Universitá di Pisa
Dipartimento di Ingegneria dell'Informazione

Laurea Magistrale in

Ingegneria Robotica e dell'Automazione

Deconfliction of Motion Paths in Robot-Robot and Human-Robot Interactions with Barrier Certificate Functions

Laureando

Federico Celi

Relatrice:

Prof.ssa Lucia Pallottino

Advisor:

Prof. Magnus Egerstedt

Anno Accademico 2017-2018

To my parents, R & A
To my systers, A & F

Contents

List of Figures

1 Introduction

Always do the right thing.

That's it?

-Spike Lee

Guidance navigation algorithms are designed in compliance with the traffic rules that apply to the particular scenario they are developed for. However, although strictly observed and enforced, road traffic rules are not the same worldwide, with the most notable example in the existing difference in left- and right-hand driving. In the 1st century BC all roads lead to Rome left-handedly (Walters, 1998), possibly to keep the riders' right hand free to greed (or defend from) an upcoming traveler; in the Medieval times Pope Boniface VIII enjoined pilgrims to keep the left (Hamer, 1986). In post Revolution France, Napoleon imposed right handed driving over most of Europe - as left-hand driving was a heritage of royal France. Right-hand driving also became the custom in XVIII century northern America following the spread of wagons - right-handed teamster would ride on the rear left-most horse (Pop, 1935). As of 1919, 104 out of 208 of the world's territories drove on the left side of the road (Watson, 1999), with the obviously noticeable pattern of colonial influences. On top of this, there was no real consistency even in the positioning of the driving wheel with respect to the road rule's side. In 1908, Ford's Model T was the first manufacturer's car to feature a left-driver seat meant for right-handed roads and the popularity of the car in the 20's set the standard for all cars manufacturers (Lay, 1992). It is worth noting that marine (Sea, 1972; Cockcroft and Lameijer, 2011) and air traffic (Federal Aviation Administration, 2013) both observe right-hand traffic rules.

Since the history of the rule of the road is complex and often dictated by chance, it is unclear weather humans have a particular predisposition for one side or the other, especially when related to pedestrian traffic as a left-side preference has been suggested in some studies (Gordon et al.,

1992; Mohr et al., 2004) but confuted by others (Bracha et al., 1987; Robertson et al., 2015) and seems to be only slightly correlated to driving habits (Thomas et al., 2017). It is clear—despite differences in the standards—that roads are governed by a strict set of rules that stay consistent throughout the ride.

However, there are some environments where rules are not as firm and they are often broken in order to improve the overall driving experience, e.g. in parking lots. Here it is accepted for a driver, for example, to overtake on the *wrong* side of the road, or to resolve a head on scenario by steering to the left—or to the right—without a particular preference, since road surface markings are not always in place and speeds are low—a scenario more relatable to the pedestrian's instinct.

Self driving vehicles, also referred to as autonomous or driverless vehicles, have been around us for a while in some limited—environment controlled—forms, such as autonomous airport shuttles and trains. Recently, self driving technology has been applied to more complex scenarios, ultimately in the form of cars operating on regular roads; this lead to the first commercial production of partially self driving, or driver assist cars, corresponding to a Society of Automation Engineers (SAE) Level 3 or 4 of Autonomy (International, 2016). One important factor of success for a technology who's final goal is to be adopted among Non-Expert Users (Gough et al., 2014) is to build users' trust, especially as one of the issues that will need to be addressed is the coexistence of autonomous vehicles (AV) and human operated vehicles (HV). Potential users will build their opinion based on the interaction experience with other AVs, therefore it seems worthwhile to build tools that allow AVs to be provably safe for the user while *feeling* safe and *familiar* to an external HV's driver that interacts with it. It is with this premises in mind that we argue that navigation algorithms should be more flexible in those environments where rules are fuzzy, by combining safety with user experience. In the present work we set the path to start investigating the rules, tools and solutions proposed toward achieving such goal.

We can define a set of rules of thumbs for AV's parking lot behavior, based on the Pedestrian and Driver Etiquettes: i) *watch out for others*, i.e., prevent collisions; ii) *avoid sudden stops*; iii) *drive consistently with the traffic rules*, e.g., if in a right-hand driving environment, tend to resolve conflict scenarios by steering right; vi) *adapt to the surrounding users' behavior*, e.g., if another agent, perhaps human, acts in discord with traffic rules, take action accordingly.

Rule one is the most critical of the four and numerous tools have been developed to solve it, e.g., artificial potential functions (Park et al., 2001), bug algorithms (Choset et al., 2005), edge

energy functions (Ji and Egerstedt, 2007), barrier functions. The choice of a particular technique is crucial, as it will dictate how the rest of the rules shall be implemented. Decentralized Safety Barrier Certificates (SBC), introduced by in (Borrmann et al., 2015) and based on Zeroing Control Barrier Functions (ZCBF) (Ames et al., 2014), are a particularly promising technology, as they do not require the knowledge of the higher level objective controller (e.g. go-to-goal) and offer tools such as multi-objective composition (Wang et al., 2016b; Ames et al., 2017), while always guaranteeing safety. Barrier certificates for multirobot collision avoidance were introduced in (Borrmann et al., 2015) and (Wang et al., 2016b) and were extended to a non-conservative decentralization, guaranteeing safety, feasibility and providing approaches for deadlock resolution in (Wang et al., 2017). The main idea for the deadlock resolution technique introduced in (Wang et al., 2017) is to perturb the nominal control input with symmetry-breaking traffic rules; we are going to build on this results and extend this concept to achieve a more *dynamic* behavior for the AV. In particular, in this work we are going to focus on the deconfliction of symmetric and asymmetric head-on scenarios for a two agent system, first in the case of both agents being autonomously controlled, and finally when one agent is autonomous and a second agent is controlled by a human user.

The solution here proposed is a double-edged tool: from one side it resolves the conflicting scenarios with *user-friendly* traffic inspired rules; on the other side it offers a first approach to building a human driving behavior model based on the observed reaction of the driver in a conflicting scenario.

1.1 Problem Description

Picture two cars driving in opposite directions down a narrow corridor, as usually seen in parking lots and garages. When the road is clear, the drivers may tend to stay closer to the center of the roadway (see Figure 2.1); notice that the road is wide enough to allow both cars to pass at the same time. When the two cars are facing *close enough*, they will start moving away from the center of the road in order to keep heading towards their respective goal; we will refer to this scenario as head-on scenario. This is outlined in more detail in Figure 1.1, set in a right-hand driving environment. In the first example, Figure 1.1a, the two agents are driving far enough from the center of the road, allowing both cars to pass; since no conflict is taking place, no further action needs to be taken. In Figure 1.1b and 1.1c, the two cars are heading misaligned, therefore should avoid collision by

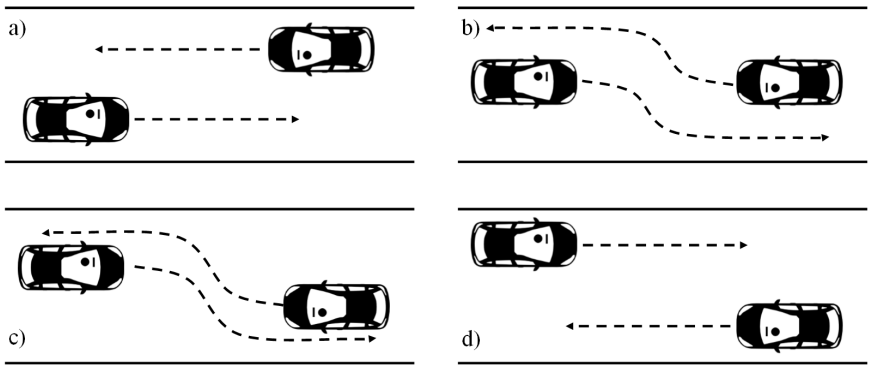


Figure 1.1: Four different head-on scenarios (right-hand driving). From a) to c) the cars overtake as the road rules impose; in d) the misalignment is strong enough for them to overtake regardless of the rules.

steering to the right, in compliance with local traffic rules. Finally, in Figure 1.1d, the two agents are, again, far enough from the center of the road, yet they are on the wrong side of the road, respectively. However, it seems of little use to force both cars to steer right, as road rules would require, since no conflicting motion is taking place. This example shows how we would like our navigation algorithm to resolve conflicting head-on scenarios.

On a different level, consider the scenario of Figure 1.1b, where one agent is autonomous and one is human; a second design goal is to make the AV *aware* of the HV driving direction bias, e.g., if HV steers to the left, so should AV, even if in contrast to local traffic rules.

Figure 1.2: Typical parking lot scenario, with absence of road surface markings, low driving speeds and where head-ons are likely to occur.

2 Path Planning and Safety

You just landed at Hartsfield−Jackson Atlanta International Airport, the busiest airport in the World; it is your first time you are here, the gate is F24: on your left there is Gate F22, on your right Gate F26 and you cannot find an exit sign. Where do you go? Some time later, you move into your new home; you need to move your new wheeled armchair from one room to another, without obviously damaging the chair or the other pieces of furniture or walls. How do you do it? These are two examples of problems that are historically included under the wide umbrella of Motion Planning: the Airport Terminal problem, and the the Piano Mover problem. Planning has been studied in the intersection of three major different fields over the last decades: Control, Robotics, and Artificial Intelligence (AI) all worked towards solving similar problems, however focusing on field−related issues and usually exploiting different tools and techniques.

Control theory is the science of finding inputs for physical systems represented by a model, usually under the form of differential equations. These inputs should drive the system to behave in a predictable fashion, and make it accomplish some high level goal. Once a time evolution of the system is designed—as close as possible to the real system of study, may it be a rocket or a social network (Ruf, 2018)—a feedback control law has to be found that satisfies the control goal criterion. A particular branch of control theory is that of Optimal Control Theory, that, on top of the classical control theory problems, tries to minimize the consumption of a particular set of resources, such as time, energy or a combination of more.

Robotics is more concerned with the problem of automation of mechanical systems, with the inclusion of sensing, actuation and computation capabilities. The mentioned Piano Mover is a classic Path Planning Problem in Robotics, where the chair is more often represented by some kind of small vacuum cleaner that needs to navigate from room to room, avoiding static and moving obstacles.

In the framework of Path Planning, safety has always received extensive focus; in the years, different methods to ensure the safety of multi-robot systems, in the form of collision avoidance, have been developed. A complete overview of this methods, however interesting, is beyond the scope of this work; instead, a narrow set of examples will be presented, selected on a combination of their success over the years and relatability to the arguments presented later on in this work.

2.1 Artificial Potential Fields

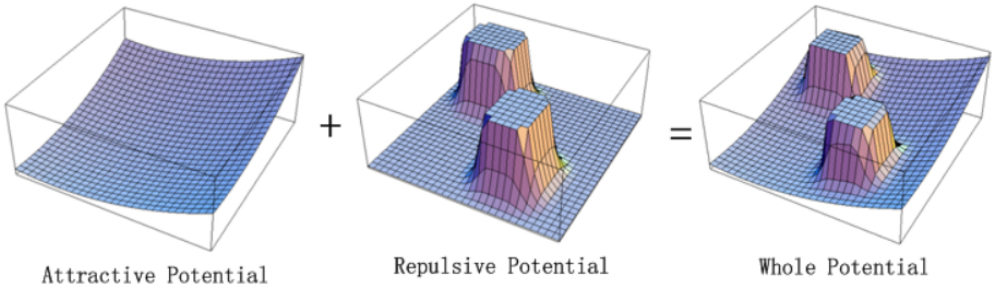


Figure 2.1: Typical Artificial Potential Field example: on the left ∇U_{att}, with the lower left corner being the goal point; in the central figure ∇U_{rep} is shown for two obstacles.

The Artificial Potential Field method (**?**) is one of the most popular collision avoidance algorithms, as it is generally easy to implement and computationally efficient. The main idea of this approach is to treat a robot as a point in space; the goal point $\mathbf{q}_{goal} \in \mathbb{R}^3$ exerts an fictitious attractive force on the robot, whereas an obstacle generates a repulsive force. These forces are expressed as the gradient of an artificial field U and they can be straightforwardly summed together: under the influence of the field U the robot will be attracted to q_{goal} and rejected to obstacles. In short:

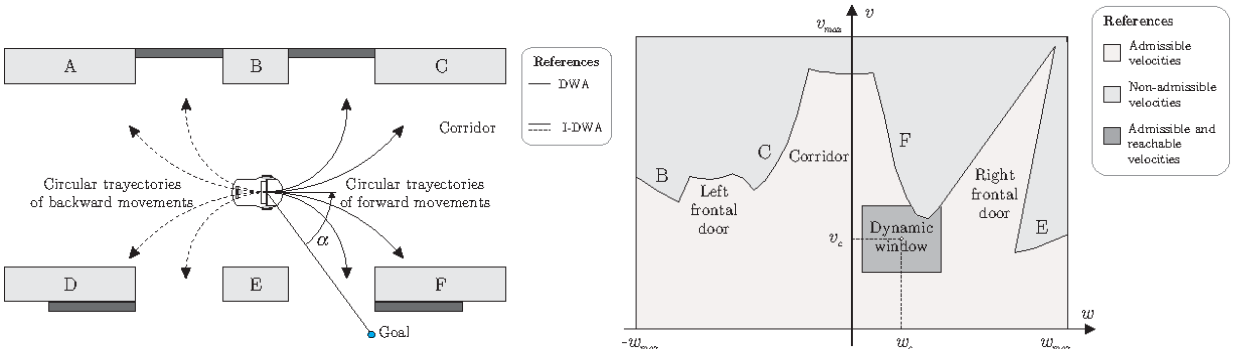
$$F(\mathbf{q}) = -\nabla U(\mathbf{q}) = -\nabla U_{att}(\mathbf{q}) - \nabla U_{rep}(\mathbf{q}). \tag{2.1}$$

There are several ways to select a strategy for the potential fields U_{att} and U_{rep}, a deeper analysis of which can be found in (**?**). Once the nature of U is selected, a Gradient Descent algorithm can be chosen so that the time evolution of the robot, following the gradient of the field, will take it to the goal. One of the issues of this approach, shared in general with the gradient descend strategies, is that of local minima, for which the robot, given some pathological configuration of the gradient, could get stuck in a point of locally minimum gradient, different from the actual goal point; solutions such as simulated annealing has been proposed and implemented (Park et al.,

2001).

2.2 Dynamical Window

The Dynamical Window approach controls the transnational and rotational velocity of the robot directly in the space of velocities, thus incorporating the dynamics of the robot. This is done by reducing the search space to the dynamic window, which consists of the velocities reachable within a short time interval. Within the dynamic window the approach only considers admissible velocities yielding a trajectory on which the robot is able to stop safely (Fox et al., 1997). The issue of local minima presents itself again in this approach therefore a Convergent Dynamical Window was developed by adding a navigation function (Ögren and Leonard, 2002).



(a) Illustration of a robot navigation environment.

(b) Velocity map corresponding to the scenario on the left

Figure 2.2: Dynamical Window Approach (**?**).

3 Barrier Functions

Ensuring safety of a system is one of the crucial design tasks when building controllers, since potential conflicts in the control objectives and safety constraints may arise: it is often the case that safety is implemented in a second moment, as a result of a high-level (nominal) control goal, e.g., go-to-goal. The nominal control law is executed most of the time, while the secondary, safe, controller takes over when some defined events occur, e.g., two agents are too close. In multiagent systems, however, as the number of agents increases, the collision avoidance strategy starts to dominate the system's behavior, resulting in the agents *waisting* computational power over avoiding collisions, and making little to no progress towards the high level control goal.

Typical safety verification methods for linear and nonlinear systems require the explicit computation of the reachable set. Barrier Certificates, or Barrier Functions, were introduced in (Prajna, 2005) as a tool to validate models with parametric uncertainties for nonlinear systems. Similar to the Lyapunov stability theory, the main idea behind barrier certificates is to study properties of the system without the need of computing the flow explicitly. Instead, it will be sufficient to prove that, defined a set of initial states and unsafe states, the system will never evolve towards the unsafe region. Barrier Certificates were extended to Control Barrier Functions (CBF) by (Wieland and Allgöwer, 2007), based on the theory of Control Lyapunov Functions (CLF) and the set invariance theory. CBF based quadratic programs were proposed in (Ames et al., 2014), with the goal of unifying safety conditions (CBF) and control objectives (CLF). Safety Barrier Certificates (SBC) were introduced in (Borrmann et al., 2015) in order to synthesize a minimally invasive collision avoidance tool for multiagent systems.

The present Chapter is devoted to illustrating the mentioned results, as they are required background tools for the correct understanding of the upcoming Chapters.

3.1 Original Formulation

A first formulation of safety via barrier certificates is found in (Prajna, 2005) and related work, see (Prajna et al., 2004; Prajna and Jadbabaie, 2004a; Prajna and Jadbabaie, 2004b; Prajna and Rantzer, 2005; Prajna, 2006). Such formulation is then extended to control barrier functions in (Wieland and Allgöwer, 2007) and (Tang et al., 2013). Although a new approach is presented in (Ames et al., 2014), see Section 3.2, we find that this original formulation is useful to gain better insights on the intuitions that brought to the latter formulation, that will be used in the remaining of the work.

3.1.1 Barrier Certificates

Consider the issue of safety verification of a nonlinear continuous system; the goal is to ensure that, given a set of initial states and a set of unsafe states, there is no trajectory of the system that starts from the first set and reaches the second.

Consider a nonlinear system of the form

$$\dot{x} = f(x), \tag{3.1}$$

where $x \subseteq \mathcal{X} \in \mathbb{R}^n$ are the states of the system and $f \in C(\mathcal{X}, \mathbb{R}^n)$. We make the assumption that f is forward complete, that is that, for any initial condition $x_0 := x(t_0) \in \mathbb{R}^n$, there exists a maximum time interval $I(x_0) = [t_0, \infty)$ such that $x(t)$ is the unique solution to the system in (3.1) on $I(x_0)$.

Definition 3.1.1 (Forward invariant set)**.** A set $\mathcal{S}$ is said to be forward invariant with respect to (3.1) if, for every $x_0 \in \mathcal{S}$, then $x(t) \in \mathcal{S}$ for all $t \in I(x_0)$, with $I(x_0)$ being the maximal interval of existence of $x(t, x_0)$.

To ensure safety, we first need to define a set of unsafe states, i.e., the set of states where we wish our system not to go, as $\mathcal{X}_u \subseteq \mathcal{X}$; we also define the set of possible initial states as $\mathcal{X}_0 \subseteq \mathcal{X}$.

Definition 3.1.2 (Safety)**.** Given the system in (3.1), the state set $\mathcal{X}$, the initial set $\mathcal{X}_0$ and the unsafe set $\mathcal{X}_u$, we say that the system is safe if, for any trajectory $x : [0, T] \to \mathbb{R}^n$, there exists no $T > 0$ such that $x(0) \in \mathcal{X}_0$, $x(T) \in \mathcal{X}_u$ and $x(t) \in \mathcal{X}$ for all $t \in [0, T)$.

The following result, adapted from (Prajna, 2005), ensures safety, thanks to the definition of a barrier certificate.

Definition 3.1.3 (Barrier Certificate)**.** A function $B(x) : \mathbb{R}^n \to \mathbb{R}$, function of the states of the system, is said to be a Barrier Certificate, or Barrier Function, if

$$B(x) \leq 0 \qquad \forall x \in \mathcal{X}_0, \tag{3.2}$$

$$B(x) > 0 \qquad \forall x \in \mathcal{X}_u, \tag{3.3}$$

$$\frac{\partial B}{\partial x} f(x) = L_f B(x) \leq 0 \qquad \forall x \in \mathcal{X}. \tag{3.4}$$

Theorem 1. *Given the system in (3.1), $\mathcal{X}_u$ and $\mathcal{X}_0$, the safety of the system is guaranteed if there exist a barrier certificate.*

The zero level set of a barrier certificate provides a barrier between states, in the sense that no trajectory can cross such a barrier; in particular, following Definition 3.1.3, if a barrier certificate can be found, no trajectory will take the system form $x(0) \in \mathcal{X}_0$ to $x(T) \in \mathcal{X}_u$. If the set $\mathcal{X}_u$ can never be reached, then the system is provably safe, without the need of explicitly computing the system trajectories or reachable set.

We will illustrate this result with a practical example, inspired by (Prajna and Jadbabaie, 2004a). Consider the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_1 \\ x_2 \end{bmatrix} \tag{3.5}$$

and let $\mathcal{X}_0 = \{x \in \mathbb{R}^2 \mid \|x - x_A\|^2 \leq 1\}$, and $\mathcal{X}_u = \{x \in \mathbb{R}^2 \mid \|x - x_B\|^2 \leq 1\}$, with $x_A = [-1.5, 2.5]^T$ and $x_B = [-3.5, 1]^T$. Let the barrier certificate be $B(x) = 2x_1^2 - x_2^2$; in Figure 3.1 the dashed line zero level set of $B(x)$, and we note that conditions (3.2)–(3.3) are satisfied by the particular choices of $B(x)$, $\mathcal{X}_0$ and $\mathcal{X}_u$. To verify condition (3.4), we simply note that $L_f B(x) = [4x_1 \ -2x_2][-x_1 \ x_2]^T = -2(2x_1^2 + x_2^2) \leq 0$, for all $x \in \mathbb{R}^2$. This concludes that if the system starts in $\mathcal{X}_0$ it will never reach any state in $\mathcal{X}_u$, as desired.

3.1.2 Control Barrier Functions

The concept of barrier certificates was expanded to Control Barrier Functions in (Wieland and Allgöwer, 2007) in order to consider a wider class of systems, not limited to closed nonlinear systems as in (3.1).

In order to be able to design a safety feedback controller of a not necessary safe system, consider

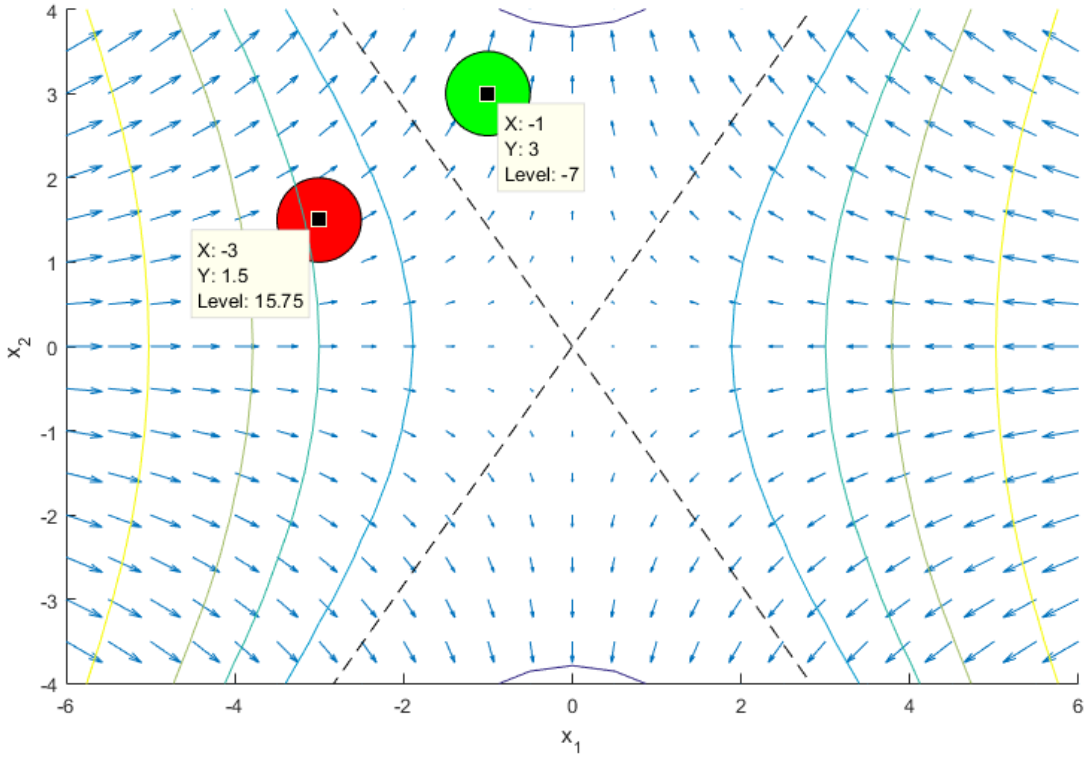


Figure 3.1: Flow representation of the system in (3.5); the circles represent $\mathcal{X}_u$ (leftmost, red) and $\mathcal{X}_0$. The dashed lines show the level set for $B(x) = 0$.

the nonlinear input affine system

$$\dot{x} = f(x) + g(x)u, \tag{3.6}$$

where, as before, the states $x \in \mathcal{X} \subseteq \mathbb{R}^n$, the function $f \in C(\mathcal{X}, \mathbb{R}^n)$, the control inputs are $u \in \mathbb{R}^m$ and the function $g \in C(\mathcal{X}, \mathbb{R}^{n \times m})$. The idea now is to find, if they exist, the control inputs u such that the time evolution of the trajectory of 3.6 stays safe, for all times t, or which is the same, a set $\mathcal{U} = \{u \in \mathbb{R}^m \mid \nexists \tau \in \mathbb{R} : x(\tau) \in \mathcal{X}_u,\ x(0) \in \mathcal{X}_0\}$, where the time evolution of $x(t)$ is (3.6).

Here we make use of the control input u as a safety condition; however, the control input u can also be used as a control objective. Let $\hat{u}$ be the original control objective, for which no safety property can be guaranteed; let $u = k(x, \hat{u})$, that is, the actual control input to the system. We wish for the control input u to stay as close as possible to the original control goal $\hat{u}$, while always being safe, i.e., if $\hat{u}$ endangers the system, change it: make it safe, but keep it as close as possible to $\hat{u}$.

Definition 3.1.4 (Control Barrier Function)**.** Given a system (3.6) and a set of unsafe states

$x \in \mathcal{X}_u \subseteq \mathcal{X}$, a function $B \in C^1(\mathcal{X})$ satisfying

$$B(x) > 0 \quad \forall x \in \mathcal{X}_u, \tag{3.7}$$
$$L_g B(x) = 0 \Rightarrow L_f B(x) < 0, \tag{3.8}$$
$$\exists x \in \mathcal{X} \mid B(x) \leq 0. \tag{3.9}$$

is called control barrier function.

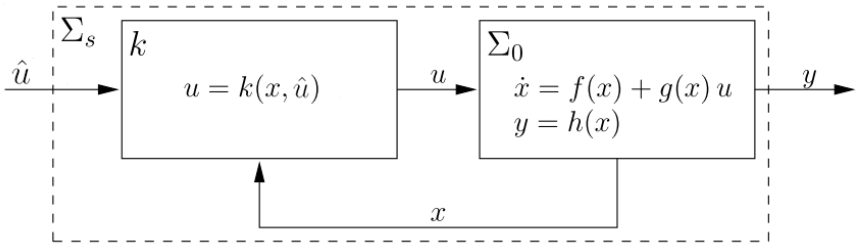


Figure 3.2: Block diagram of safe system Σ_s with original system Σ_O and safety feedback k.

It is easy to show that a CBF serves as a barrier certificate for the closed loop system. In fact, conditions (3.2)–(3.3) are satisfied by the definition of CBF when we define $\mathcal{X}_0 = \{x \in \mathcal{X} \mid B(x) \leq 0\}$. To verify condition (3.4), we define the closed loop system as $c(x) := f(x) + g(x)u$

$$L_c B(x) = L_f B(x) + L_g B(x) u. \tag{3.10}$$

The closed loop system $L_c B(x) < 0$ for $L_g B(x) = 0$, if and only if $L_f B(x) < 0$.

3.2 New Formulation

The formulation of Section 3.1 helps grasping the main idea underneath barrier certificates and their relationship with Lyapunov functions—and Lyapunov control functions—consequentially. However, the constraints this approach requires are too strict. For this reason, (Ames et al., 2014) introduces a new way of defining barrier certificates, using the forward invariance properties of sets. For the profs of the results in this section, we refer to (Ames et al., 2014; Ames et al., 2017; Wang et al., 2016b; Wang et al., 2016a; Wang et al., 2017; Xu et al., 2015), and the references therein.

Consider a nonlinear system of the form

$$\dot{x} = f(x) \tag{3.11}$$

for $x \in \mathbb{R}^n$, with f assumed to be locally Lipschitz. Given a set $\mathcal{C} \subset \mathbb{R}^n$ of states that we consider to be safe, we now determine conditions on a function $B : \mathcal{C} \to \mathbb{R}$ such that all solutions of (3.11) are guaranteed to stay in $\mathcal{C}$.

3.2.1 Barrier Functions

Let the set of safe states be defined as $\mathcal{C} \subset \mathbb{R}^n$, that is the set where we wish that the aggregate robot sates $x \in \mathbb{R}^n$ should stay. The goal is then to design a controller that guarantees the forward invariance of the set $\mathcal{C}$.

Definition 3.2.1 (Forward invariance)**.** A set $\mathcal{C}$ is said to be forward invariant when, if $x(0) \in \mathcal{C}$ then $x(t) \in \mathcal{C}, \forall t \geq 0$.

To be able to work with a mathematically manipulable formulation, the set $\mathcal{C}$ is encoded through $h(x)$:

$$\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}, \tag{3.12}$$
$$\partial\mathcal{C} = \{x \in \mathbb{R}^n : h(x) = 0\}, \tag{3.13}$$
$$\text{Int}(\mathcal{C}) = \{x \in \mathbb{R}^n : h(x) > 0\}, \tag{3.14}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a smooth (continuously differentiable) function and $\partial\mathcal{C}$, $\text{Int}(\mathcal{C})$ denote respectively the boundary and interior of the set $\mathcal{C}$. How can we relate $h(x)$ to the forward invariance properties of $\mathcal{C}$?

Before pursuing with the results, we first need to introduce two classes of functions.

Definition 3.2.2 (Class-$\mathscr{K}$ function)**.** A continuous function $\alpha : [0, a) \to [0, \infty)$ for some $a > 0$ is said to be a class-$\mathscr{K}$ function if it is strictly increasing and $\alpha(0) = 0$.

Definition 3.2.3 (Extended class-$\mathscr{K}$ function)**.** A continuous function $\beta : (-b, a) \to \mathbb{R}$ for some $a, b > 0$ is said to be an extended class-$\mathscr{K}$ function if it is strictly increasing and $\beta(0) = 0$.

We can now give a new definition of barrier certificates, as found in (Ames et al., 2014).

Definition 3.2.4 (Barrier Function)**.** For the dynamical system in (3.11), a function $B : \mathcal{C} \subset \mathbb{R}^n \to \mathbb{R}$ is a barrier function for the set $\mathcal{C}$ defined by (3.12)–(3.14) for a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ if there exist locally Lipschitz class-$\mathscr{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ such that, for all

$x \in \text{Int}(\mathcal{C})$,

$$\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))} \tag{3.15}$$

$$\dot{B}(x) \leq \alpha_3(h(x)). \tag{3.16}$$

Condition (3.15) implies that the barrier function B essentially behaves like the function $\frac{1}{\alpha(h(x))}$, for α a class-$\mathscr{K}$ function which satisfies the essential conditions:

$$\inf_{x \in \text{Int}(\mathcal{C})} \frac{1}{\alpha(h(x))} \geq 0, \qquad \lim_{x \to \partial\mathcal{C}} \frac{1}{\alpha(h(x))} = \infty. \tag{3.17}$$

Moreover, the condition on $\dot{B}$ in (3.16) allows for B to grow quickly when solutions are far away from $\partial\mathcal{C}$, with this growth approaching to zero as solution approach $\partial\mathcal{C}$.

This new formulation allows us to relate the forward invariance of set $\mathcal{C}$ with barrier certificates thanks to the following result.

Theorem 2 ((Ames et al., 2014)). *Given a set $\mathcal{C} \subset \mathbb{R}^n$ defined by (3.12)–(3.14), if there exists a barrier function $B : \mathcal{C} \to \mathbb{R}^n$, then $\mathcal{C}$ is forward invariant.*

3.2.2 Control Barrier Functions

Given the new formulation of barrier certificates, the notion of Control Barrier Certificate must be updated as well.

Consider a nonlinear system in the control affine form

$$\dot{x} = f(x) + g(x)u, \tag{3.18}$$

with f and g locally Lipschitz, $x \in \mathbb{R}^n$ and $u \in \mathcal{U} \subset R^m$. How can a controller u be designed that keeps the time evolution of 3.18 in the safe set $\mathcal{C}$?

Definition 3.2.5. Let $\mathcal{C} \subset \mathbb{R}^n$ be defined by (3.12)–(3.14) for a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, then a function $B : \mathcal{C} \to \mathbb{R}$ is a control barrier function if there exists class-$\mathscr{K}$ functions α_1, α_2 and $0 < \gamma$ such that

$$\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))} \tag{3.19}$$

$$\inf_{u \in \mathcal{U}} \left[L_f B(x) + L_g B(x) u - \frac{\gamma}{B(x)} \leq 0 \right] \tag{3.20}$$

for all $x \in \text{Int}(\mathcal{C})$.

Given a CBF, we can consider all control values that satisfy (3.20):

$$K_{cbf}(x) = \{u \in \mathcal{U} \mid L_f B(x) + L_g B(x)u - \frac{\gamma}{B(x)} \leq 0\} \tag{3.21}$$

Theorem 3. *Given a set $\mathcal{C} \subset \mathbb{R}^n$ defined by (3.12)–(3.14) with associated barrier function B, any Lipschitz continuous controller $u(x) \in K_{cbf}(x)$ for the system in (3.18) renders the set $\mathcal{C}$ forward invariant.*

This result allows us to relax the conditions on the change in B, since in the original formulation it was required that $\dot{B} \leq 0$, while the new formulation only requires that:

$$\dot{B} \leq \frac{\gamma}{B} \tag{3.22}$$

with $\gamma < 0$.

Definition 3.2.6 (Zeroing Control Barrier Function)**.** Given a dynamical system in (3.18) and a set $\mathcal{C} \subset \mathbb{R}^n$, defined by (3.12)–(3.14) for a smooth function $h : \mathcal{D} \to \mathbb{R}$, with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$. The function h is called a Zeroing Control Barrier Function (ZCBF), if there exists an extended class-$\mathscr{K}$ function κ such that

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x)u + \kappa(h(x))\} \geq 0 \tag{3.23}$$

for all $x \in \mathcal{D}$.

Given a ZCBF, the admissible control space $S(x)$ can be defined as

$$S(x) = \{u \in \mathcal{U} \mid L_f h(x) + L_g h(x)u + \kappa(h(x)) \geq 0\},\ x \in \mathcal{D}. \tag{3.24}$$

Now, we need to guarantee that the set $\mathcal{C}$ is forward, and this is ensured by the following theorem.

Theorem 4 ((Xu et al., 2015))**.** *Given a set $\mathcal{C} \subset \mathbb{R}^n$ defined by (3.12)–(3.14) and a ZCBF h defined on $\mathcal{D}$, with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$, any Lipschitz continuous controller $u : \mathcal{D} \to \mathbb{R}$ such that $u \in S(x)$ for the system (3.18) renders the set $\mathcal{C}$ forward invariant. And $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$.*

As noted in (Ames et al., 2014), the extend class-$\mathscr{K}$ function κ regulates how fast the state of the system can approach the boundary of $\mathcal{C}$. Noting that $\kappa(r) = r^p$ is always an extended class-$\mathscr{K}$ function when $p = 2n + 1$, for all $n \in \mathbb{Z}^+$, we adopt the particular choice of

$$\kappa(h(x)) = \gamma h^3(x), \quad \gamma \in \mathbb{R}^+, \tag{3.25}$$

which means that the controller needs to satisfy

$$L_f h(x) + L_g h(x)u + \gamma h^3(x) \geq 0 \tag{3.26}$$

so to render the set $\mathcal{C}$ forward invariant. Instead of proving Theorem 4, we are going to show how the particular choice of a controller u compatible with (3.26) indeed ensures safety, by forcing $h(x(t)) \geq 0$, for all $t > 0$.

Proof: We first notice that

$$\frac{d}{dt}h(x) = \frac{\partial h(x)}{\partial x}\frac{dx}{dt} = \frac{\partial h(x)}{\partial x}\dot{x} = \frac{\partial h(x)}{\partial x}\Big(f(x) + g(x)u\Big) = L_f h(x) + L_g h(x)u, \tag{3.27}$$

and recall the Comparison Lemma (Khalil, 2002).

Theorem 5 (Comparison Lemma)**.** *Consider the scalar differential equation*

$$\dot{u} = f(t, u), \quad u(t_0) = u_0$$

where $f(t, u)$ is continuous in t and locally Lipschitz in u, for all $t \geq 0$ and all $u \in J \subset \mathbb{R}$. Let $[t_0, T)$ be the maximal interval of existence of the solution $u(t)$, and suppose $u(t) \in J$ for all $t \in [t_0, T)$. Let $v(t)$ be a continuous function whose upper right-hand derivative $D^+v(t)$ satisfies the differential inequality

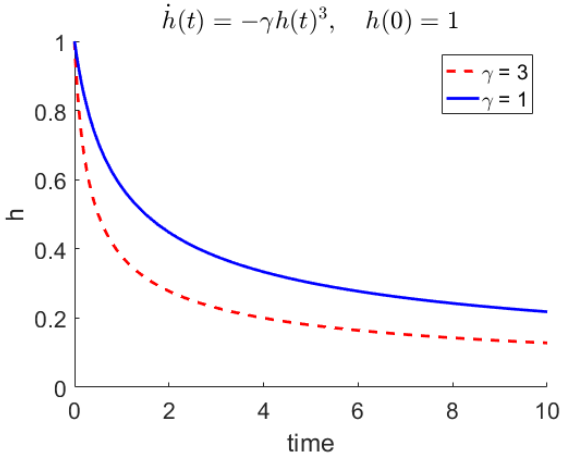
$$D^+v(t) \leq f(t, v(t)), \quad v(t_0) \leq u_0$$

with $v(t) \in J$ for all $t \in [t_0, T)$. Then, $v(t) \leq u(t)$ for all $t \in [t_0, T)$.

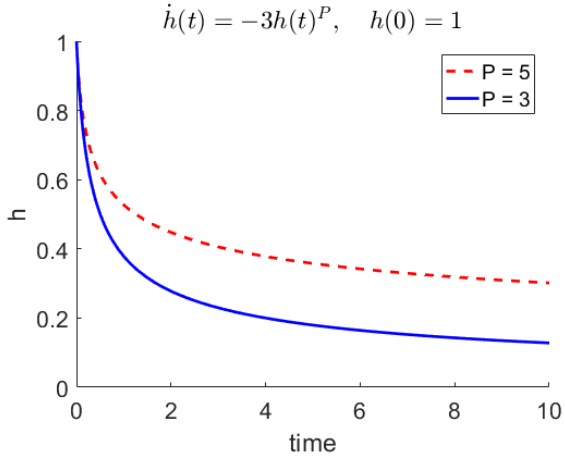
Let $\dot{k}(t) = \mathcal{F}(k(t), t) = \gamma k(t)^3$; different examples of this function are showed in Figure 3.3 and we note that $k(t) \geq 0$, for all t if $k(t_0) \geq 0$. Thanks to the Comparison Lemma, we can state that if $\dot{h}(t) \geq \mathcal{F}(h(t), t)$ and $h(t_0) \geq k(t_0)$, then $h(t) \geq k(t) \geq 0$, for all $t \in [t_0, T)$. It is trivial to see now that if

$$\dot{h} \geq -\gamma h^3, \tag{3.28}$$

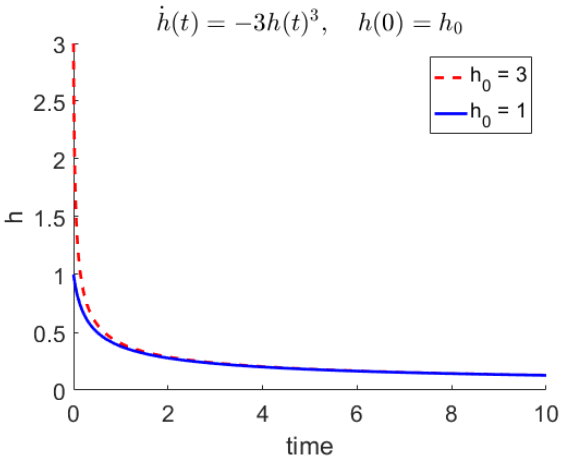
then $h(t) \geq 0$. Since, from (3.27) we have $\dot{h}(x) = L_f h(x) + L_g h(x)u$, we get (3.26). □



(a) Different values of γ.



(b) Different values of P.



(c) Different values of h_0.

Figure 3.3: Different examples of the time evolution of $\dot{h}(t) = \gamma h(t)^P$, based on different parameters combination.

3.3 Safety Barrier Certificates

The goal of the Safety Barrier Certificates is to ensure safety, avoiding agent to agent collisions, as well as agent to obstacles collisions. It is now important to determine the factors in play to, possibly, obtain a function $h(\cdot)$ that is a Zeroing Control Barrier Function, therefore guaranteeing that the state of the system, if safe at time t_0, is kept safe in the future; based on the work by Li Wang (Wang et al., 2017) a Centralized algorithm is first developed, then expanded in a Decentralized fashion in order to ensure scalability in multiagent systems.

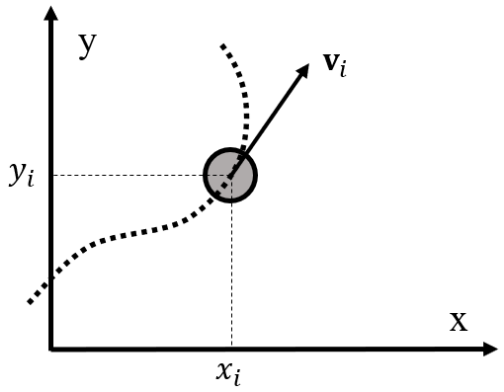


Figure 3.4: Geometric representation of agent i's trajectory on the plane $P = (x, y)$. The vector $\mathbf{p}_i = (x_i, y_i)$ encodes the coordinates of the agent's position at any given time, while $\mathbf{v}_i$ is the agent's velocity.

3.3.1 Deceleration limits and safety distance

Consider a multirobot system consisting of N planar, mobile robots, indexed by $\mathcal{N} = \{i \mid i = 1, 2, ..., N\}$. The easiest way of modeling a planar robot in the $2D$ space is that of a Single Integrator, see section A.1; however, it is reasonable to think that, especially for bigger robots, acceleration (and therefore deceleration) limits play an important role in the agent's dynamics. For this reason, in the following, the more accurate Double Integrator is used to model all agents (see section A.4), here recalled

$$\begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\mathbf{v}}_i \end{bmatrix} = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{v}_i \end{bmatrix} + \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} \mathbf{u}_i \tag{3.29}$$

where $\mathbf{p}_i \in \mathbb{R}^2$, $\mathbf{v}_i \in \mathbb{R}^2$, and $\mathbf{u}_i \in \mathbb{R}^2$ represent the positions, velocities, and inputs (acceleration commands) for agent i and velocity and acceleration of the agent are limited by $\|\mathbf{v}_i\|_\infty \leq \beta$ and $\|\mathbf{u}_i\|_\infty \leq \alpha$. The aggregate state of all N agents positions and velocities will be denoted as $(\mathbf{p}^T, \mathbf{v}^T)^T \in \mathbb{R}^{4N}$.

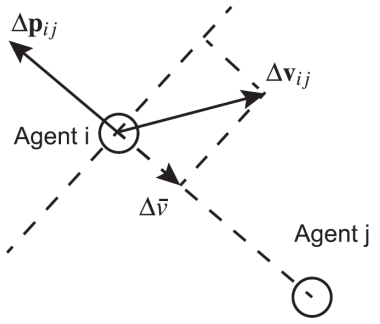


Figure 3.5: Relative position and velocity between two agents.

To guarantee robot to robot safety, we define D_s as the safety distance between any two agents i and j; our goal is to ensure that the distance between the two agents, $\Delta\,\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ is never lesser that the safety distance. Let $\Delta\,\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ be the relative velocity of agent i with respect to agent j; the actual component that will eventually lead to a collision between the two agents is actually the normal component of the agents' relative velocity:

$$\Delta\bar{v} = \|\dot{\Delta\mathbf{p}}_{ij}\| = \frac{\Delta\,\mathbf{p}_{ij}^T}{\|\Delta\,\mathbf{p}_{ij}\|}\Delta v_{ij} \tag{3.30}$$

while the tangent component of $\Delta\,\mathbf{v}_{ij}$ translates in a relative rotation about each agent; this is shown in Figure 3.5. Since we want the two agents to never get closer than D_s, we need to make sure that they can always decelerate on time to keep the safe distance, that is that the actual robot distance plus the distance needed to stop (when both agents apply maximum negative acceleration) is still greater than D_s. Formally

$$\|\Delta\mathbf{p}_{ij}\| + \int_{t_0}^{t_0+T_b} \Delta\bar{v}(t_0) + (\alpha_i + \alpha_j)dt \geq D_s, \quad \forall i \neq j \tag{3.31}$$

where $T_b = \frac{0-\Delta\bar{v}(t_0)}{\alpha_i+\alpha_j}$ is the time needed to reach zero relative velocity $\Delta\bar{v}(t_0 + T_b) = 0$ while the maximum breaking acceleration is applied to both robots $(\alpha_i + \alpha_j)$. Solving the integral in 3.31 leads us to the inequality

$$\|\Delta\mathbf{p}_{ij}\| - \frac{(\Delta\bar{v})^2}{2(\alpha_i + \alpha_j)} \geq D_s, \quad \forall i \neq j. \tag{3.32}$$

Combining equation 3.30 with equation 3.32, and noting that the constraints must be enforced only when the two agents are heading one toward the other, that is, when $\Delta\bar{v} \leq 0$, we obtain

$$-\frac{\Delta\,\mathbf{p}_{ij}^T}{\|\Delta\,\mathbf{p}_{ij}\|}\Delta v_{ij} \leq \sqrt{2(\|\Delta\mathbf{p}_{ij}\| - D_s)(\alpha_i + \alpha_j)}. \tag{3.33}$$

This condition must always hold, for every agent $i \in N$ and any time instant $t \in [t_0, +\infty)$. We now have a constraint we can use as ZCBF, so that $h(x) \geq 0$, that is

$$h_{ij}(\mathbf{p}, \mathbf{v}) = \sqrt{2(\|\Delta\mathbf{p}_{ij}\| - D_s)(\alpha_i + \alpha_j)} + \frac{\Delta\,\mathbf{p}_{ij}^T}{\|\Delta\,\mathbf{p}_{ij}\|}\Delta v_{ij} \geq 0. \tag{3.34}$$

3.3.2 Building a Zeroing Control Barrier Function candidate $h(\cdot)$

Since the ZCBF $h(x)$ in (3.12)–(3.14) is a function of the aggregate states, we want to express it as a function of each agents' position and velocity for the system in (3.29), i.e., $h_{ij}(\mathbf{p}, \mathbf{v})$. Considering

the interaction of two agents, i and j, we can define the pairwise set $\mathcal{C}_{ij}$ as

$$\mathcal{C}_{ij} = \{(\mathbf{p}_i, \mathbf{v}_i) \in \mathbb{R}^4 \mid h_{ij}(\Delta\mathbf{p}_{ij}, \Delta\mathbf{v}_{ij}) \geq 0\}, \quad \forall i \neq j. \tag{3.35}$$

Combining (3.34) and (3.26), the safety barrier constraint can be written as

$$-\Delta\mathbf{p}_{ij}^T \Delta\mathbf{u}_{ij} \leq \gamma h_{ij}^3 \left\|\Delta\mathbf{p}_{ij}\right\| - \frac{(\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij})^2}{\left\|\Delta\mathbf{p}_{ij}^2\right\|} + \left\|\Delta\mathbf{v}_{ij}\right\|^2 + \frac{(\alpha_i + \alpha_j)\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij}}{\sqrt{2(\alpha_i + \alpha_j)(\left\|\Delta\mathbf{p}_{ij}\right\| - D_s)}}, \tag{3.36}$$

that is a linear constraint in $\mathbf{u}_i$, that can be represented as

$$A_{ij}\mathbf{u}_i \leq b_{ij}. \tag{3.37}$$

We can then write $A_{ij}\mathbf{u}_i \leq b_{ij}$ as

$$A_{ij} = [0, ..., -\Delta\mathbf{p}_{ij}^T, ..., \Delta\mathbf{p}_{ij}^T, ..., 0] \tag{3.38}$$

and

$$b_{ij} = \gamma h_{ij}^3 \left\|\Delta\mathbf{p}_{ij}\right\| - \frac{(\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij})^2}{\left\|\Delta\mathbf{p}_{ij}^2\right\|} + \left\|\Delta\mathbf{v}_{ij}\right\|^2 + \frac{(\alpha_i + \alpha_j)\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij}}{\sqrt{2(\alpha_i + \alpha_j)(\left\|\Delta\mathbf{p}_{ij}\right\| - D_s)}}. \tag{3.39}$$

Therefore, for any $\mathbf{u}_i$ satisfying the inequality in (3.37), we ensure that the control input is safe, that is, the acceleration inputs will keep the state of the system in (3.18) such that the relative velocity and position of any two agents will not bring to a collision.

Given a nominal controller $\hat{\mathbf{u}}_i$ for the system in (3.18), if the condition in (3.37) holds for $\hat{\mathbf{u}}_i$, the nominal control input is safe, therefore making $\mathbf{u}_i^* = \hat{\mathbf{u}}_i$ will pose no harm to the system. However, since we have no control over the time evolution of nominal controller, no guarantee can be given. In the following section we are going to provide a tool to keep the control $\mathbf{u}_i^*$ safe when the linear inequality in (3.37) is not satisfied by the nominal controller $\hat{\mathbf{u}}_i$.

3.4 Centralized Safety Barrier Certificates

We have seen how we can express safety as an inequality constraint that is linear with respect to the control input $\mathbf{u}_i$. Recall how we wish our collision avoidance strategy to be minimally invasive, i.e., the control to our system should be as close as possible to the desired (high-level) goal when no collision is imminent—note that this is relateble to $u = k(x, \hat{\mathbf{u}})$ of Figure 3.2. We can efficiently solve this issue by means of a quadratic programming (QP) problem, where we penalize deviation

of the control input from the nominal control in the least-squares sense. In particular, let $\mathbf{u}_i$ be the **actual control command** and let $\hat{\mathbf{u}}_i$ be the **nominal controller** (high-level control, e..g, go-to-point) we ensure safety by solving

$$\begin{aligned} \mathbf{u}^* = \underset{\mathbf{u}\in\mathbb{R}^{2N}}{\arg\min} \quad & J(\mathbf{u}) = \textstyle\sum_{i=1}^{N} \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2 \\ \text{subject to} \quad & A_{ij}\mathbf{u}_i \leq b_{ij}, \quad \forall i \neq j \\ & \|\mathbf{u}_i\|_\infty \leq \alpha_i, \quad \forall i \in \mathscr{N}. \end{aligned} \tag{3.40}$$

This is a centralized approach, in the sense that the resulting optimal controller $\mathbf{u}^*$ is provided to each agent by a central computing server that is omniscient about the network. This approach, although very efficient for small network, has a number of disadvantages, that are better explained with the help of simulation results in the next chapter.

3.4.1 Simulation Results

As a comparison tool with the results presented in the upcoming sections of this work, the conclusion so far described are simulated in MATLAB and are here shown. In Figure 3.6, two agents are forced to drive one against the other, a situation we previously defined as head-on scenario, described in more details in Section 1.1 and Figure 1.1.

The high level controller $\hat{\mathbf{u}}$ is in the form

$$\hat{\mathbf{u}}_i = -k_p(\mathbf{g}_i - \mathbf{p}_i) - k_v\mathbf{v}_i \tag{3.41}$$

where $\mathbf{g}_i \in \mathbb{R}^2$ are the coordinates of the goal point in the plane and $k_p, k_v \in \mathbb{R}$ are the gains for the position and velocity, respectively.

The first agent begins at position $\mathbf{p}_1 = [0.3, -0.3]$ and has goal $\mathbf{g}_1 = [-0.3, 0.3]$, while $\mathbf{p}_2 = \mathbf{g}_1$ and $\mathbf{g}_2 = \mathbf{p}_1$. This first result is shown in Figure 3.6.

While the agents are far enough from each other they follow the nominal control input $\hat{\mathbf{u}}$, heading to the goal position. When no collision is imminent, the two agents follow the original control goal. While the agents get closer, the centralized controller in (3.40) successfully perturbs the agents' paths in order to avoid collisions and, therefore, ensuring safety. The QP problem in MATLAB is solved efficiently with the **quadprog** command, however presents scalability issues, as it is often the case for centralized approaches with an undefined growing number of agents. In particular, the

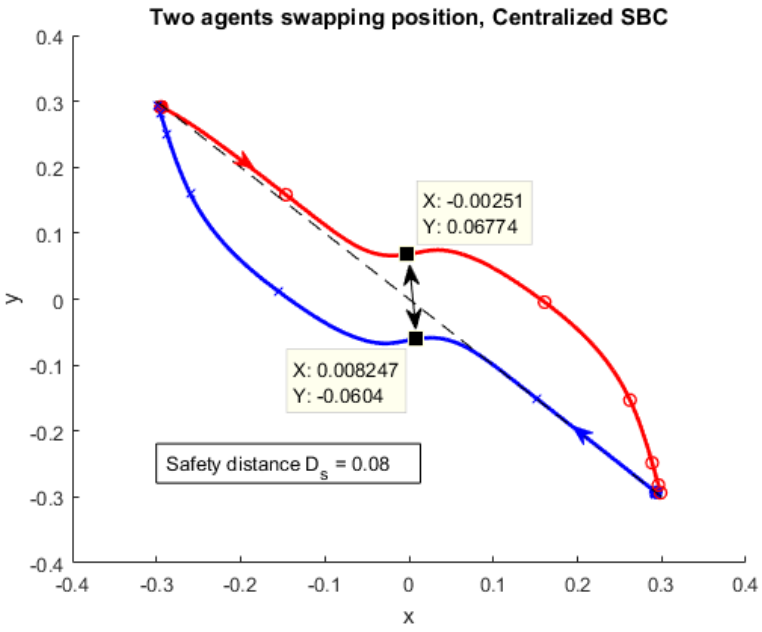
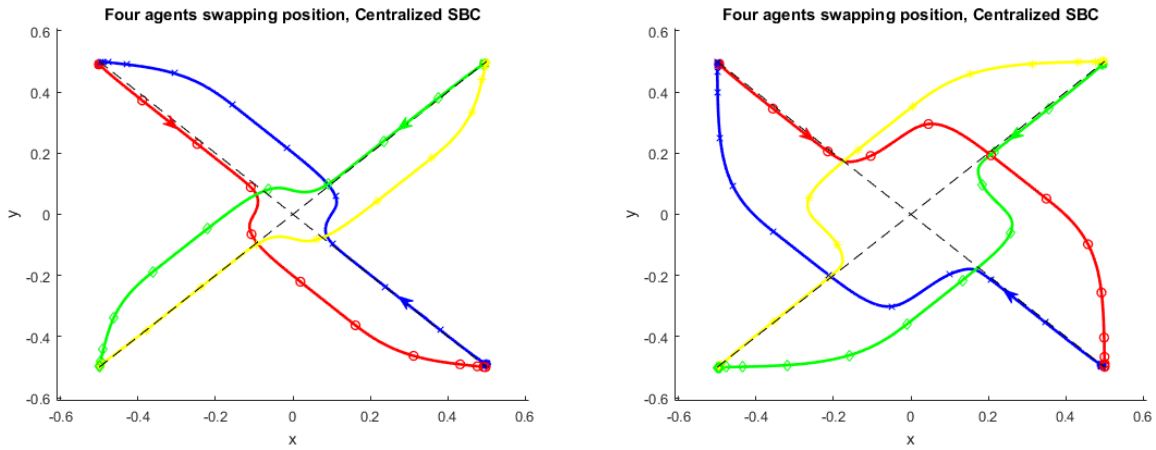


Figure 3.6: Head-on with two agents with centralized safety barrier certificates. Circles and crosses show a time lapse of 61 iterations for the respective agent (2 seconds).

number of decision variables is $2N$, with N being the number of agents in the network; the number of liner constraints is, in the worst case, $\frac{N(N-1)}{2}$.

The reader may note an inconsistency with this example and the definition of safety distance D_s. We defined the safety distance as the minimum distance any two agents must keep at any given time; however, as highlighted in Figure 3.6, at time $t = 4s$ the distance among the two agents is $\Delta \mathbf{p} = 0.069$, while we imposed $D_s = 0.08$. This is due to the model used for the software simulations. In fact, the simulation environment used for this tests is modeled with the unicycle dynamics—better suited for a future hardware implementation, while our original system of study is that of a double integrator—easier to manipulate. In Appendix A, we recall the differences between these models, while also providing a transformation tool from one to the other. Since the transformation operates thanks to approximations, the resulting agents' behaviors may be different from the expected one; for this reason it is necessary to carefully choose the safe distance as a robustness parameter to model the safety of our system.

In Figure 3.7, the same experiment is performed with four agents crossing paths, for different values of safe distance. It is worth noting that the deconflicting motion is different from Figure 3.7a to Figure 3.7b, as in the first simulation the agents describe a counterclockwise motion (by



(a) $D_s = 0.08$. (b) $D_s = 0.4$.

Figure 3.7: Head-on scenarios for a network with four agents and centralized SBC with different values of safety distance D_s.

all steering to the right), while in the second example they turn in a clockwise fashion. This behavior is not dependent on the particular choice of D_s, but it depends on the small asymmetries in the positioning of the agents in the network; in this scenario is therefore hard to predict how the deconfliction motion will be resolved, if right-handedly or left-handedly. Since our goal is to encode traffic rules in the agents' behavior, it is clear that one of the issues to be addressed is this of finding a deterministic side to deconfliction head on scenarios, possibly, depending on local road rules.

3.5 Decentralized Safety Barrier Certificates

Centralized SBC face significantly increased communication and computation burden when the size of the robotic swarm grows. To address this scalability issue, it is desirable to have decentralized barrier certificates, acting only on locally based information, while still ensuring the overall safety of the system. In the nominal case, the time evolution of the system in (3.18) will be regulated by the nominal controller $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1^T, ..., \hat{\mathbf{u}}_N^T)^T$; as collisions approach, we wish for the actual control input $\mathbf{u} = (\mathbf{u}_1^T, ..., \mathbf{u}_N^T)^T$ to be save, whereas stay, pairwise, as close a possible to $\hat{\mathbf{u}}$. Following strategy A of (Wang et al., 2016a), we distribute b_{ij} to two robot agents, rendering the pairwise

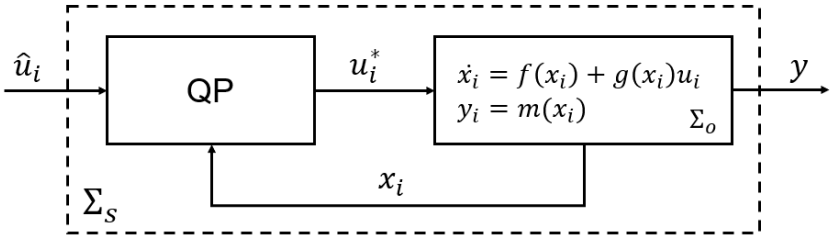


Figure 3.8: Input–output relationships for the original system Σ_o and for the safe, collision free, system Σ_s.

safety barrier constraint between agent i and j

$$-\Delta\mathbf{p}_{ij}^T\mathbf{u}_i \leq \frac{\alpha_i}{\alpha_i+\alpha_j}b_{ij}, \tag{3.42}$$

$$\Delta\mathbf{p}_{ij}^T\mathbf{u}_j \leq \frac{\alpha_j}{\alpha_i+\alpha_j}b_{ij}. \tag{3.43}$$

With the decentralized constraints, each agent can compute its own QP problem, assuming each agent has sensing capabilities for its neighbors: each agent $i \in \mathscr{N}$ runs its own version of QP-bases controller

$$\begin{aligned} \mathbf{u}_i^* = \underset{\mathbf{u}_i\in\mathbb{R}^2}{\arg\min} \quad & \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2 \\ \text{subject to} \quad & \bar{A}_{ij}\mathbf{u}_i \leq \bar{b}_{ij}, \quad \forall j \in \mathscr{N}_i \\ & \|\mathbf{u}_i\|_\infty \leq \alpha_i \end{aligned} \tag{3.44}$$

where $\bar{A}_{ij} = -\Delta\,\mathbf{p}_{ij}^T$ and $\bar{b}_{ij} = \frac{\alpha_i}{\alpha_i+\alpha_j}b_{ij}$.

$\mathscr{N}_i$ is the neighboring set of agent i, defined in (Wang et al., 2017) as

$$\mathscr{N}_i = \{j \in \mathscr{N} \mid \left\|\Delta\mathbf{p}_{ij}\right\| \leq D_{\mathscr{N}}^i,\, j \neq i\} \tag{3.45}$$

where

$$D_{\mathscr{N}}^i = D_s + \frac{1}{2(\alpha_i+\alpha_{\min})}\left(\sqrt[3]{\frac{2(\alpha_i+\alpha_{\max})}{\gamma}} + \beta_i + \beta_{\max}\right)^2 \tag{3.46}$$

is the size of the radius of the neighbors, where $\alpha_{\min} = \min_{j\in\mathscr{N}}\{\alpha_j\}$, $\alpha_{\max} = \max_{j\in\mathscr{N}}\{\alpha_j\}$ and $\beta_{\max} = \max_{j\in\mathscr{N}}\{\beta_j\}$ are the lower and upper bounds of all agents' acceleration limits and the upper bound of all agents' velocity limits, respectively. Using a controller $\mathbf{u}_i^*$ as the one defined in (3.44) and $\mathbf{u}^* = (\mathbf{u}_1^{*T}, ..., \mathbf{u}_N^{*T})^T$, ensure safety of the system since collisions are always avoided.

The resulting system is described in Figure 3.8, where Σ_O is the original system and Σ_S is the modified, provably safe, system.

3.5.1 Simulation Results

As in the case of the centralized version of SBC, we dedicate this section to show simulation results for the distributed counterpart of the safe resolution algorithm. The obvious advantage of this

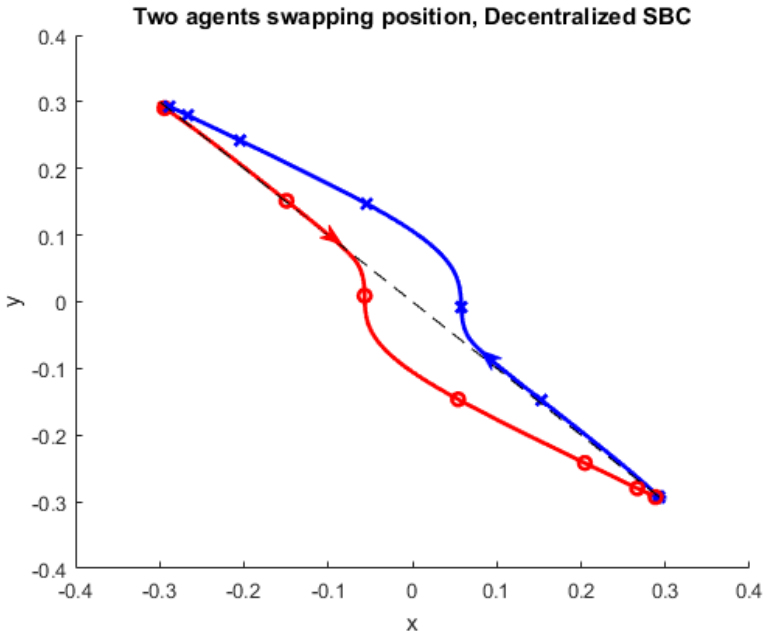


Figure 3.9: Head-on with two agents with decentralized safety barrier certificates. Circles and crosses show a time lapse of 61 iterations for the respective agent (2 seconds).

solution is that the computational effort is shared among each agent, resulting in a intrinsically scalable approach. We note that in this case the number of decision variables is always 2, since $\mathbf{u}_i \in \mathbb{R}^2$, for all $i \in \mathscr{N}$; the number of constraints, that depends on the number of neighboring agents $\mathscr{N}_i$, is always lesser or equal to $N - 1$.

The downside of this approach is that every agent has only local knowledge about the network—and therefore of the other agents' goals and constraints. This will translate in the single agents solving their QP problem regardless of what the other agents are trying to pursue.

TODO: describe two agents case

Consider Figure 3.10, where four agents are heading towards the same point in space, and confront it with Figure 3.7, where the same scenario is solved with centralized SCB. The original

objective of ensuring safety is achieved, however the goals of the four agents result in a conflicting motion, not allowing the agents to *separately* solve such conflict. Note how the agents lose a significant amount of time trying to go to the goal position, while keeping the sytem safe, basically stopping—small motions still take place due to the network asymmetries. This scenario, where the motion to keep the agent safe and the agent's goal control result in the agent stopping, is referred to as deadlock in (Wang et al., 2016b).

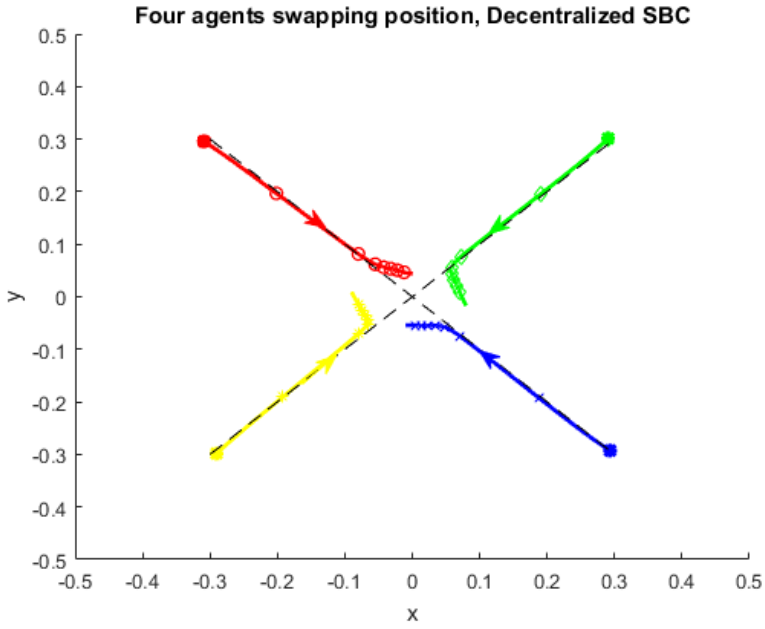


Figure 3.10: Head-on for a four agents network, with decentralized safety barrier certificates. The agents goals are in conflict, resulting in the agents essentially stopping (note that the agents spend around 12 seconds facing one another).

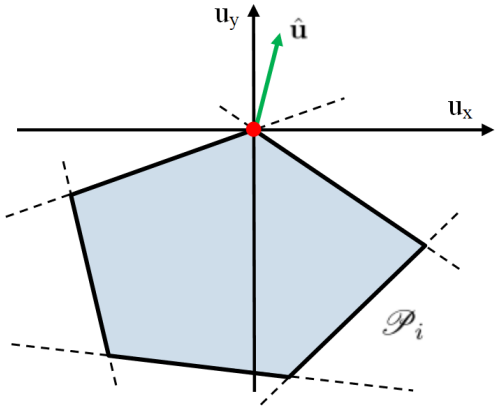
3.6 Deadlock detection and resolution

Although safety is guaranteed, there are situations where the constraints to the Decentralized SBC QP problem in (3.44) are too strong; this could result in one or more agents stopping (acceleration and velocity are both zero), therefore preventing the fulfillment of the original agents' goal $\hat{\mathbf{u}}_i$.

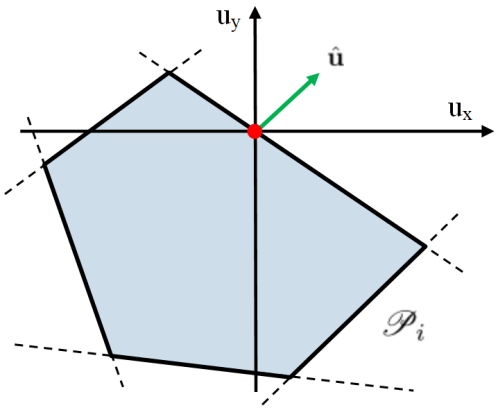
Consider the decentralized admissible control space, $\mathscr{P}_i$, for agent i

$$\mathscr{P}_i = \{\mathbf{u}_i \in \mathbb{R}^2 \mid \bar{A}_{ij}\mathbf{u}_i \leq \bar{b}_{ij}, \quad \forall i \neq j\}. \tag{3.47}$$

The size of the feasible control space, termed *width of the feasible set* (Morris et al., 2013), can be



(a) Type one deadlock: the optimal solution ∈ vertex of the admissible control space



(b) Type two deadlock: the optimal solution ∈ edge of the admissible control space

Figure 3.11: Graphical representation of the QP problem in (3.44). The plane is the two dimensional control input space $\mathbf{u} = (u_x, u_y)$. The solid polygon is the admissible control space $\mathscr{P}_i$ of agent i, generated from the intersection of the inequality constraints (dotted lines). The cost function is the vector $\|\mathbf{u}_i - \hat{\mathbf{u}}_i\|$, and in these particular cases, the optimal solution is $\mathbf{u}_i^* = \mathbf{0}$.

evaluated with a Linear Program (LP)

$$\begin{aligned} \min_{\mathbf{u}_i \in \mathbb{R}^2, \delta_{LP} \in \mathbb{R}} \quad & \begin{bmatrix} \mathbf{0}_{1\times 2} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \delta_{LP} \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} \bar{A}_{ij} & -1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \delta_{LP} \end{bmatrix} \leq \bar{b}_{ij}, \quad \forall j \in \mathscr{N}_i \\ & \|\mathbf{u}_i\|_\infty \leq \alpha_i. \end{aligned} \tag{3.48}$$

The solution of the LP characterizes how much control margin is left for the strict safety barrier constraint. If $\delta_{LP} \leq 0$, the corresponding QP is solvable.

Definition 3.6.1 (Deadlock)**.** A robot agent i is said to be in deadlock if it remains stationary, that is, if the solution to (3.44) is $\mathbf{u}_i^* = 0$ and the speed is zero ($\mathbf{v}_i = 0$), while the nominal control command is $\|\hat{\mathbf{u}}_i\| \neq 0$.

In (Wang et al., 2017), three different deadlock scenarios are identified:

1. Type 1 deadlock: $\delta_{LP} \leq 0, u_i = 0 \in \text{vertex}(\mathscr{P}_i)$;
2. Type 2 deadlock: $\delta_{LP} \leq 0, u_i = 0 \in \text{edge}(\mathscr{P}_i)$;
3. Type 3 deadlock: $\delta_{LP} > 0$.

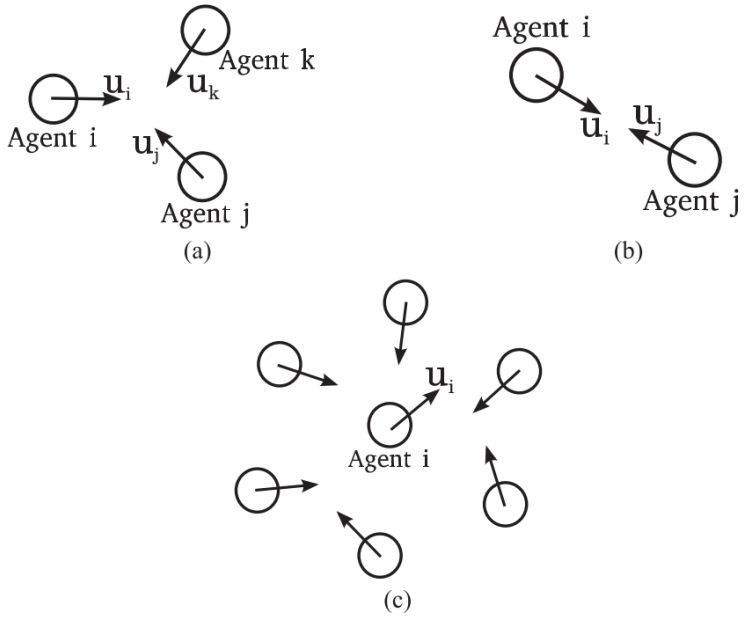


Figure 3.12: Three types of Deadlocks for robot agent i in a multirobot system. (a) Type 1 deadlock. (b) Type 2 deadlock. (c) Type 3 deadlock.

In the present work, we are going to build on the results in (Wang et al., 2017) to define deconfliction tools compatible with traffic rules.

4 Two agents head-on deconfliction

As a first step towards understanding the general motion deconfliction problem, we consider a two agents system, $N = 2$, where the robots are forced to interact in a pattern reletable to Figure 1.1. With this assumption of $N = 2$, $\mathscr{N}_i = \{j\},\ i \neq j$ and $i, j = 1, 2$ if $\left\|\Delta \mathbf{p}_{ij}\right\| \leq D^i_{\mathscr{N}}$; $\mathscr{N}_i = \{\emptyset\}$, otherwise. This allows us to limit the deadlock types as seen in Definition 3.6.1, since we will never have more than one inequality constraint in the admissible control space in $\mathscr{N}_i$, for all $i \in \mathscr{N}$. For this reason, any solution $\mathbf{u}_i^* \in \text{edge}(\mathscr{P}_i)$ and Type 1 deadlock cannot occur.

Definition 4.0.1 (Quasi-deadlock)**.** A robot agent i is said to be in a quasi-deadlock if $\|\mathbf{u}_i\| \leq \underline{\alpha}$, $\|\mathbf{v}_i\| \leq \underline{\beta}$, and the nominal control $\|\hat{\mathbf{u}}_i - \mathbf{u}_i\| > \underline{\epsilon}$,

where $\underline{\alpha}$, $\underline{\beta} \in \mathbb{R}^+$ are the acceleration and velocity thresholds, respectively, and $\underline{\epsilon} \in \mathbb{R}^+$ is the control threshold. By introducing a lower bound on the acceleration and velocity, we wish to identify those agents that are slowing down, while the difference between the nominal and the actual controller, ϵ, ensures that this slowing evolution is due to the constraints introduced in (3.37) and not by the actual control goal.

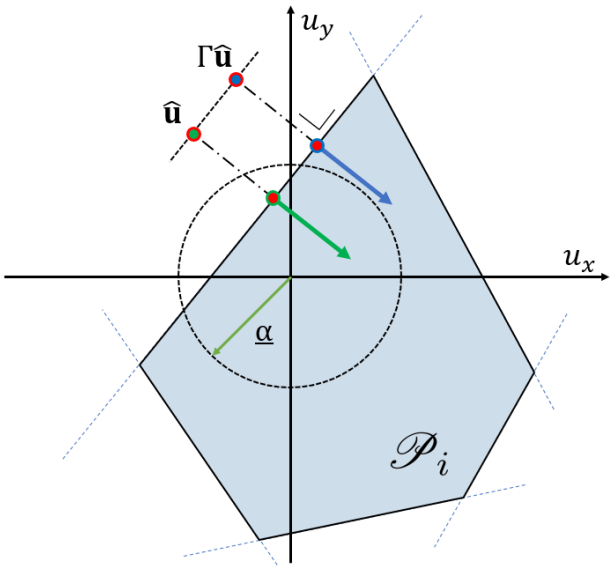


Figure 4.1: Graphical representation of $\mathscr{P}_i$, $(N_i = 5)$ with quasi-deadlock resolution: $\hat{\mathbf{u}}$ is projected resulting in $\Gamma\hat{\mathbf{u}}$ and consequently the optimal solution $\mathbf{u}^*$ changes.

4.1 Robot–robot interaction

We mentioned, in the problem statement in Section 1.1, that our goal is to find a conflict resolution tool of motion paths for mixed human–autonomous networks. However, all the tools introduced so far are meant for fully autonomous networks, and safety is ensured only if every agent acts accordingly to the defined rules—distributed SBC.

In this Sections, we are going to start developing the new tools, building on the prerequisites presented in Chapter 3, limited to a strictly autonomous network. In Section 4.1 we will generalize this results to be compatible with a mixed network, allowing one agent to be remotely operated by a human, a scenario more similar to the original problem statement.

4.1.1 Type 2 quasi-deadlock resolution

If an agent enters a deadlock it stops; (Wang et al., 2017) presents a deadlock resolution tool, however this requires that the agent enters a deadlock. This results in a slow reacting system as, before taking any deconfliciting motion, any agent must come to a complete stop. The quasi-deadlock aims at detecting when an agent is about to enter a deadlock, allowing the controller to take preventive actions without the need for the robot agent to come to a complete alt.

The conflict resolution tool proposed in the present work can be resumed as follows: given the nominal control $\hat{\mathbf{u}}_i$, if agent i finds itself to be in quasi-deadlock, perturb $\hat{\mathbf{u}}_i$ such that the new nominal control is $\Gamma_i\hat{\mathbf{u}}_i$ where $\Gamma_i = I + k_{\gamma i}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}$ and $k_{\gamma i} \in \mathbb{R}$ for all $i \in \mathscr{N}$. This is graphically explained in Figure 4.1. $\Gamma\hat{\mathbf{u}}_i$ is a nominal perturbation to the left, or to the right, of $\hat{\mathbf{u}}_i$, depending on the sign of $k_{\gamma i}$.

This translates in $k_{\gamma i}$ encoding the concept of left-hand and right-hand driving, based on its sign. In fact, if $k_{\gamma i} > 0$ ($k_{\gamma i} < 0$) the control input is perturbed to the left (right): when an agent i slows down due to the constraint imposed by the presence of another agent j, agent i will reshape its control input, steering slightly to the left (or right), bases on sign$\{k_{\gamma i}\}$. Moreover, according to the module of $k_{\gamma i}$, the perturbation will be more or less aggressive, that is, $\left\|\hat{\mathbf{u}}_i - \Gamma_i^{(1)}\hat{\mathbf{u}}_i\right\| \geq \left\|\hat{\mathbf{u}}_i - \Gamma_i^{(2)}\hat{\mathbf{u}}_i\right\|$, if $k_{\gamma i}^{(1)} > k_{\gamma i}^{(2)}$. For this reasons, we will also refer to $k_{\gamma i}$ as agent i's *direction bias*. The quasi-deadlock resolution is summarized in Algorithm 1.

Algorithm 1 Type 2 quasi-deadlock resolution

input $\mathbf{u}_i, \hat{\mathbf{u}}_i, \mathbf{v}_i$
$\gamma \leftarrow Decentralized\_LP$
if $\|\mathbf{u}_i\| \leq \underline{\alpha}$ & $\|\mathbf{v}_i\| \leq \underline{\beta}$ & $\|\hat{\mathbf{u}}_i - \mathbf{u}_i\| > \underline{\epsilon}$ & $\gamma \leq 0$ **then**
  $\hat{\mathbf{u}}_i \leftarrow \Gamma_i\hat{\mathbf{u}}_i$
$\mathbf{u}_i^* \leftarrow Decentralized\_QP(\hat{\mathbf{u}}_i)$ **return** $\mathbf{u}_i$

4.1.2 Simulation results

This translates in $k_{\gamma i}$ encoding the concept of left-hand and right-hand driving, based on its sign. According to the module of $k_{\gamma i}$, the perturbation will be more or less aggressive. For this reasons, we will also refer to $k_{\gamma i}$ as agent i's *direction bias*.

In Figure 4.3a, a software simulation is showed, where both agents are hading on with $k_{\gamma i} = 1$. As expected, when the robots enter the quasi-deadlock, they start perturbing their controller to the left, resulting in both agents steering to the left.

It is interesting to investigate how the two agents behave with different values of $k_{\gamma i}$, as described in Figures 4.3b and 4.3c, where agent 1 (dots) has $k_\gamma = -1$ and agent 2 (crosses) has $k_\gamma = 1$. As expected, both agents steer specularly, resulting in a new conflicting motion that will be resolved differently based on the particular boundary conditions, as can be seen from the differences in Figure 4.3b and Figure 4.3c.

Another important result is that, if the acceleration and velocity limits are properly tuned, the quasi-deadlock detection, as defined in Definition 4.0.1, will affect the control even when the two agents are not perfectly aligned, in a pattern that matches our goal designed in Figure 1.1. In Figure 4.2, computer simulations for misaligned head-ons are shown.

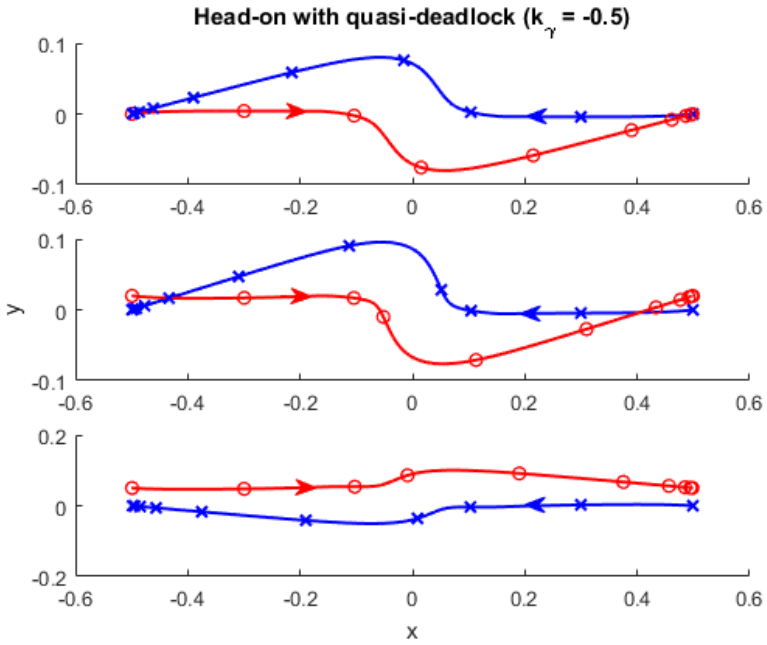


Figure 4.2: Quasi-deadlock resolution to different head-on scenarios; each marker indicates 61 iterations ($\sim$ 2 seconds).

In Figure 4.2a, the agents are perfectly aligned, resulting in a specular deconfliction to the one seen in Figure 4.3a, as expected from the negative sign of $k_{\gamma i}$ for both agents. In Figure 4.2b, a slight misalignment on the y-axis is present, resulting in a scenario similar to Figure 1.1b—c: the two agents steer to the right, in compliance to the encoded road rules. Finally, in Figure 4.2c the misalignment is significant, resulting in both agents keeping their side, again in compliance to the original goal.

4.1.3 Direction bias estimation

Assume that every agent's goal is shared on the network, therefore $\hat{\mathbf{u}}_j$ is known by every agent i, while $k_{\gamma j}$ is not. We also assume that the final optimal control $\mathbf{u}_j^*$ in Σ_T is known by every agent. As a first approach, we are going to ignore limits on the acceleration input, i.e. $\|\mathbf{u}_j\|_\infty = +\infty$. Can agent i learn $k_{\gamma j}$ from observing j's behavior?

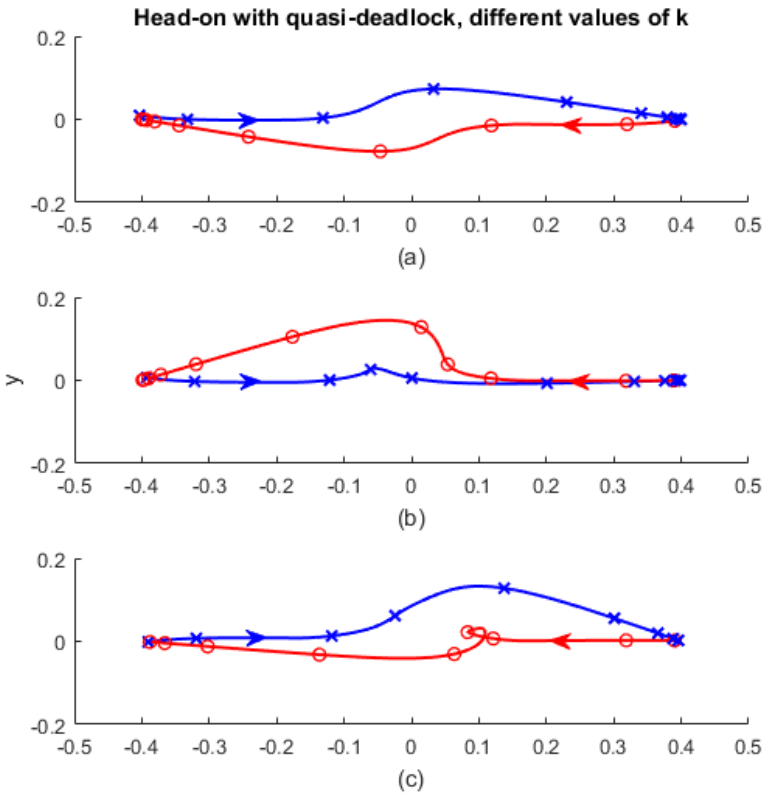


Figure 4.3: Reaction of SBC to quasi-deadlock in head-on scenario for two agents moving in the plane $P = (x, y)$; agent 1 (dots) starts at (0.4, 0) and drives to (-0.4, 0), swapping position with agent 2 (crosses). a) k_γ positive: left-hand driving; b–c) different values of $sign\{k_\gamma\}$ result in a conflicting motion. Each marker indicates 61 iterations ($\sim$ 2 seconds).

Proposition 1. Consider the QP problem in (3.44), where $\alpha_i = +\infty$, $\forall j \in \mathcal{N}$, with the quasi-deadlock resolution defined in Algorithm 1. If the nominal controller $\hat{\mathbf{u}}_j$, and the solution to the QP problem $\mathbf{u}_j^*$ are known for all $j \in \mathcal{N}_i$, then agent i can compute $k_{\gamma j}$ as in (4.6), where $k_{\gamma j} = k_{\gamma j}^{\text{est}}$.

Proof: Let us again consider the simple scenario of two agents driving toward each other (position swapping) as described in the previous sections. Since we are solving the problem in (3.44), a QP problem in $\mathbb{R}^2$, we know that the solution $\mathbf{u}_j^*$ will solve all inequality constraints and at least one of the inequalities will actually be solved as an equality (Boyd and Vandenberghe, 2004). Using the assumption that only two agents are present in the network, we can conclude that the matrix A_{ji} will always have one row, and therefore the solution will always be along the line $\bar{A}_{ji}\,\mathbf{u}_j^* = \bar{b}_j$, where $\bar{A}_{ji} \in \mathbb{R}^{1\times 2}$ and $\bar{b}_j \in \mathbb{R}$. We can express the cost function in (3.44) as

$$\|\mathbf{u} - \Gamma\,\hat{\mathbf{u}}\|^2 = \mathbf{u}^T\,\mathbf{u} + \hat{\mathbf{u}}^T\Gamma^T\Gamma\,\hat{\mathbf{u}} - 2\mathbf{u}^T\Gamma\,\hat{\mathbf{u}}\,. \tag{4.1}$$

where, to ease up notation, we define $\hat{\mathbf{u}}_j = \hat{\mathbf{u}} = (\hat{u}_x, \hat{u}_y)^T$ and $\mathbf{u}_j = \mathbf{u} = (u_x, u_y)^T$. Using the Lagrangian multiplier $\lambda \in \mathbb{R}$, we can express the equality

$$H = \mathbf{u}^T \mathbf{u} + \hat{\mathbf{u}}^T \Gamma^T \Gamma \hat{\mathbf{u}} - 2\mathbf{u}^T \Gamma \hat{\mathbf{u}} + \lambda(A\mathbf{u} - b) \tag{4.2}$$

and, differentiating H along $\mathbf{u}$ we get

$$\frac{\partial H}{\partial \mathbf{u}} = 2\mathbf{u} - 2\Gamma \hat{\mathbf{u}} + \lambda A^T = 0. \tag{4.3}$$

Defining $\bar{A}_{ji} = A = [A_{11}, A_{12}]$ and $\bar{b}_j = b$, and recalling that $A\mathbf{u} = b$, we obtain $\mathbf{u} = \Gamma \hat{\mathbf{u}} - \frac{1}{2}\lambda A^T$ from the previous equation and we get

$$\lambda = 2(AA^T)^{-1}(A\Gamma \hat{\mathbf{u}} - b). \tag{4.4}$$

Finally, giving that $\mathbf{u} = \Gamma \hat{\mathbf{u}} - \frac{1}{2}\lambda A^T$ we conclude that

$$\mathbf{u} = \Gamma \hat{\mathbf{u}} - (AA^T)^{-1}(A\Gamma \hat{\mathbf{u}} - b)A^T \tag{4.5}$$

that is linear with respect to the free parameter Γ. We define $(AA^T)^{-1} = \gamma$ and note that $\gamma \in \mathbb{R}$, $\forall A \in \mathbb{R}^{1\times N}$ and $AA^T \neq 0$ if $A \neq \mathbf{0}$. Equation (4.5) is a two equations system with one unknown parameter, k_γ; solving for $\mathbf{u}_x$, we obtain:

$$k_\gamma^{\text{est}} = \frac{\hat{u}_x - u_x - \gamma A_{11}(A_{11}\hat{u}_x + A_{12}\hat{u}_y - b)}{\hat{u}_2 + \gamma A_{11}(A_{12}\hat{u}_x - A_{11}\hat{u}_y)} \tag{4.6}$$

This concludes that, for a two agents system, if we know the objective controller $\hat{\mathbf{u}}$ and the optimal safe controller $\mathbf{u}^*$ for every agent, it is possible to obtain k_γ of the agents once they enter a quasi-deadlock scenario, i.e., $\Gamma \neq 1$, since if $\Gamma = 1 \rightarrow k_\gamma = 0 \rightarrow \|\mathbf{u} - \Gamma\hat{\mathbf{u}}\|^2 = \|\mathbf{u} - \hat{\mathbf{u}}\|^2$. □

Can we conclude that agent j's direction bias is always computable by agent i in the two agents case? In our analysis we did not take into consideration an important constraint of the QP problem in (3.44): the acceleration limits $\|\mathbf{u}_j\|_\infty \leq \alpha_j$. These limits introduce a saturation in the system in the form of inequality constraints, limiting the admissible control space $\mathscr{P}_j \cap \mathscr{Q}_i$, where $\mathscr{Q}_i = \{\mathbf{u}_i \in \mathbb{R}^2 \mid \|\mathbf{u}_i\|_\infty \leq \alpha_i\}$. This limits the $k_{\gamma j}$ estimation tool we developed.

Proposition 2. Consider the QP problem in (3.44), with the quasi-deadlock resolution defined in Algorithm 1. If the nominal controller $\hat{\mathbf{u}}_j$, and the solution to the QP problem $\mathbf{u}_j^*$ are known for all $j \in \mathscr{N}_i$, then agent i can compute a lower bound on $k_{\gamma j}$, where the solution to (4.6) is such that

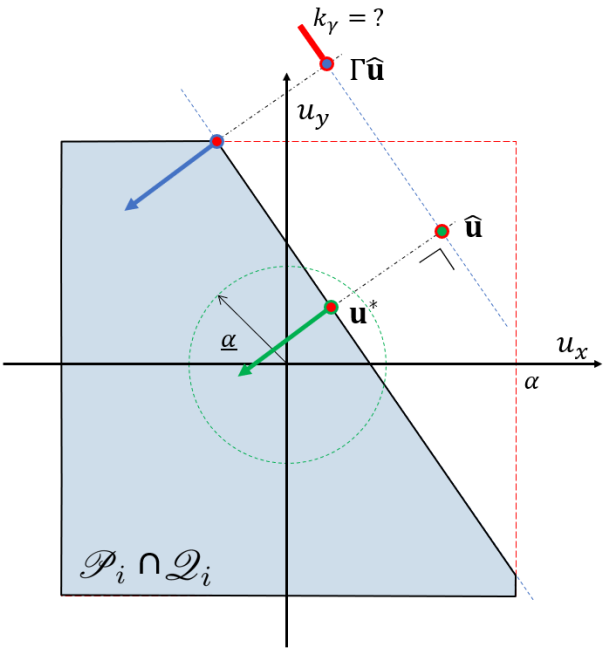


Figure 4.4: Graphical representation of $\mathscr{P}_i$ with quasi-deadlock resolution. k_γ can be computed precisely as long as the optimal solution to $\Gamma\hat{\mathbf{u}}$ is $\in$ edge($\mathscr{P}_i \cap \mathscr{Q}_i$).

$\left\| k_{\gamma j}^{\text{est}} \right\| \leq \|k_{\gamma j}\|$.

Proof: As described in Figure 4.4, if the projection $\Gamma\hat{\mathbf{u}}_j$ falls along the half-line on the right of $\Gamma\hat{\mathbf{u}}_j$, the new optimal solution $\mathbf{u}_j^*$ will fall on the vertex of $\mathscr{P}_j \cap \mathscr{Q}_i$, regardless of the actual $k_{\gamma j}$. However, the sign of $k_{\gamma j}$ will still be computed correctly and the resulting $|k_{\gamma j}|$, although wrong, will provide a lower bound on the actual value of $k_{\gamma j}$. □

It is worth noting that, since we narrowed our case of study to parking lots, it is unlikely that users will drive close to the acceleration limits.

In this Section we proposed a simple mathematical model to encode rule–based quasi-deadlock resolution, thanks to the direction bias k_γ: left vs right handed, $sign\{k_\gamma\}$, smoother vs rougher, $abs\{k_\gamma\}$. We then provided a way of estimating the other agents' driving parameters: this could allow agents to update their own parameters based on other users in the network, therefore avoiding situations as the one described in Figure 4.3b–c.

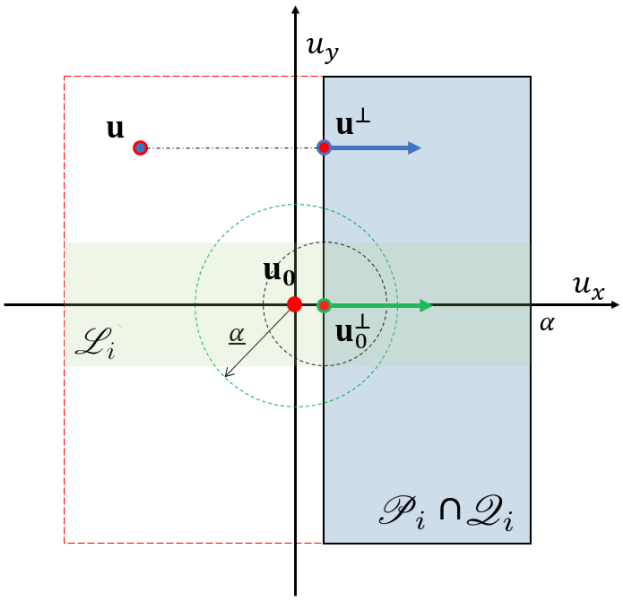


Figure 4.5: Graphical representation of $\mathscr{P}_i$ of HV as seen from AV in a simple two agents network. in this example, $\mathbf{u} \notin \mathscr{L}_i$, therefore the control $\mathbf{u}$ is considered to be not goal compatible, revealing a left-hand biased for HV.

4.2 Human–robot interaction

Let us now consider the case of a two agents network, $N = 2$, where the first agent is autonomous and implements Decentralized SBC with quasi-deadlock resolution as described in Section 4.1, and the second agent is remotely controlled by a human driver. The human driver controls the robot agent via a joystick or keyboard and is able to set the acceleration input at any instant; the human is asked to drive the robot *as straight as possible* to a specific goal point in the xy–plane, resulting in a head-on scenario as the one described in Figure 1.1b. When the human agent considers it necessary, e.g., a collision is about to happen, she or he can steer the agent to the left or to the right, according to personal preference, even if not in compliance with road rules.

Can the autonomous agent use the tools previously developed in Section 4.1 for the fully autonomous case, to deduce the human driver left- or right-hand driving bias and, potentially, updating its control law accordingly?

It is necessary to highlight that a human operator will not necessary drive enforcing the notion of SBC as discussed in the previous sections of this work. Safety of the systems Σ_S and Σ_T can be guaranteed only when all agents respect the conditions of the problem in (3.44). In the experiments

presented in the next section, we will assume that the human drives without the intent of harming the system, avoiding collision by steering (changing direction) when she or he considers it to be necessary. More even, though we have no reason to assume that human drivers employ SBC, the autonomous vehicle will act as if the human was actually using SBC. As will be seen in the results, this assumption proves to be a good insight in a human driver's behavior when related to hers or his driving direction bias.

Assume that the AV has perfect knowledge of the HV input controller and of the HV control goal, e.g., a position in the xy-plane. Let $\mathbf{u}_i$ be the control input for HV; HV implements a modified version of SBC, i.e., it computes the inequality constraints for the problem in (3.44) as if in the autonomous case, without however applying the computed optimal control, since it is directly imposed by the outside user. The AV can, as seen in Section 4.1, observe the behavior of HV and has all the necessary tools to mimic its QP problem, as seen in Section 4.1 for the fully autonomous case.

Let $\mathbf{u}_{0i}(t)$ be the control input expected from agent i at any instant t and, dropping the notion of time for ease of notation, let $\mathbf{u}_{0i}^{\perp}$ and $\mathbf{u}_{i}^{\perp}$, be the projection of $\mathbf{u}_{0i}$ and $\mathbf{u}_i$, respectively, on the equality constraint $A_{ij}\,\mathbf{u}_i^* = b_i$, where $A_{ij} \in \mathbb{R}^{1\times 2}$ and $b \in R$. Let $B_\epsilon(\mathbf{u}_{0i}^{\perp})$ be the ball of radius ϵ centered at $\mathbf{u}_{0i}^{\perp}$ and let $\mathscr{L}_i = \{\mathbf{u}_i \mid \mathbf{u}_i^{\perp} \in B_\epsilon(\mathbf{u}_{0i}^{\perp})\}$.

Definition 4.2.1. A user's control input $\mathbf{u}_i$ is said to be *goal compatible* with the control goal $\mathbf{u}_{0i}$ if $\mathbf{u}_i \in \mathscr{L}_i$.

If the autonomous agent finds that HV is driving with goal compatible inputs, no further actions will be taken, as AV considers that the human driver is trying to achieve her or his original goal. On the other hand, if $\mathbf{u}_i \notin \mathscr{L}_i$, AV considers HV to be perturbing its control away from the original goal, e.g., to avoid a collision.

With this new tools we can construct the human–robot interaction problem as the robot–robot problem of Section 4.1, where $\mathbf{u}_0^{\perp} = \mathbf{u}^*$ and $\mathbf{u}^{\perp} = \mathbf{u}_Q^*$.

5 The Multiagent head-on deconfliction

In Chapter 3 we recalled the necessary tools to understand and develop safety via Control Barrier Functions; in Definitions 3.6.1 and 4.0.1 we described the conditions for deadlock and quasi-deadlock, respectively: a deadlock occurs when an agents has to stop—due to a conflicting motion between other agents; a quasi-deadlock tries to anticipate a deadlock, allowing the deconfliction tool to take action sooner, therefore avoiding the need for an agent to come to a complete stop. In Chapter 3.6 we identified three different types of deadlock, based on the nature of the QP problem in (3.44), and we provided a deconfliction tool specifically for a Type 2 deadlock.

In Chapter 4 we developed and evaluated a tool to deconflict a pair of agents when they are entering a quasi-deadlock; such tool proved to behave in a similar fashion to the goal motion described in the problem formulation in Section 1.1. The two agent case is particularly simple to analyze, since only one of the three aforementioned quasi-deadlock types can occur. It is clear, however, how this solution is limited and improvements need be made for truly multiagent systems, with any number of agents.

In this Chapter we are going to analyze how the deconflicting tools so far developed apply to a truly multiagent network, and its limitation. Improving on these results, we are then going to develop a more exhaustive tool that allows an efficient deconfliction motion for the more general case.

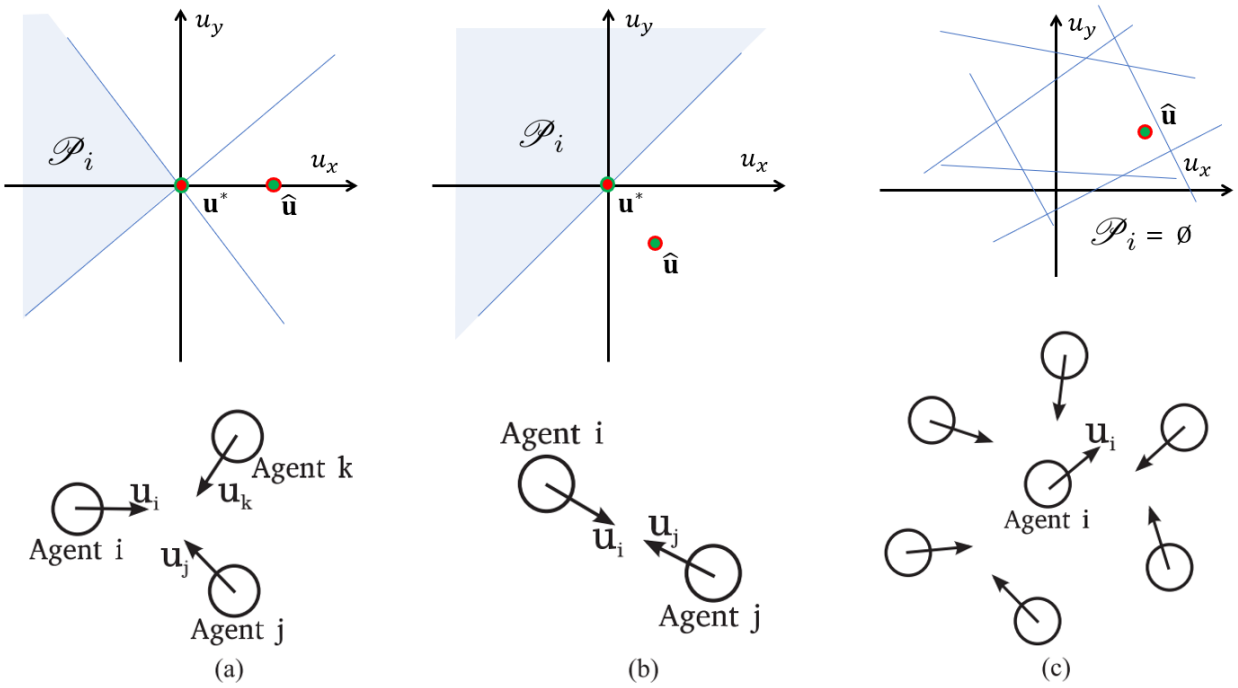


Figure 5.1: Three types of Deadlocks for robot agent i in a multirobot system with the respective admissible control space $\mathscr{P}_i$. (a) Type 1 deadlock. (b) Type 2 deadlock. (c) Type 3 deadlock.

5.1 Limits of applying Type 2 quasi-deadlock resolution to a Type 1 quasi-deadlock

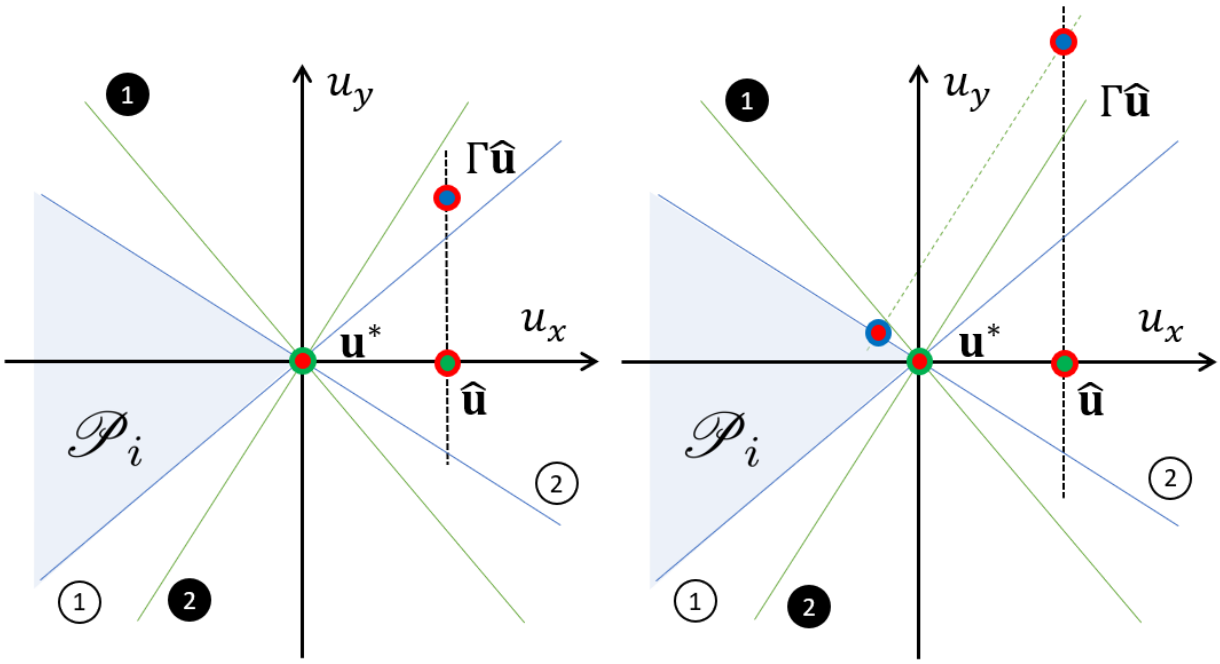
In Figure 5.1 we recall the three different deadlock and quasi deadlock types, and how they relate to the QP problem of (3.44). In particular, in the Type 1 deadlock the optimal solution belongs to a vertex of the admissible control space $\mathscr{P}_i$; in the Type 2 deadlock the optimal solution belongs to an edge of $\mathscr{P}_i$; in the Type 3 deadlock, $\mathscr{P}_i$ is empty, therefore no optimal solution exists.

In Chapter 4 we focused only on Type 2 deadlocks; as a conflict resolution tool (see Algorithm 1) we introduced the direction bias k_γ, allowing us to perturb the original control input. In Chapter 4.1.3 we provided a tool to estimate other agents' direction bias (see Proposition 1). This was possible since (4.6) always exists and it is unique when both the original and the perturbed optimal solutions belong to the edge of $\mathscr{P}_i$; in Proposition 2 we introduced a limit (in the form of saturation) on the acceleration inputs, therefore providing a bound on the estimation of k_γ.

There is no formal limitation in applying Algorithm 1 to any type of deadlock; however, we will show, by mean of an example, how this will be practically inconvenient.

Consider a Type 1 deadlock, as in Figure 5.1a; what will happen if we use the same resolution

tool as for a Type 2 deadlock? This is explained graphically in Figure 5.2. In this particular example, agent i has two neighbours, therefore two constraints on $\mathscr{P}_i$; the two constraints are shown and numbered in Figure 5.2 with a white circle. With a black circle we indicate the normal to the respective constraint. In Figure 5.2a the direction bias k_γ is small enough that the perturbed



(a) The perturbed $\Gamma\hat{\mathbf{u}}$ is such that the original optimal solution and the new optimal solution match.

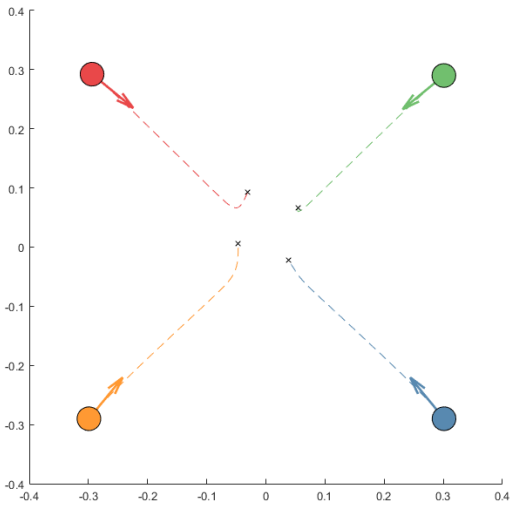
(b) The perturbed $\Gamma\hat{\mathbf{u}}$ is such that the new optimal solution differs from the original.

Figure 5.2: Type 1 deadlock; the nominal controller $\hat{\mathbf{u}}$ is perturbed with different absolute values of direction bias k_γ

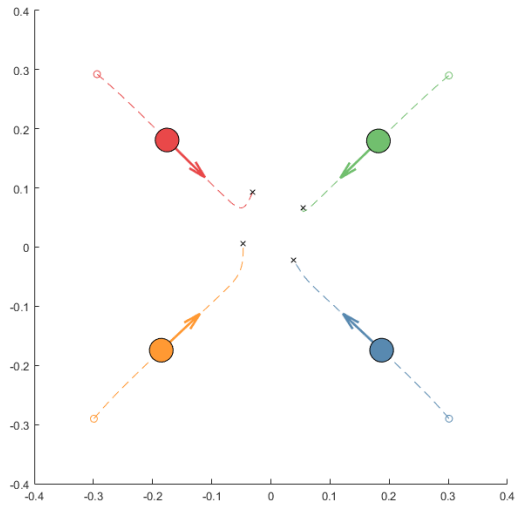
controller $\Gamma\hat{\mathbf{u}}$ produces the same optimal solution as the non perturbed controller $\hat{\mathbf{u}}$. This is a limitation since the resolution tool will provide no practical change to the control input, therefore providing no conclusive way to deconflict a deadlock. Differently, in Figure 5.2b k_γ is big enough to push the projection $\Gamma\hat{\mathbf{u}}$ such that the new optimal solution is different from the original optimal solution, therefore providing—in this particular case—a valid deconflicting tool.

This example shows that, for a Type 1 deadlock, this resolution tool is not always effective, like seen previously for the Type 2 deadlock. For this reason, we will now build a deconfliction tool for a Type 1 deadlock; we will see that the two methods will not be in conflict but will, in fact, be complementary towards the resolution of the more general case. In Figure 5.3 and Figure 5.4 we show simulation results of the problem in Figure 5.2 with a four agents network; in particular, as expected, in Figure 5.3, the agents are not able to solve the impasse since k_γ is too small, whereas,

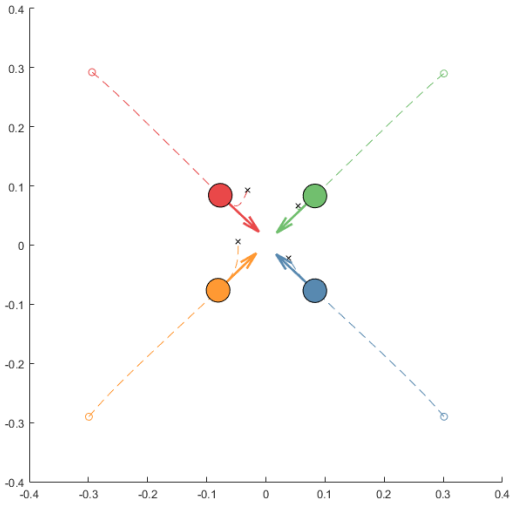
in Figure 5.4 they behave similarly to the desired goal, given a value of k_γ big enough, that projects the new optimal solution away from the original optimal solution.



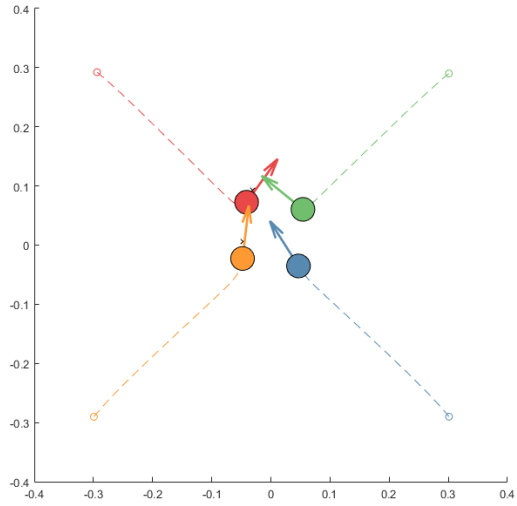
(a) t = 0 s



(b) t = 2 s

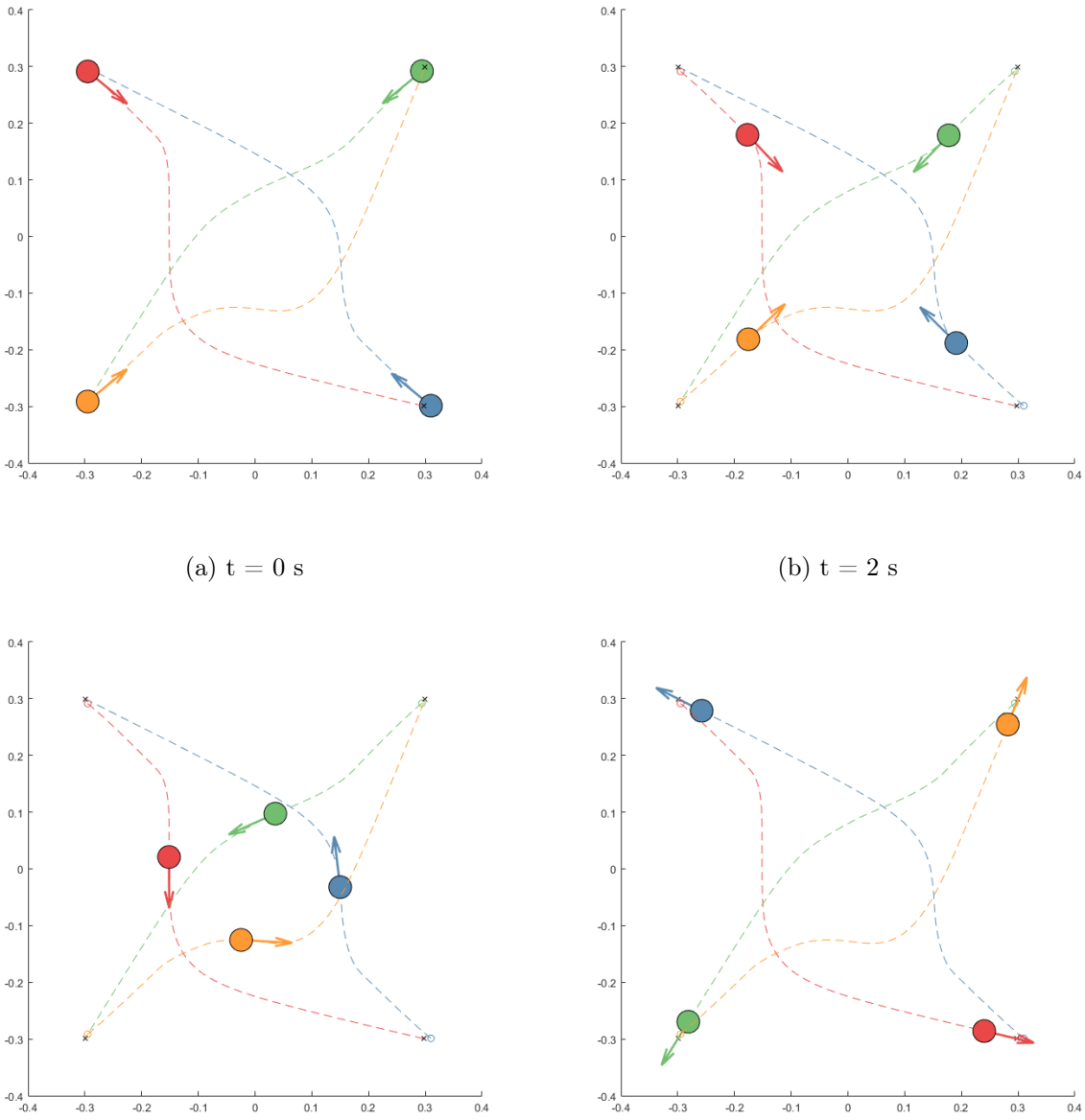


(c) t = 2.5 s



(d) t = 3.5 s

Figure 5.3: Head-on scenario with four agents; given the geometry of the system, all agents enter a Type 1 deadlock. $k_\gamma = -1$, is not enough to push the new optimal solution away from the original, therefore the agents are not able to deconflict their motion and are unable to achieve the goal of swapping positions.



(a) t = 0 s

(b) t = 2 s

(c) t = 2.5 s

(d) t = 3.5 s

Figure 5.4: Head-on scenario with four agents; given the geometry of the system, all agents enter a Type 1 deadlock. $k_\gamma = -10$, is enough to push the new optimal solution away from the original, allowing agents to deconflict their motion and achieve the goal of swapping positions.

5.2 Type 1 deadlock resolution

We noticed how perturbing the cost function of the QP problem in (3.44) with a projection of the nominal controller $\Gamma\hat{\mathbf{u}}$ works best for Type 2 deadlocks, but provides mixed results for a Type 1

deadlock. We here expand the deconflicting tools so far developed, introducing a different resolution for this different kind of deadlock.

The proposed strategy for resolving a Type 1 deadlock or quasi-deadlock is to perturb the admissible control space, instead of the cost function, of the QP problem. We notice that if we perturb the admissible control space $\mathscr{P}_i$ such stat the new admissible control space $\mathscr{P}_i'$ is a subset of $\mathscr{P}_i$, that is

$$\mathscr{P}_i' \subset \mathscr{P}_i, \tag{5.1}$$

then any solution generated by the QP problem in (3.44) with $\mathscr{P}_i'$ belongs to the space of solutions of the original QP problem with $\mathscr{P}_i$. Intuitively, to satisfy (5.1), we are looking to shrink the original admissible control space; this can be done by modifying consistently—that is, such that the new control space is included in the former—one of the constraints $A_{ij}\,\mathbf{u}_i \leq b_{ij}$.

Definition 5.2.1 (Perturbed admissible control space)**.** Given a QP problem as in (3.44), and given an index $k \in \mathbb{R}$, then the perturbed admissible control space $\mathscr{P}_i'$ is defined as

$$\mathscr{P}_i' = \{\mathbf{u}_i \in \mathbb{R}^2 \mid \bar{A}_{ij}\mathbf{u}_i \leq \bar{b}_{ij} \ \wedge \ \bar{A}_{ik}\mathbf{u}_i \leq \bar{b}_{ik} - k_\pi \quad \forall\, i \neq j,\ j \neq k\},$$

where $k_\pi \in \mathbb{R}^+$.

We notice that we did not give any indication on how to choose which constrain k to relax; a possible strategy will be discussed shortly.

Proposition 3. Given a perturbed admissible control space as defined in 5.2.1, it is always true that $\mathscr{P}_i' \subset \mathscr{P}_i$.

Proof: By absurd, assume that there exists a $\mathbf{u}^o \in \mathscr{P}_i'$ and such that $\mathbf{u}^o \notin \mathscr{P}_i$. By definition $\mathbf{u}^o$ is such that $\bar{A}_{ik}\mathbf{u}^o{}_i \leq \bar{b}_{ik} - k_\pi$, but, since $k_\pi > 0$, also by definition, then $\bar{A}_{ik}\mathbf{u}^o{}_i \leq \bar{b}_{ik}$; since all the other constraints remain unchanged, we can conclude that $u^o \in \mathscr{P}_i$. □

This is shown graphically in Figure 5.5, where one of the five constraints (in particular, $k = 2$) is compressed, therefore giving the perturbed admissible control space.

5.2.1 How to choose the index k

The goal is to choose the index k such that the perturbed admissible control space for the QP problem in (3.44) produces a new optimal solution that will deconflict the motion of an agent.

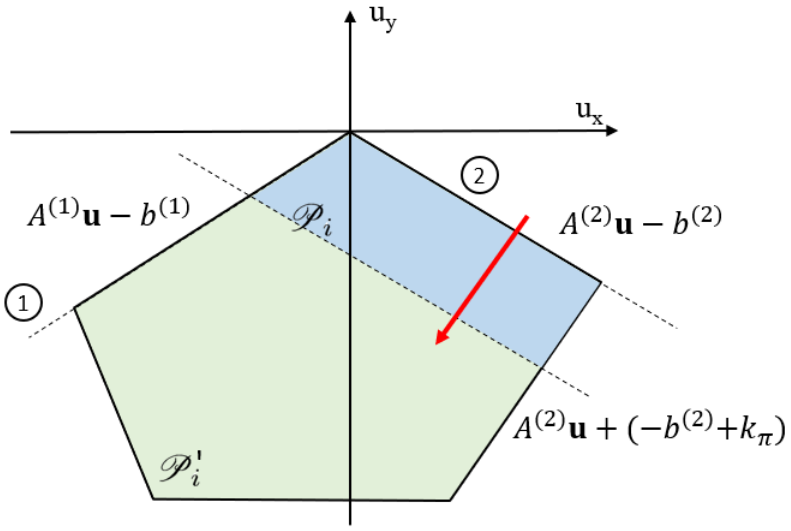


Figure 5.5: Admissible control space $\mathscr{P}_i$ and perturbed admissible control space $\mathscr{P}_i'$. The compressing index is $k = 2$.

Since we are focusing on Type 1 quasi-deadlocks, we know by definition that the original optimal solution (that is going to be $\mathbf{u}^* = 0$ in the case of pure deadlock or $\|\mathbf{u}^*\| < \underline{\alpha}$ in case of quasi-deadlock) will belong to at least two inequality constraints.

It is intuitive to see that the most direct way of finding a new optimal solution is to modify one of such constraints, that is, k needs to be one of the inequality constraints that solve the original optimal solution $\mathbf{u}^*$ with an equality $\bar{A}_{ij}\,\mathbf{u}_i^* = \bar{b}_{ij}$. Moreover, we would like to link this deconfliction tool to the one of Chapter 4, therefore we would like to be able to encode once again a left-handed or right-handed driving bias in the way this compression takes place.

Consider Figure 5.6, that is the same scenario as Figure 5.5 but where now the QP problem is complete, as we have a nominal control input $\hat{\mathbf{u}}$. If we compress the constraint $k = 2$, as in this example, the new optimal solution will fall to the left of the original optimal solution; similarly, if we compress the constraint $k = 1$, the new optimal solution will fall on the right of the former solution. We will assign to every constraint a circled L (R) if compressing such constraint will push the new optimal solution to the left (right) of the original optimal solution.

The question we want to address now is: given a Type 1 quasi-deadlock, is it always possible to find a constraint that, once compressed, will steer the agent to the left and one that will steer the agent to the right? Before addressing this question, we need to find a formal way of determining weather compressing a particular constraint will steer the agent to one side or to the other, and how to discriminate between the two.

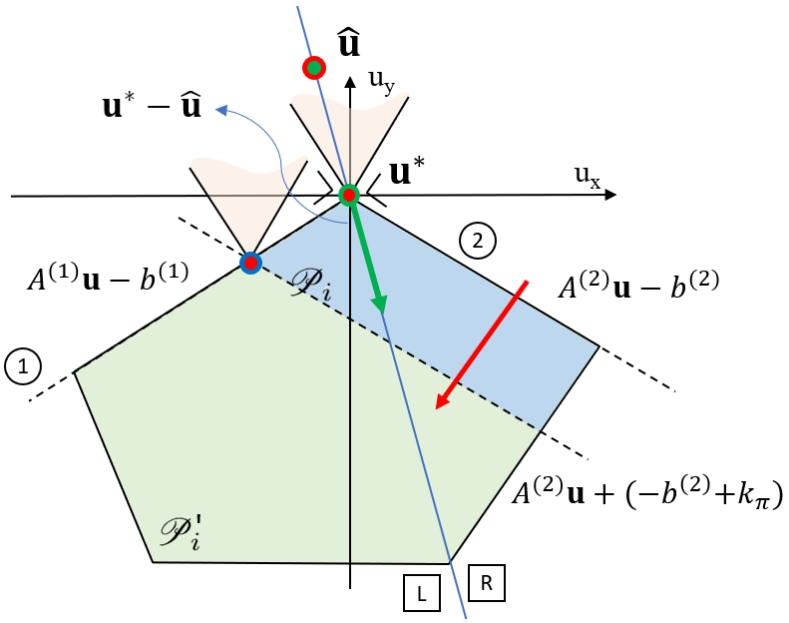


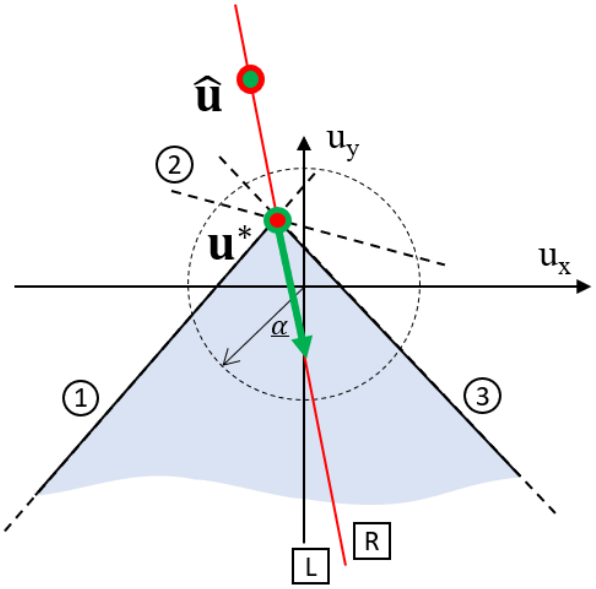
Figure 5.6: Admissible control space $\mathscr{P}_i$, perturbed admissible control space $\mathscr{P}_i'$, nominal control $\hat{\mathbf{u}}_i$ and optimal control $\mathbf{u}^*$. The compressing index is $k = 2$.

We notice that, by left or right, we mean relatively to the agent's direction of motion, in body frame; geometrically, since we are dealing with a 2 dimensional control space, we can divide the plane in two half-planes along the direction of the cost function vector (changed in sign) $\hat{\mathbf{u}} - \mathbf{u}^*$. Both the nominal control $\hat{\mathbf{u}}$ and the original optimal solution $\mathbf{u}^*$ lye on such line, therefore have no direction bias. When the new optimal solution falls on the left (right) half plane, due to the compression of a constraint, such constraint will be categorized as a left-biased (right-biased) constraint.

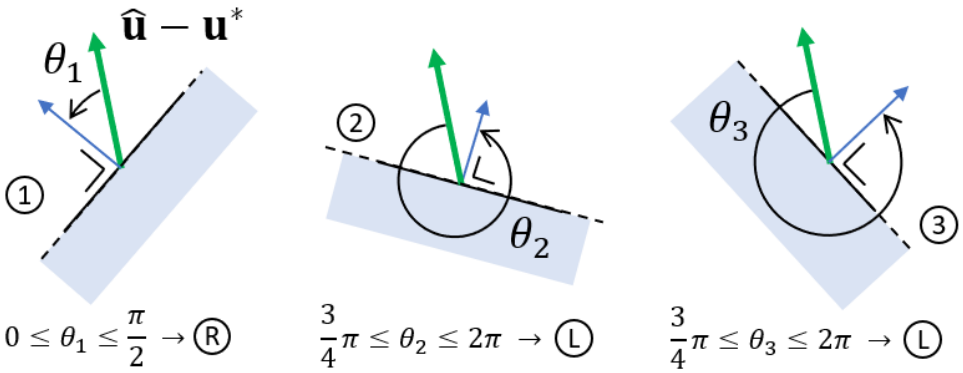
Definition 5.2.2 (Left-biased (right-biased) constraint)**.** A constraint, indexed with k, to the QP problem in (3.44) is said to be left-biased (right-biased) if the solution to the problem with the perturbed admissible control space (5.2.1), $\mathscr{P}_i'$, falls to the left (right) of the optimal solution with the complete admissible control space, $\mathscr{P}_i$. Geometrically, a constraint k is left-biased (right-biased) if the angle θ_k described by the vector $\hat{\mathbf{u}} - \mathbf{u}^*$ and the normal to the constraint half plane $\bar{A}_{ik}$ is $3/4\pi \leq \theta_k \leq 2\pi$ $(0 \leq \theta_k \leq \pi/2)$.

An example of a Type 1 quasi-deadlock, where the optimal solution belongs to three different constraints, is shown in Figure 5.7. The three constraints are divided in left and right biased, based on the angle between the cost function and the optimal solution.

Proposition 4. Given a Type 1 quasi-deadlock, there is always a left-biased and a right-biased constraint associated to the optimal solution.



(a)



(b)

Figure 5.7: Decomposition of a Type 1 quasi-deadlock with three equality constraints.

We finally have all tools to handle Type 1 deadlocks and quasi-deadlocks, with a strategy whose outcome matches that of the Type 2 quasi-deadlock resolution. This is resumed in Algorithm 2.

Algorithm 2 Type 1 and 2 quasi-deadlock resolution

input $\mathbf{u}_i, \hat{\mathbf{u}}_i, \mathbf{v}_i$
is_deadlock $\leftarrow$ **false**
$\gamma \leftarrow Decentralized\_LP$

if $\|\mathbf{u}_i\| \leq \underline{\alpha}$ & $\|\mathbf{v}_i\| \leq \underline{\beta}$ & $\|\hat{\mathbf{u}}_i - \mathbf{u}_i\| > \underline{\epsilon}$ & $\gamma \leq 0$ **then**
    is_deadlock $\leftarrow$ **true**

if is_deadlock **then**
    if Type_1 **then**
        $k \leftarrow getContraintIndex(\Gamma_i)$
        $\bar{b_k} \leftarrow \bar{b_k} - k_i$
    if Type_2 **then**
        $\hat{\mathbf{u}}_i \leftarrow \Gamma_i \hat{\mathbf{u}}_i$

$\mathbf{u}_i^* \leftarrow Decentralized\_QP(\hat{\mathbf{u}}_i)$
return $\mathbf{u}_i$

6 Hardware experiments

In this Chapter we are going to walk through the results presented in the previous sections of this work, implementing the solutions on hardware and evaluating their performance. All experiments are performed on the Robotarium at Georgia Tech (Pickem et al., 2016; Pickem et al., 2017), an open access hardware multirobot simulation environment.

6.1 The Robotarium

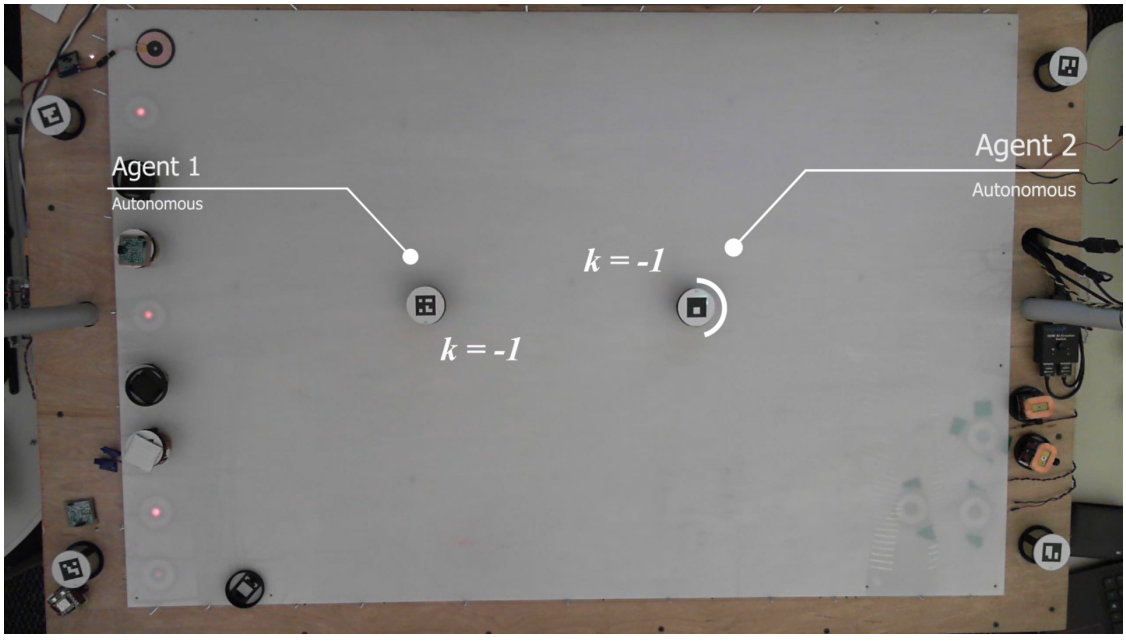
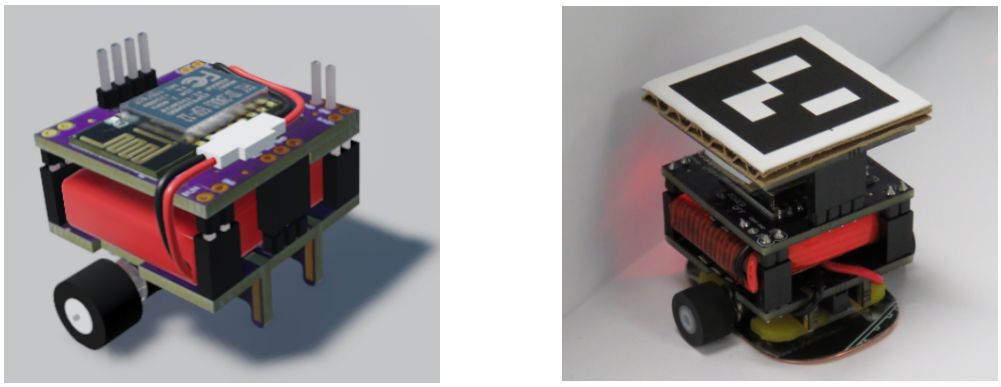


Figure 6.1: Frame from a video recording of an experiment on the Robotarium.

The Robotarium is a multirobot simulation testbed, developed by the Institute for Robotics and Intelligent Machines (formally Georgia Robotics and InTelligent Systems Laboratory) at Georgia Tech. During the stages of development, different iterations of the Robotarium were created; for

this work, a smaller version was used (around $2m^2$); an improved version of the Robotarium is now available, measuring around 4 by 4.5 meters and is publicly accessible on-line.

The Robotarium is an extremely powerful tool that allows for fast and effortless implementation and exhaustive testing of complex distributed algorithms on networks up to dozens of robots. In Figure 6.2 a model and a picture of the custom developed GRITSbot is showed, measuring about 4 centimeters per side.



(a) 3D rendering of a GRITSBot. (b) Picture of GRITSBot.

Figure 6.2: 3D model and current hardware implementation of the GRITSBot.

The Robotarium offers a high level programming interface trough MATLAB and Python, and takes care of the lower level aspects, like providing each agent with the control commands, agent tracking and security countermeasures (Figure 6.3). For the purpose of this experiments the default barrier certificates that the Robotarium framework imposes are disabled.

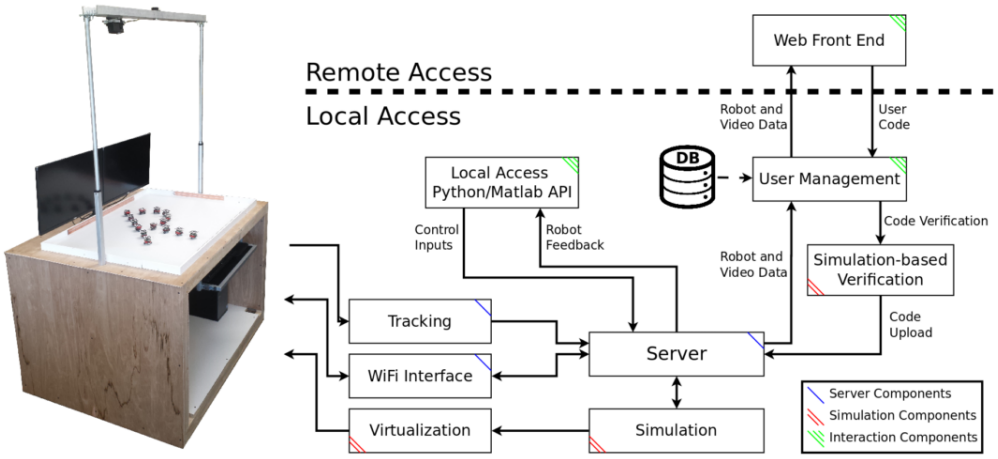


Figure 6.3: Robotarium system architecture overview.

6.2 Two agents networks

6.2.1 Robot–Robot Head-On

Two AVs are controlled with the goal of swapping positions on the plane; three series of experiments are performed, in close relation to the software simulations of Figure 4.2. The results, shown in

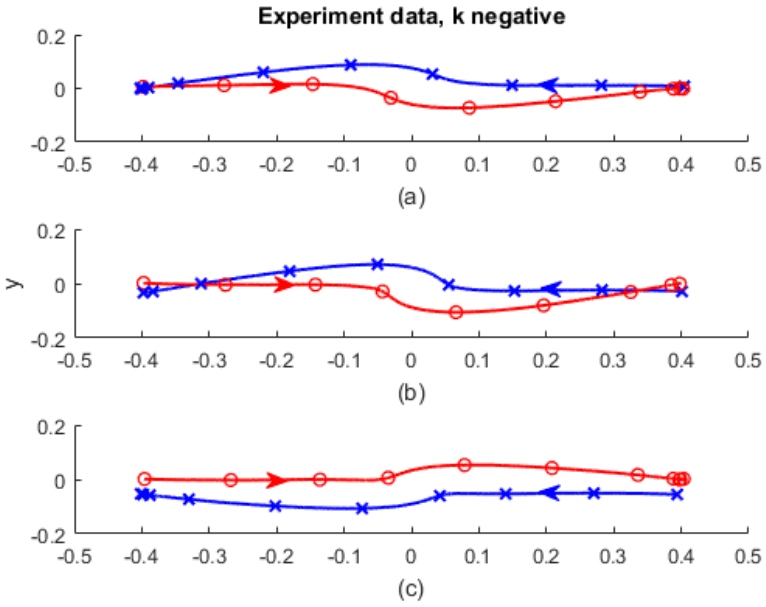
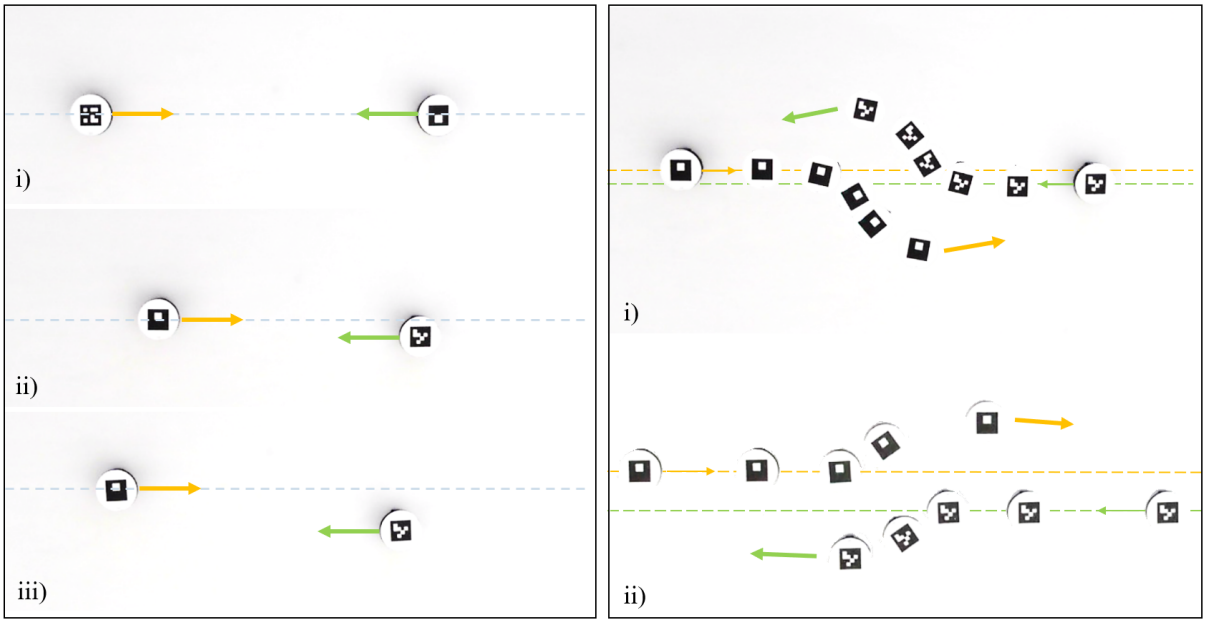


Figure 6.4: Experimental quasi-deadlock resolution to different head-on scenarios; each marker indicates 61 iterations ($\sim$ 2 seconds).

Figure 6.4, match the simulations'; another result is that of Figure 6.4b, proposed as a consecutive series of snapshots in Figure 6.5. Here, the two agents are slightly misaligned to their left on the vertical axis; however, since the misalignment is not significant, in accordance to the rules defined in Figure 1.1, both agents steer to the right. In the experiment of Figure 6.4c, instead, the misalignment is significant, as in Figure 1.1d, resulting in both agents holding their side of the road. As a performance comparison, in Figure 6.6 the same experiments are performed with the original formulation of Decentralized SBC and with Decentralized SBC with deadlock detection and resolution as defined in (Wang et al., 2017). The absence of any kind of prearranged deconfliction in the experiment of Figure 6.6a, translates in the two agents getting stuck for about four seconds, before naturally resolving the conflict in an unpredictable fashion—in perfect conditions the two agents would get stuck indefinitely. Comparing Figure 6.4a and Figure 6.6b, we notice that the



(a) Different alignment options for the fully autonomous head on conflict motion: (i) perfect alignment, (ii) slight misalignment; (iii) significant misalignment.

(b) Consecutive snapshots of quasi-deadlock deconfliction: (i) slightly misaligned case, (ii) significantly misaligned case.

Figure 6.5: Experimental result of different head on scenarios, set in a right-handed driving environment, when both agents are autonomous and implement SBC with quasi-deadlock resolution.

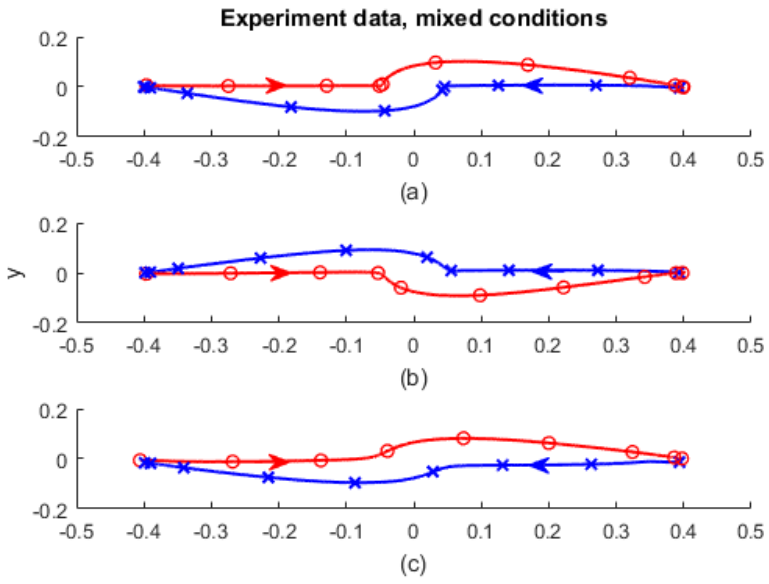
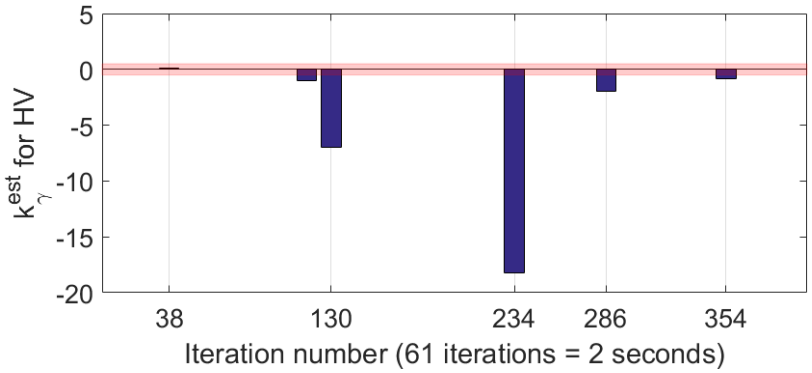


Figure 6.6: Experimental head-on resolution without quasi-deadlock: a) no deadlock and no quasi-deadlock resolution; b) deadlock resolution ($k_\gamma = -1$); c) slight misalignment with quasi-deadlock ($k_\gamma = -1$).

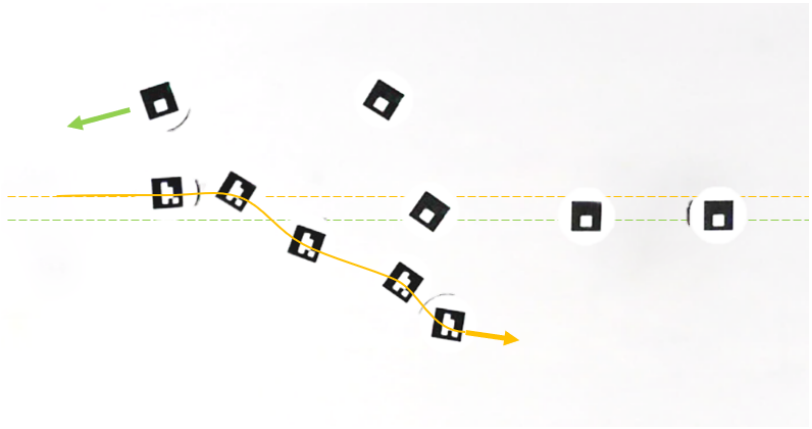
quasi-deadlock detection perturbs the controller earlier, resulting in a faster decoflicting motion. Finally, in Figure 6.6c, since the alignment is not perfect, the two agents never enter a deadlock and the deconflicting motion is only given by the Decentralized SBC, resulting in the agents straying from one other in order to keep a safety distance, but with no regards for road rules (compare that to Figure 6.4b). A video of the experiments is available online at [todo:ref].

6.2.2 Human–Robot Head On

Experimental results for the scenario described in Section 4.2 are shown here in Figure 6.7. In particular, the AV agent is programmed with $k_{\gamma A} > 0$ (left-hand driving) and the HV is allowed, once she or he considers it necessary, to steer to the left or to the right. The autonomous agent detects the human driving direction bias and updates its internal value of $k_{\gamma H}$ accordingly.



(a) k_γ of HV, estimated by AV. The shaded area higligts the actions that are considered *goal compatible* and, therefore, ignored.



(b) Snapshot evolution of head-on interaction between HV (left) and AV (right).

Figure 6.7: Head on scenario with mixed human-robot interaction. The AV starts with a left driving bias, but updates it based on HV's behavior.

In the experiment of Figure 6.7, the autonomous agents knows that the HV needs to reach a point in the plane along its current trajectory of motion; any acceleration input along that direction will not influence the computation of $k_{\gamma H}$. At about 1 second from the start (iteration 38) a *goal compatible* acceleration input is given: the user commands the agent to accelerate forward, towards the goal. Later in the experiment, just after the 4 seconds mark (iteration 125) the user starts steering the robot agent significantly, resulting in a well defined sign of $k_{\gamma H}$, suggesting that the HV is, indeed, steering to the negative values of k_{γ}, i.e. to the right of the goal.

7 Conclusions

A tool to solve conflicting motion paths for networks of multiagent robots was presented. Focusing on the interaction of two robot vehicles heading towards a collision, we provided a tool to deconflict the agents' motion, while ensuring safety with the notion of traffic rules. Based on the agents' relative position and velocity, the agents will act according to road rules, or decide to break them if particular conditions are present. We introduced the notion of driving direction bias, as the direction (left or right) that will preferably be used by the autonomous agent to deconflict its motion, together with a tool to estimate other agents' direction bias.

Finally, we proposed a strategy to adapt the model, presented for the autonomous vehicles, to a mixed human and robot interaction, where the robot vehicle morphs its driving direction bias based on the observed human behavior.

The proposed algorithms were tested on hardware on the Robotarium.

Even though there is no reason to believe that human drivers employ barrier certificates, the construction in this work shows that this assumption still allows for an effective deconfliction strategy between human and autonomous drivers. Moreover, we believe that these results can be further generalized, removing the need to share the agents' state on the network and exploiting tools already present in the literature to estimate the local agents' states from sensor data (Prajna et al., 2007). Indeed, the performed experiments in the Human-Robot interaction scenario are limited to just one human subject. Further work is required to present a statistically significant set of experiments that should aim at benchmarking the behaviour of this resolution tool with different human subjects and, therefore, driving behaviours.

A Models of mobile robots in the plane

The focus of the present work is on multi agent systems consisting of two wheeled robots moving on en even plane surface, $z(t) = 0, \forall t$, as described in figure A.2; this results, however, can be extend to a three dimensional scenario, such as UAVs (Unmanned Aerial Vehicles), or more complex models, at the cost of finding a valid ZCBF for the case of interest.

In the following section I am going to describe three different ways of modeling such robots and how it is possible to transform from a model to the other.

A.1 Single Integrator

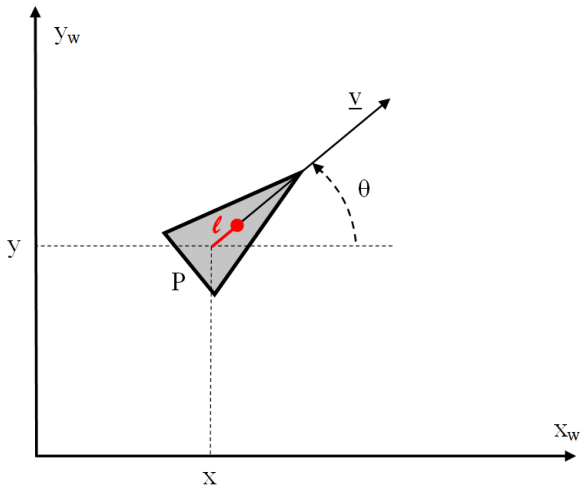


Figure A.1: Global reference frame and robot representation, single integrator

Probably the simplest model for a moving robot is that of a wheeled rover of unit mass moving along a straight infinite road running through a 2D plane (Francis and Maggiore, 2016). Let $\mathbf{x} = (x, y)$ be the position of the agent (or rover) in $\mathbb{R}^2$, see figure A.1. The agent has an on-board

motor that drives the wheel without slipping, therefore controlling its speed (and also measuring it by means of en encoder); neglecting viscous friction, we can simply apply Newton's second law by saying:

$$\ddot{\mathbf{x}} = f \tag{A.1}$$

or, equivalently, by defining the velocity v as $\dot{\mathbf{x}} = v$

$$\dot{v} = f. \tag{A.2}$$

Since we can impose the velocity directly, the model is ready; defining $u \in \mathbb{R}^2$ as our velocity command input, being $\mathbf{x} \in \mathbb{R}^2$ our state vector the models is

$$\dot{\mathbf{x}} = u. \tag{A.3}$$

A.2 Unicycle

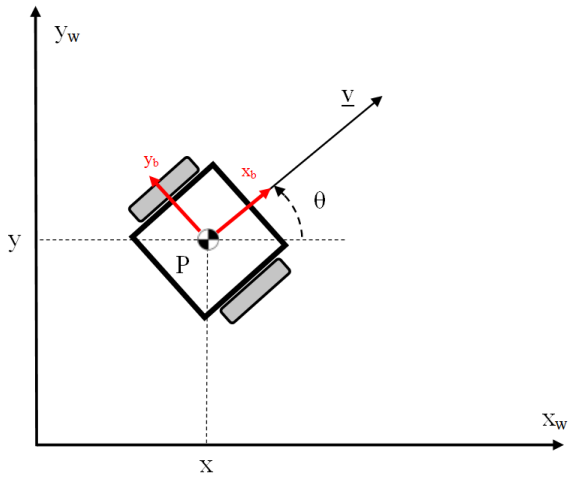


Figure A.2: Global and body reference frame, two wheeled robot

Throughout this analysis we model the robot as a more complex rigid body on two wheels, operating on a horizontal plane. The total dimensionality of this robot chassis on the plane is three, two for position in the plane $P = (x, y)$ and one for orientation θ along the vertical axis, which is orthogonal to the plane; the pose of the robot, our system state, can be defined as $\mathbf{x} = [x, y, \theta]^T$. The constraint the wheel imposes on the plane translational movement of the vehicle is

$$\frac{\dot{y}}{\dot{x}} = \tan\theta = \frac{\sin\theta}{\cos\theta} \tag{A.4}$$

that can be expressed in the Pfaffian form as

$$A(\mathbf{x})\dot{\mathbf{x}} = \begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0. \tag{A.5}$$

It is clear that $A(\mathbf{x})$ has Rank one everywhere, therefore it's null space has dimension two; using the Quasi Velocities Approach we define $S(\mathbf{x})$, base of $A(\mathbf{x})$'s null space, so such that $A(\mathbf{x})S(\mathbf{x}) = 0$. A suitable choice of $S(\mathbf{x})$ is

$$S(\mathbf{x}) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{A.6}$$

Thanks to the Quasi Velocities Method we can conclude that our state evolution is $\dot{\mathbf{x}} = S(\mathbf{x})\upsilon$ that is

$$\dot{\mathbf{x}} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix} \tag{A.7}$$

The equation in A.7 has a clear physical interpretation: the first component, υ_1, corresponding to the first column of $S(\mathbf{x})$, represent the forward velocity of the vehicle, v; the second component, υ_2, is the angular velocity $\omega = \dot{\theta}$ of the vehicle about the vertical axis in (x, y) (Bicchi, 2015).

In conclusion, the unicycle dynamics are:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{A.8}$$

A.3 Single Integrator to Unicycle transformation

It is important to understand the differences between the two models and why they are both valid, although extremely different: the first one, the Single Integrator model, is a good high level approximation of any moving robot living on the plane (or 3D space), from a four wheels car (or quad-copter) to a rolling ball (or aircraft); however, it is not a good enough model to control the agents at lover levels. For instance, a two wheel robot cannot move along the y_b axes (see figure A.2) due to the wheels constraints; this could instead be done in a Single Integrator model (for an exhaustive description of this classic problem see (Bicchi, 2015)). For this reason, the problem of controlling the path of a robot is usually broken down into three layers, or levels:

- Strategic level (or High-Level Planning): where to go?
- Operational level (or Low-Level Planning): where to go?
- Tactical level (or Execution): how to go there?

To put things in prospective, consider the following example. A self-driving car has to go from Atlanta to Boston; at the strategic level is the actual goal, the need to go from point A to point B and can be designed with any kind of high level path planning algorithm, like Dijkstra's algorithm (Dijkstra, 1959); at the operational level, the work can be done using a general model, like the one in A.3; this will generate a trajectory (a desired direction and magnitude) that the robot has to follow. Finally, at the lowest level, the tactical level, we need to translate the information from the operational level to something that particular car can use, that is, a model of that particular car; in the case of a two wheel car, the one in A.8. At this point we need a way of tracking a Single Integrator model to a Unicycle model.

Recall equation A.8: here we have the dynamics, known the position, (x, y), and the heading, θ. Assume we do not need the heading angle and consider a point $P' = (\tilde{x}, \tilde{y})$ as in figure A.1, at distance l from the center of mass of the robot along the heading direction, where l is a small number. The equations for the new point are

$$\begin{cases} \tilde{x} = x + l\cos\theta \\ \tilde{y} = y + l\sin\theta \end{cases} \tag{A.9}$$

To study the system's dynamics it is sufficient do find the time derivative of $\tilde{x}$ and $\tilde{y}$ and substituting from equation A.8

$$\begin{cases} \dot{\tilde{x}} = \dot{x} - l\dot{\theta}\cos\theta = v\cos\theta - l\omega\sin\theta \\ \dot{\tilde{y}} = \dot{y} + l\dot{\theta}\sin\theta = v\sin\theta + l\omega\cos\theta \end{cases} \tag{A.10}$$

Assume now that we have before generated the trajectories for P' instead of P; we would simply have $\dot{\tilde{x}} = u_1, \quad \dot{\tilde{y}} = u_2$ that gives us the system

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v \\ l\omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{A.11}$$

where we recognize a rotational matrix around the z axis of an angle θ, $R_z(\theta) \in SO(2)$; we can rewrite the previous equation as

$$R_z(\theta) \begin{bmatrix} 1 & 0 \\ 0 & l \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{A.12}$$

and, since a rotational matrix is always invertible and is equal to $R_z(\theta)^{-1} = R_z(-\theta)$ (Siciliano et al., 2009), we get our final transformation[1]:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/l \end{bmatrix} R_z(-\theta) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{A.13}$$

This way, we can actually control perfectly the point P', therefore the smaller the distance between P and P', l, is, the better the transformation's result.

A.4 Double Integrator

Although very general, and usually accurate enough, sometimes the single integrator model could be insufficient for a Low-Level planning purpose, for example when we want to design energy aware algorithms [] or when limits on the accelerations levels are preset []. This is generally the case when designing goals require to control the variation over time of the velocity, that is - the integral of the velocity itself - the acceleration; this is why this model is known as the Double Integrator (Rao and Bernstein, 2001). The control input u is now an acceleration input and equation A.3 can be easily extend as

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} + \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} u \tag{A.14}$$

where $p \in \mathbb{R}^2$, $v \in \mathbb{R}^2$, and $u \in \mathbb{R}^2$ represent the positions, velocities, and inputs (acceleration commands) of a single agent. Generally, we can limit the velocity and acceleration of the agent by saying $\|v\|_\infty \leq \beta$ and $\|u\|_\infty \leq \alpha$ (Wang et al., 2017).

[1] A Mapping function maps every element of a given set to a unique element of another set; a Transformation is the replacement of the variables in an algebraic expression by their values in terms of another set of variables.

Bibliography

(1935). Why We Drive on the Right of the Road. *Popular Science Monthly*, 126(1):37.

(1972). Convention on the International Regulations for Preventing Collisions at Sea.

Ames, A. D., Grizzle, J. W., and Tabuada, P. (2014). Control barrier function based quadratic programs with application to adaptive cruise control. *53rd IEEE Conference on Decision and Control*, pages 6271–6278.

Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P. (2017). Control Barrier Function Based Quadratic Programs for Safety Critical Systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876.

Bicchi, A. (2015). Appunti di Robotica II.

Borrmann, U., Wang, L., Ames, A. D., and Egerstedt, M. (2015). Control Barrier Certificates for Safe Swarm Behavior. *IFAC-PapersOnLine*, 48(27):68–73.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bracha, H. S., Seitz, D. J., Otemaa, J., and Glick, S. D. (1987). Rotational movement (circling) in normal humans: sex difference and relationship to hand, foot and eye preference. *Brain Research*, 411(2):231–235.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of Robot Motion*. MIT Press.

Cockcroft, A. and Lameijer, J. (2011). *A Guide to the Collision Avoidance Rules*. 7th edition.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. 271(1):269–271.

Federal Aviation Administration (2013). Title 14 of the Code of Federal Regulations.

Fox, D., Burgard, W., and Thrun, S. (1997). The Dynamic Window Approach t o Collision Avoidance. *Robotics & Automation Magazine, ...*, 4(March):1–23.

Francis, B. A. and Maggiore, M. (2016). Flocking and Rendezvous in Distributed Robotics.

Gordon, H. W., Busdiecker, E. C., and Bracha, H. S. (1992). The Relationship Between Leftward Turning Bias and Visuospatial Ability in Humans. *International Journal of Neuroscience*, 65(1-4):29–36.

Gough, P., de Berigny Wall, C., and Bednarz, T. (2014). Affective and Effective Visualisation: Communicating Science to Non-Expert Users. *IEEE Pacific Visualization Symposium Affective*, pages 335–339.

Hamer, M. (1986). Left is right on the road. *New Scientist*, pages 16–18.

International, S. A. E. (2016). U.S. Department of Transportation's New Policy on Automated Vehicles Adopts SAE International's Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles. *SAE international*, page 1.

Ji, M. and Egerstedt, M. (2007). Distributed Coordination Control of Multiagent Systems While Preserving Connectedness. *IEEE Transactions on Robotics*, 23(4):693–703.

Khalil, H. (2002). *Nonlinear Systems*. Prentice Hall, third edition.

Lay, M. G. (1992). *Ways of the World: A History of the World's Roads and the Vehicles that Used Them*.

Mohr, C., Brugger, P., Bracha, H. S., Landis, T., and Viaud-Delmon, I. (2004). Human side preferences in three different whole-body movement tasks. *Behavioural Brain Research*, 151(1-2):321–326.

Morris, B., Powell, M. J., and Ames, A. D. (2013). Sufficient conditions for the lipschitz continuity of QP-based multi-objective control of humanoid robots. *Proceedings of the IEEE Conference on Decision and Control*, pages 2920–2926.

Ögren, P. and Leonard, N. E. (2002). A provably convergent dynamic window approach to obstacle avoidance. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 15, pages 115–120.

Park, M. G., Jeon, J. H., and Lee, M. C. (2001). Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. *Proceedings. ISIE 2001. IEEE International Symposium on Industrial Electronics, 2001.*, pages 1530–1535.

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., and Egerstedt, M. (2017). The Robotarium: A remotely accessible swarm robotics research testbed. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1699–1706.

Pickem, D., Wang, L., Glotfelter, P., Diaz-Mercado, Y., Mote, M., Ames, A., Feron, E., and Egerstedt, M. (2016). Safe, Remote-Access Swarm Robotics Research on the Robotarium. (April):1–13.

Prajna, S. (2005). Optimization-based methods for nonlinear and hybrid systems verification. *System*, 2005:110.

Prajna, S. (2006). Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126.

Prajna, S. and Jadbabaie, A. (2004a). Barrier Certificates for Nonlinear Model Validation. (December):477–492.

Prajna, S. and Jadbabaie, A. (2004b). Saftey verification of hybrid systems using barrier certificates. *Hybrid Systems: Computation and Control*, 2993.

Prajna, S., Jadbabaie, A., and Pappas, G. (2004). Stochastic safety verification using barrier certificates. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, pages 929–934 Vol.1.

Prajna, S., Jadbabaie, A., and Pappas, G. J. (2007). A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428.

Prajna, S. and Rantzer, A. (2005). *on the Necessity of Barrier Certificates*, volume 38. IFAC.

Rao, V. G. and Bernstein, D. S. (2001). Naive control of the double integrator. *IEEE Control Systems Magazine*, 21(5):86–97.

Robertson, J. S., Forte, J. D., and Nicholls, M. E. (2015). Deviating to the right: Using eye-tracking to study the role of attention in navigation asymmetries. *Attention, Perception, and Psychophysics*, 77(3):830–843.

Ruf, S. F. (2018). A Control Theoretic Perspective on Social Networks. (May).

Siciliano, B., Schiavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics Modelling Planning and Control*. Springer.

Tang, Z. L., Tee, K. P., and He, W. (2013). Barrier Lyapunov functions for the control of output-constrained nonlinear systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 3(PART 1):449–455.

Thomas, N. A., Churches, O., White, I., Mohr, C., Schrag, Y., Obucina, S., and Nicholls, M. E. (2017). An investigation of left/right driving rules on deviations while walking. *PLoS ONE*, 12(10):1–11.

Walters, B. (1998). Huge Roman Quarry found in North Wiltshire. *ARA The Bulletin of The Association for Roman Archaeology*, pages 8–9.

Wang, L., Ames, A., and Egerstedt, M. (2016a). Safety barrier certificates for heterogeneous multi-robot systems. In *Proceedings of the American Control Conference*, volume 2016-July, pages 5213–5218.

Wang, L., Ames, A. D., and Egerstedt, M. (2016b). Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 2659–2664.

Wang, L., Ames, A. D., and Egerstedt, M. (2017). Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674.

Watson, I. (1999). The rule of the road, 1919-1986 A case study of standards change.

Wieland, P. and Allgöwer, F. (2007). Constructive safety using control barrier functions. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 7(PART 1):462–467.

Xu, X., Tabuada, P., Grizzle, J. W., and Ames, A. D. (2015). Robustness of Control Barrier Functions for Safety Critical Control. *IFAC-PapersOnLine*, 48(27):54–61.

Ringraziamenti

Ho aspettato tanto per scrivere queste due righe, eppure ora fatico a trovare le parole. Sono soddisfatto dei risultati che sono riuscito ad ottenere da questi anni di magistrale e, anche se per molti potrebbe sembrare banale, pensarmi qui non mi è sempre parso scontato: devo questo mio piccolo risultato a tutte le persone che mi sono state sempre vicine.

Il primo ringraziamento è per la mia famiglia: i miei genitori, le mie sorelle, le mie zie. Mi avete dato tutto, soprattutto la serenità.

A second thought goes to Pedro V., you were with me especially when I needed it the most, always.

I am grateful to the three professors that helped me the most to achieve this result, Dr. Magnus Egerstedt for the care he always showed in my work and whose enthusiasm moves mountains; Prof.ssa Lucia Pallottino for the precious feedback and constant interest; Prof. Antonio Bicchi for always taking the time in guiding me towards the right direction with candor and affection.

Grazie ad Umberto B. e Lorenzo Q. amici e fratelli, faro e bussola.

Grazie a Federico N., Pietro G., Eugenio F. inesauribili fonti di ispirazione, e grazie ad Edwin H. per la pazienza, la perseveranza, la motivazione. Siete stati la mia seconda anima ed il mio modello di riferimento in questi lunghi e cortissimi anni.

Grazie ad Alessandro V. che ha creduto in me.

Grazie ad Alessandro B. che non si è mai stufato di me.

Grazie a tutti i ragazzi e le ragazze di U-PHOS, l'avventura più incredibile della mia vita. E soprattutto grazie al Prof. Sauro F. e al Prof. Mauro M. che, senza lavagna e gesso, mi hanno insegnato le lezioni più importanti.

A warm thank-you goes to all my friends in IRIM, who welcomed me like one of their own: it was a blast and an honour.

Grazie anche a chi, con piccole azioni quotidiane, mi è stato vicino nell'ultimo periodo, grazie di cuore a Francesca G., Alessandro P. e Antonio A.

Grazie, infine, agli amici di casa, sempre presenti attraverso gli anni, àncora del tempo. *Cà nisciun è fes!*