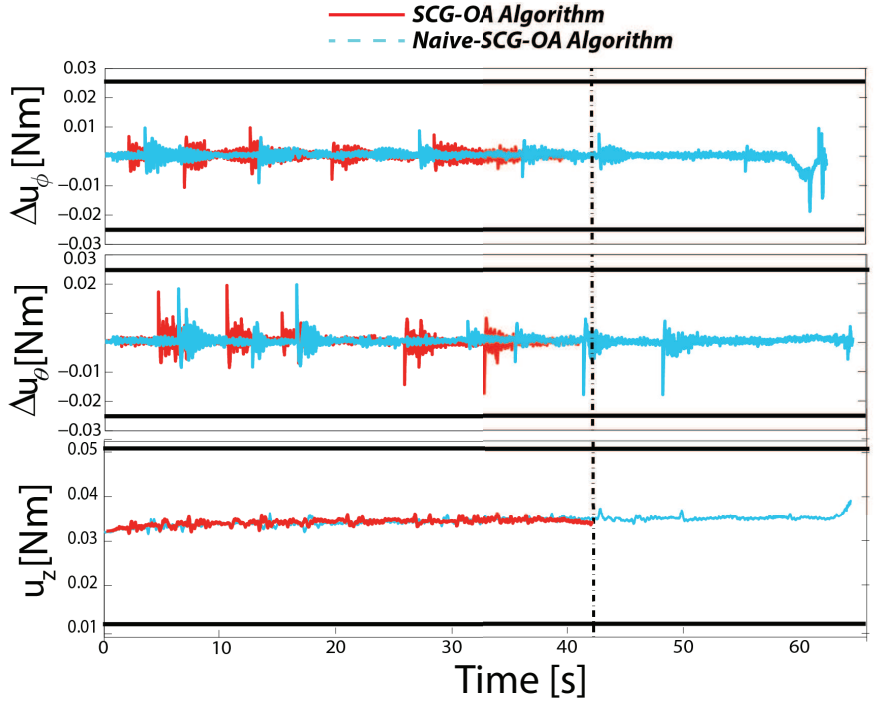

Fig. 5.20. Command inputs: Δu_ϕ, Δu_θ and u_z

Finally, further relevant simulation and experimental demos can be found at the following four web links:

- *Naive SCG-OA*-Algorithm:
  `https://www.dropbox.com/s/ylkrlsjtkio5lzu/basic_algorithm_behaviour.wmv`
- *SCG-OA*-Algorith:
  `https://www.dropbox.com/s/sn35tnffq9ke53n/fast_algorithm_behaviour.wmv`
- *Qball-X4 and Naive SCG-OA*-Algorithm:
  `https://www.dropbox.com/s/wv2kpx1rg7y5cxb/basic_quadrotor_05_11_movie.wmv`
- *Qball-X4 and SCG-OA*-Algorithm:
  `https://www.dropbox.com/s/0w8tv9i8gbxz37o/Fast_quad_movie_horizontal_22_11.wmv`

5.6 Conclusions

In this Chapter an extension of the basic CG strategy for solving the obstacle avoidance motion planning problem for unnamed aerial vehicles has been presented.
Linearity and constraint sets overlapping properties are the key ingredients to prove that feasibility retention and constraints fulfillment are preserved in spite of constraint structure modifications. By taking advantage from viability arguments, the proposed control architecture does not require the on-line CG re-design when *a-priori* unknown obstacles block the UAV navigation.
The experimental set-up involving the Qball-X4 quadrotor allows to clearly show the effectiveness and applicability of the proposed hybrid strategy when obstacle scenarios are taken into consideration.

Conclusions

In this dissertation, predictive based control architectures devoted to solve obstacle avoidance motion planning problems for autonomous vehicles subject to input and state constraints have been presented and below summarized:

- **Unmanned Ground Vehicles -**

  √ *Obstacle Avoidance MPC (**OA-MPC**)*. The receding horizon control strategy concerns with autonomous vehicles described by of LTI or multi-model state space descriptions and operating within structured dynamic environments. Set-theoretic ideas have been used to take care of all admissible time varying obstacle scenarios and move into the off-line phase most of the computation pertaining to the RHC controller. As a consequence, the resulting dual-mode controller has a modest on-line computational burden and it is appealing in real-time applications.

  √ *Sum-of-Squares Obstacle Avoidance MPC (**SOS-OA-MPC**)*. It provides an extension of the ***OA-MPC*** framework in order to consider polynomial vehicle descriptions and to take care of nonholonomic constraints. Semi-algebraic arguments have been exploited with the aim to adapt the ***OA-MPC*** algorithm to the polynomial modelling. Moreover, adequate relaxations of the resulting on-line non-linear optimization problems have been provided so that simple SDPs result.

  √ *Moving Obstacle Avoidance MPC (**MOA-MPC**)*. This scheme extends the application range of the basic ***OA-MPC*** strategy: moving agents are allowed so that non-structured dynamic environments come out. By jointly exploiting one-step controllable set and RPI region concepts, a novel scheduling policy to on-line take care of the admissible moving obstacle scenarios has been derived. The required computational resources (spatial and memory complexities) are modest and therefore the strategy could be affordable in practical applications.

- **Unmanned Aerial Vehicles -**

  √ *Switching command governor obstacle avoidance algorithm* ***(SCG-OA)***. The control architecture is a pertinent extension of the basic CG strategy in order to deal with *UAV obstacle avoidance motion planning problems* ***(UAV-OAMPs)***. By taking advantage from viability arguments, the proposed control scheme does not requires the on-line CG re-design during the UAV navigation. Finally, linearity and constraint set overlapping properties are used to prove that feasibility retention and constraints fulfilment are preserved despite of any admissible constraint structure modifications.

Future directions

The core of this thesis can be stated as: find novel solutions to the obstacle avoidance motion planning problem for a single unmanned vehicle. The extension of the proposed control strategies to control and coordination of multi-agent systems could be the natural future research direction.
As it is well-known, this topic has been recently attracted a lot of attention due to their broad possible applications: cooperative control of unmanned air vehicles, scheduling of automated highway systems, formation control of satellite clusters and so on, see [62], [72], [117], [124].
On the other hand, this extension seems to be not straightforward because the following key aspects need to be carefully investigated:

- Scalability to multi-agent descriptions;
- Command input distributed computation;
- Feasibility retention;

Along these lines, a preliminary result has been presented in [45] where leader-follower formations are considered and the obstacle avoidance motion planning problem is solved within static environments.

A

One step ahead controllable set: computation details

In this appendix, by considering the model description (2.9), a computational algorithm for computing recursion (2.11) is derived by means of Linear Matrix Inequalities (LMI) techniques provided that the following assumptions are satisfied :

Assumption A.1. *$\mathcal{U}$ and $\mathcal{X}$ can be expressed as the intersection of (possible degenerate) ellipsoidal sets: $\mathcal{U} = \cap \mathcal{T}_h^{\mathcal{U}}$ and $\mathcal{X} = \cap \mathcal{T}_h^{\mathcal{X}}$. Notice that any convex compact set can be represented as the intersection of ellipsoids [65]. In particular this allows for intervals and boxes to be considered.* □

Assumption A.2. *$\mathcal{D}$ is itself an ellipsoid (or an interval in the case of scalar disturbances).* □

If Assumptions (A.1)-(A.2) hold true then a simple algorithm can be implemented once the following set manipulations have been carried out:

$$\begin{aligned} &\{\ x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \forall d \in \mathcal{D} : \Phi x + Gu + G_d d \in \mathcal{T}_{k-1}\} \\ &\quad \supset \{x \in \mathcal{X} : \exists\, u \in \mathcal{U} : \Phi x + Gu + G_d d \in In[\mathcal{T}_{k-1} \sim G_d D]\} \\ &\quad = Proj_x\{[x,u] : u \in \mathcal{U}, x \in \mathcal{X}, [x,u] \in \tilde{\mathcal{T}}_{k-1}\} \end{aligned} \tag{A.1}$$

where $Proj_x$ denotes the projection over the x subspace and the ellipsoidal set $\tilde{\mathcal{T}}_{k-1}$ is defined as follows in the extended space x, u :

$$\tilde{\mathcal{T}}_{k-1} := \{[x,u] : \Phi x + Gu \in In[\mathcal{T}_{k-1} \sim G_d D]\}.$$

Therefore

$$\mathcal{T}_k := Proj_x \left[In \left[\tilde{\mathcal{T}}_{k-1} \bigcap \left(\bigcap \mathcal{T}_h^{\mathcal{X}} \times \bigcap \mathcal{T}_k^{\mathcal{U}} \right) \right] \right] \tag{A.2}$$

Notice that the Cartesian product $\left(\bigcap \mathcal{T}_h^{\mathcal{X}} \times \bigcap \mathcal{T}_h^{\mathcal{U}}\right)$ is again an ellipsoidal set in the extended space (x, u). With the aim of computing a projection into the x subspace it is more convenient to refer to the dual ellipsoidal parametrization (2.8). In fact, when a variables vector z is partitioned as $z = [x^T, u^T]^T$, the projected ellipsoid over x is obtained by just restricting the support function to the desired subspace:

$$\max_{x\in Proj_x[\mathcal{T}(P)]} c^T x = \max_{x\in Proj_x[\mathcal{T}(P)]} [c^T, 0]z$$
$$= \sqrt{[c^T, 0]P[c^T, 0]^T} = \sqrt{c'[P]_{11}c}$$

where $[P]_{11}$ is the $n \times n$ principal minor of P. In order to maximize, according to some criterion, the size of the projected ellipsoid (for instance its volume) the following LMI optimization problem can be used

$$\begin{aligned} &\max \ \log\det Q_{11} \quad s.t \\ &0 < \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \leq P \end{aligned} \tag{A.3}$$

and then the ellipsoidal set $\mathcal{T}_i$ can be computed by means of the following projection formula

$$Proj_x \left\{ \begin{bmatrix} x^T & u^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix}^{-1} \begin{bmatrix} x \\ u \end{bmatrix} \leq 1 \right\} = \left\{ x : x^T Q_{11}^{-1} x \leq 1 \right\} \tag{A.4}$$

Remark A.3. If of interest, once can enforce the following nesting property amongst the ellipsoidal set $\mathcal{T}_k$

$$\mathcal{T}_{k-1} \subseteq \mathcal{T}_k$$

by adding to problem (A.3) the following constraint

$$Q_{11} \geq Q_{11}^{k-1}$$

where Q_{11}^{k-1} denotes the matrix Q_{11} corresponding to $\mathcal{T}_{k-1}$ □

Remark A.4. The optimization problem (A.3) is valid only for non-degenerate ellipsoidal sets, whereas typically ellipsoid defined in the extended space $[x, u]$ are unbounded along some directions. Therefore, in order for (A.3) to make sense, it is necessary to project the LMI along the directions where they are non-degenerate, see [120] for further details.
In particular, as P is symmetric it can be factorized by using the singular value decomposition (SVD) as

$$P = U^T \Sigma U$$

where U is an orthogonal matrix and Σ is partitioned as follows:

$$\Sigma = \begin{bmatrix} \tilde{\Sigma} & 0 \\ 0 & 0 \end{bmatrix}, \quad \tilde{\Sigma} \text{ diagonal and invertible}$$

Therefore, the projection of (A.3) along the non-degenerate directions yields:.

$$\begin{aligned} &\max \ \log\det Q_{11} \quad s.t \\ &\begin{bmatrix} I & 0 \end{bmatrix} U \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} U^T \begin{bmatrix} I \\ 0 \end{bmatrix} \leq \tilde{\Sigma}, \quad \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \geq 0 \end{aligned} \tag{A.5}$$

with I is the identity matrix of dimension $rank(P)$ □

In conclusion recursion (2.10) can be approximately computed by means of ellipsoidal sets (2.11), regardless of any disturbance occurrences $d(t) \in \mathcal{D}$, by performing intersection (A.2) and projections (A.4) of ellipsoids. Both the required steps are numerically well established and with a problem dimension that grows quadratically with the size of the state and the input.

B

Nonholonomic ground vehicles: non linear models and polynomial recasting

The aim of this appendix is twofold: 1) provide detailed model descriptions of autonomous ground vehicles and 2) show that such nonlinear models can be recast as polynomial systems. To this end two well-known model descriptions of the unicycle car-like mobile robot (also known as differential drive) are considered, see Fig. B.1. These models are first detailed and then recast using the ***PRA*** procedure presented in Section 2.3.4.

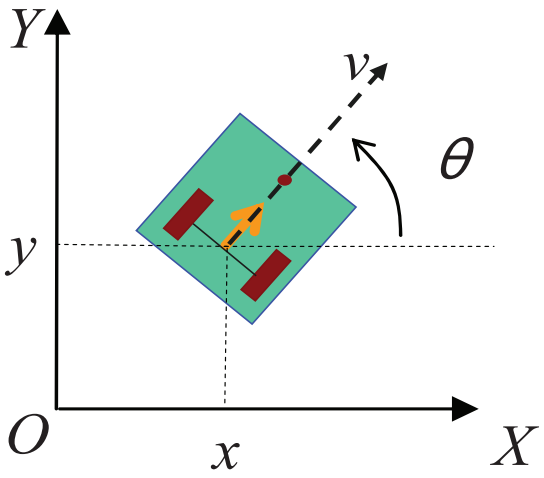

Fig. B.1. Nonholonomic mobile robot

B.1 Kinematic unicycle model

A differential drive robot is a typical nonholonomic wheeled vehicle, which has two rear drive wheels and a front castor for body support. It is assumed that the motion of the robot cannot slip laterally so that the translational velocity is in the direction of heading, i.e. a pure rolling contact between the wheels and the ground. The speed of the two rear wheels (v_l and v_r) are used to impose the translation ($v = (v_l + v_r)/2$) and angular ($\omega = (v_r - v_l)/B$) speeds of the robot (B is the wheelbase). The robot pose is described by its position (x_r, y_r), the midpoint of the rear axis of the robot, and its orientation (θ_r). Then, the kinematics equation is

$$\begin{bmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \\ \dot{\theta}_r(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \\ 0 \end{bmatrix} v(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega(t) \tag{B.1}$$

where maximum linear and angular velocities are prescribed

$$|v(t)| \leq V_{MAX} \quad \text{and} \quad |\omega(t)| \leq W_{MAX}, \ \forall t \in \mathbb{R}_+ .$$

Notice that the kinematic model (B.1) is nonintegratable and, as a consequence, kinematics constraints can not be converted into geometrical requirements [36]. Moreover since the number of control variables is less than the number of state variables, a nonholonomic constraint holds and a continuous time-invariant feedback control law cannot be used [52], [34]. On the other hand, since the accessibility rank condition is globally satisfied [89]:

$$Rank \left(\begin{bmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \left[\begin{bmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right] \right) = 3 \tag{B.2}$$

the model plant (B.1) is controllable by means of a nonlinear or time-varying controller.

B.1.1 Polynomial recast

Model (B.1) includes non-polynomial non-linearities, i.e. $\sin(\theta_r(t))$ and $\cos(\theta_r(t))$, but it can be straightforwardly converted through ***PRA*** to a polynomial system.

First of all , as prescribed by Step 1 of ***PRA***, all the state variables of the original model (B.1) are imposed as state components of (2.19), i.e. $x_1 = x_r$, $x_2 = y_r$ and $x_3 = \theta_r$. Then, along the line dictated by Steps 2-3 of ***PRA*** two further state variables are introduced

$$\begin{aligned} x_4(t) &= \cos(\theta_r(t)) \\ x_5(t) &= \sin(\theta_r(t)) \end{aligned} \tag{B.3}$$

and their derivative computed

$$\begin{aligned}\dot{x}_4(t) &= -sin(\theta_r(t))\dot{\theta}_r(t) = -x_5(t)\omega(t)\\ \dot{x}_5(t) &= \cos(\theta_r(t))\dot{\theta}_r(t) = x_4(t)\omega(t)\end{aligned} \tag{B.4}$$

Finally substituting (B.3)-(B.4) into (B.1) the following system equations result

$$\begin{cases} \dot{x}_a = \begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\end{bmatrix} = \begin{matrix} v(t)x_4(t)\\ v(t)x_5(t)\\ \omega(t)\end{matrix} \\ \dot{x}_b = \begin{bmatrix}\dot{x}_4(t)\\ \dot{x}_5(t)\end{bmatrix} = \begin{matrix} -x_5(t)\omega(t)\\ x_4(t)\omega(t)\end{matrix} \end{cases} \tag{B.5}$$

where only polynomial terms are involved. Notice that solutions of (B.5) lie into a 6-dimensional manifold but it can be restricted within a 4-dimensional manifold by simply introducing the equality constraint

$$G_a(z_a, z_b) := x_4^2(t) + x_5^2(t) = 1, \tag{B.6}$$

which typically come out whenever $\sin(\cdot)$ and $\cos(\cdot)$ functions are involved (see Section 2.3.4)

B.2 Dynamical unicycle model

Assume that the unicycle motion and orientation are achieved by two independent actuators (e.g. DC motors) providing the necessary torques τ_1 and τ_2 to the driving wheels. The robot pose in the inertial Cartesian frame $\{O, X, Y\}$ is completely specified by the $3-$dimensional vector of generalized coordinates $q(t) := (x_r(t), y_r(t), \theta_r(t))$. Moreover, the autonomous vehicle is subject to an independent velocity constraint:

$$\dot{x}_r(t)\sin(\theta_r(t)) - \dot{y}_r(t)\cos(\theta_r(t)) = 0 \tag{B.7}$$

This represents a nonholonomic constraint and states that the robot can only move in the direction normal to the axis of the driving wheels: the mobile base satisfies the *pure rolling* and *non slipping* conditions, see [35]. Then, the vehicle can be described by the following dynamical model:

$$\begin{cases} \ddot{x}_r(t) = -\dot{y}_r(t)\dot{\theta}_r(t) + \frac{\cos(\theta_r(t))}{mr}\left(\tau_1(t) + \tau_2(t)\right) \\ \ddot{y}_r(t) = \dot{x}_r(t)\dot{\theta}_r(t) + \frac{\sin(\theta_r(t))}{mr}\left(\tau_1(t) + \tau_2(t)\right) \\ \ddot{\theta}_r(t) = \frac{R}{Ir}\left(\tau_1(t) - \tau_2(t)\right) \end{cases} \tag{B.8}$$

where m, I, R are the robot mass, moment of inertia and length, respectively, and r is the wheels radius. Finally, considering as state variables the robot pose $q(t)$, as

well as its derivative, $\dot{q}(t) = [\dot{x}(t)\ \dot{y}(t)\ \dot{\theta}(t)]^T$ system (B.8) can be translated into the next nonlinear state space form

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_5(t) \\ \dot{x}_6(t) \end{bmatrix} = \begin{bmatrix} x_4(t) \\ x_5(t) \\ x_6(t) \\ -x_5(t)x_6(t) \\ x_4(t)x_6(t) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{\cos(x_3(t))}{mr} & \frac{\cos(x_3(t))}{mr} \\ \frac{\sin(x_3(t))}{mr} & \frac{\sin(x_3(t))}{mr} \\ \frac{R}{Ir} & -\frac{R}{Ir} \end{bmatrix} \begin{bmatrix} \tau_1(t) \\ \tau_2(t) \end{bmatrix} \tag{B.9}$$

where $x_1(t) = x_r(t)$, $x_2(t) = y_r(t)$, $x_3(t) = \theta_r(t)$, $x_4(t) = \dot{x}_r(t)$, $x_5(t) = \dot{y}_r(t)$, $x_6(t) = \dot{\theta}_r(t)$.

B.2.1 Polynomial recast

Because B.9 exhibits the same nonlinearities involved in (B.1), the recasting procedure follows the same arguments used in Section B.1.1. By denoting with

$$\begin{aligned} x_7(t) &= \cos(x_3(t)) \\ x_8(t) &= \sin(x_3(t)) \end{aligned} \tag{B.10}$$

the additional state variables, the following polynomial model results

$$\begin{cases} \dot{x}_a = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_5(t) \\ \dot{x}_6(t) \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ -x_5x_6 + \frac{x_7}{mr}(\tau_1 + \tau_1) \\ x_4x_6 + \frac{x_8}{mr}(\tau_1 + \tau_1) \\ \frac{R}{Ir}(\tau_1(t) - \tau_2(t)) \end{bmatrix} \\ \dot{x}_b = \begin{bmatrix} \dot{x}_7(t) \\ \dot{x}_8(t) \end{bmatrix} = \begin{bmatrix} -x_8(t)x_6 \\ x_7(t)x_6 \end{bmatrix} \end{cases} \tag{B.11}$$

Once again the solution manifold of (B.11) is restricted by imposing the equality constraint

$$G_a(x_a, x_b) := x_7^2(t) + x_8^2(t) = 1 \tag{B.12}$$

In conclusion, it is worth to underline that also the non holonomic constraint (B.7) simply translates into a polynomial condition

$$\dot{x}_1(t)x_8(t) - \dot{x}_2(t)x_7(t) = 0 \tag{B.13}$$

C

Command Governor computational details

This appendix describes all the computational details required to design and implement the CG strategy.

Consider the plant model (2.22) and linear constraints for the sets $c(t) \in \mathcal{C}$ and $d(t) \in \mathcal{D}$. They can be represented as the intersection of a finite number of half spaces:

$$\mathcal{C} = \{c \in \mathbb{R}^{n_c} : Tc \leq b\} \tag{C.1}$$

$$T = \begin{bmatrix} T_1^T \\ T_2^T \\ \vdots \\ T_z^T \end{bmatrix} \in \mathbb{R}^{z \times n_c}, \quad \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ g_z \end{bmatrix} \in \mathbb{R}^z, \; z \geq n_c, \quad \text{rank}(T) = n_c$$

and

$$D = \left\{ d \in \mathbb{R}^d : Vd \leq h \right\} \tag{C.2}$$

$$V = \begin{bmatrix} V_1^T \\ V_2^T \\ \vdots \\ V_{n_u}^T \end{bmatrix} \in \mathbb{R}^{n_u \times d}, \quad h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n_u} \end{bmatrix} \in \mathbb{R}^{n_u}, \; n_u \geq d, \; \text{rank}(V) = d, \; h_i \geq 0$$

where T, V are shaping matrices and b, h vectors accounting for the set dimensions. Exploiting the above set definitions, recursion (2.26) can be computed as follows

$$
\begin{aligned}
\mathcal{C}_0 &= \mathcal{C} \sim L_d\mathcal{D} = \left\{ c \in \mathbb{R}^{n_c} \ : T_i^T \leq b_i - \sup_{d\in\mathcal{D}} T_i^T L_d d,\ i = 1, \ldots, z \right\} \\
&= \left\{ c \in \mathbb{R}^{n_c} \ : T_i^T c \leq g_i^0, i = 1,\ \ldots, z \right\} \\
\mathcal{C}_1 &= \mathcal{C}_0 \sim H_c G_d \mathcal{D} = \left\{ c \in \mathbb{R}^{n_c} \ : T_i^T \leq b_i^0 - \sup_{d\in\mathcal{D}} T_i^T H_c G_d d,\ i = 1, \ldots, z \right\} \\
&= \left\{ c \in \mathbb{R}^{n_c} \ : T_i^T c \leq g_i^1, i = 1,\ \ldots, z \right\} \\
&\ldots \\
\mathcal{C}_k &= \mathcal{C}_{k-1} \sim H_c \Phi^{k-1} G_d \mathcal{D} = \Big\{ c \in \mathbb{R}^{n_c} : T_i^T \leq b_i^{k-1} - \sup_{d\in\mathcal{D}} T_i^T H_c \Phi^{k-1} G_d d, \\
&\quad i = 1, \ldots, z \} \\
&= \left\{ c \in \mathbb{R}^{n_c} \ : T_i^T c \leq g_i^k, i = 1,\ \ldots, z \right\}
\end{aligned}
\tag{C.3}
$$

In order to compute $\mathcal{W}^\delta$ an evaluation of $\mathcal{C}_\infty^\delta$ in necessary. To this end, a convenient approximation of $\mathcal{C}_\infty$ can be obtained by looking for a convenient set

$$\mathcal{C}_\infty(\varepsilon) \subset \mathcal{C}_\infty \subset \mathcal{C}_\infty(\varepsilon) + \mathcal{B}_\varepsilon$$

with $\mathcal{B}_\varepsilon$ a ball of radius ε
Such a set, unlike $\mathcal{C}_\infty$, is computable in a finite number of steps. In fact, it can be shown that

$$\mathcal{C}_\infty = C_k \sim \left(\sum_{i=k}^{\infty} H_c \Phi^i G_d \mathcal{D} \right)$$

Moreover, because of stability of Φ (see Assumption 2.29 *a*), there exist two constants $M > 0$ and $\lambda \in (0, 1)$ such that

$$||\Phi^k|| \leq M\lambda^k$$

Boundedness of $\mathcal{D}$ also implies that there exists finite

$$d_{max} := \max_{d\in\mathcal{D}} ||d||_2$$

The above facts are enough to ensure that, once chosen desired accuracy ε, there exists an index k_ε such that

$$\sum_{i=k_\varepsilon}^{\infty} H_c \Phi^i G_d \mathcal{D} \subset \mathcal{B}_\varepsilon$$

In fact, it is suffices that

$$d_{max} \bar{\sigma}(H_c) \bar{\sigma}(G_d) M \sum_{i=k_\varepsilon}^{\infty} \lambda^i \leq \varepsilon$$

which, after direct steps, gives rise to

$$k_\varepsilon = \frac{\log \varepsilon + \log(1-\lambda)[\bar{\sigma}(H_c)\bar{\sigma}(G_d) M d_{max}]}{\log \lambda}$$

Then, the desired approximation can be computed as

$$\mathcal{C}_\infty(\varepsilon) = \mathcal{C}_{k_\varepsilon} \sim \mathcal{B}_\varepsilon$$

and, in turn,

$$\begin{aligned}\mathcal{C}_\infty^\delta(\varepsilon) &= (\mathcal{C}_{k_\varepsilon} - \mathcal{B}_\varepsilon) \sim \mathcal{B}_\delta \\ &= \left\{c \in \mathbb{R}^m \ : b^{k_\varepsilon} - (\varepsilon+\delta)\left[T_i^T T_i\right]\right\}\end{aligned}$$

$$\begin{aligned}\mathcal{W}^\delta &= \left\{\omega \in \mathbb{R}^m : \bar{c}_\omega \in \mathcal{C}_\infty^\delta(\varepsilon)\right\} \\ &= \left\{\omega \in \mathbb{R}^m \in T(H_c(I-\Phi)^{-1}G + L)\omega \le b^{k_\varepsilon} - (\varepsilon+\delta)\left[\sqrt{T_i^T T_i}\right]\right\}\end{aligned} \tag{C.4}$$

where $(\varepsilon+\lambda)\left[\sqrt{T_i^T T_i}\right]$ is the support function of the ball $\mathcal{B}_{\varepsilon+\delta}$

In this case $\mathcal{V}(x)$ can be characterized as

$$\mathcal{V}(x) = \left\{\omega \in \mathcal{W}^\delta : \bar{c}(k,x,\omega) \in \mathcal{C}_k,\ k = 0,\dots,\bar{k}\right\} \tag{C.5}$$

where the integer $\bar{k}$ will be specified in while. Then, by exploiting the next disturbance-free predictions

$$\begin{aligned}\bar{x}(k,x,\omega) &= \Phi^k x + \left(\sum_{i=0}^{k-1}\Phi^i G\right)\omega \\ &\quad \Phi^k x + R_k^x \omega \\ \bar{c}(k,x,\omega) &= H_c\bar{x}(k,x,\omega) + L\omega \\ &\quad H_c\Phi^k x + (H_c R_k + L)\omega \\ &\quad H_c\Phi^k x + R_k^c\omega\end{aligned}$$

the set (C.5) can rewritten as

$$\mathcal{V}(x) = \left\{\omega \in \mathcal{W}^\delta : TH_c\Phi^k x + TR_k^c\omega \le b^k,\ k = 0,\dots,\bar{k}\right\} \tag{C.6}$$

Finally the CG action (2.30) consists of solving the following QP optimization problem

$$\begin{aligned}g(t) = \arg\min_w &(\omega - r(t))^T\Psi(\omega - r(t)) \\ &\text{subject to:} \\ TH_c\Phi^k x(t) + TR_k^c\omega \le b^k,\ k = 0,\dots\bar{k} &\quad , \ \Psi = \Psi^T \ge 0 \\ T(H_c(I-\Phi)^{-1}G + L)\omega \le b^{k_\varepsilon} - (\varepsilon+\delta)&\left[T_i^T T_i\right]\end{aligned} \tag{C.7}$$

By defining the function

$$\begin{aligned}G_k(j) := \arg \max_{\omega\in\mathcal{W}, x\in\mathbb{R}^n}\ & T_j^T\bar{c}(k,x,\omega) - b_j^k \\ &\text{subject to} \\ T_j^T\bar{c}(i,x,\omega) \le b_j^k,\ & \ i = 0\dots,k-1\end{aligned} \tag{C.8}$$

the computation of the constraint horizon $\bar{k}$ can be accomplished via the following procedure [48]:

$\bar{k}$−*Procedure*

1: *k=1;*
2: ***if*** $G_k(j) < 0, \quad \forall j = 1, \ldots, z$ ***then***
3: $\bar{k} = k$;
4: *Stop $\bar{k}$-procedure;*
5: ***else***
6: $k = k + 1$;
7: *Goto Step 2;*
8: ***end if***

References

1. E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull. Morse Decompositions for Coverage Tasks, 2002.
2. D. Angeli, A. Casavola, G. Franzè, and E. Mosca. An ellipsoidal off-line MPC scheme for uncertain polytopic discrete-time systems. *Automatica*, 44(12):3113–3119, December 2008.
3. F. Bacconi, E. Mosca, and A. Casavola. Hybrid constrained formation flying control of micro-satellites. *IET Control Theory & Applications*, 1(2):513–521, 2007.
4. A. Bemporad. Reference governor for constrained nonlinear systems. *Automatic Control, IEEE Transactions on*, 43(3):415–419, 1998.
5. A. Bemporad, F. Borrelli, and M. at al. Morari. Model predictive control based on linear programming˜ the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.
6. A. Bemporad, A. Casavola, and E. Mosca. Nonlinear control of constrained linear systems via predictive reference management. *Automatic Control, IEEE Transactions on*, 42(3):340–349, 1997.
7. A. Bemporad, A. De Luca, and G. Oriolo. Local incremental planning for a car-like robot navigating among obstacles. In *Robotics and Automation. Proceedings IEEE International Conference on*, volume 2, pages 1205–1211. IEEE, 1996.
8. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.
9. A Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit solution of model predictive control via multiparametric quadratic programming. In *American Control Conference, 2000. Proceedings of the 2000*, volume 2, pages 872–876. IEEE, 2000.
10. F. Blanchini and S. Miani. *Set-theoretic methods in control*. Springer, 2007.
11. J. Bochnak, Coste M., and M.-F. Roy. *Géométrie algébrique réelle*, volume 12. Springer, 1987.
12. F. Borrelli. Constrained optimal control of linear and hybrid systems. 2003.
13. S. A Bortoff. Path planning for uavs. In *American Control Conference, 2000. Proceedings of the 2000*, volume 1, pages 364–368. IEEE, 2000.
14. S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.
15. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

16. RW. Brockett. Asymptotic stability and feedback stabilization. In *Differential Geometric Control Theory*. Citeseer, 1983.
17. T. Brogårdh. Present and future robot control developmentAn industrial perspective. *Annual Reviews in Control*, 31:69–79, 2007.
18. C. W Brown and H. Hong. Qepcad-quantifier elimination by partial cylindrical algebraic decomposition. *http://www. es. usna. edu/-qepead/B/QEPCAD, html*, 2004.
19. J. Byrne, M. Cosgrove, and R. Mehra. Stereo based obstacle detection for an unmanned air vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2830–2835. IEEE, 2006.
20. G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, O. Nguyen, R. Moser, P. Schoeneich, and F. Mondada. Highly compact robots for inspection of power plants. *Journal of Field Robotics*, 29(1):47–68, 2012.
21. L.R.G. Carrillo and G.R. Flores Colunga. Quad Rotorcraft Switching Control: An Application for the Task of Path Following. 22(4):1255–1267, 2013.
22. J. M. Carson, B. Açkmee, R. M. Murray, and Douglas G. M. A robust model predictive control algorithm augmented with a reactive safety mode. *Automatica*, 49(5):1251–1260, May 2013.
23. A. Casavola, E. Mosca, and D. Angeli. Robust command governors for constrained linear systems. *Automatic Control, IEEE Transactions on*, 45(11):2071–2077, 2000.
24. D. W. Casbeer, RW. Beard, TW McLain, Sai-Ming Li, and R. K. Mehra. Forest fire monitoring with multiple small uavs. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3530–3535. IEEE, 2005.
25. G. Chesi. Estimating the domain of attraction via union of continuous families of lyapunov estimates. *Systems & control letters*, 56(4):326–333, 2007.
26. M.-D. Choi, T. Y. Lam, and B. Reznick. Sums of squares of real polynomials. In *Proceedings of Symposia in Pure mathematics*, volume 58, pages 103–126. American Mathematical Society, 1995.
27. Matlab Coder. `http://www.mathworks.com/products/matlab-coder/`.
28. B. Coifman, M. McCord, R. G Mishalani, and K. Redmill. Surface transportation surveillance from unmanned aerial vehicles. In *Proc. of the 83rd Annual Meeting of the Transportation Research Board*, 2004.
29. A. Dax. The distance between two convex sets. *Linear Algebra and Its Applications*, 416:184–213, 2006.
30. A. De Luca and G. Oriolo. *Modelling and control of nonholonomic mechanical systems*. Springer, 1995.
31. B. Ding, Y. Xi, M. T. Cychowski, and T. OMahony. Improving off-line approach to robust MPC based-on nominal performance cost. *Automatica*, 43(1):158–163, January 2007.
32. N.E. Du Toit and J.W. Burdick. Robot Motion Planning in Dynamic, Uncertain Environments. *Robotics, IEEE Transactions on*, 28:101 –115, 2012.
33. W. B. Dunbar and R. M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 4, pages 4631–4636. IEEE, 2002.
34. J. M. Eklund, J. Sprinkle, and S. Sastry. Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft. *Control Systems Technology, IEEE Transactions on*, 20(3):604–620, 2012.
35. R. Fierro and F. L. Lewis. Robust practical point stabilization of a nonholonomic mobile robot using neural networks. *Journal of intelligent and Robotic Systems*, 20(2-4):295–317, 1997.

36. R. Fierro and F.L. Lewis. Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In *Decision and Control, Proceedings of the 34th IEEE Conference on*, volume 4, pages 3805–3810, Dec 1995.
37. C. A. Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.
38. T. Fraichard and H Asama. Inevitable collision statesa step towards safer robots? *Advanced Robotics*, 2004.
39. G. Franzè. A nonlinear sum-of-squares model predictive control approach. *Automatic Control, IEEE Transactions on*, 55(6):1466–1471, 2010.
40. G. Franzè and W. Lucia. A model predictive control scheme for mobile robotic vehicles in dynamic environments. In *Decision and Control (CDC), IEEE 52nd Annual Conference on*, pages 5752–5757. IEEE, 2013.
41. G. Franzè and W. Lucia. The obstacle avoidance motion planning problem for autonomous vehicles: a low-demanding receding horizon control scheme. (*Provisionally Accepted*).
42. G. Franzè and W. Lucia. An obstacle avoidance model predictive control scheme for mobile robots subject to nonholonomic constraints: a sum-of-squares approach. (*Under Review*).
43. G. Franzè and W. Lucia. A receding horizon control strategy for autonomous vehicles in dynamic environments. (*Under Review*).
44. G. Franzè, W. Lucia, and P. Muraca. An obstacle avoidance receding horizon control scheme for autonomous vehicles. In *American Control Conference (ACC), 2013*, pages 3948–3953. IEEE, 2013.
45. G. Franzè, W. Lucia, F. Tedesco, and V. Scordamaglia. A distributed obstacle avoidance MPC strategy for leader-follower formations. In *In 19th IFAC World Congress*. Citeseer, 2014.
46. E. Frazzoli, M. A Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *Robotics, IEEE Transactions on*, 21(6):1077–1091, 2005.
47. C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348, 1989.
48. E. G. Gilbert, I. Kolmanovsky, and K. T. Tan. Discrete-time reference governors and the nonlinear control of systems with state and control constraints. *International Journal of Robust and Nonlinear Control*, 5(5):487–504, 1995.
49. Elmer G Gilbert and Ilya Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. *International Journal of Robust and Nonlinear Control*, 9(15):1117–1141, 1999.
50. A. R. Girard, A. S. Howell, and J. K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. In *Decision and Control, CDC, 43rd IEEE Conference on*, volume 1, pages 620–625. IEEE, 2004.
51. S. Gray. Cooperation between uavs in search and destroy mission. In *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conf. and Exhibit, Austin, USA*, 2003.
52. D. Gu and H. Hu. Receding horizon tracking control of wheeled mobile robots. *Control Systems Technology, IEEE Transactions on*, 14(4):743–749, 2006.
53. J. Hauser and A. Jadbabaie. Aggressive maneuvering of a thrust vectored flying wing: A receding horizon approach. In *Decision and Control, Proceedings of the 39th IEEE Conference on*, volume 4, pages 3582–3587. IEEE, 2000.
54. D. Henrion and J.-B. Lasserre. Gloptipoly: Global optimization over polynomials with matlab and sedumi. *ACM Transactions on Mathematical Software (TOMS)*, 29(2):165–194, 2003.

55. M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/~mpt`.
56. SR. Herwitz, LF. Johnson, SE. Dunagan, RG. Higgins, DV. Sullivan, J. Zheng, BM. Lobitz, JG. Leung, BA. Gallmeyer, and et al. Aoyagi. Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. *Computers and Electronics in Agriculture*, 44(1):49–61, 2004.
57. P. Hou, A. Saberi, Z. Lin, and P. Sannuti. Simultaneous External and Internal Stabilization for Continuous and Discrete-Time Critically Unstable Linear Systems with Saturating Actuators. *Automatica*, 34(12):1547–1557, December 1998.
58. C.-H. Hsieh and JS. Liu. Nonlinear model predictive control for wheeled mobile robot in dynamic environment. *Advanced Intelligent Mechatronics (AIM), ...*, pages 363–368, July 2012.
59. D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.
60. J. Huang, C. Wen, W. Wang, and ZP Jiang. Adaptive stabilization and tracking control of a nonholonomic mobile robot with input saturation and disturbance. *Systems & Control Letters*, 62:234–241, 2013.
61. Quanser Inc. Qball user manual `http://www.quanser.com/english/html/UVS_Lab/fs_Qball_X4.htm`.
62. A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988–1001, 2003.
63. Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard. Control applications of sum of squares programming. In *Positive Polynomials in Control*, pages 3–22. Springer, 2005.
64. M.V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.
65. A. Kurzhanski and I. Valyi. *Ellipsoidal calculus for estimation and control*. Berlin: Birkhäuser, 1997.
66. A. A Kurzhanskiy and P. Varaiya. Ellipsoidal toolbox (et). In *Decision and Control, 2006 45th IEEE Conference on*, pages 1498–1503. IEEE, 2006.
67. Y. Kuwata, T. Schouwenaars, A. Richards, and J. How. Robust constrained receding horizon control for trajectory planning. In *Proceedings of the AIAA guidance, navigation and control conference*, 2005.
68. U. Lages. Laserscanner for obstacle detection. In *Advanced Microsystems for Automotive Applications Yearbook 2002*, pages 136–140. Springer, 2002.
69. L. Lapierre, R. Zapata, and P. Lepinay. Combined path-following and obstacle avoidance control of a wheeled robot. *The International Journal of Robotics Research*, 26(4):361–375, April 2007.
70. Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
71. Steven M LaValle. Rapidly-exploring random trees a ew tool for path planning. 1998.
72. Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *Automatic Control, IEEE Transactions on*, 49(4):622–629, 2004.
73. R.-W. Liu. Convergent systems. *Automatic Control, IEEE Transactions on*, 13(4):384–391, 1968.
74. B. Llanas. Efficient computation of the Hausdorff distance between polytopes by exterior random covering. *Computational Optimization and Applications*, 30(2):161–194, February 2005.

75. J. Lofberg. Pre-and post-processing sum-of-squares programs in practice. *Automatic Control, IEEE Transactions on*, X(X):1–5, 2009.
76. W. Lucia, G. Franzè, and P. Muraca. An obstacle avoidance model predictive control scheme: A sum-of-squares approach. In *Control Automation (MED), 21st Mediterranean Conference on*, pages 1575–1582. IEEE, 2013.
77. W. Lucia, G. Franzè, and M. Sznaier. An obstacle avoidance and motion planning command governor based scheme: the qball-x4 quadrotor case of study. In *Decision and Control (CDC), 2014 IEEE 53nd Annual Conference on*. IEEE, 2014.
78. W. Lucia, G. Franzè, and M. Sznaier. An hybrid command governor scheme for unmanned aerial vehicles: obstacle avoidance and motion planning. (*Under review*).
79. J. M. Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.
80. A. S. Matveev, C. Wang, and A. V. Savkin. Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous systems*, 60(6):769–788, 2012.
81. D. Q. Mayne, J. B Rawlings, C. V Rao, and P. OM. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.
82. P. C. Merrell, Dah-Jye Lee, and R. W. Beard. Obstacle avoidance for unmanned air vehicles using optical flow probability distributions. *Mobile Robots XVII*, 5609(1):13–22, 2004.
83. J. Minguez, F. Lamiraux, and J.P. Laumond. Motion planning and obstacle avoidance. *Springer handbook of robotics*, pages 827–852, 2008.
84. J. Minguez and L. Montano. Robot Navigation in very Complex, Dense, and Cluttered Indoor/Outdoor Evironments. In *In 15th IFAC World Congress*. Citeseer, 2002.
85. M. Morari and J. H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.
86. Dennis Nieuwenhuisen. Efficient path planning in changing environments. *Intelligent Robots and Systems (IROS 2007)*, pages 3295–3301, October 2007.
87. P Ogren and NE Leonard. A convergent dynamic window approach to obstacle avoidance. *Robotics, IEEE Transactions on*, 21(2):188–195, 2005.
88. T. Ohtsuka. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.
89. G. Oriolo, A. De Luca, and M. Vendittelli. Wmr control via dynamic feedback linearization: design, implementation, and experimental validation. *Control Systems Technology, IEEE Transactions on*, 10(6):835–852, 2002.
90. A. Papachristodoulou and S. Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. In *Positive Polynomials in Control*, pages 23–43. Springer, 2005.
91. P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 320:293–320, 2003.
92. S. Petti and T. Fraichard. Safe motion planning in dynamic environments. *Intelligent Robots and Systems*, pages 2210–2215, 2005.
93. S. Prajna, A. Papachristodoulou, and P. Parrilo. Introducing SOSTOOLS: a general purpose sum of squares programming solver. *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 1(2):741–746, 2002.
94. James A Primbs, Vesna Nevistić, and John C Doyle. Nonlinear optimal control: A control lyapunov function and receding horizon perspective. *Asian Journal of Control*, 1(1):14–24, 1999.
95. M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

96. Zhihua Qu, Jing Wang, and CE Plaisted. A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *Robotics, IEEE Transactions on*, 20(6):978–993, 2004.
97. P. Raja and S. Pugazhenthi. Path planning for a mobile robot in dynamic environments. *International Journal of Physical Sciences*, 6(20):4721–4731, 2011.
98. S. V. Raković and D. Q. Mayne. Robust model predictive control for obstacle avoidance: discrete time case. In *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 617–627. Springer, 2007.
99. J. B. Rawlings. Tutorial overview of model predictive control. *Control Systems, IEEE*, 20(3):38–52, 2000.
100. W. Ren, J.-S. Sun, R. W. Beard, and T. W. McLain. Nonlinear tracking control for nonholonomic mobile robots with input constraints: An experimental study. In *Proceedings of the American Control Conference*, pages 4923–4928. IEEE, 2005.
101. B. Reznick. Extremal PSD forms with few terms. *Duke mathematical journal*, 1978.
102. B. Reznick. Some concrete aspects of Hilbert's 17th problem. *Contemporary Mathematics*, 2000.
103. A. Ryan and J K. Hedrick. A mode-switching path planner for uav-assisted search and rescue. In *Decision and Control and European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 1471–1476. IEEE, 2005.
104. A. Saberi, P. Hou, and A. A. Stoorvogel. On simultaneous global external and global internal stabilization of critically unstable linear systems with saturating actuators. *Automatic Control, IEEE Transactions on*, 45(6):1042–1052, 2000.
105. I. Sadeghzadeh, A.t Mehta, Y. Zhang, and C.-A. Rabbath. Fault-tolerant trajectory tracking control of a quadrotor helicopter using gain-scheduled pid and model reference adaptive control. In *Annual Conference of the Prognostics and Health Management Society*, volume 2, 2011.
106. Z. Sarris and S. Atlas. Survey of uav applications in civil markets. In *IEEE Mediterranean Conference on Control and Automation*, page 11, 2001.
107. M.A. Savageau and E.O. Voit. Recasting nonlinear differential equations as S-systems: a canonical nonlinear form. *Mathematical biosciences*, 115, 1987.
108. K. Schmüdgen. Thek-moment problem for compact semi-algebraic sets. *Mathematische Annalen*, 289(1):203–206, 1991.
109. T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European control conference*, volume 1, pages 2603–2608. Citeseer, 2001.
110. Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm: Follow the Gap Method. *Robotics and Autonomous Systems*, 60(9):1123–1134, September 2012.
111. A. Sgorbissa and R. Zaccaria. Planning and obstacle avoidance in mobile robotics. *Robotics and Autonomous Systems*, 60(4):628–638, April 2012.
112. D. Hyunchul Shim and S. Sastry. An evasive maneuvering algorithm for uavs in see-and-avoid situations. In *American Control Conference, ACC'07*, pages 3886–3891. IEEE, 2007.
113. B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer, 2009.
114. L. Singh and J. Fuller. Trajectory generation for a uav in urban terrain, using nonlinear mpc. In *Proceedings of the American Control Conference*, volume 3, pages 2301–2308. IEEE, 2001.
115. J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 1999.

116. W. Tan and A. Packard. Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming. *IEEE Transactions on Automatic Control*, 53(2):565–570, 2008.
117. H. G. Tanner, G. J Pappas, and V. Kumar. Leader-to-formation stability. *Robotics and Automation, IEEE Transactions on*, 20(3):443–455, 2004.
118. F. Tedesco, D. M Raimondo, and A. Casavola. Collision avoidance command governor for multi-vehicle unmanned systems. *International Journal of Robust and Nonlinear Control*, 2013.
119. P. Varaiya. Reach set computation using optimal control. *Verification of Digital and Hybrid Systems*, 19(1 Suppl A):S3, January 2000.
120. P. Varaiya. Reach set computation using optimal control. In *Verification of Digital and Hybrid Systems*, pages 323–331. Springer, 2000.
121. Vicon. `http://www.vicon.com/`.
122. S. Waki, H.and Kim, M. Kojima, M. Muramatsu, and H. Sugimoto. Algorithm 883: Sparsepop-a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):15, 2008.
123. Z. Wan and M. V. Kothare. An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39(5):837–846, May 2003.
124. Wei Wang and J-JE Slotine. A theoretical study of different leader roles in networks. *Automatic Control, IEEE Transactions on*, 51(7):1156–1161, 2006.
125. Y. Wang and S. Boyd. Fast model predictive control using online optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, 2010.
126. Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry. Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, 17(7):741–750, July 2009.
127. YM. Zhang, A. Chamseddine, CA. Rabbath, BW. Gordon, C-Y. Su, S. Rakheja, C Fulford, J Apkarian, and P Gosselin. Development of advanced fdd and ftc techniques with application to an unmanned quadrotor helicopter testbed. *Journal of the Franklin Institute*, 350(9):2396–2422, 2013.
128. Y. Zhu, T. Zhang, J. Song, X. Li, and M. Nakamura. A new method for mobile robots to avoid collision with moving obstacle. *Artificial Life and Robotics*, 16(4):507–510, 2012.