Coordinated Path Following Control and Formation Control of Mobile Robots

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften

der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

M.Sc. Kiattisin Kanjanawanishkul

aus Trang, Thailand

Tübingen

2010

Tag der mündlichen Qualifikation:	23.07.2010
Dekan:	Prof. Dr.-Ing. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr.-Ing. Frank Allgöwer

Abstract

Rapid advances in sensing, computing and communication technologies have led to considerably increased research activities in multi-robot systems over the last decade. Topics include multi-robot motion planning, cooperative manipulation, aerial applications involving cooperative exploration of the unknown environment, automated highway systems, software architectures for multi-robot systems, and formation control. Multi-robot systems have been proven to offer additional advantages in terms of flexibility in operating a group of robots and failure tolerance due to redundancy in available mobile robots. However, the benefits of using multi-robot teams do not come without cost. Coordinating teams of autonomous robots is much more challenging than maneuvering a single robot.

This dissertation addresses formation control problems, which are among the most active research topics in multi-robot systems. Over the last two decades, there have been a large number of publications on this field, and it is still growing. Recently, this research has been extended to some related research areas, e.g., consensus problems and distributed control systems, imposing new challenges on formation control problems. In general, formation control subproblems addressed in the literature can be classified as formation shape generation, formation reconfiguration/selection, formation tracking, and role assignment in formation. The main purpose of this dissertation is to address two important and correlated subproblems in formation control: formation tracking and role assignment in formation. The goal of the former is that a team of mobile robots is required to maintain a geometric formation while tracking a reference or a set of references. The latter arises when a mobile robot in the team must decide what role to take on in a desired formation configuration.

In particular, we study coordinated path following control of omnidirectional mobile robots and unicycle mobile robots. This problem can be seen as a subtask of formation tracking. Path following is one of the three basic motion control tasks in mobile robot research. The others are trajectory tracking and point stabilization. Even though less attention is drawn to this problem in the literature, it offers some advantages over trajectory tracking in some cases. The objective of path following control is to be on the path rather than at a certain point at a particular time. To solve this problem, we employ a model predictive control (MPC) technique to generate a sequence of optimal velocities of a so-called virtual vehicle which is followed by a real robot. This approach can eliminate stringent initial condition constraints because the velocity of a virtual vehicle is controlled explicitly. Using this technique, we can gain some benefits over other available control schemes, e.g., the ability to incorporate generic models, linear and nonlinear, and constraints in the optimal control problem and the ability to use future values of ref-

erences when they are available, allowing to improve system performance. However, the main drawback is significant computational burden associated with solving a set of nonlinear differential equations and a nonlinear dynamic optimization problem online.

Then, we extend path following control to *coordinated path following control*. A group of mobile robots not only follow a reference path but also maintain a geometric formation shape. The main challenge is to design a decentralized control law using only local information to achieve a formation tracking objective. In this study, we propose two solutions. In the first solution, the MPC framework for path following control is extended to the coordinated path following control problem. In spite of great theoretical properties of such MPC controllers, the stability and feasibility of decentralized schemes are rather conservative. The second solution is computationally simple so that it may be suitable for low-computational systems when the advantages of MPC schemes including constraint handling are not a dominating factor. Its controller design is based on a Lyapunov technique and a second-order consensus protocol with a reference velocity. It is worth noting that the path variable has been used as a coupling variable synchronizing each member in formation in both solutions.

In the second formation control subproblem, we study role assignment in formation. This problem becomes more challenging when robots in the team do not have complete information and they do not know the number of robots participating in the formation tasks. With the assumption that the formation graph is connected and bidirectional, we propose an online and distributed role assignment. This approach is proven by extensive simulation and experimental results.

Acknowledgments

I would first like to acknowledge my advisor, Prof. Dr. Andreas Zell, for giving me an opportunity to do my research work on a team of mobile robots and for his continuing support and guidance throughout my years at the University of Tübingen. I would also like to thank the dissertation committee member, Prof. Dr.-Ing. Frank Allgöwer, for his constructive advice and efforts in reading and commenting on my dissertation.

I am thankful to the Attempto Tübingen Robot Soccer team members for developing the RobocCup robots which I have used for validating my control algorithms. I owe special thanks to Dr. Xiang Li for helping me use the RoboCup software framework and control the RoboCup robots during my first year and for introducing me to path following control problems and model predictive control.

I would like to thank all colleagues at the Department of Computer Architecture for an excellent working atmosphere. I appreciate the help from Matthias Müller (from the Institute for Systems Theory and Automatic Control of the University of Stuttgart), Philipp Vorst, Marius Hofmeister, Philippe Komma, and Karl E. Wenzel for their proofreading of this dissertation and many valuable suggestions. I am especially indebted to my office mate, Philipp Vorst, for always willing to give me suggestions and assistance, the interesting discussions, and for answering all my programming and Linux questions. I gratefully acknowledge his help and support all the time. Furthermore, I am very thankful to Marius Hofmeister for helping me during my first year lectures, and for his support in experiments. Last but not least I would like to thank Dr. Christian Weiss and Karsten Bohlmann for their help.

Additionally, I would like to thank Vita Serbakova for doing a great job as our department's secretary. Many thanks go to Klaus Beyreuther who is always helpful with all computer problems.

Most importantly, I profoundly thank my parents and my sisters for their love, their encouragement and constant support throughout the years.

This dissertation was made possible by a Thai government scholarship. I am grateful to the Thai government for the scholarship which enabled me to undertake a PhD study at the University of Tübingen.

Contents

Chapter 1

Introduction

Robots have become of increasingly more importance in human daily lives in the last decade and apparently the number of robots will increase and get more involved in the human society in the near future [28]. Real-world applications employing robots have already shown the effectiveness and usefulness of robots, especially in industry. However, many unsolved problems still exist in many robotic research areas.

1.1 Motivation

In recent years, multi-robot systems (MRS) have been the object of widespread research interest in the scientific community (see [13, 34, 172] for surveys on this topic), given their application in different fields of robotics, such as service, military, or educational robotics. The interest in using MRS is due to their characteristics of redundancy and flexibility of mission execution, and tolerance to possible robot faults. Moreover, MRS have been realized with different types of autonomous vehicles such as ground mobile robots [36], underwater vehicles [159], unmanned aerial vehicles [215], aircraft [133], space craft [11], and marine surface vessels [208]. The research in MRS has matured to the point where systems with hundreds of robots [97, 183] or teams of heterogeneous robots [30] are proposed.

In this dissertation, formation control, which has been an important issue in coordinated control for a group of autonomous robots for decades [42], is studied. The formation problem is defined as the coordination of a group of robots to get into and to maintain a formation with a certain shape. Current application areas of formation control include search and rescue operations, security patrols, landmine removal, remote terrain and space exploration, control of arrays of satellites and UAVs, area coverage and reconnaissance in military missions. There are a large number of publications in the fields of motion coordination and formation control. The motivations that draw the attention of researchers to this problem are as follows:

- Biological inspirations: researchers have observed the remarkable group-level characteristics that are exhibited as emergent properties from individual-level behaviors, such as flocking and schooling. Rule-based distributed group motion control

and their emergent behavior was first identified in 1986 by Reynolds [198] when he showed homogeneous animal motion can be created using computer graphic models based on the behavior of schooling fish and flocking bird. Using elementary control rules of various animals (e.g., ants, bees, birds, and fishes) and trying to imitate their group behavior (e.g., foraging, flocking, homing, and dispersing) became the key principle in cooperative behaviors [155].

- Challenging control problems: design of control algorithms for decentralized coordinated systems presents a number of challenges not present in single vehicles or centralized systems. Some of these challenges are due to high system dimensionality, complex interactions, inherent parallelism, incomplete information, and uncertainties.

- Demands of multi-robot systems: in many applications, a given task is too complex to be achieved by a single robot acting alone, or a given task cannot physically be executable at all by a single robot, or multiple robots can achieve the same mission of a single robot while reducing the execution time and increasing the performance.

In general, basic tasks in robotic research are mapping, controlling, planning and localizing [171]. Usually, a robot creates a map of the environment. Using this map, it can localize itself. Then it plans the reference if it wants to travel. The controller is designed to control it to move to the target. However, to accomplish those missions is not an easy task. The problem focused on in this dissertation is control, particularly coordinated path following control and formation control of wheeled mobile robots (WMR). Path following is one of three generic problems of motion control of a vehicle [171]. The main characteristics of these three basic motions are as follows:

1. point stabilization, where the objective is to stabilize a vehicle at a desired robot posture,

2. trajectory tracking, where the vehicle is required to track a time-parameterized reference, and

3. path following, where the vehicle is required to converge to and follow a desired path-parameterized reference, without any temporal specifications.

Typically, in path following controllers smoother convergence to a path is achieved compared to trajectory tracking controllers, and the control signals are less likely pushed to saturation. Many solutions of this problem have been proposed and applied in a wide range of applications. For example, Samson [200] described a path following problem for a car pulling several trailers. In [9], Altafini addressed a path following controller for an n trailer vehicle. Path following in an urban electric vehicle and a car-like vehicle were studied in [223] and [212], respectively. Furthermore, path following controllers for aircraft and marine vehicles were reported in [6] and [68], respectively. This problem

becomes more difficult if a group of mobile robots are required to follow a path and to maintain a desired formation at the same time. The coordination among team members poses significant theoretical and practical challenges. Each member has to communicate with its neighbors in order to achieve a coordinated path following mission using only local information.

In this study, we first develop a framework for controlling a single mobile robot to follow a reference path, and we then extend this framework to coordinated path following and formation control of a group of mobile robots through two approaches. One is based on nonlinear model predictive control (NMPC), and the other is based on a Lyapunov function and a second-order consensus protocol with a reference velocity. In this first approach, we employ two strategies, i.e., a leader-following strategy and a distributed MPC strategy. We also further investigate distributed role assignment in formation and formation switching. This problem arises when a mobile robot in the team must decide what role to take on in a desired formation configuration.

1.2 Dissertation Organization

The remainder of this dissertation is structured as follows:

Chaper 2: Background Control Theory

This chapter gives a brief review on nonlinear system theory, model predictive control, and consensus protocols. These theoretical aspects have been greatly employed in the dissertation. Since robot motions are classified as a nonlinear control system, basic principles in nonlinear control are explained in this chapter. Our solutions of coordinated path following and formation control are based on model predictive control, which is one of the most popular control techniques, and consensus protocols, which have been very active research in multi-robot systems for a decade.

Chapter 3: Robot Systems

This chapter describes system architectures, robot hardware, and software frameworks. System structures provide the infrastructure upon which multi-robot systems are implemented. They can be categorized as centralized control vs. decentralized control, communication structures, and heterogeneity vs. homogeneity. Robot hardware, on which our proposed approaches have been evaluated, include omnidirectional mobile robots and unicycle mobile robots, while robot software systems are the RoboCup framework and the CARMEN framework.

Chapter 4: Path Following Control

A path following control problem for an omnidirectional mobile robot and a unicycle mobile robot is studied in this chapter. The proposed algorithms based on model predictive control (MPC) are presented. It is shown that the optimal velocity of a so-called

virtual vehicle can be obtained explicitly by using MPC algorithms. We validate our proposed solutions on physical mobile robots. We also give a comparison between trajectory tracking and path following, and between nonlinear model predictive control and linear model predictive control using a linearized model.

Chapter 5: Coordinated Path Following Control

We extend the results from Chapter 4 to multi-robot systems. Current literature on formation control strategies, distributed MPC, and coordinated path following control are briefly reviewed in this chapter. Two solutions are proposed. In the first solution, the nonlinear MPC framework is tailored to coordinated path control problems using two alternative choices, a leader-following approach and a distributed approach. MPC formulations are discussed and real-world experiments are presented. The other solution is based on a Lyapunov function and a consensus protocol. The cooperative strategies between an omnidirectional mobile robot and unicycle mobile robots are also developed in this chapter.

Chapter 6: Role Assignment and Formation Switching

Finally, contributions to role assignment in the role formation and formation switching are given. Two examples using a team of physical nonholonomic mobile robots are provided. First, robots reconfigure themselves from one formation to another. Second, formation switching happens, while each robot is following a reference path.

Chapter 7: Conclusions and Future Work

This chapter summarizes the contributions of this dissertation and outlines possible directions of future research.

Chapter 2

Background Control Theory

This chapter provides basic knowledge of nonlinear control systems and theories concerning model predictive control and consensus protocols, which are mainly employed to achieve our goal in coordinated path following control and formation control. We also introduce relevant literature and background material on these topics. Major issues on nonlinear model predictive control are discussed and nonlinear optimization solvers are reviewed in Subsection 2.2.2 and Subsection 2.2.3, respectively. This is followed by a discussion of the literature on decentralized MPC algorithms and their advantages over centralized ones. Section 2.3 is devoted to consensus problems, in which a variety of algorithms have been proposed over the past few years, such that a group of robots can agree upon certain quantities of interest, such as direction, position, and decision, with only local information.

2.1 Nonlinear System Theory

In order to be self-contained, some fundamental control theory about stability of equilibrium points of autonomous and nonautonomous nonlinear systems are recalled. The theorems and definitions are mainly borrowed from [123]. However, the proofs are not reported here.

2.1.1 Lipschitz Functions

To ensure the existence and the uniqueness of the solution of the initial-value problem

$$\dot{x} = f(t,x), \quad x(t_o) = x_0$$

a key constraint is the Lipschitz condition, whereby $f(t,x)$ satisfies the inequality

$$\|f(t,x) - f(t,y)\| \leq L\|x-y\| \tag{2.1}$$

for all (t,x) and (t,y) in some neighborhood of (t_0,x_0).

A function satisfying the inequality (2.1) is said to be *Lipschitz* in x, and the positive constant L is called a *Lipschitz constant*.

Definition 1. A function $f(t,x)$ is said to be

- *locally Lipschitz* in x on $[a,b] \times D \subset \mathbb{R} \times \mathbb{R}^n$ if each point $x \in D$ has a neighborhood D_0 such that f satisfies (2.1) on $[a,b] \times D_0$ with some Lipschitz constant L_0.
- *Lipschitz* in x on $[a,b] \times W$ if it satisfies (2.1) for all $t \in [a,b]$ and all points in W with the same Lipschitz constant L.
- *globally Lipschitz* in x if it is *Lipschitz* in x on $[a,b] \times \mathbb{R}^n \subset \mathbb{R} \times \mathbb{R}^n$.

The Lipschitz property of a function is stronger than continuity and, as stated in the following lemmas, weaker than continuous differentiability.

Lemma 1. If $f(t,x)$ and $[\partial f/\partial x](t,x)$ are continuous on $[a,b] \times D$, for some domain $D \subset \mathbb{R}^n$, then f is locally Lipschitz in x on $[a,b] \times D$.

Lemma 2. If $f(t,x)$ and $[\partial f/\partial x](t,x)$ are continuous on $[a,b] \times \mathbb{R}^n$, then f is globally Lipschitz in x on $[a,b] \times \mathbb{R}^n$ if and only if $[\partial f/\partial x]$ is uniformly bounded on $[a,b] \times \mathbb{R}^n$.

2.1.2 Lyapunov Stability

Consider the autonomous system

$$\dot{x} = f(x) \tag{2.2}$$

where $f : D \to \mathbb{R}^n$ is a locally Lipschitz map from a domain $D \subset \mathbb{R}^n$ into $\mathbb{R}^n$. Suppose this system has an equilibrium point at the origin of $\mathbb{R}^n$, that is, $f(0) = 0$. There is no loss of generality in doing so because any equilibrium point can be shifted to the origin via a change of variables.

Definition 2. The equilibrium point $x = 0$ of (2.2) is

- *stable*, if, for each $\varepsilon > 0$, there is a $\delta = \delta(\varepsilon) > 0$ such that

$$\|x(0)\| < \delta \Rightarrow \|x(t)\| < \varepsilon, \ \forall t \geq 0.$$

- *unstable*, if it is not stable.
- *asymptotically stable*, if it is stable and δ can be chosen such that

$$\|x(0)\| < \delta \Rightarrow \lim_{t \to \infty} x(t) = 0.$$

In order to demonstrate that the origin is a stable equilibrium point, for each selected value of ε one must produce a value of δ, possibly dependent on ε, such that a trajectory starting in a δ neighborhood of the origin will never leave the ε neighborhood. It is possible to determine the stability by examining the derivatives of some particular functions, without having to know explicitly the solution of (2.2).

Theorem 1. *(Lyapunov's stability theorem) Let $x = 0$ be an equilibrium point for (2.2) and $D \subset \mathbb{R}^n$ be a domain containing $x = 0$. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(0) = 0 \ \text{ and } \ V(x) > 0 \ \text{ in } \ D - \{0\} \tag{2.3}$$

$$\dot{V}(x) \leq 0 \ \text{ in } \ D \ . \tag{2.4}$$

Then, $x = 0$ is stable. Moreover, if

$$\dot{V}(x) < 0 \ \text{ in } \ D - \{0\} \tag{2.5}$$

then $x = 0$ is asymptotically stable.

A continuously differentiable function $V(x)$ satisfying (2.3) and (2.4) is called a *Lyapunov function*, after the Russian mathematician who laid the foundation of this theory. A function $V(x)$ satisfying condition (2.3) is said to be *positive definite*. If it satisfies the weaker condition $V(x) \geq 0$ for $x \neq 0$ it is said to be *positive semidefinite*. A function $V(x)$ is said to be *negative definite* or *negative semidefinite* if $-V(x)$ is positive defnite or positive semidefinite, respectively. With this terminology, we can rephrase Lyapunov's theorem to say that the origin is stable if there is a continuously differentiable positive definite function $V(x)$ so that $\dot{V}(x)$ is negative semidefinite, and it is asymptotically stable if $\dot{V}(x)$ is negative definite.

When the origin $x = 0$ is asymptotically stable, we are often interested in determining how far from the origin the trajectory can be and still converge to the origin as $t \to \infty$. This gives rise to the definition of the *region of attraction*. Let $\phi(t;x)$ be the solution of (2.2) that starts at initial state x at time $t = 0$. Then, the region of attraction is defined as the set of all points x such that $\phi(t;x)$ is defined for all $t \geq 0$ and $\lim_{t\to\infty} \phi(t;x) = 0$. Furthermore, it will be the case if we can show that for any initial state x, the trajectory $\phi(t;x)$ approaches the origin as $t \to \infty$, no matter how large $\|x\|$ is. If an asymptotically stable equilibrium point at the origin has this property, it is said to be *globally asymptotically stable*.

Theorem 2. *Let $x = 0$ be an equilibrium point for (2.2). Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(0) = 0 \ \text{ and } \ V(x) > 0, \ \ \forall x \neq 0 \tag{2.6}$$

$$\|x\| \to \infty \ \Rightarrow \ V(x) \to \infty \tag{2.7}$$

$$\dot{V}(x) < 0, \ \ \forall x \neq 0 \tag{2.8}$$

then $x = 0$ is globally asymptotically stable.

A function satisfying condition (2.7) is said to be *radially unbounded*.

2.1.3 The Invariance Principle

The stability theorems of subsection 2.1.2 require to find a Lyapunov function whose time derivative is negative definite. If in a domain around the origin, however, a Lyapunov function can be found whose derivative along the trajectories of the system is only negative semidefinite, asymptotic stability of the origin might still be proved, provided that no trajectory can stay identically at the points where $\dot{V}(x) = 0$, except at the origin. This idea follows from LaSalle's *invariance principle*.

Definition 3. A set M is called *invariant set* for an autonomous system if every trajectory starting from a point in M will remain in the set for all future time.

Definition 4. A set M is said to be a *positively invariant set* if

$$x(0) \in M \;\Rightarrow\; x(t) \in M, \;\; \forall t \geq 0. \tag{2.9}$$

Theorem 3. *(LaSalle's theorem) Let $\Omega \subset D$ be a compact set that is positively invariant with respect to $\dot{x} = f(x)$. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $\dot{V}(x) \leq 0$ in Ω. Let E be the set of all points in Ω where $\dot{V}(x) = 0$. Let M be the largest invariant set in E. Then every solution starting in Ω approaches M as $t \to \infty$.*

When we want to show that $x(t) \to 0$ as $t \to \infty$, we need to establish that the largest invariant set in E is the origin. Specializing Theorem 3 to this case and taking $V(x)$ to be positive definite, we obtain the following two corollaries:

Corollary 4. *Let $x = 0$ be as an equilibrium point for (2.2). Let $V : D \to \mathbb{R}$ be a continuously differentiable positive definite function on a domain D containing the origin $x = 0$, such that $\dot{V}(x) \leq 0$ in D. Let $S = \{x \in D | \dot{V}(x) = 0\}$ and suppose that no solution can stay identically in S, other than the trivial solution $x(t) \equiv 0$. Then, the origin is asymptotically stable.*

Corollary 5. *Let $x = 0$ be as an equilibrium point for (2.2). Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable, radially unbounded, positive definite function such that $\dot{V}(x) \leq 0$ for all $x \in \mathbb{R}^n$. Let $S = \{x \in \mathbb{R}^n | \dot{V}(x) = 0\}$ and suppose that no solution can stay identically in S, other than the trivial solution $x(t) \equiv 0$. Then, the origin is globally asymptotically stable.*

2.1.4 Nonautonomous Systems

The notions of stability and asymptotic stability of equilibrium points of nonautonomous systems are very similar to those introduced in Definition (2.2) for autonomous systems. The difference is that while the solution of an autonomous system depends only on $(t - t_0)$, the solution of the nonautonomous system

$$\dot{x} = f(t,x), \;\; x(t_0) = x_0 \tag{2.10}$$

depends on both t and t_0. To cope with this new situation, the definitions of stability and asymptotic stability need to be redefined so that they hold uniformly in the initial time t_0. The origin is an equilibrium point of (2.10) at $t = 0$ if

$$f(t,0) = 0, \ \ \forall t \geq 0.$$

Again, there is no loss of generality since an equilibrium point at the origin could be a translation of a nonzero equilibrium point.

Definition 5. The equilibrium point $x = 0$ of (2.10) is

- *stable*, if, for each $\varepsilon > 0$, there is $\delta = \delta(\varepsilon, t_0) > 0$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| < \varepsilon, \ \ \forall t \geq t_0 \geq 0.$$

- *uniformly stable*, if, for each $\varepsilon > 0$, there is $\delta = \delta(\varepsilon) > 0$, independent of t_0, such that (2.10) is satisfied.

- *unstable*, if it is not stable.

- *asymptotically stable*, if it is stable and there is a positive constant $c = c(t_0)$ such that $x(t) \to 0$ as $t \to \infty$, for all $\|x(t_0)\| < c$.

- *uniformly asymptotically stable*, if it is uniformly stable and there is a positive constant c, independent of t_0, such that for all $\|x(t_0)\| < c$, $x(t) \to 0$ as $t \to \infty$, uniformly in t_0; that is, for each $\eta > 0$, there is $T = T(\eta) > 0$ such that

$$\|x(t)\| < \eta, \ \ \forall t \geq t_0 + T(\eta), \ \ \forall \|x(t_0)\| < c.$$

- *globally uniformly asymptotically stable*, if it is uniformly stable, $\delta(\varepsilon)$ can be chosen to satisfy $\lim_{\varepsilon \to \infty} \delta(\varepsilon) = \infty$, and, for each pair of positive numbers η and c, there is $T = T(\eta, c) > 0$ such that

$$\|x(t)\| < \eta, \ \ \forall t \geq t_0 + T(\eta, c), \ \ \forall \|x(t_0)\| < c.$$

Equivalent definitions can be given using two comparison functions, known as class $\mathcal{K}$ and $\mathcal{KL}$ functions.

Definition 6. A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasingly and $\alpha(0) = 0$. It is said to belong to class $\mathcal{K}_\infty$ if $a = \infty$ and $\alpha(r) \to \infty$ as $r \to \infty$.

Definition 7. A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ is said to belong to class $\mathcal{KL}$ if, for each fixed s, the mapping $\beta(r, s)$ belongs to class $\mathcal{K}$ with respect to r and, for each fixed r, the mapping $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to \infty$.

The next lemma redefines uniform stability and uniform asymptotic stability using class $\mathcal{K}$ and class $\mathcal{KL}$ functions.

Lemma 3. The equilibrium point $x = 0$ of (2.10) is

- *uniformly stable*, if and only if there exist a class $\mathcal{K}$ function α and a positive constant c, indepedent of t_0, such that

$$\|x(t)\| < \alpha(\|x(t_0)\|), \ \forall t \geq t_0 \geq 0, \ \forall \|x(t_0)\| < c. \tag{2.11}$$

- *uniformly asymptotically stable*, if and only if there exist a class $\mathcal{KL}$ function β and a positive constant c, independent of t_0, such that

$$\|x(t)\| < \beta(\|x(t_0)\|, t - t_0), \ \forall t \geq t_0 \geq 0, \ \forall \|x(t_0)\| < c. \tag{2.12}$$

- *globally uniformly asymptotically stable*, if and only if inequality (2.12) is satisfied for any initial state $x(t_0)$.

A special case of uniform asymptotic stability arises when the class $\mathcal{KL}$ function β in (2.12) takes the form $\beta(r,s) = kre^{-\lambda s}$.

Definition 8. The equilibrium point $x = 0$ of (2.10) is exponentially stable if there exist positive constants c, k and λ such that

$$\|x(t)\| \leq k\|x(t_0)\|e^{-\lambda(t-t_0)}, \ \forall \|x(t_0)\| < c \tag{2.13}$$

and globally exponentially stable if (2.13) is satisfied for any initial state $x(t_0)$.

The Lyapunov theory for autonomous systems extended to nonautonomous systems is given in [123].

2.1.5 Barbalat's Lemma and Stability of Time-varying Systems

In the case of autonomous systems, LaSalle's invariance theorem (Theorem 3) shows that the trajectory of the system approaches the largest invariant set in E, where E is the set of all points in Ω where $\dot{V}(x) = 0$. In the case of nonautonomous systems, it may not even be clear how to define a set E, since $\dot{V}(t,x)$ is a function of both t and x. This is where Barbalat's lemma comes into picture. It says

Lemma 4. (Barbalat's Lemma) If $V(t,x)$ satisfies the following conditions

1. $V(t,x)$ is lower bounded,
2. $\dot{V}(t,x)$ is negative semidefinite, and
3. $\dot{V}(t,x)$ is uniformly continuous in time (satisfied if $\ddot{V}$ is finite)

then $\dot{V}(t,x) \to 0$ as $t \to \infty$.

2.1.6 Boundedness

Lyapunov analysis can be used to show boundedness of the solution of the state equation, even when there is no equilibrium point at the origin.

Definition 9. The solution of

$$\dot{x} = f(t,x) \tag{2.14}$$

where $f : [0,\infty) \times D \to \mathbb{R}^n$ is piecewise continuous in t and locally Lipschitz in x on $[0,\infty) \times D$, and $D \subset \mathbb{R}^n$ is a domain that contains the origin, is

- *uniformly bounded*, if there exists a positive constant c, independent of $t_0 \geq 0$, and for every $a \in (0,c)$, there is $\beta = \beta(a) > 0$, independent of t_0, such that

$$\|x(t_0)\| \leq a \Rightarrow \|x(t)\| \leq \beta, \ \ \forall t \geq t_0. \tag{2.15}$$

- *globally uniformly bounded*, if (2.15) holds for arbitrarily large a.

- *uniformly ultimately bounded with ultimate bound b* if there exist positive constants b and c, independent of $t_0 \geq 0$, and for every $a \in (0,c)$, there is $T = T(a,b) \geq 0$, independent of t_0, such that

$$\|x(t_0)\| \leq a \Rightarrow \|x(t)\| \leq b, \ \ \forall t \geq t_0 + T. \tag{2.16}$$

- *globally uniformly ultimately bounded* if (2.16) holds for arbitrarily large a.

In the case of autonomous systems, we may drop the word "uniformly" since the solution depends only on $t - t_0$.

2.2 Model Predictive Control (MPC)

Model predictive control (MPC), also referred to as receding horizon control (RHC) and moving horizon optimal control, has been widely adopted in process control industry because the control objectives and operating constraints can be integrated explicitly in the optimization problem that is solved at each instant. Many successful MPC applications have been reported in the last three decades [169, 192].

Although it is traditionally applied to plants with dynamics slow enough to permit computations between samples, recently, due to the combination of advanced research results and the advent of faster computers, it has become possible to extend the implementation of MPC design to systems governed by faster dynamics. Examples of such applications to systems other than process control problems have begun to emerge, including neural network based MPC applied to an underwater vehicle [126], mobile robots [180], wind-tunnel experiments [204], helicopter experiments [229, 206], thrust-vectored flight control [165], and aircraft gas turbine engines [29].

2.2.1 Principles and Formulation

The conceptual structure of MPC is illustrated in Figure 2.1. As its name suggests, an MPC algorithm employs an explicit *model* of the plant to be controlled which is used to *predict* the future output behavior. This prediction capability allows computing a sequence of manipulated variable adjustments in order to solve optimal control problems online, where the future behavior of a plant, e.g., tracking error, namely, the difference between the predicted output and the desired reference, is optimized or minimized over a future horizon, possibly subject to constraints on the manipulated inputs and outputs [8, 22, 157]. The result of the optimization is applied according to a *receding horizon* philosophy: At time t only the first input of the optimal command sequence is actually applied to the plant. The remaining optimal inputs are discarded, and a new optimal control problem is solved at time $t+\delta$, where δ is the sampling period. As new measurements are collected from the plant at each time t, the receding horizon mechanism provides the controller with the desired feedback characteristics.

When the model is linear, the optimization problem is a strictly convex quadratic programming problem if the performance index is expressed through the l_2-norm, or a linear programming problem if expressed through the l_1/l_∞-norm. It has a unique, global minimum which can be quickly and reliably computed numerically in a constrained case. In an unconstrained case the solution can be computed analytically as a linear feedback control law. Two most popular linear MPC algorithms using input-output type process models include dynamic matrix control (DMC) [50] and generalized predictive control (GPC) [45, 46]. If the process model is in the form of a discrete step response, DMC can be obtained, whereas if it is in the form of a discrete transfer function or equivalently the

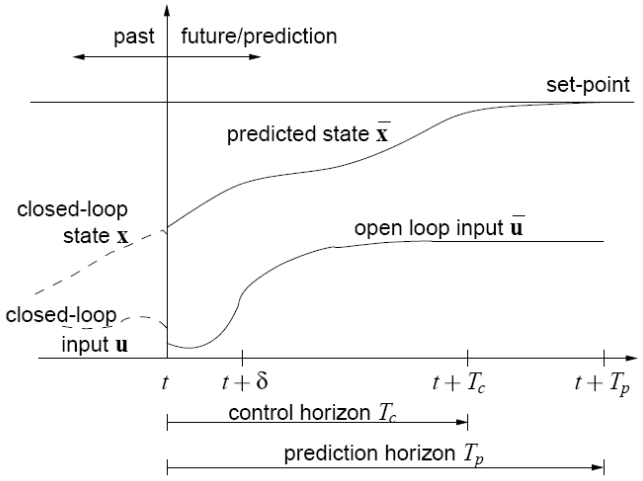


Figure 2.1: Principle of model predictive control [8].

difference equation (ARX-type model), GPC can be derived. By now, important issues of linear MPC theory (e.g., online computation, modeling/identification, and stability) are well addressed [130, 169]. However, many systems are inherently nonlinear and linear MPC is inadequate for highly nonlinear systems. Therefore, nonlinear models must be used [8]. However, the optimization problem is certainly not linear or quadratic, it is generally a nonconvex and even multi-modal one when the model is nonlinear. For such problems, there are no sufficiently fast and reliable numerical optimization procedures, i.e., procedures yielding always an optimal point and within predefined time limit as is required in online control applications. Therefore, many attempts have been made to construct simplified (and generally suboptimal) nonlinear MPC algorithms avoiding full online nonlinear optimization. One possibility is to use model linearization or multiple linear models, in which only a QP problem is solved online [169]. There are also many designs of predictive algorithms based on nonlinear optimization and also using artificial neural network (ANN) techniques [5, 38, 137, 188, 221].

In general, nonlinear model predictive control (NMPC) refers to MPC schemes that are based on nonlinear models and/or consider non-quadratic cost functionals and general nonlinear constraints on the states and inputs. A nonlinear system is normally described by the following nonlinear differential equation:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) \\ \text{subject to:} \quad & \mathbf{x}(t) \in \mathcal{X},\ \mathbf{u}(t) \in \mathcal{U},\ \forall t \geq 0 \end{aligned} \tag{2.17}$$

where $\mathbf{x}(t) \in \mathbb{R}^n,\ \mathbf{u}(t) \in \mathbb{R}^m$ are the n dimensional state vector and the m dimensional input vector of the system, respectively. $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ denote the set of feasible states and inputs of the system, respectively. In NMPC, the input applied to the system is usually given by the solution of the following finite horizon open-loop optimal control problem (FHOCP), which is solved at every sampling instant:

$$\min_{\bar{\mathbf{u}}(\cdot)} \int_t^{t+T_p} F(\bar{\mathbf{x}}(\tau),\bar{\mathbf{u}}(\tau))\,d\tau + V(\bar{\mathbf{x}}(t+T_p)) \tag{2.18}$$

$$\begin{aligned} \text{subject to:} \quad & \dot{\bar{\mathbf{x}}}(\tau) = \mathbf{f}(\bar{\mathbf{x}}(\tau),\bar{\mathbf{u}}(\tau)) \\ & \bar{\mathbf{u}}(\tau) \in \mathcal{U} \ \ \forall \tau \in [t, t+T_c] \\ & \bar{\mathbf{x}}(\tau) \in \mathcal{X} \ \ \forall \tau \in [t, t+T_p] \\ & \bar{\mathbf{x}}(t+T_p) \in \Omega \end{aligned} \tag{2.19}$$

where $F(\bar{\mathbf{x}},\bar{\mathbf{u}}) = \bar{\mathbf{x}}^T Q \bar{\mathbf{x}} + \bar{\mathbf{u}}^T R \bar{\mathbf{u}}$. The bar denotes an internal controller variable. T_p represents the length of the *prediction horizon* or *output horizon*, and T_c denotes the length of the *control horizon* or *input horizon* ($T_c \leq T_p$). When $T_p = \infty$, we refer to this as the *infinite horizon problem*, and similarly, when T_p is finite, as a *finite horizon* problem. $V(\bar{\mathbf{x}}(t+T_p))$ is the terminal penalty and Ω is the terminal region. The deviation from the desired values is weighted by the positive definite matrices Q and R. Note that

in our implementation, the control horizon T_c is set to be equal to the prediction horizon T_p.

A standard MPC scheme works as follows [8]:

1. Obtain measurements/estimates of the states of the system at time instant t.

2. Calculate an optimal input series $\bar{u}(t, t+T_c)$ by minimizing the desired cost function over the predictive horizon in the future using the system model, the generated predictive state sequence $\bar{x}(t, t+T_p)$ from $\bar{u}(t, t+T_c)$ should contain the terminal state $\bar{x}(t+T_p)$ that falls in the required terminal state region.

3. Implement the first part of the optimal input series $u = \bar{u}$ until the new measurement/estimates of the states are available.

4. Continue with 1. at the next time instant $t = t + \delta$.

The most significant tuning parameters of the NMPC include the sampling period δ, the control horizon T_c, prediction horizon T_p, and the penalty weight matrices Q, R. To ensure good closed-loop performance, the sampling period should be small enough to capture the process dynamics. Using a small sampling period generally improves performance but requires a longer prediction horizon to adequately capture the process dynamics which means an increase in online computation time. For a fixed prediction horizon, a smaller control horizon yields more sluggish output response and more conservative input moves. Large control horizons have the opposite effect on performance. In addition, large values of a control horizon lead to an increase of online computation as the control horizon is linearly related to the number of decision variables in the nonlinear programming (NLP) problem. In practice, the control horizon often must be chosen to provide a balance between performance and computation. The prediction horizon has similar effects as the control horizon. In fact, nominal stability is strongly affected by the prediction horizon length. However, the advantages of longer prediction horizons are outweighed by the increase in computation time and result in more aggressive control. The weighting matrices Q, R, can be the most difficult tuning parameters to select because their values depend both on the scaling of the problem and the relative importance of the variables.

The main advantages of using the nonlinear version of MPC include

- its ability to incorporate generic models, linear and nonlinear, and constraints in the optimal control problem;

- its formulation that can be extended to handle multiple-variable, nonlinear, time-varying plants in a single control formulation;

- its ability to redefine cost functions and constraints as needed to reflect changes in the system and/or the environment;

- its ability to use future values of references when they are available, allowing MPC to improve the performance in navigation such as waypoint trajectory tracking;
- its ability to tune parameters that are directly related to a cost function.

Until now, nearly all aspects regarding the NMPC theory (e.g., stability, performance, nonlinearity, and robustness) are well developed, see [22, 151, 157, 169]. In the next subsection, we review some important system theoretic issues, while the issue of real-time optimization is addressed in Subsection 2.2.3. Finally, the literature on decentralized MPC schemes is discussed in Subsection 2.2.4.

2.2.2 Issues on Nonlinear MPC

In this subsection, we review two important issues, i.e., feasibility and stability.

Feasibility of the optimization problem at each time t must be ensured. Typically one assumes feasibility at time $t = 0$ and chooses the cost function and the stability constraints such that feasibility is preserved at the following time steps. This can be done, for example, by ensuring that the shifted optimal sequence $\{\bar{u}(t+\delta), ..., \bar{u}(t+\delta T_p), 0\}$ is feasible at time $t+\delta$. Furthermore, typically the constraints in (2.19) which involve state components are treated as *soft* constraints, for instance by adding the slack variable ε, while pure input constraints in (2.19) are maintained as *hard* because they stem from actuator saturation and/or physical, safety or economical requirements. Relaxing the state constraints removes the feasibility problem at least for stable systems. Keeping the state constraints tight does not make sense from a practical point of view because of the presence of noise, disturbances, and numerical errors.

The next major concern in the use of a predictive control horizon is that whether such an open-loop control can guarantee *system stability*. It is shown that an infinite predictive control horizon can guarantee stability of a system, but the infinite predictive horizon may not be feasible for a nonlinear system in practice [8]. Mayne et al. [157] have presented the essential principles for the stability of model predictive control of constrained dynamical systems. Different approaches to attain closed-loop stability using finite horizon lengths exist. We review some of the popular techniques proposed in the literature to *enforce* stability. For reasons of a simple presentation, no detailed proofs are given.

The first result of proving continuous MPC with a terminal equality constraint, i.e., $\bar{x}(t+T_p) = 0$, was done by Chen and Shaw [40] in 1982 for nonlinear time-invariant systems. Another original work by Keerthi and Gilbert [120] in 1988 employed a terminal equality constraint on the state for time-varying, constrained, nonlinear, discrete-time systems. In this variant $V(x) = 0$ in (2.18) and $\Omega = \{0\}$ in (2.19), Mayne and Michalska [156] showed that the finite horizon constrained optimal control problem can be posed as minimizing a standard quadratic objective subject to an additional terminal state equality constraint, i.e., $\bar{x}(t+T_p) = 0$. The main drawback of using terminal equality constraints

is that the control effort required to steer the state to the origin can be large, especially for short T_p, and therefore feasibility is more critical. The domain of attraction of the closed-loop system is limited to the set of initial state x_0 that can be steered to 0 in T_p time-unit. Also, performance can be negatively affected because of the artificial terminal constraint. Thus, researchers looked for relaxations that would still guarantee stability. The version of MPC utilizing a terminal inequality constraint provides some relaxation. Michalska and Mayne [163] proposed terminal inequality constraints such that the states are on the boundary of a terminal region at the end of a variable prediction horizon. They suggested a dual-mode receding horizon control scheme with a local linear state feedback controller inside the terminal region and a receding horizon controller outside the terminal region. Closed-loop control with this scheme is implemented by switching between these two controllers, depending on the states being inside or outside the terminal region. However, the switching would result in unsmoothed control signals. It was found later in multiple studies that there is a good reason to incorporate a terminal cost. Specially, it is generally possible to set $V(\cdot)$ to be exactly or approximately equal to the infinite horizon value function in a suitable neighborhood of the origin. Thus, most recent MPC controllers use a terminal cost and enforce a terminal constraint set. The following stability theorem given in [8] provides a way to find the suitable terminal penalty and constraints.

Theorem 6. *Suppose*

1. $\mathcal{U} \subset \mathbb{R}^m$ *is compact,* $\mathcal{X} \subseteq \mathbb{R}^n$ *is connected and the origin is contained in the interior of* $\mathcal{U} \times \mathcal{X}$.

2. *The vector field* $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ *is continuous in* $\boldsymbol{u}$ *and locally Lipschitz in* $\boldsymbol{x}$ *and satisfies* $f(0,0) = 0$.

3. $F : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}$ *is continuous in all arguments with* $F(0,0) = 0$ *and* $F(x,u) > 0,\ \forall (x,u) \in \mathbb{R}^n \times \mathcal{U} \setminus \{0,0\}$.

4. *The terminal penalty* $V : \Omega \to \mathbb{R}$ *is continuous with* $V(0) = 0$ *and that the terminal region* Ω *is given by* $\Omega := \{x \in \mathcal{X} \mid V(x) \leq e_1\}$ *for some* $e_1 > 0$ *such that* $\Omega \subset \mathcal{X}$.

5. *There exists a continuous local control law* $u = k(x)$ *such that* $k(x) \in \mathcal{U}$ *for all* $x \in \Omega$ *and*

$$\frac{\partial V}{\partial x} f(x,k(x)) + F(x,k(x)) \leq 0,\ \forall x \in \Omega. \tag{2.20}$$

6. *The NMPC open-loop optimal control problem (2.18) has a feasible solution for* $t = 0$.

Then for any sampling time $0 < \delta \leq T_p$ *the nominal closed-loop system is asymptotically stable and the region of attraction* $\mathcal{R}$ *is given by the set of states for which the open-loop optimal control problem has a feasible solution.*

The terminal cost V is, basically, assumed to be an F-conform control Lyapunov function for the system in the terminal region Ω, enforcing a decrease in the value function. The terminal region constraint enforces feasibility at the next sampling instants and allows to show that the value function is strictly decreasing. Thus, stability can be established [8]. Many NMPC schemes follow this theorem to guarantee stability. Generally, they differ in how the terminal region and terminal penalty terms are obtained. For example, Chen and Allgöwer [41] proposed a quasi-infinite horizon NMPC scheme, where the terminal state penalty term approximates the infinite horizon cost of the nonlinear system starting from the terminal region and controlled by the local linear state feedback controller. The terminal region is calculated around the origin that can be stabilized by a linear control law. However, the local state feedback control is never implemented. In this case, instead of requiring the final states to be at the origin, the final states will be in the region Ω. Kothare and Morari [128] addressed nonlinear constrained MPC for continuous-time systems by employing an additional state constraint called a contractive constraint. For full state information and in the absence of disturbances, the contractive constraint is defined as a Lyapunov function that is used to prove exponential stability of the closed-loop system. MPC with no state or control constraints is dealt with by [187] (discrete time) and [101] (continuous time). Both studies employ a local stabilizing control law, a local control Lyapunov function for $V(\cdot)$ and choose Ω to be a level set of $V(\cdot)$ and to be positively invariant under a local controller. The terminal constraint is omitted from the online optimization but it satisfied implicitly for all initial states in a level set of the value function. Jadbabaie et al. [102] also showed that the region of attraction for a finite horizon problem can be made larger by choosing a larger horizon. For the horizon approaching ∞, this leads to the region of attraction of the infinite horizon problem. A mixture of enforced contraction and the control Lyapunov function approach is considered by Primbs et al. [190].

To summarize, the main differences of all these approaches using the terminal region constraint and/or a terminal penalty term in the cost function are feasibility, computational burden and performance. Basically, the terminal penalty and the terminal region are determined off-line such that the cost function gives an upper bound on the infinite horizon cost and guarantees a decrease in the value function as the horizon recedes in time. Various ways to determine a suitable terminal penalty term and terminal region exist. Examples are the use of a control Lyapunov function as terminal penalty [102, 190] for the system in the terminal region, enforcing a decrease in the value function, or the use of a local nonlinear or linear control law to determine a suitable terminal penalty and a terminal region [41, 150, 163]. The terminal region constraint is added to enforce that if the open-loop optimal control problem is feasible once, that it will remain feasible, and to allow establishing the decrease using the terminal penalty (see [41, 101, 157, 163] for more details). In general, it is not necessary to find always an optimal solution in order to guarantee stability [41, 102, 203]. Only a feasible solution resulting in a decrease in the value function is necessary. This can be utilized to decrease the necessary online solution time [8].

2.2.3 Optimization Solvers

Although rigorous stability results for nonlinear MPC have been well-established, it is not applicable in practical implementation. Since a constrained nonlinear optimization problem has to be solved online, the heavy online computational burden causes two important issues in implementation of nonlinear MPC [8]. One is the computational delay. The other is the global optimization solution which cannot be guaranteed in each optimization procedure since it is, in general, a non-convex, constrained nonlinear optimization problem.

In practice, linear models are most often used and the resulting optimizations are linear or quadratic programs. In the nonlinear constrained optimization, the solution can be obtained using either an indirect or a direct method. *Indirect methods* aim to solve the multi-point boundary value problem (MPBVP). The solution of the MPBVP can be very difficult even for small problems. *Direct methods*, on the other hand, optimize the objective criterion directly by discretizing the original problem to finite dimensional approximation. Basically this is done by parameterizing the input (and possibly the states) by a finite number of parameters and to solve/approximate the differential equations during the optimization [8]. In principle, any parameterization of the input can be chosen, i.e., the parameterized input is given by

$$\bar{u}(\tau;q), \quad \tau \in [t, t+t_p]$$

where q is the vector of parameterization parameters. The parameterized $\bar{u}(\tau;q)$ might, for example, be given by a sum of basis functions such as a Fourier series or the input is parameterized as piecewise constant [8]. The discretized version of the optimal control problem (OCP) – a sparse NLP problem – can be solved with well known NLP algorithms, such as sequence quadratic programming (SQP) [7, 19, 75, 222] or interior-point (IP) methods [39]. To apply parameterization, there are mainly three strategies [8]:

1. Sequential approach or feasible path approach: the control is finitely parameterized in the form $\bar{u}(\tau;q)$ and the state trajectories are eliminated by numerically integrating the differential equation and cost. Only the control parameterization parameters remain as degree of freedom in a standard mathematical program. For each evaluation of the cost $J(\{\bar{u}_1, ...\bar{u}_N\}, x(t); t, t+T_p)$ in the solution of the mathematical program the differential equation and the cost function are numerically integrated using the current guess of the input parameterization parameters of the optimizer. Thus the name sequential or feasible path approach, since the optimization steps and the simulation are performed sequentially leading to a valid/feasible state trajectory [136].

2. Simultaneous approach: the solution to the differential equation and the optimization is obtained simultaneously. For this purpose, the differential equations are discretized and then treated as additional constraints in the optimization problem.

Typical simultaneous approaches use collocation methods to discretize the differential equations. The resulting nonlinear programming problem is very large but also very sparse. The sparseness can be exploited to achieve an efficient solution.

3. Direct multiple shooting approach: the optimization horizon of interest is divided into a number of sub-intervals with local control parameterizations. The differential equations and cost on these intervals are integrated independently during each optimization iteration based on the current guess of the control. The continuity/-consistency of the final state trajectory at the end of the optimization is enforced by adding consistency constraints to the nonlinear programming problem. The resulting nonlinear program takes a special sparse structure which can be utilized for an efficient solution.

These strategies have different advantages and disadvantages [8]. For example, the introduction of initial states as optimization variables in simultaneous approaches result in a sparse structure of the underlying QP problem. This structure leads to a fast solution strategy. In comparison, the matrices for the sequential approach are often dense and thus the solution is expensive to obtain. A drawback of the simultaneous approach and multiple shooting approach is that only at the end of the iteration a valid state trajectory for the system is available. Thus, if the immediate answer for early termination is needed, a single shooting approach is more favored.

To solve NLP problems, the following key points need to be considered:

- *Interior point vs. active set* to handle bounds and inequality constraints in generating search directions: classical active set methods are iterative processes that try to guess at each iteration which are the active constraints. They usually consider active constraints one at a time, inducing a computation time directly related to the number of active constraints. Unfortunately, they are non-polynomial, i.e., the worst-case number of performed steps increases faster than any polynomial with the number of inequality constraints. On the other hand, we have interior-point methods with a polynomial bound of computational complexity. The computation time of modern interior point methods is relatively constant, regardless of the number of active constraints. Interior point algorithms are based on the idea of identifying active inequality constraints numerically, with the help of a nonlinear barrier function. The resulting nonlinear equation system is then solved using a Newton method. Interior point methods show excellent numerical properties for large-scale QP problems. In comparing these approaches, both methods possess clear trade-offs [19, 26, 193]. If there are few inequality constraints (particularly in case of small and medium problems) or an active set is known (from a good starting guess, or the warm-start QP solution from a previous iteration) then the active set method is favored. On the other hand, for problems with many inequality constraints, interior point methods are often faster as they avoid the combinatorial problem of selecting the active set.

- *Primal vs. dual*: primal strategies ensure that all the constraints are satisfied at every iteration. The iterative process can be interrupted and still produce a feasible motion at any moment, satisfying all the constraints, if there is a limit on computation time. One limitation of primal strategies is that they require an initial value for the variable $\bar{u}$ which already satisfies all the constraints. For a general QP, computing such an initial value can take as much time as solving the QP afterwards. This is why dual methods are usually preferred, they satisfy all the constraints only at the last iteration but they do not require such an initial value.

- *Null space vs. range space algebra*: there exist mainly two ways of making computations with the linear constraints, either considering the null space of the matrix orthogonal to the constraints, or the range space of this matrix, parallel to the constraints.

- *First and second derivatives*: there are a number of modeling and simulation platforms with accurate first and second derivatives that can be accessed for optimization. An exact or approximated (such as forward finite differences and centered finite difference) first derivative is used to compute the objective function gradient and constraint gradient. If the second derivative is not available, positive definite quasi-Newton approximations to the reduced Hessian (such as quasi-Newton BFGS, quasi-Newton SR1, and limited-memory quasi-Newton BFGS) are quite successful.

- *Line search vs. trust region methods* to enforce global convergence of the SQP iterations: these two methods are commonly used for the search directions calculated from the above QP subproblems. In a trust region approach, the constraint is added to the QP. The step is taken if there is sufficient reduction of a merit function (e.g., the objective function weighted with some measure of the constraint violations). Popular merit functions for SQP methods include the augmented Lagrangian function or exact penalty functions. On the other hand, line search methods can be more efficient on problems with reasonably good starting points and well-conditioned QP subproblems.

In this work, we have employed two free optimization solver packages. One is the DONLP2 software package written by P. Spellucci [214] for solving NLP problems. The other is the OOQP package written by E. M. Gertz and S. J. Wrightis [81] for solving convex quadratic programming problems. DONLP2 is a set of C subroutines for minimizing a smooth function subject to constraints, which may include simple bounds on the variable, linear constraints and smooth nonlinear constraints. A user provides subroutines to define the objective and constraint functions and (optionally) their first derivatives. DONLP2 uses a sequential quadratic programming (SQP) algorithm with an active set technique, in which each search direction is the solution of a QP subproblem. Bounds, linear constraints and nonlinear constraints are treated separately. It also has

a built-in feature for finding a gradient numerically if an analytic gradient is not given. In general, the SQP method has a two-level structure of iterations [75]. At each major iteration, a QP subproblem is obtained from the nonlinear program through a quadratic approximation of the Hessian of the Lagrangian function and a linear approximation of the constraints. This leads to a search direction and a line-search step size, which determines the next iterate [85]. In minor iterations we simply iterate the solution of the QP problem.

The OOQP package is used in Section 4.3, where a linear MPC controller with a linearized model is obtained. In this QP problem, the objective is a convex quadratic function and the constraints are linear functions of a vector of real variables. In OOQP, object-oriented quadratic programming techniques are used to implement a primal-dual, interior-point algorithm.

2.2.4 Centralized MPC vs. Decentralized MPC

MPC is usually implemented in a centralized fashion. One controller has the full knowledge about the entire system and computes all the control inputs for the entire system. In large-scale interconnected systems, such as water distribution systems, traffic systems, power systems, manufacturing systems, economic systems such a centralized-control, non-convex optimization scheme may be too complex or not even possible for technical or commercial reasons [207]. Thus, with the rapid development of communication networks, centralized control has been gradually replaced by distributed control, such as formation control [65, 121], applications in the manufacturing and process industry where multiple units cooperatively produce a product [1], and large-scale power systems [32, 90, 95]. In distributed or decentralized control schemes, the local control inputs are computed using local measurements and reduced-order models of the local dynamics [207]. The main challenge is to formulate simpler decentralized problems which result in a behavior similar to what is obtained with a centralized approach.

In general, stability and feasibility of decentralized schemes are very difficult to prove and/or too conservative [121]. Even if we assume T_p to be infinite, the decentralized MPC approach does not guarantee that solutions computed locally are globally feasible and stable. The decentralization of the control is further complicated when disturbances act on the subsystems making the prediction of future behavior uncertain. The key point to guarantee feasibility and stability is that when decisions are made in a decentralized fashion, the actions of each subsystem must be consistent with those of the other subsystems [199]. Thus, decisions taken independently do not lead to a violation of the coupling constraints. For example, to provide some information about anticipated effects of interactions between subsystems, a wide variety of approaches include exchange of predicted state trajectories [65], robustness to the actions of others [32, 107, 106], penalty functions [206], and partial grouping of computations [121]. The ways, in which a decentralized system model and the coupling between subsystems is formed, can be found in [173].

To summarize, approaches to decentralized control design differ from each other in the assumptions they make on [122]

- the kind of interaction between different systems or different components of the same system (dynamics, constraints, objective),
- the model of the system (linear, nonlinear, constrained, continuous-time, discrete-time),
- the model of information exchange between the systems,
- the control design technique used.

For instance, decoupled nonlinear dynamics and constraints and coupling in a quadratic cost function is presented by Dunbar and Murray [65]. Dynamically coupled subsystems are given by Venket [226], though it requires that subsystems be linear, time-invariant and coupled solely through the control inputs. In [32], Camponogara et al. proposed a scheme with stability guarantees for dynamically coupled systems, with information exchange between nodes and contractive stability constraints in the distributed MPC subproblems.

2.3 Consensus Protocols

Information consensus has been the center of much attention recently since advances in computation and communication technology over the past few years have provided efficient and inexpensive ways to share and compute information. A variety of algorithms have been proposed such that a group of robots can agree upon certain quantities of interest, such as direction and position with only local information (see [72, 178, 197] and a myriad of references for details on consensus algorithms).

One avenue of the research in consensus protocols relies on algebraic graph theory, in which graph topologies are connected with the algebraic properties of the corresponding graph matrices. Communication links among agents are described by *Laplacian* matrices. Each agent is treated as a vertex and the communication links between agents are treated as edges. It is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, ..., N\}$ is the set of agents and $\mathcal{E} \subset \mathcal{V}\text{x}\mathcal{V}$ is the set of relative vectors between agents. Two agents i and j are called neighbors if $(i, j) \in \mathcal{E}$, and the set of neighbors of agent i is denoted by $N_i \subseteq \mathcal{V}$. Graphs considered can be either undirected or directed. When the graph is undirected and connected, the *Laplacian* matrix L, constructed from $L = D - A$, where the adjacency matrix $A = (a_{ij})$ and the diagonal degree matrix D, is symmetric positive semi-definite. It has a simple zero eigenvalue and all the other eigenvalues are positive if and only if the graph is connected [72].

We first show a simple formulation given by Fax and Murray [72] and Olfati-Saber and Murray [177]. A solution to the consensus problem is to let the behavior of each linear system/agent be governed by the first order differential equation

$$\dot{x}_i = -\frac{1}{|N_i|}\sum_{j=1}^{|N_i|}(x_i - x_j) \tag{2.21}$$

where $x_i \in \mathbb{R}$ is the internal state of the agent. For this system, one can show that if the information flow is bidirectional (if agent i is a neighbor of agent j, then j is a neighbor of i), the states of the individual vehicles asymptotically converge to the average of the initial state values for any connected graph $\mathcal{G}$.

If $\mathcal{G}$ is not bidirectional (so that there are asymmetries in the information available to each agent), then the interaction above does not necessarily lead to average consensus. We define a graph to be *balanced* if the in-degree and out-degree of all nodes are equal. In the case of balanced graphs, one can once again show that any connected graph solves the average consensus problem using the interaction rules above [177]. Furthermore, even if the connections are changing as a function of time, it can be shown that the average consensus is still reached.

We intentionally do not collect all published contribution to consensus-based approaches. Some work reported on the consensus problem in the literature are as follows.

Jadbabaie et al. [103] gave a theoretical explanation for consensus of the heading angles of a group of agents using nearest neighbor rules under undirected switching information exchange topologies. The stability of the consensus algorithms were analyzed with the aid of results from graph theory. It is shown that consensus is achieved asymptotically if the union of the information exchange graphs for the team is connected most of the time as the system evolves. Lin et al. [145] extended the consensus algorithms to the case where the information exchange graphs were directed. In [146], Lin et al. derived general conditions on the network topology stabilizing a group of unicycle robots. It was also shown how to make a group of robots converge to a line or general geometric form by solving the consensus problem. Lafferriere et al. [131] investigated the connection between the spectral graph theory and the control problem in vehicle formations. The vehicles exchange information according to a pre-specified undirected communication graph. A state-space approach was developed to stabilize the formation. It is proven that a linear stabilizing feedback law always exists provided that the communication graph is connected. The rate of convergence to formation is governed by the size of the smallest positive eigenvalue of the Laplacian of the communication graph. Fax and Murray [72] studied information exchange techniques to improve stability margins and formation performance for vehicle formations. In [195], Ren and Beard considered the problem of information consensus among multiple agents in the presence of limited and unreliable information exchange with dynamically switching topologies. Updated algorithms were proposed for information consensus in both discrete and continuous cases. The consensus problem for networks of dynamic agents with fixed and switching topologies was

discussed by Olfati-Saber and Murray [177]. They proposed two consensus protocols for networks with and without time-delays. Olfati-Saber et al. [178] investigated consensus algorithms with emphasis on robustness, time-delays and performance guarantee. In [220], Tanner et al. proposed a decentralized controller which is stable under arbitrary changes in the connected network.

In contrast to the algebraic graph approach, some other researchers make use of nonlinear mathematical tools to study consensus problems. In [170], Moreau used a set-valued Lyapunov approach to study the stability of the consensus problems with unidirectional time-dependent communication links. Necessary and/or sufficient conditions for the convergence of the state of each individual agent to a consensus vector were presented with the aid of graph theory and convexity. In [44], Chung and Slotine used nonlinear contraction theory to study synchronization, which are related to the consensus problem.

Optimality issues related to consensus problems are also studied in the literature. Xiao and Boyd [236] addressed the fastest distributed linear averaging (FDLA) problems in the context of consensus seeking among multiple autonomous agents. On asynchronous communication networks, a distributed iterative procedure under the eventual update assumption was developed by Mehyar et al. [160] for calculating average consensus. The asynchronous consensus problem with zero time delay was studied by Cao at al. in [33] where the union of the communication graphs is assumed to have a common root spanning tree.

2.4 Summary

This chapter provided the basic knowledge in nonlinear control systems, model predictive control (MPC), and consensus protocols. In Section 2.2, an overview of the theoretical and computational aspects of nonlinear MPC has shown some of the challenging issues, including stability, feasibility, nonlinear optimization problems, and decentralized implementation. Although MPC is suitable for low-process systems, such as chemical factories, with new optimization solvers, more powerful computers and more advanced MPC frameworks, MPC can be implemented in real-time applications, as seen in our experimental results in Chapter 4 and Chapter 5. In Section 2.3, consensus protocols, which have proven to be effective tools for performing network-wide distributed computation tasks, are reviewed. Their usefulness and effectiveness to coordinated motions are shown in Section 5.5 and Chapter 6.

Chapter 3

Robot Systems

To illustrate the usefulness of our proposed algorithms, both simulation and physical robot experiments have been conducted in this dissertation. The control framework has been evaluated on omnidirectional mobile robots and unicycle mobile robots. The omnidirectional mobile robots were originated from the Attempto Tübingen Robot Soccer team [93]. They have been rebuilt and used for supporting the service robot research (see Figure 3.5, compared to the old structure in Figure 3.1). The unicycle mobile robots, called c't-Bots, were originated from the German c't magazine, and further developed by M. Hofmeister [96]. Both types of robots have differences in sensors, models, software structures, and computational power.

The first section in this chapter reviews system architectures, i.e., what infrastructure is behind the multi-robot system implementation. Next, robot hardware of the omnidirectional mobile robots and unicycle mobile robots are described in detail. Software frameworks are then given in Section 3.3.

3.1 System Architectures

The system architectures provide the infrastructure upon which the multi-robot systems are implemented. They furthermore determine the capabilities and limitations of the system. Although not comprehensive, the criteria given here demonstrate some of the key features that must be addressed in solving coordinated path following and formation control problems. We here briefly discuss some of the key architectural features of multi-robot systems. The reader is referred to [13, 34, 64] for more details.

3.1.1 Heterogeneity vs. Homogeneity

In general, team composition can be divided in two main classes, heterogeneous and homogeneous teams. Homogeneous teams are composed of team members that have the same hardware and control software, while in heterogeneous teams the robots differ either in the hardware or in the software control procedures. Heterogeneity generally introduces complexity since task assignment becomes more difficult and robots have a

need to model other individuals in the group. Using homogeneous robots makes the system robust because no single robot is critical to the mission.

Using heterogeneous robots in MRS tasks may be necessary in some applications. For instance, the formation can involve different kinds of robots equipped with different sensors: vision, sonars, lasers, and GPS. Only few robots may possess all the sensors and thus could serve as leader of the whole team, providing higher level information, such as mapping or exploration.

3.1.2 Communication Structures

Clearly, the usage of communication among the robots can improve team performance, allowing the robots to acquire more information and to self-organize in a more reliable way. The communication may take place directly via an explicit communication channel or indirectly through one robot sensing a change of other robots in its environment. Communication configuration can be further categorized as follows (see [64] for details):

- *Communication range*: the maximum distance between two robots of the team such that communication is still possible. We list three key classes for this dimension: (i) no direct communication: robots cannot communicate with other robots directly, but it is possible for robots to communicate with each other indirectly by observing their presence, absence or behavior, (ii) local communication: robots can only communicate with other robots which are sufficiently nearby, and (iii) global communication: robots can communicate with any other robot. This is a classical assumption, which is probably impractical if the number of robots is large.

- *Communication topology*: it captures physical interconnections among team members. Robots may not be able to communicate with an arbitrary individual of the team regardless of its proximity. Individual robots may have names and messages may be sent to them directly, or messages may be broadcasted to all robots. The topology can be represented as a tree, in which robots only communicate through this hierarchy or as a graph. A graph is a more general connectivity scheme than a tree and is more robust since redundant links can prevent the entire group from becoming disconnected. Moreover, the topology can be either static if the topology is fixed, or dynamic if the relationship of members of the team can change arbitrarily. The interconnection structure can also be either bidirectional or unidirectional.

- *Communication bandwidth*: it indicates the amount of data a communication link can transmit in a given period of time.

Studies that require global information or broadcast communication may suffer from lack of scalability or high costs of the physical setup but allow more accurate coordination of

multi-robot tasks. On the other hand, studies using only local communication and sensor data tend to be more scalable, more robust, and easier to build; but their cooperative tasks may be limited. Currently, a lot of work in the literature has dealt with achieving robustness and fault tolerance in multi-robot communication, such as setting up and maintaining distributed communication networks and ensuring reliability in multi-robot communication.

3.1.3 Centralization vs. Decentralization

Centralized controllers deal with systems in that a single controller processes all the information needed to achieve the desired control objectives (including stability and performance requirements). However, in many applications, because of the nature of the inter-robot communication network and due to the highly distributed nature of robots' sensing and actuation modules, it is impossible to tackle the problems in the centralized control framework. For these reasons, there have been a lot of activities in the area of multi-agent networks over the past few years. It is widely claimed that decentralized systems under local sensing, control, and interactions among robots and environments have several inherent advantages, including robustness/fault tolerance against single robot failures, natural exploitation of parallelism, and scalability [34].

In formation control problems, centralized control laws can ideally yield superior performance and optimal decisions for both the individual members and the formation as a whole. For example, the control scheme uses a single controller that oversees the decision process for the whole group, generates collision free trajectories in the workspace, and plans the motion of the group members. The motion of each member is then transmitted to the robot controller via a communication channel. Although this guarantees a complete solution, centralized algorithms require high computational power, massive communication flow of information (e.g., state measurements, sensory information, and guidance signals), and are not robust because of heavy dependence on a single controller. Such requirements are not feasible for numerous applications. On the other hand, in decentralized control, each formation member has its own controller and is completely autonomous in the decision process. This can significantly reduce the number of signals being communicated, is more flexible, and robust, requires less computational effort, and is more scalable. Furthermore each robot can quickly respond to problems occurring in its environment.

Nevertheless, there is also the need to provide some degree of centralization with an interface to human operators for programming, tasking, and monitoring of the system. There are also some hybrid centralized/decentralized architectures wherein there is a central planner that exerts high-level control over mostly autonomous robots.

3.2 Robot Hardware

In this section, two different robot structures, i.e., a unicycle mobile robot and an omnidirectional mobile robot and their basic motion control are examined in detail. Motion control approaches of mobile robots can, in general, be designed based on the robots' dynamic or kinematic models. To design a control law, the kinematic model is simpler than the dynamic one. In particular, it does not involve a certain number of matrix-valued functions whose precise determination relies on the knowledge of numerous parameters associated with the vehicle and its actuators (geometry of bodies, masses, moments of inertia, coefficients of reduction in the transmission of torques produced by the motors, etc.) [171]. For many applications, it is not necessary to know all these terms precisely. Moreover, most commercial available robots do not allow the direct control of forces or torques. Instead, they incorporate motor controllers that allow the specification of translation and rotation, which we treat as control inputs. In this case, for the low-level velocity control loop, a simple PID controller can be used to stabilize the motor angular velocity. If the regulation loop is efficient, the difference between the desired and actual velocities remains small, even when the desired velocity and the motor load vary continuously. However, in case that decoupling the kinematics from the dynamics of the vehicle is not present, we can design them by using the information of the terms involved in the dynamic equation. The reader is referred to [31, 171] for more details in the structural properties and classification of kinematic and dynamic models of wheeled mobile robots. In this dissertation, it is assumed that the plane of each wheel is perpendicular to the ground and that the contact between the wheels and the ground is pure rolling and non-slipping. Thus, kinematic models can be used in our controller design. We neglect actuator dynamics by assuming that the low-level velocity control loop is much faster than the desired closed loop system dynamics.

3.2.1 Omnidirectional Mobile Robots

Omnidirectional mobile robots are becoming increasingly popular in mobile robot applications, since they have some distinct advantages over nonholonomic mobile robots. They have a full omnidirectionality with simultaneously and independently controlled rotational and translational motion capabilities, which means that they can move at each instant in any direction without reorientation [31]. The middle-size league of the annual RoboCup competition wherein teams of autonomous mobile robots compete in the game of soccer, is an example of a highly dynamic environment where omnidirectional vehicles have been employed successfully (see RoboCup Official Site: http://www.robocup.org).

The omnidirectional motion is enabled via special wheels used in the mobile robot designs [189]. One of the most popular arrangements utilizes so-called Swedish wheels, shown in Figure 3.4, mounted on the periphery of the chassis, thus allowing freedom of motion without the necessity of reconfiguring its internal state. A Swedish wheel differs from a common wheel in the fact that rollers are mounted on its perimeter (see

Figure 3.3). If all the rollers are parallel to each other and misaligned with respect to the wheel hub axis, they will provide an extra degree of mobility with respect to a traditional perfectly rolling wheel. The kinematics analysis of Swedish wheel robots has been addressed in several papers [15, 124, 148, 189, 213], while dynamic models of omnidirectional mobile robots equipped with Swedish wheels are discussed in [109, 191].

We validate our control algorithms by using physical omnidirectional mobile robots, shown in Figure 3.1, later rebuilt, shown in Figure 3.5. The architecture utilizes a Pentium-M 2 GHz onboard PC with 1 GB RAM for the main processing unit. There are three Swedish wheels placed in such orientation that their axes of rotation point towards the center of the robot and there is an angle of 120° between them. Each wheel is driven by a 24-Volt Maxon DC motor of 60 Watt each with 18:1 gear ratio, and has the same distance L_w from its center to the robot's center of mass (point R in Figure 3.2). Figure 3.3 shows a photograph of the real hardware with the top view of the general three-wheeled omnidirectional mobile robot model. Optical encoders are mounted on each motor shaft to provide feedback for the motor speed controller. The control system consists of two loops: the outer loop is an external kinematic loop, while the inner loop is the low-level actuator velocity servo loops. In particular, the low-level motor speed feedback control is realized by a three-channel digital PID board (the TMC200 designed and built at AiS - Fraunhofer Autonomous Intelligent Systems Institute) with pulse-width modulation (PWM) output. The TMC200 board is interfaced through a serial RS232 link with an onboard PC. In the old structure designated for the RoboCup competition, there was an omnidirectional camera as a sole sensor, which was used for self localization within the RoboCup field, developed by Heinemann et al. [94]. This self-localization algorithm was based on probabilistic Monte-Carlo localization (MCL). The vision system consisting of a Marlin F-046C camera and a hyperbolic mirror employed on the mobile robot platform provides a 360° field of view. The Marlin F-046C has a resolution of 780x580 pixels and is able to capture and transmit 50 fps at a reduced resolution of 580x580 pixels in the 16 bit YUV4:2:2 format via the IEEE 1394a FireWire bus [93]. In the new structure designated for robot service purposes, the sensory systems consist of six sonar sensors, one laser scanner, and one vision system. The overall system uses 24-Volt NiMH batteries.

As a first step to develop a model-based robot controller, the equations of robot motion need to be derived. The kinematic model of an omnidirectional mobile robot can be easily obtained by considering the geometric representation. The basic architecture of the wheeled platform is illustrated in Figure 3.2, where there are two coordinate frames used in the modeling: the body frame (X_m, Y_m) and the world frame (X_w, Y_w). The body frame is fixed on the moving robot with the origin at its center of mass, whereas the world frame is fixed at the ground. We assume that the center of mass for the robot is located at the center of the robot chassis, which is the origin of (X_m, Y_m). With the assumption that no slip in all the three wheels occurs, sensors have high accuracy and ground is sufficiently planar, the velocity component with respect to the world frame is obtained by

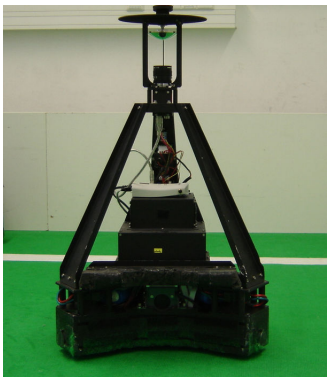


Figure 3.1: The old structure of the omnidirectional mobile robot.

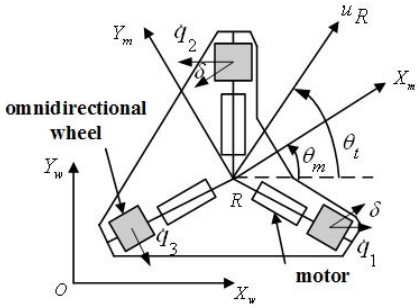


Figure 3.2: Coordinate frames of the omnidirectional mobile robot.

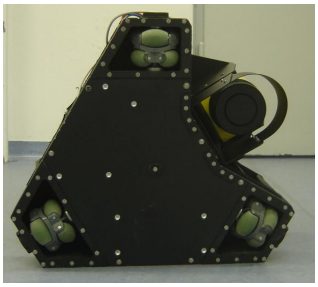


Figure 3.3: Real base of the omnidirectional mobile robot.

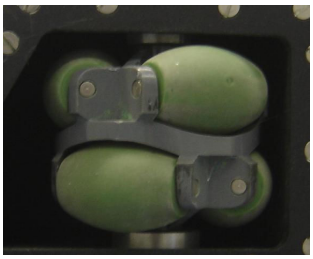


Figure 3.4: A Swedish wheel.

Figure 3.5: The new structure of omnidirectional mobile robots.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega \end{bmatrix} \tag{3.1}$$

where the point (x, y) is the position of the center of the robot on the axes (X_w, Y_w) and θ is the angular position with respect to the axis X_w. The input signals are given by $u,\ v,\ \omega$ with $u,\ v$ being two orthogonal velocity vectors, where u is aligned with the reference axis of the robot. ω corresponds to the rotational velocity of the robot. Angle θ_t denotes the robot's moving direction in the world frame. From Figure 3.2, where we defined that wheel 1, wheel 2, and wheel 3 are front right wheel, front left wheel, and real wheel, respectively, it is easy to see that the wheel velocities are related to the components over the axis (X_m, Y_m) and the rotational velocity as

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta & L_w \\ -\cos\delta & \sin\delta & L_w \\ 0 & -1 & L_w \end{bmatrix} \begin{bmatrix} u \\ v \\ \omega \end{bmatrix} \tag{3.2}$$

where $\dot{\mathbf{q}}(t) = [\dot{q}_1,\ \dot{q}_2,\ \dot{q}_3]^T$ is the vector of wheel velocities, which is equal to the wheel's radius multiplied by the wheel's angular velocity. L_w gives the distance between each wheel and the center of the robot (point R in Figure 3.2) and δ refers to the wheel orientation in the body frame. As the motor's voltage and current are magnitude-limited, the maximum wheel velocity is limited by $\dot{q}_{\max}$, i.e., $|\dot{q}_i| \leq \dot{q}_{\max}$, where $i = 1, 2, 3$. It is worth noting that the transformation matrices in the above kinematic models are all full rank, which denotes that the translation and rotation of the omnidirectional mobile robot are decoupled, and guarantees the separate control of these two movements [31].

Related work on motion control of omnidirectional mobile robots is reviewed as follows. Oubbati et al. [184] replaced the PID controllers for the motors with a recurrent neural network to convert the desired wheel velocities to PWM commands. Purwin and D'Andrea [191] presented a trajectory generation algorithm, which computes the minimum time trajectory from a given initial state to a given final state while taking limited friction and weight transfer into account. Watanabe [230] reviewed the omnidirectional mobile robot using a variety of mechanisms. The use of resolved acceleration control method, PID method, fuzzy model method, and stochastic fuzzy servo method have been highlighted. Liu et al. [147] implemented a method called trajectory linearization control, which is based on linearization along the desired trajectory and inversion of the dynamics. Kalmar-Nagy et al. [109] developed a trajectory generation algorithm which computes a minimum time path based on the dynamics of the vehicle and the motor characteristics. Velasco-Villa et al. [225] addressed the path-tracking problem on an omnidirectional mobile robot subject to transport delays. Xiang and Zell [142] proposed a control method based on the inverse input-output linearized kinematic model. Recently,

a general kinematic model of an N Swedish wheeled vehicle was derived and analyzed by Indiveri [100]. He addressed the trajectory tracking motion control problem in the presence of actuator velocity saturation. Furthermore, some researchers [59, 217, 233] have considered slipping motion between the wheels and motion surface and included it into a dynamic model.

3.2.2 Unicycle Mobile Robots

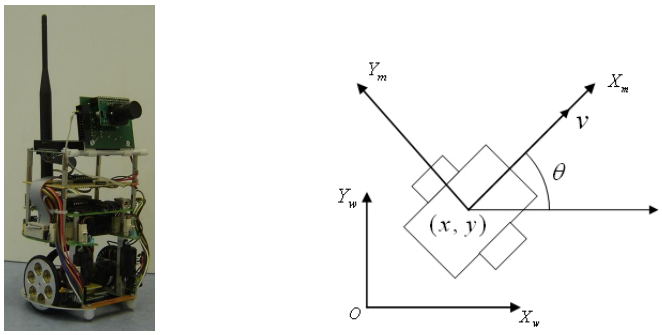
The unicycle robot, shown in Figure 3.6(a), has two identical parallel rear wheels, which are controlled by two independent motors on the same axle and one caster wheel. It is assumed that the center of mass of the mobile robot is located in the middle of the axis connecting the rear wheels. A robot with this kind of wheel configuration has an underlying nonholonomic property that restricts the mobility of the robot in the sideways direction. This adds to the complexity of the motion control problem.

A simple kinematic model of a unicycle mobile robot is the following:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{3.3}$$

where (x, y) indicates the position of the robot center in the world frame (X_w, Y_w) and θ is the heading angle of the robot (angle between the X_w-axis and the axis of the robot X_m). v and ω stand for the linear and angular velocities, respectively. The wheel velocities can be derived by $v_r = v + b\omega$, $v_l = v - b\omega$, where v_r and v_l are the velocities of right and left wheels, whose unit is m/s, respectively, and b is the length of the axis from the robot center to the wheel.

The robot controller is an ATMEGA644 microprocessor with 64 KB flash program memory, 16 MHz clock frequency, and 4 KB SRAM. The robot orientation is measured

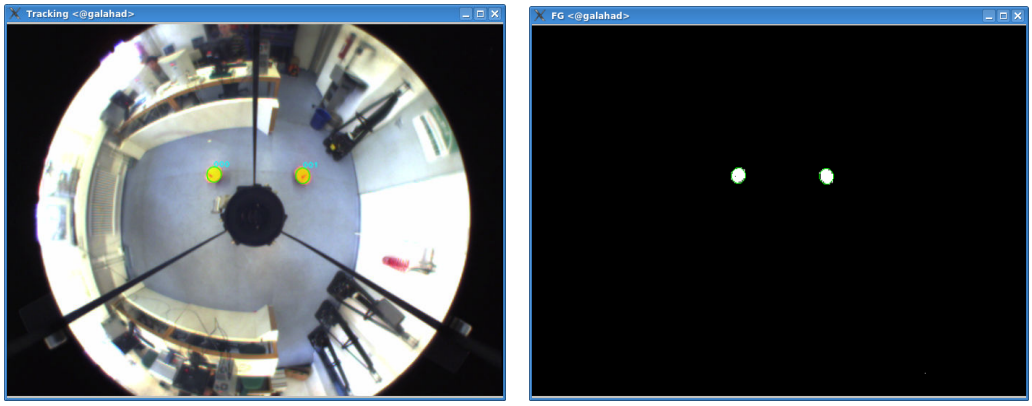


(a) (b)

Figure 3.6: (a) A unicycle mobile robot (12 cm diameter), and (b) coordinate frames of a unicycle mobile robot.

by a Devantech CMPS03 compass. The robot is identified by a colored circle placed on the top of its platform. Using omnidirectional images from a camera looking down upon the robot's workplace, the robot position can be estimated through color segmentation and a Kalman filter. This tracking software was implemented based on the OpenCV library [48]. Figure 3.7 shows images from the robot detection and tracking software. White regions in Figure 3.7(b) involve the colored circle segmented from an omnidirectional image. This position information is incorportated into the motion control structure (see Figure 3.8), used in our real-world experiments.

Motion control of this kind of robots has been, and still is, the subject of numerous research studies. In particular, nonholonomy constraints associated with these systems have motivated the development of highly nonlinear control techniques. Nonholonomic constraints mean the perfect rolling constraints without longitudinal or lateral slipping of



(a) (b)

Figure 3.7: Robot detection and tracking software: (a) an original omnidirectional image (b) blobs of color-segemented robots.

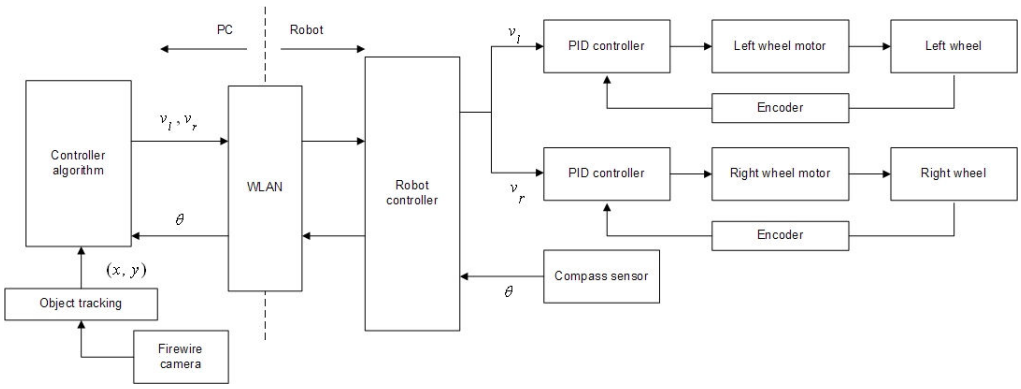


Figure 3.8: Block diagram illustrates the control structure of the unicycle mobile robot.

the wheels. In other words, they are the constraints on the velocity of the system which cannot be integrated into position constraints. Brockett's theorem [27] states that nonholonomic systems cannot be stabilized via smooth time-invariant state feedback. The asymptotic stabilization of fixed points is mainly achieved via discontinuous feedback and/or continuous time-varying feedback. This has given rise recently to an abundant literature dealing with the derivation of planning and control algorithms especially dedicated to either kinematic or dynamic models of unicycle, trailer-like, or car-like wheeled mobile robots. Usually, they are also classified as underactuated systems because of less control inputs than system states. Many nonlinear techniques have been proposed in the literature to solve these basic motion problems, e.g., dynamic feedback linearization [181], sliding mode control [212], backstepping techniques [2, 211], intelligent control [73], optimal control [88], passivity based approaches [108], etc., to name some. An extensive review of nonholonomic control problems can be found in [127].

3.3 Software Frameworks

In this section, we present two software architectures. The first one was designed for the Attempto Tübingen Robot Soccer Team [93] to play robot soccer in RoboCup tournaments. The framework was divided into several functional processes that provide data to other processes according to a client/server architecture. Figure 3.9 [93] gives an overview of the data flow in the software framework. The framework can be divided into three functional layers, the low-level data layer containing all processes that provide the low-level access to sensor systems and serve basic sensor data, the intermediate layer with processes that pre-process the basic sensor data and extract features like landmarks and obstacles, and finally the high-level layer that includes the collection of the pre-processed data in a model of the environment and the high-level control of the robot. Details of each layer are given in [93], including graphical user interface and tools.

The second framework is the CARMEN [167, 168] toolkit, which is an open-source software for mobile robot control. It offers a distributed collection of modules designed to provide basic navigation primitives including: base and sensor control, logging, obstacle avoidance, localization, path planning, and mapping. Modules communicate with each other over an interprocess communication protocol (IPC). This software toolkit has been adopted to our rebuilt omnidirectional mobile robots. Figure 3.10(a) shows the *robotgui* module, in which the robot is displayed as a circle with a small line designing the front and also laser information. This program provides a simple graphical interface for the robot, allowing direct motion control and a display of current sensor information. The *navigatorgui* module shown in Figure 3.10(b) provides a graphical interface which shows the robot's position and destination on the pre-built map and allows the setting of the current position and orientation, and the selection of destinations. Our software development including graphical user interface and tools to visualize and to control both omnidirectional mobile robots and unicycle mobile robots is under construction.

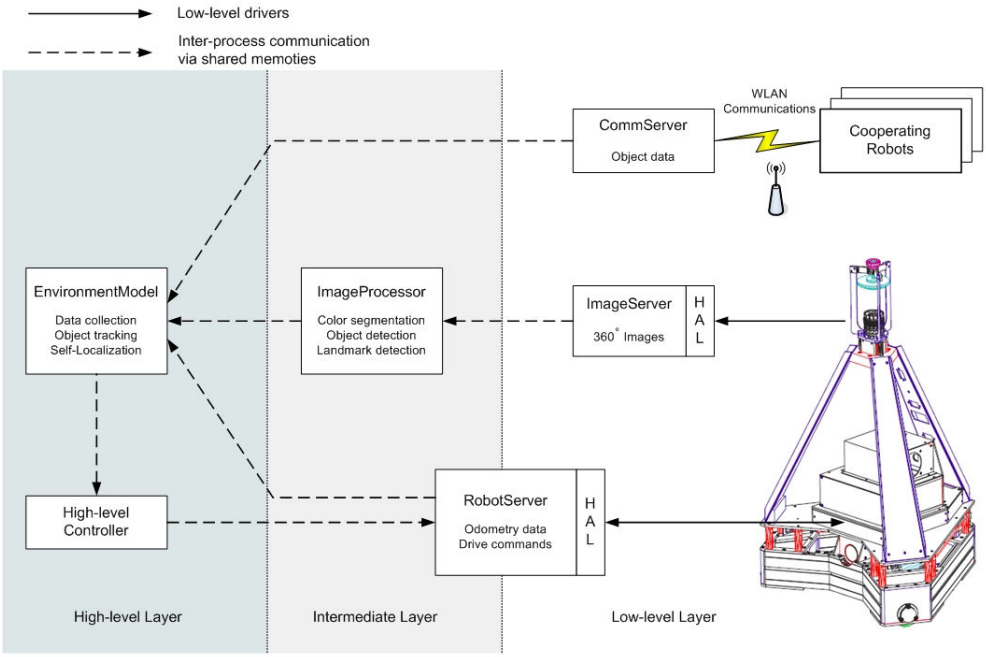
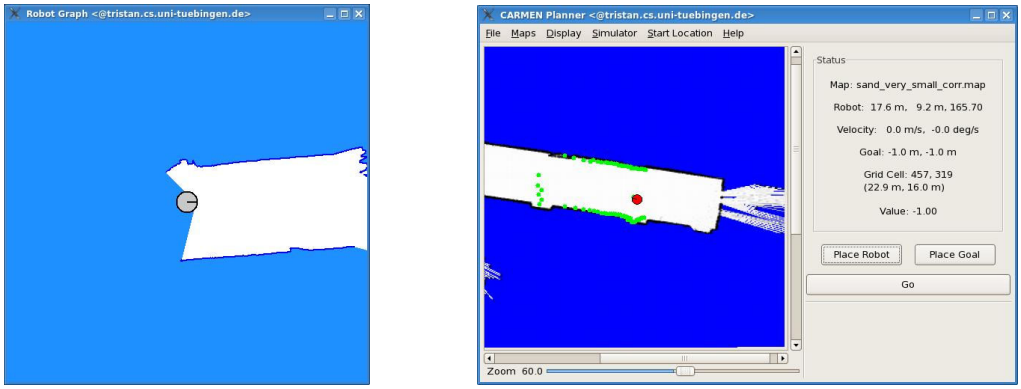


Figure 3.9: Block diagram illustrates an overview of the RoboCup software framework [93].



(a) (b)

Figure 3.10: CARMEN toolkit (a) *robotgui* module, and (b) *navigatorgui* module.

3.4 Summary

In this chapter, we described the infrastructure of our robot systems, in which our real-world experiments are conducted. Decentralization and homogeneity are employed in the control laws proposed in Section 5.3, 5.4, and 5.5, and in Chapter 6, while centralization

and heterogeneity are utilized in the control laws proposed in Subsection 5.5.5. Global communication is implemented in Section 5.3 and Subsection 5.5.5, whereas local communication with a connectivity assumption is carried out in Section 5.4, in Section 5.5, and in Chapter 6.

Omnidirectional mobile robots are employed in Section 5.3 and in Section 5.4. Unicycle mobile robots are used in Section 5.5 and Chapter 6. The cooperative strategy for both kinds of robots is given in Subsection 5.5.5.

Chapter 4

Path Following Control

As mentioned in Chapter 1, the problem of motion control addressed in the literature can be roughly classified into three groups, for which either kinematic or dynamic solutions are derived: 1) point stabilization, which refers into the problem of steering a vehicle to a final target point with a desired orientation, 2) trajectory tracking, which requires a vehicle to track a time-parameterized reference curve, and 3) path following, which aims at forcing a vehicle to converge to and to follow a desired spatial path without any temporal specifications [171].

In this dissertation, we explore the path following problem of a single mobile robot and of multiple mobile robots, where the objective is to be *on the path* rather than at a certain point at a particular time. Even though less attention is drawn to this problem in the literature, it offers some advantages over trajectory tracking in some cases.

Typically, trajectory tracking problems for autonomous vehicles are solved by designing control laws that make the vehicles track predetermined feasible trajectories, i.e., trajectories that specify the time evolution of the position, orientation (spatial dimension), as well as the linear and angular velocities (temporal dimension) [3]. However, this approach suffers from the drawback that usually the vehicles' dynamics exhibit complex nonlinear terms and significant uncertainties, which make the task of computing a feasible trajectory difficult. Also, in the presence of tracking errors, the controller attempts to make the outputs catch up with the time-parameterized desired outputs. This may lead to closed loop performance difficulties and too large control signals. One approach to eliminate such problems is to use a path following controller instead of a tracking controller.

Path following has recently been formulated to replace the standard trajectory tracking as it is more suitable for certain applications. In particular, Aguiar et al. [3] highlighted a fundamental difference between the path following and the standard trajectory tracking by demonstrating that performance limitations due to unstable zero-dynamics can be removed in the path following problem. As illustrated in [209], the path following control also provides natural settings for many engineering applications. With path following, the time-dependence of the problem is removed, smoother convergence to the path is achieved, and the control signals are less likely pushed into saturation when compared to trajectory tracking.

Path following problems [3] are primarily concerned with the design of control laws that steer an object (robot arm, mobile robot, ship, aircraft, etc.) to reach and to follow a geometric path, i.e., a manifold parameterized by a continuous scalar s (called a geometric task), while a secondary goal is to force the object moving along the path to satisfy some additional dynamic specifications (called a dynamic assignment task). This dynamic behavior is further specified via time, speed, or acceleration assignments [209]. This setting is more general than the common trajectory tracking problem, in which the path variable s is left as an extra degree of freedom for the secondary goal.

To determine the path variable s, Micaelli and Samson [161] used a numerical projection from the current state onto the path. This point plays the role of a virtual vehicle that should be tracked by the real vehicle. The main problem of a numerical projection is that singularities occur when the distance to a path is not well-defined. This problem can be solved by controlling explicitly the timing law for s to be tracked along the path.

Usually, the choice of the timing law for the path variable s has the following desired feature: When the path following error is large, the virtual vehicle will wait for the real one, when the path error is small, the virtual vehicle will move along the path at the speed close to the desired speed assignment. This feature is suitable in practice because it avoids the use of a high gain control for large path error signals. Diaz del Rio et al. [58] proposed a method, called error adaptive tracking, in which the tracking adapts to the errors. They defined the function of $\dot{s}$ as $\dot{s} = g(e)$, where e is the distance error. They also proposed $\dot{s} = g(t,e)$ in order to preserve time determinism of trajectory tracking. In [4], Aicardi et al. specified the target motion to be a continuous radial function centered on the ellipsoidal domain, which attains its maximum value equal to the largest admissible one in correspondence with the origin, and a null minimum value in correspondence with the border of the ellipsoid itself. Soeanto et al. [211] controlled explicitly the rate of progression of a virtual vehicle by modeling the kinematic equations of motion with respect to the Frenet frame. A virtual vehicle concept was also employed by Egerstedt et al. [67], whose control law ensures global stability by determining the dynamics of the parameterized reference point $\dot{s}$. The motion of the virtual vehicle on the desired path is governed by a differential equation containing error feedback.

Besides using the path parameterization strategy, various approaches have also been proposed for path tracking, such as pure-pursuit [10], sliding-mode control [237], nonlinear proportional control [242], and vector pursuit [234]. Moreover, the control design in vision-based path following allows a mobile robot equipped with a camera to track a line on the ground using visual feedback [43, 49, 149].

In this work, we design an MPC controller to receive the reference point on the path and to generate an optimal velocity of a virtual vehicle. This controller takes input constraints into account. In contrast, other control techniques are usually designed under the assumption that there are no limitations on inputs or states. The main drawback of nonlinear model predictive control (NMPC) schemes is related to its computational burden. With the development of increasingly faster processors and efficient numerical algorithms, however, the use of an NMPC controller in faster applications (e.g., wheeled

mobile robots) becomes possible.

Related work on motion control using MPC is first briefly reviewed in the next section and three experimental scenarios are then presented. All of the proposed algorithms are based on MPC schemes which differ in strategies and models. The first experiment shows a comparison between trajectory tracking and path following for an omnidirectional mobile robot using NMPC [144]. The path following control for a linearized model of an omnidirectional mobile robot is carried out in the second experiment [113]. In this case, a time varying convex quadratic optimization problem is formulated and solved at each time step, leading to the reduction of the computational burden. The last experiment compares trajectory tracking and path following of a unicycle mobile robot, including obstacle avoidance and a time-parameterized penalty [117].

4.1 Related Work on Motion Control Using MPC

In the field of mobile robotics, MPC approaches to path tracking seem to be very promising because the reference path is known beforehand. In the literature, most MPC controllers use a linear model of mobile robot kinematics to predict future system outputs. Ollero and Amidi [179] used generalized predictive control (GPC) to solve the path following problem to obtain an appropriate steering angle taking into account the vehicle velocity. A GPC approach with a Smith predictor was presented by Normey-Rico et al. [174] to cope with an estimated system time delay. In [174, 179], it was assumed that the control acts only in the angular velocity, while the linear velocity is constant. Hence, an input-output linear model is used to compute the distance between the robot and a reference path. Lages and Alves [132] used a successive linearization approach, yielding a linear, time-varying description of the system that can be controlled through linear MPC. Then, by considering the control inputs as the decision variables, it is possible to transform the optimization problem to be solved at each sampling time into a QP problem. Since the latter is a convex problem, the QP problem can be easily solved by numerically robust solvers, leading to global optimal solutions. Klančar and Škrjanc [125] applied an MPC controller based on a linearized error dynamics model obtained around the reference trajectory. The analytic control law is explicitly obtained without using any optimization solver. Iterative model predictive control was used by Wen and Sooyong [231]. Their approach is similar to nonlinear MPC but does not consider optimization or constraints explicitly; instead it uses a gradient based algorithm to reduce the predicted state error after a fixed number of trajectory points. Vougioukas [228] presented a reactive trajectory tracking controller based on nonlinear MPC, along with an iterative gradient descent algorithm for its real-time implementation. Seyr and Jakubek [205] solved a nonholonomic control problem consisting of nonlinear predictive control in conjunction with linear state space control. Falcone et al. [70] implemented an MPC-based approach for controlling an active front steering system in an autonomous vehicle. They presented two approaches with different computational complexities. In

the first approach, the MPC problem is formulated by using a nonlinear vehicle model. The second approach is based on successive online linearization of the vehicle model. A suboptimal MPC controller based on successive online linearization of the nonlinear vehicle model is designed for the resulting linear time-varying (LTV) system. Moreover, the nonlinear version of MPC schemes for a trajectory tracking problem was proposed by Gu and Hu [88] and by Hedjar et al. [92]. A neural network also helps to solve the optimization problem. Yang et al. [237] solved the path following problem by using a neural network to predict the future behavior of a car-like robot. The modeling errors are corrected online with the neural network model. Essen and Nijmeijer [69] developed a nonlinear MPC algorithm in a state-space representation, which is applied to both problems of point stabilization and trajectory tracking. A modified cost function to be minimized was proposed. Gu and Hu [87] proposed the nonlinear predictive controller scheme for a path-tracking problem, where the neural network is employed to model the nonlinear kinematic behavior of a mobile robot. We can classify all these publications as trajectory tracking [88, 92, 125, 132, 205] and path following [70, 87, 174, 179, 237].

The key points of our proposed algorithms based on MPC schemes include (i) in path following control, we propose another possibility to obtain an optimal velocity of a virtual vehicle to be followed along a path in order to overcome stringent initial condition constraints [211], (ii) the comparison between path following and trajectory tracking on both an omnidirectional mobile robot and a unicycle mobile robot is drawn, (iii) compared to other path following controllers in the literature, input constraints are handled straightforwardly in the optimization problem so that the robot can travel safely at a high velocity, (iv) future information is utilized to improve the system performance because the reference path is known beforehand, and (v) in case of a unicycle mobile robot, we achieve smooth convergence towards the reference including time constraints and obstacle avoidance.

4.2 Path Following Control of an Omnidirectional Robot

In this section, we address the path following control problem of an omnidirectional mobile robot. We also compare the results with trajectory tracking. The comparison illustrates the differences between these two fundamental motion control problems. We first formulate error dynamic models, which represent the errors between the real robot states and its desired states according to a reference path. We then analyze and design NMPC controllers with guaranteed stability to drive the errors to zero, and we also show the feasibility of applying NMPC to a physical omnidirectional mobile robot.

4.2.1 Problem Formulation

The path following and trajectory tracking problems are illustrated in Figure 4.1 and Figure 4.2, respectively. Point Q is the desired reference point of the robot. In trajectory

tracking, point Q is time-parameterized, whereas in path following, the motion of point Q is controlled by the velocity of the virtual vehicle moving along the path, which is parameterized by the path variable s.

We begin with the path following problem. The error state vector $\mathbf{x}_e$ can be defined as follows:

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix} \tag{4.1}$$

where $[x,\ y,\ \theta]^T$ is the robot state vector and $[x_d,\ y_d,\ \theta_d]^T$ is the reference state vector. By using the error state and its kinematic model (3.1), the path following control problem is converted into a regulation problem. The error state dynamic model with respect to the rotated coordinate frame (4.1) can be derived as follows:

$$\begin{aligned} \dot{x}_e &= y_e \kappa(s)\dot{s} - \dot{s} + u_R \cos\alpha_e \\ \dot{y}_e &= -x_e \kappa(s)\dot{s} + u_R \sin\alpha_e \\ \dot{\alpha}_e &= \dot{\theta}_t - \kappa(s)\dot{s} \end{aligned} \tag{4.2}$$

where α_e represents the angular error between the robot moving direction θ_t and the path tangent direction θ_d. $\kappa(s)$ denotes the path curvature at point Q. u_R refers to the translation velocity. It is noticed from (4.2) that u_R is decoupled from $\dot{s}$ and $\dot{\theta}_t$, because the errors can remain on the equilibrium $(\mathbf{x}_e = 0)$ when $\dot{s}$ approaches u_R. Therefore, the objective is to find suitable values of $\dot{s}$ and $\dot{\theta}_t$ driving the errors $\mathbf{x}_e$ to zero with u_R to be steered to track any desired velocity.

In trajectory tracking, the desired translational velocity of the robot cannot be chosen freely, but it is determined by the reference trajectory parameterized through time t. Therefore, the trajectory tracking problem requires the robot to track the specific position and velocity defined at each given time step. If we control the robot orientation to track the tangent direction of the reference trajectory, i.e., $v_d = 0$, and use the robot coordinate system as the reference frame, the following error dynamic model of the trajectory

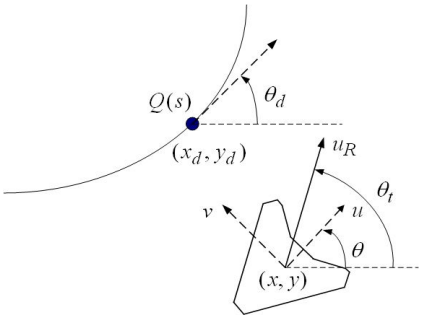


Figure 4.1: Illustration of the path following problem.

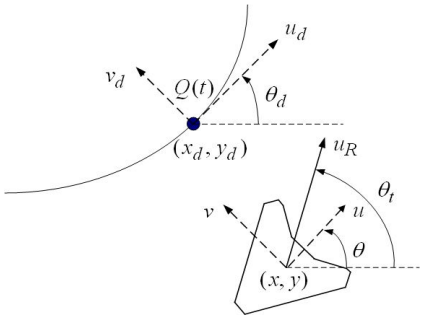


Figure 4.2: Illustration of the trajectory tracking problem.

tracking problem can be deduced as

$$\begin{aligned}\dot{x}_e &= \omega y_e - u + u_d \cos\theta_e \\ \dot{y}_e &= -\omega x_e - v + u_d \sin\theta_e \\ \dot{\theta}_e &= \omega_d - \omega\end{aligned} \tag{4.3}$$

where the error vector $\mathbf{x}_e$ is given with respect to the robot coordinate system and θ_e represents the orientation error between the robot orientation θ and the trajectory tangent direction θ_d. With respect to the error model (4.3), the objective of trajectory tracking is to choose suitable robot inputs u, v and ω, such that the tracking error $\mathbf{x}_e$ converges to zero.

4.2.2 Controller Design

We first design the path following controller based on NMPC. As the translation and rotation of an omnidirectional mobile robot are completely decoupled, the angular error α_e in (4.2) can be directly controlled. Therefore, the path following problem can be solved by finding suitable $\dot{s}$ and α_e. The robot orientation can be seen as an additional degree of freedom, i.e., we can drive the robot orientation θ to any desired orientation θ_d at the same time. Defining the orientation error θ_e and a new vector $\mathbf{u}_e = [u_1,\ u_2,\ u_3]^T$, we obtain the following error kinematic model whose equilibrium is reached if $\mathbf{x}_e = 0$ and $\mathbf{u}_e = 0$:

$$\dot{\mathbf{x}}_e = \begin{bmatrix}\dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e\end{bmatrix} = \begin{bmatrix}0 & \kappa(s)\dot{s} & 0 \\ -\kappa(s)\dot{s} & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_e \\ y_e \\ \theta_e\end{bmatrix} + \begin{bmatrix}u_1 \\ u_2 \\ u_3\end{bmatrix} \tag{4.4}$$

with

$$\mathbf{u}_e = \begin{bmatrix}u_1 \\ u_2 \\ u_3\end{bmatrix} = \begin{bmatrix}-\dot{s} + u_R\cos\alpha_e \\ u_R \sin\alpha_e \\ \omega_d - \omega\end{bmatrix} \tag{4.5}$$

where ω_d denotes the desired rotation velocity.

As the errors are required to converge to zero, we select the following cost function for (2.18):

$$F(\mathbf{x}_e, \mathbf{u}_e) = \mathbf{x}_e^T Q \mathbf{x}_e + \mathbf{u}_e^T R \mathbf{u}_e \tag{4.6}$$

with positive weight matrices $Q = \mathrm{diag}(q_{11}, q_{22}, q_{33})$ and $R = \mathrm{diag}(r_{11}, r_{22}, r_{33})$. To guarantee the control stability, the following Lyapunov function can be chosen as the terminal penalty:

$$V(\mathbf{x}_e(t+T_p)) = \tfrac{1}{2}\mathbf{x}_e(t+T_p)^T \mathbf{x}_e(t+T_p) \tag{4.7}$$

where $\mathbf{x}_e(t+T_p) = [x_{eT_p},\ y_{eT_p},\ \theta_{eT_p}]^T$ denotes the terminal state.

If we choose the terminal feedback controller $\mathbf{u}^L = [u_1^L,\ u_2^L,\ u_3^L]^T$ as:

$$u_1^L = -\alpha x_{eT_p} \qquad u_2^L = -\beta y_{eT_p} \qquad u_3^L = -\gamma\theta_{eT_p} \tag{4.8}$$

with parameters $\alpha \geq 0$, $\beta \geq 0$, and $\gamma \geq 0$, the stability condition becomes

$$\begin{aligned}(\dot{V}+F)(\mathbf{x}_e,\mathbf{u}^L) &= x_{eT_p}\dot{x}_{eT_p} + y_{eT_p}\dot{y}_{eT_p} + \theta_{eT_p}\dot{\theta}_{eT_p} + F(t+T_p) \\ &= x_{eT_p}u_1^L + y_{eT_p}u_2^L + \theta_{eT_p}u_3^L + F(t+T_p) \\ &= x_{eT_p}u_1^L + y_{eT_p}u_2^L + \theta_{eT_p}u_3^L + q_{11}x_{eT_p}^2 + q_{22}y_{eT_p}^2 \\ &\quad + q_{33}\theta_{eT_p}^2 + r_{11}u_1^{L^2} + r_{22}u_2^{L^2} + r_{33}u_3^{L^2} \leq 0\end{aligned} \tag{4.9}$$

for all $\mathbf{x}_e \in \Omega$.

To satisfy the stability condition, the following requirements are necessary:

$$\begin{aligned}\alpha - q_{11} - \alpha^2 r_{11} &\geq 0 \\ \beta - q_{22} - \beta^2 r_{22} &\geq 0 \\ \gamma - q_{33} - \gamma^2 r_{33} &\geq 0\end{aligned} \tag{4.10}$$

The requirements (4.10) are used to define the terminal region Ω. Furthermore, the outputs of feedback control (4.8) have to satisfy the system constraints, which are the bounded wheel velocities in our case. With the definitions of the input vector $\mathbf{u}_e$, the terminal feedback controller $\mathbf{u}^L$, and the robot's kinematic model given in (3.1) and (3.2), we obtain the second part of terminal constraints as:

$$-\begin{bmatrix}\dot{q}_{\max}\\ \dot{q}_{\max}\\ \dot{q}_{\max}\end{bmatrix} \leq \begin{bmatrix}\cos(\theta+\delta) & \sin(\theta+\delta) & L_w\\ -\cos(\theta-\delta) & -\sin(\theta-\delta) & L_w\\ \sin\theta & -\cos\theta & L_w\end{bmatrix}\begin{bmatrix}-\alpha x_{eT_p}+\dot{s}\\ -\beta y_{eT_p}\\ -\gamma\theta_{eT_p}+\omega_d\end{bmatrix} \leq \begin{bmatrix}\dot{q}_{\max}\\ \dot{q}_{\max}\\ \dot{q}_{\max}\end{bmatrix} \tag{4.11}$$

Now we move on to trajectory tracking, with the same idea of solving the path following problem: We transfer the error kinematic model of the trajectory tracking problem (4.3) by introducing the following control variables u_1, u_2, and u_3:

$$\dot{\mathbf{x}}_e = \begin{bmatrix}\dot{x}_e\\ \dot{y}_e\\ \dot{\theta}_e\end{bmatrix} = \begin{bmatrix}0 & \omega & 0\\ -\omega & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_e\\ y_e\\ \theta_e\end{bmatrix} + \begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix} \tag{4.12}$$

where

$$\mathbf{u}_e = \begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix} = \begin{bmatrix}u_R\cos\theta_e - u\\ u_R\sin\theta_e - v\\ \omega_d - \omega\end{bmatrix}. \tag{4.13}$$

When we select the cost function, the terminal penalty and the feedback controller with the same forms as in (4.6), (4.7), and (4.8), respectively. The corresponding terminal constraints can be deduced by following the same process.

4.2.3 Experimental Results

Two kinds of experiments were performed to test the above NMPC method: One was the path following control with a constant desired velocity $u_R = 1.0$ m/s. Here, the desired robot orientation θ_d was the path tangent direction. The other was the trajectory tracking control, where the desired robot orientation θ_d was changing with respect to time t. An eight-shaped curve was adopted as the reference path and trajectory, because its geometrical symmetry and sharp changes in curvature render the test challenging. With respect to the world coordinate system, the coordinates of the reference path and trajectory are given by

$$x_d(\phi) = 1.8\sin(0.4\phi), \quad y_d(\phi) = 1.2\sin(0.8\phi), \tag{4.14}$$

where $\phi = t$ in case of trajectory tracking, while this reference was numerically parameterized by the path variable s in case of the path following problem. The same

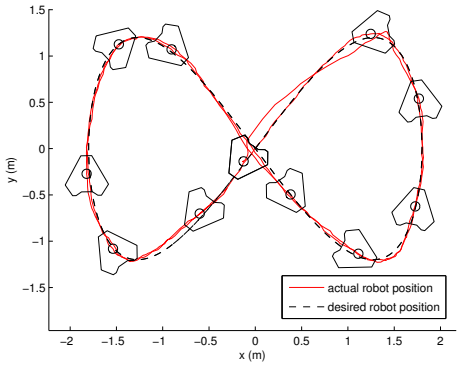


Figure 4.3: The reference trajectory and the robot trajectory.

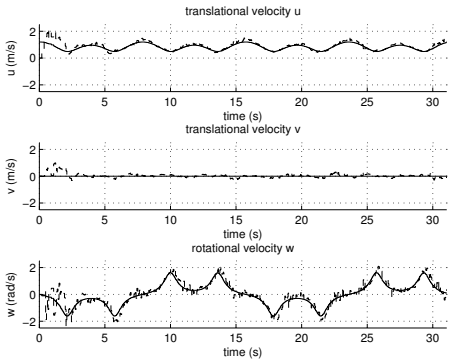


Figure 4.4: Velocities with respect to the robot frame.

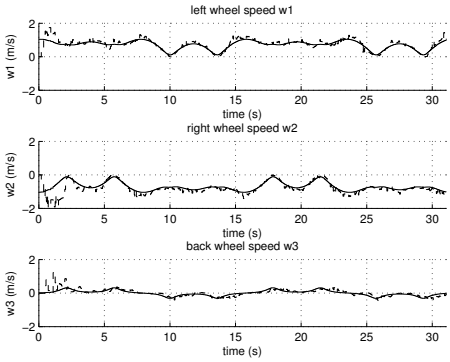


Figure 4.5: Wheel velocities.

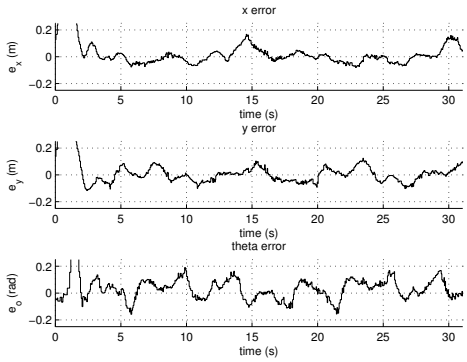


Figure 4.6: Tracking errors.

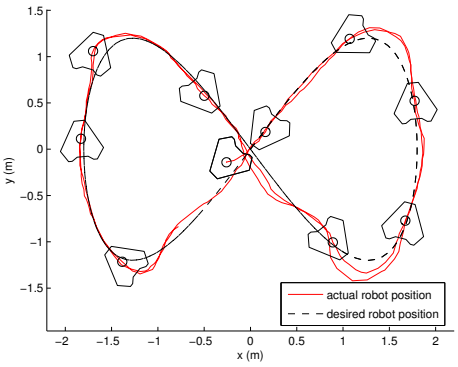


Figure 4.7: The reference path and the robot path.

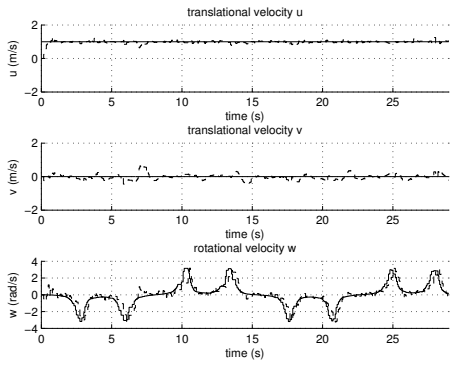


Figure 4.8: Velocities with respect to the robot frame.

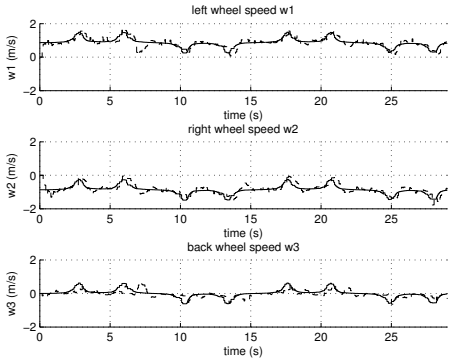


Figure 4.9: Wheel velocities.

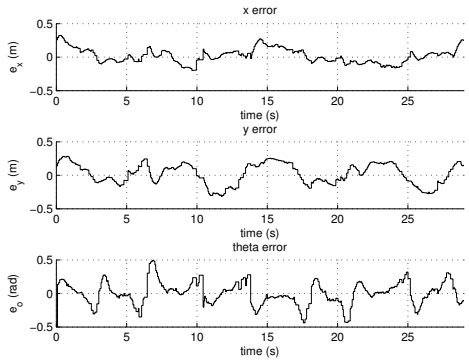


Figure 4.10: Following errors.

parameters for both experiments were chosen as follows: $Q = \text{diag}(0.5, 0.5, 0.5)$, $R = \text{diag}(0.1, 0.1, 0.1)$, $\alpha = \beta = \gamma = 2$, $\tau = 0.07$ s, $N_p = N_c = 3$ steps.

Figure 4.3 and Figure 4.7 illustrate the results of trajectory tracking and path following experiments. Figure 4.4 and Figure 4.8 show that the desired translation velocity is changing according to the trajectory's curvature, but remaining constant in the path following problem, which increases the difficulty in following the sharp turning part of the given path. Figure 4.5 and Figure 4.9 show that the controller guarantees that the required wheel velocities are under the boundaries. Figure 4.6 and Figure 4.10 show that the NMPC solved these real-time motion control problems with good performance, because at most time steps, the distance errors are less than 0.15 m and 0.3 m, the orientation errors are less than 0.2 rad and 0.5 rad in the trajectory tracking and path following tasks, respectively.

4.3 Linearized Path Following Control of an Omnidirectional Robot

Since one possibility to reduce the computational time of solving nonlinear optimization problems is to use linearization techniques, the linearized version of the path following problem is considered in this section. With this linearized time-varying system, the optimization problem can be transformed into a QP problem. Since it turns into a convex problem, solving the QP problem results in global optimal solutions. After solving the QP problem at each time instant, an optimal velocity of a virtual vehicle can be obtained. We also integrate input constraints into the QP problem so that we can steer the robot safely with a high forward velocity. Furthermore, the velocity profile, which the robot's forward velocity will track, is shaped to comply with the robot and path constraints and is employed along the prediction horizon of the MPC controller. Hence, the MPC controller exploits future information to generate optimal control inputs at each time instant. This linear MPC controller is computationally effective and can be easily used in fast real-time implementations.

4.3.1 Problem Formulation

Since the constraints in a QP problem have to be linear, we need to decouple the maximum rotation from the maximum translation. First, we substitute (3.2) into (3.1) resulting in

$$\dot{\mathbf{x}}(t) = P(\theta)\dot{\mathbf{q}}(t) \tag{4.15}$$

with

$$P(\theta) = \frac{2}{3}\begin{bmatrix} \cos(\theta+\delta) & -\cos(\theta-\delta) & \sin\theta \\ \sin(\theta+\delta) & -\sin(\theta-\delta) & -\cos\theta \\ \frac{1}{2L_w} & \frac{1}{2L_w} & \frac{1}{2L_w} \end{bmatrix}.$$

Since translations and rotations are coupled via θ in (4.15), we employ a similar method proposed in [109] to decouple the θ-equation from those of the translational ones. For a given θ, the linear transformation $P(\theta)$ maps the cube $\mathcal{Q}(t) = \{\dot{\mathbf{q}}(t) \mid |\dot{q}_i(t)| \leq \dot{q}_{i,\max}\}$ into the tilted cuboid $P(\theta)\mathcal{Q}(t)$. The matrix $P(\theta)$ can be decomposed into a product of a rotation and a θ-independent linear transformation

$$P(\theta) = R_z(\theta)P(0) \tag{4.16}$$

where

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P(0) = \begin{bmatrix} \sqrt{3}/3 & -\sqrt{3}/3 & 0 \\ 1/3 & 1/3 & -2/3 \\ 1/(3L_w) & 1/(3L_w) & 1/(3L_w) \end{bmatrix}$$

with $\delta = \frac{\pi}{6}$ rad.

The linear transformation $P(0)$ maps the cube $\mathcal{Q}(t)$ (see Figure 4.11) into the tilted cuboid $P(0)\mathcal{Q}(t)$ (see Figure 4.12) with a diagonal $\dot{q}_3 \leq 9.7436$ rad/s, calculated by using $L_w = 0.195$ m and $|\dot{q}_i(t)| \leq 1.9$ m/s, along the ω axis. The transformation $R_z(\theta)$ then rotates this cuboid about the ω axis. The problem is to find the solid of revolution that is the intersection of all possible rotations $R(\theta)_z P(0)\mathcal{Q}(t)$ of the cuboid. This solid of revolution is characterized by (see Figure 4.13 and [109] for details)

$$\dot{x}^2(t) + \dot{y}^2(t) \leq r^2(\omega) \tag{4.17}$$

where the radius is

$$r(\omega) = \frac{9.7436 - |\omega|}{5.1283}. \tag{4.18}$$

Equation (4.17) shows the relationship between translational and rotational velocities. For example, in our experiments we set $\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \leq u_{\max} = 1.315$ m/s as the maximum forward velocity and then the maximum allowable rotational velocity becomes $|\omega| \leq \omega_{\max} = 3$ rad/s. When the desired rotational velocity is larger than this maximum allowable value, the forward velocity needs to be decreased in order to achieve the rotational velocity. Using this relationship, we ensure that the maximum wheel velocity constraints will not be violated. These constraints on translational and rotational velocities can be either self-imposed due to desired behaviors and safety concerns or physical due to actual limitations such as currents and voltages in the motors [16].

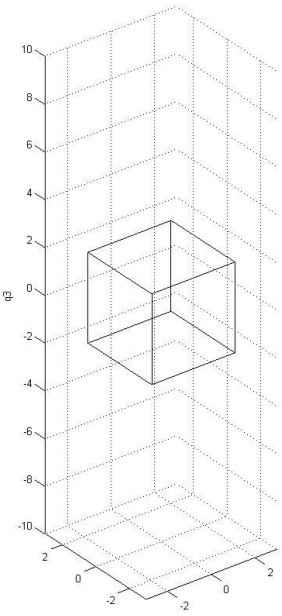


Figure 4.11: Illustration of the cube defined by $\mathcal{Q}(t) = \{\dot{\mathbf{q}}(t) \mid |\dot{q}_i(t)| \leq \dot{q}_{i,\max}\}$.

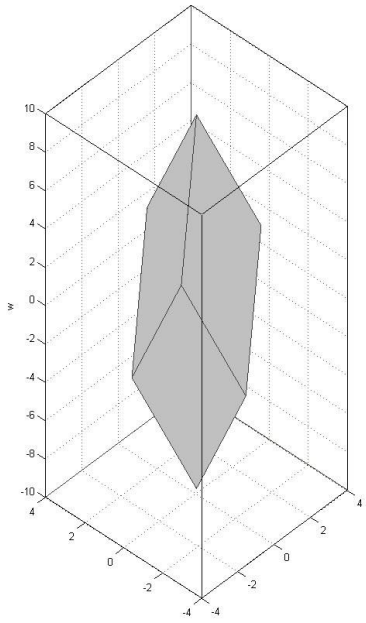


Figure 4.12: Illustration of the tilted cuboid $P(0)\mathcal{Q}(t)$.

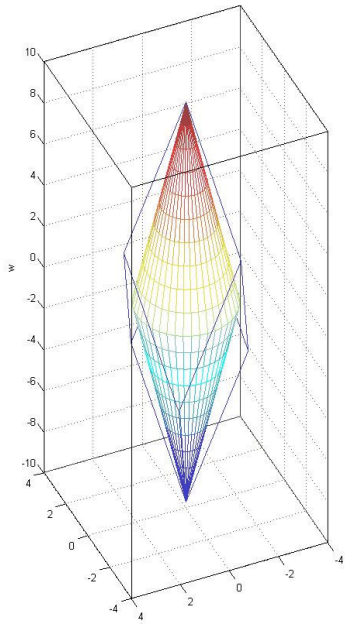


Figure 4.13: Illustration of the solid of revolution that is defined as the intersection of all possible rotations $R(\theta)_z P(0)\mathcal{Q}(t)$ of the cuboid.

To apply MPC schemes, we need to derive the system model. In this section, the kinematic model of an omnidirectional mobile robot can be formulated with respect to a Frenet frame moving along the reference path. This frame plays the role of the body frame of a virtual vehicle that must be followed by the real vehicle. In our path following problem shown in Figure 4.14, we let the forward velocity u_R track a desired velocity profile, while the velocity of a virtual vehicle $\dot{s}$ converges to u_R. The error state vector $\mathbf{x}_e$ between the robot state vector $\mathbf{x}$ and a virtual vehicle's state vector $\mathbf{x}_r = [x_r, y_r, \theta_r, \theta_b]^T$, where θ_b is the desired orientation and θ_r is the tangent angle to the path, can be expressed in the frame of the path coordinate as follows:

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 & 0 \\ -\sin\theta_r & \cos\theta_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \theta_t - \theta_r \\ \theta - \theta_b \end{bmatrix} \tag{4.19}$$

Since translations and rotations of omnidirectional mobile robots can be controlled separately, we can rewrite (3.1) by decoupling translation and rotation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta}_t \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_R \cos\theta_t \\ u_R \sin\theta_t \\ \Phi u_R \\ \omega \end{bmatrix} \tag{4.20}$$

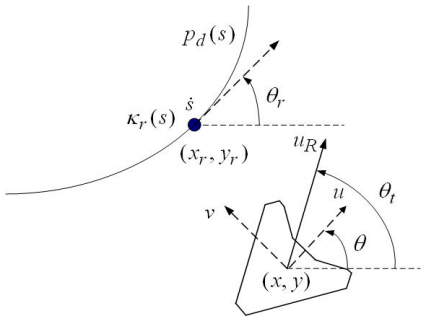


Figure 4.14: Illustration of the path following problem.

where Φ is the curvature. The robot translational velocities can be determined by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_R \cos(\theta_t - \theta) \\ u_R \sin(\theta_t - \theta) \end{bmatrix} . \tag{4.21}$$

Using (4.20) and (4.21), the error state dynamic model chosen in the rotated coordinate frame (4.19) is derived as follows:

$$\begin{aligned} \dot{x}_e &= y_e \kappa_r(s)\dot{s} - \dot{s} + u_R \cos\alpha_e \\ \dot{y}_e &= -x_e \kappa_r(s)\dot{s} + u_R \sin\alpha_e \\ \dot{\alpha}_e &= \Phi u_R - \kappa_r(s)\dot{s} \\ \dot{\theta}_e &= \omega - \omega_b \end{aligned} \tag{4.22}$$

where $\omega_b = \dot{\theta}_b$, $\dot{\theta}_r = \kappa_r(s)\dot{s}$, and $\kappa(s)$ is the path curvature.

Linearizing the error dynamics (4.22) around the reference path $\mathbf{x}_d = [x_d, y_d, \theta_d]^T$, we obtain the following linear model:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & \kappa_d u_R & 0 & 0 \\ -\kappa_d u_R & 0 & u_R & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \alpha_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{4.23}$$

where

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} -\dot{s} + u_R \cos\alpha_e \\ \Phi u_R - \kappa_r(s)\dot{s} \\ \omega - \omega_b \end{bmatrix} . \tag{4.24}$$

In [31], it is shown that the posture kinematic model of omnidirectional robots is completely controllable.

Besides converging the vehicle to a desired path, assigning a velocity profile to the path is an additional task, in which the forward velocity is used as an extra degree of freedom. For example, the forward velocity should be decreased as the vehicle rotates

around a sharp corner. This allows the integration of constraints by scaling the forward velocity as proposed by Bak et al. [16]. Lapierre et al. [135] combined path following with obstacle avoidance. The forward velocity has to be controlled when an obstacle is detected. In this section, the velocity profile is generated by considering path and robot constraints. Then, our proposed controller will steer the robot to follow that path with a predefined velocity profile. In general, the paths are chosen to be C^2 continuous, so that they have continuous curvature and no cusps. The curvature κ of a continuous curvature path is upper bounded by $\kappa_{\max}$. That is, the steering radius $\rho \geq \rho_{\min} = 1/\kappa_{\max}$. Additionally, there is an upper bound on the curvature derivative, $\dot{\kappa}$, since a robot must reorient its moving direction with a finite steering velocity. In this section, we consider the following velocity constraints:

- Maximum velocity constraints, $u \leq u_{\max}$
- Given the desired forward velocity u_R and the desired rotational velocity ω_b, we can obtain the forward velocity by using

$$u_d = \min(u_R, (4.18)) \tag{4.25}$$

  where ω in (4.18) is replaced by ω_b.

- To prevent the robot slipping off the path, we have the following constraint

$$-\sqrt{\frac{\mu g}{|\kappa|}} \leq u \leq \sqrt{\frac{\mu g}{|\kappa|}} \tag{4.26}$$

  where μ is the friction coefficient, g is the acceleration due to gravity, and κ is the curvature.

Thus, the velocity constraints can be given as

$$0 \leq u \leq \min(u_{\max}, u_d, \sqrt{\frac{\mu g}{|\kappa|}}) \ . \tag{4.27}$$

The strategy for generating a velocity profile can be integrated into an online path planner. The velocity can be monitored within some lookahead distance. The slower the desired forward velocity u_R, the shorter the lookahead distance can be considered. The number of steps of a velocity profile must be at least equal to the prediction horizon N of the MPC, detailed in the next subsection.

4.3.2 Controller Design

Equation (4.23) can be given in the state-space form $\dot{\mathbf{x}}_e = A_c\mathbf{x}_e + B_c\mathbf{u}_e$. To design the MPC controller for path following, the linearized system (4.23) will be written in a discrete state space system as

$$\mathbf{x}_e(k+1) = A\mathbf{x}_e(k) + B\mathbf{u}_e(k) \tag{4.28}$$

where $A \in \mathbb{R}^n \times \mathbb{R}^n$, n is the number of the state variables and $B \in \mathbb{R}^n \times \mathbb{R}^m$, m is the number of input variables. The discrete matrices A and B can be obtained as follows:

$$A = I + A_c T_s, \quad B = B_c T_s \tag{4.29}$$

where T_s is a sampling time.

It is possible to use MPC to control a state space model (4.28) of a system. The goal is to find the control-variable values that minimize the quadratic objective function by solving a quadratic program (QP). The quadratic objective function with a prediction horizon N is given by

$$J(k) = \sum_{j=1}^{N} \{\mathbf{x}_e^T(k+j|k) Q \mathbf{x}_e(k+j|k) + \mathbf{u}_e^T(k+j-1|k) R \mathbf{u}_e(k+j-1|k)\} \tag{4.30}$$

where $Q \in \mathbb{R}^n \times \mathbb{R}^n$ and $R \in \mathbb{R}^m \times \mathbb{R}^m$ are the weighting matrices, with $Q \geq 0$ and $R \geq 0$. The double subscript notation $(k+j|k)$ denotes the prediction made at time k of a value at time $k+j$.

First, the following matrices are defined in order to be able to write the problem in a form, which a QP solver can solve. By introducing matrices such that all $\mathbf{u}_e(\cdot)$ and $\mathbf{x}_e(\cdot)$ are conveniently stored, we obtain a more compact expression. Defining the prediction-error vector

$$X(k) = [\mathbf{x}_e^T(k+1|k), \mathbf{x}_e^T(k+2|k), \ldots, \mathbf{x}_e^T(k+N|k)]^T$$

where $X \in \mathbb{R}^{n \cdot N}$ and the control error vector

$$U(k) = [\mathbf{u}_e^T(k|k), \mathbf{u}_e^T(k+1|k), \ldots, \mathbf{u}_e^T(k+N-1|k)]^T$$

where $U \in \mathbb{R}^{m \cdot N}$.

It can be shown that

$$X(k) = G(k)\mathbf{x}_e(k|k) + S(k)U(k) \tag{4.31}$$

where $G(k) \in \mathbb{R}^{n \cdot N} \times \mathbb{R}^{n \cdot N}$ and $S(k) \in \mathbb{R}^{n \cdot N} \times \mathbb{R}^{m \cdot N}$ are defined as follows

$$G(k) = [\alpha(k+1,k), \alpha(k+2,k), \cdots, \alpha(k+N,k)]^T$$

and

$$S(k) = \begin{bmatrix} \beta_{11}(k) & 0 & \cdots & 0 \\ \beta_{21}(k) & \beta_{22}(k) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1}(k) & \beta_{N2}(k) & \cdots & \beta_{NN}(k) \end{bmatrix}$$

with β_{ij} and $\alpha(k,j)$ defined as:

$$\beta_{ij} = \alpha(k+i, k+j) B(k+j-1)$$

$$\alpha(k_1, k_0) = \begin{cases} \mathbf{I} & \text{if } k_1 = k_0 \\ \prod_{i=0}^{N-1} A(k+i) & \text{if } k_1 > k_0 \end{cases}$$

After defining $\bar{Q} \in \mathbb{R}^{n \cdot N} \times \mathbb{R}^{n \cdot N}$ and $\bar{R} \in \mathbb{R}^{m \cdot N} \times \mathbb{R}^{m \cdot N}$ as block diagonal matrices containing Q and R repeated N times, the objective function of the QP problem can be rewritten

$$\begin{aligned} J(k) &= \sum_{j=1}^{N} \{\mathbf{x}_e^T(k+j|k) Q \mathbf{x}_e(k+j|k) + \mathbf{u}_e^T(k+j-1|k) R \mathbf{u}_e(k+j-1|k)\} \\ &= X^T \bar{Q} X + U^T \bar{R} U \\ &= (G\mathbf{x}_e(k|k) + SU)^T \bar{Q} (G\mathbf{x}_e(k|k) + SU) + U^T \bar{R} U. \end{aligned} \tag{4.32}$$

After some algebraic manipulations, we can rewrite the objective function (4.30) in a standard quadratic form:

$$\bar{J}(k) = \tfrac{1}{2} U^T(k) H(k) U(k) + \mathbf{f}^T(k) U(k) \tag{4.33}$$

with

$$\begin{aligned} H(k) &= 2(S(k)^T \bar{Q} S(k) + \bar{R}) \\ \mathbf{f}(k) &= 2S(k)^T \bar{Q} G(k) \mathbf{x}_e(k|k) \end{aligned}$$

where $H \in \mathbb{R}^{m \cdot N} \times \mathbb{R}^{m \cdot N}$ and $\mathbf{f} \in \mathbb{R}^{m \cdot N}$. Since all constants, which do not contain the variable U, do not affect the optimum, they have been excluded in (4.33). The matrix $H(k)$ is a Hessian matrix which is always positive definite. It describes the quadratic part of the objective function, and the vector $\mathbf{f}(k)$ describes the linear part.

To handle the input constraints, we consider the existence of bounds in the amplitude of the control variables:

$$\mathbf{u}_{\min} \leq \mathbf{u}(k+j|k) \leq \mathbf{u}_{\max} \tag{4.34}$$

where $j \in [0,\, N-1]$, and $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ denote the lower and upper bounds, respectively. In our path following problem, we consider the following constraints:

$$\begin{array}{ccccc} 0 & \leq & \dot{s} & \leq & u_{\max} \\ -\omega_c - \omega_b & \leq & \omega & \leq & \omega_c - \omega_b \end{array} \tag{4.35}$$

with $\omega_c = \min(\omega_l, \omega_{\max})$, where ω_l is calculated by using (4.18), in which $r(\omega)$ is replaced by u_R.

Thus, this standard expression is used in QP problems and the optimization problem to be solved at each sampling time is stated as follows:

$$\begin{aligned} U^\star &= \arg \min_{\mathbf{u}_e} \bar{J}(k) \\ \text{subject to} \quad & \mathbf{u}_{\min} \leq \mathbf{u}(k+j|k) \leq \mathbf{u}_{\max} \end{aligned} \tag{4.36}$$

After the QP problem at time t_k is solved, an optimal control sequence is generated. The velocity of a virtual vehicle $\dot{s}$, the steering angle θ_t, and the rotational velocity ω can be calculated from the first element of this sequence and is then applied to the system.

4.3.3 Experimental Results

We validated our proposed control algorithm with a real omnidirectional mobile robot, shown in Figure 3.1. An eight-shaped reference path and the user-defined parameters were given as follows

$$x_d(t) = 1.8\sin(t), \quad y_d(t) = 1.2\sin(2t),$$
$$\theta_b = 0, \ \ \omega_b = 0, \ \ \mu = 0.18, \ \ g = 9.81 \text{ m/s}^2, \ \ u_R = 1 \text{ m/s},$$
$$u_{\max} = 1.315 \text{ m/s}, \ \ \omega_{\max} = 3 \text{ rad/s}.$$

The reference path was numerically parameterized by the curvilinear abscissa s. All tunable parameters used in our experiments are listed as follows: $Q = \text{diag}(300, 300, 7, 70)$, $R = \text{diag}(1, 0.001, 3)$, $N = 3$, $T_s = 0.05$ s.

In our experiments, the package *OOQP* [81] has been used to solve the QP problem. The experimental results are shown in Figure 4.15. As seen from the experimental results, we achieve a real-time implementation of our control law. The forward velocity u_R was decreased in order to preserve the curvature radius when the robot made sharp turns, while the velocity commands did not exceed the velocity constraints, as expected.

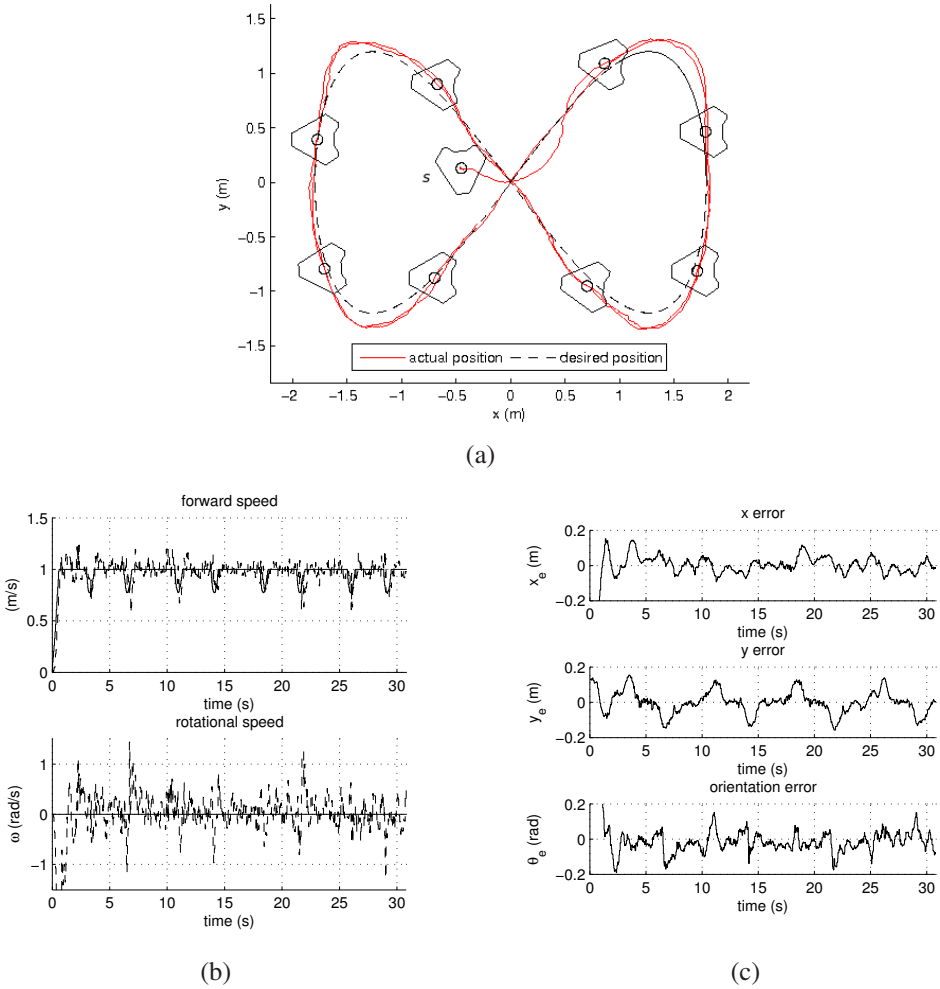
By using the MPC law, it is well known that the shorter the prediction horizon N, the less costly the solution of the online optimization problem. Thus it is desirable from a computational point of view to implement MPC schemes using short horizons. However, this may lead to poor performance. In case of fast dynamic systems like our system, the prediction horizon must be chosen in such a way that the computing time is smaller than the sampling period. From our experiments, shown in Table 4.1, we have found that $N = 3 - 5$ steps is a reasonable choice for the prediction horizon.

4.4 Smooth Reference Tracking of a Unicycle Mobile Robot

In this section, the path following control and the trajectory tracking control of a unicycle mobile robot are studied. Reference convergence in a path following problem and time

Table 4.1: Computing time depending on the prediction horizon.

Prediction Horizon	Computing Time (ms)
1	0.54
3	1.12
5	2.19
10	9.74
15	29.62



(a)

(b)

(c)

Figure 4.15: The experimental results by using the linear MPC law: (a) the superimposed snapshots, (b) the forward velocity and the rotational velocity, and (c) the pose errors with respect to the path coordinate. *S* in (a) denotes the initial position of the robot.

convergence in a trajectory tracking problem are considered in the cost function of the NMPC framework. The advantage of the path following controller is that the path following controller eliminates aggressiveness of the tracking controller by forcing convergence to the desired path in a smooth way. Thus, we incorporate this benefit to the trajectory tracking problem to achieve smooth convergence to the reference and to achieve time convergence of trajectory tracking. This is accomplished by modifying the cost function of the MPC framework through the addition of a time dependent penalty term. Based on this concept, our controller is able to optimize the reference point between the virtual vehicle (path-parameterized) and the trajectory point (time-parameterized). Furthermore,

input constraints and obstacle avoidance are taken into account. In the presence of obstacles, the controller deviates from the reference by incorporating obstacle information from range sensors into the optimization, while respecting motion constraints.

4.4.1 Problem Formulation

First, we derive control inputs $\dot{s}$ and ω such that the robot follows a virtual vehicle with position $\mathbf{x}_d = [x_d, y_d, \theta_d]^T$. The kinematic model of a mobile robot can be formulated with respect to a Frenet frame moving along the reference path. This frame plays the role of the body frame of a virtual vehicle that must be followed by the real robot, together with a spatial path Γ as depicted in Figure 4.16. In the path following problem, we normally let the forward velocity v track a desired velocity profile v_d, while the velocity of a virtual vehicle $\dot{s}$ converges to v. The error state vector $\mathbf{x}_e$ between the robot state vector $\mathbf{x}$ and a virtual vehicle's state vector $\mathbf{x}_d$ can be expressed in the frame of the path coordinate as follows

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix} . \tag{4.37}$$

Using (3.3) and (4.37), the error state dynamic model chosen in a rotated coordinate frame becomes

$$\begin{aligned} \dot{x}_e &= y_e \dot{s}\kappa - \dot{s} + v\cos\theta_e \\ \dot{y}_e &= -x_e \dot{s}\kappa + v\sin\theta_e \\ \dot{\theta}_e &= \omega - \dot{s}\kappa \end{aligned} \tag{4.38}$$

where κ is the path curvature and $\dot{s}$ is the velocity of a virtual vehicle, bounded by $0 \leq \dot{s} \leq \dot{s}_{\max}$.

However, the robot's translation velocity v has to be controlled in order to achieve trajectory tracking. Thus, we introduce an acceleration control input a, where $a = \dot{v}$.

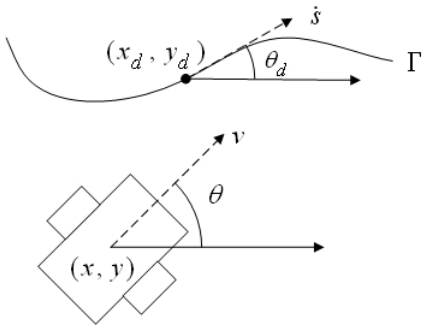


Figure 4.16: A graphical representation of a unicycle mobile robot and a reference path.

Then, we obtain

$$\dot{\eta}_e = a - \dot{v}_d \tag{4.39}$$

where $\eta_e = v - v_d$.

Similar to [88], we redefine the control signals

$$\mathbf{u}_e = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} -\dot{s} + v\cos\theta_e \\ \omega - \dot{s}\kappa \\ a - \dot{v}_d \end{bmatrix} . \tag{4.40}$$

The control input vector $\mathbf{u}_e$ is used as the control input in our NMPC framework. When the open-loop optimal control problem is solved, the system control input signals $\dot{s}$, a, ω can be obtained from (4.40).

Consequently, the error state dynamic model becomes

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \\ \dot{\eta}_e \end{bmatrix} = \begin{bmatrix} 0 & \dot{s}\kappa & 0 & 0 \\ -\dot{s}\kappa & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \\ \eta_e \end{bmatrix} + \begin{bmatrix} u_1 \\ v\sin\theta_e \\ u_2 \\ u_3 \end{bmatrix} . \tag{4.41}$$

4.4.2 Controller Design

We select the following cost function for (2.18)

$$F(\mathbf{x}_e, \mathbf{u}_e) = \mathbf{x}_e^T Q \mathbf{x}_e + \mathbf{u}_e^T R \mathbf{u}_e \tag{4.42}$$

with positive weight matrices $Q = \mathrm{diag}(q_{11}, q_{22}, q_{33}, q_{44})$ and $R = \mathrm{diag}(r_{11}, r_{22}, r_{33})$. To guarantee the control stability, the following Lyapunov function, similar to [88], can be selected as the terminal penalty in (2.18)

$$V(\mathbf{x}_e(t+T_p)) = \tfrac{1}{2}\mathbf{x}_e(t+T_p)^T P \mathbf{x}_e(t+T_p) \tag{4.43}$$

where $P = \mathrm{diag}(p_{11}, p_{22}, p_{33}, p_{44})$ is a positive definite matrix, under the terminal-state controller $\mathbf{u}_e^L(t)$ such that the following condition is satisfied:

$$\dot{V}(\mathbf{x}_e(t)) + F(t, \mathbf{x}_e(t), \mathbf{u}_e(t)) \leq 0, \ \ \forall \mathbf{x}_e(t) \in \Omega \tag{4.44}$$

The terminal state feedback controller $\mathbf{u}_e^L = [u_1^L, u_2^L, u_3^L]^T$ is defined as follows:

$$u_1^L = -\alpha x_{eT} \quad u_2^L = -\beta\theta_{eT} \quad u_3^L = -\gamma\eta_{eT} \tag{4.45}$$

where $\alpha,\ \beta,\ \gamma > 0$, and $\mathbf{x}_e(t+T_p) = [x_{eT}, y_{eT}, \theta_{eT}, \eta_{eT}]^T$. The subscript T denotes the terminal state. All weight parameters have to be selected such that (4.44) is satisfied.

Then, the stability condition becomes

$$\begin{aligned}
&(\dot{V}+F)(\mathbf{x}_e,\mathbf{u}_e^L)\\
&\quad= p_{11}x_{eT}\dot{x}_{eT}+p_{22}y_{eT}\dot{y}_{eT}+p_{33}\theta_{eT}\dot{\theta}_{eT}+p_{44}\eta_{eT}\dot{\eta}_{eT}+F(t+T_p)\\
&\quad= p_{11}x_{eT}u_1^L+p_{22}y_{eT}v\sin\theta_{eT}+p_{33}\theta_{eT}u_2^L+p_{44}\eta_{eT}u_3^L+F(t+T_p)\\
&\quad= p_{11}x_{eT}u_1^L+p_{22}y_{eT}v\sin\theta_{eT}+p_{33}\theta_{eT}u_2^L+p_{44}\eta_{eT}u_3^L+q_{11}x_{eT}^2+q_{22}y_{eT}^2+q_{33}\theta_{eT}^2\\
&\qquad+q_{44}\eta_{eT}^2+r_{11}u_1^{L^2}+r_{22}u_2^{L^2}+r_{33}u_3^{L^2}\ \leq 0
\end{aligned} \tag{4.46}$$

for all $\mathbf{x}_e \in \Omega$.

Substituting the terminal state feedback controller (4.45) into (4.46), we obtain

$$\begin{aligned}
\dot{V}(x_e(t+T_p))+F(t+T_p) = x_{eT}^2(-p_{11}\alpha+q_{11}+\alpha^2 r_{11})+\theta_{eT}^2(-p_{33}\beta+q_{33}+\beta^2 r_{22})\\
+\eta_{eT}^2(-p_{44}\gamma+q_{44}+\gamma^2 r_{33})+p_{22}y_{eT}v\sin\theta_{eT}+q_{22}y_{eT}^2.
\end{aligned} \tag{4.47}$$

Similar to [88], the following conditions for the weight parameters are required to have a negative derivative of the value function:

$$\begin{aligned}
p_{11}\alpha-q_{11}-\alpha^2 r_{11} &\geq q_{22}\\
p_{33}\beta-q_{33}-\beta^2 r_{22} &\geq 0\\
p_{44}\gamma-q_{44}-\gamma^2 r_{33} &\geq 0
\end{aligned} \tag{4.48}$$

and the terminal-state region is defined as follows:

$$\begin{aligned}
|x_{eT}| &\geq |y_{eT}|\\
p_{22}y_{eT}v\theta_{eT} &< 0
\end{aligned} \tag{4.49}$$

The terminal-state region is further bounded by the control signal region. From (4.40) and (4.45), we have the following results at the terminal state:

$$\begin{bmatrix} u_1^L\\ u_2^L\\ u_3^L \end{bmatrix} = \begin{bmatrix} -\alpha x_{eT}\\ -\beta\theta_{eT}\\ -\gamma\eta_{eT} \end{bmatrix} = \begin{bmatrix} -\dot{s}^L+v^L\cos\theta_{eT}\\ \omega^L-\dot{s}^L\kappa\\ a^L-\dot{v}_d \end{bmatrix} \tag{4.50}$$

Then, the system control inputs $\dot{s}^L$, ω^L, a^L at the terminal state can be obtained by

$$\begin{bmatrix} \dot{s}^L\\ a^L\\ \omega^L \end{bmatrix} = \begin{bmatrix} \alpha x_{eT}+v\cos\theta_{eT}\\ \beta\theta_{eT}+\dot{s}\kappa\\ -\gamma\eta_{eT}+\dot{v}_d \end{bmatrix} \tag{4.51}$$

with the control input constraints

$$\begin{bmatrix} \omega_{\min}\\ a_{\min} \end{bmatrix} \leq \begin{bmatrix} \omega^L\\ a^L \end{bmatrix} \leq \begin{bmatrix} \omega_{\max}\\ a_{\max} \end{bmatrix}. \tag{4.52}$$

To achieve time convergence of trajectory tracking, we add an extra penalty. In the trajectory tracking problem, the vehicle is required to track a time parameterized reference. We normally feed a desired posture $\mathbf{x}_{d,t}$ to a tracking controller. In this work, we wish to combine trajectory tracking behaviors in a path following control law, thus achieving smooth spatial convergence to the trajectory as well as time convergence. We penalize the cost function with

$$F_t = (\mathbf{x}(T_p) - \mathbf{x}_{d,t}(T_p))^T K_t (\mathbf{x}(T_p) - \mathbf{x}_{d,t}(T_p)) \tag{4.53}$$

where K_t is a positive definite matrix. This matrix weighs the relative importance of convergence in time over spatial convergence to the path. If $K_t = 0$ is chosen, pure path following is achieved.

To avoid obstacles, obstacle information has to be incorporated into the cost function, so that the computed control follows the desired reference, while staying away from the obstacles. Typically, the desired reference is generated by a planning algorithm based on a map of the environment and this reference is assumed to be collision-free. During the actual motions it is possible that obstacles appear in the vehicle's path, which had not been present in the planning phase. This may also happen because of the imprecise map, or localization errors. In this work, we assume that the simulated sensors mimic infra-red sensors placed in a ring around the robot, spaced by 30° and they have a distance range of 50 cm. The obstacle points detected by these sensors then contribute to the cost function with a term which penalizes states as follows:

$$F_{obs} = \sum_{i=1}^{N_p} \sum_{j=1}^{N_s} K_{obs} \frac{e^{-c_1|\theta_{obs,ij}|}}{e^{c_2 d_{obs,ij}}} \tag{4.54}$$

where K_{obs}, c_1, and c_2 are positive constants. N_s is the number of range sensors. N_p is the number of predictive steps, given by $N_p = T_p/\delta$, where δ is the sampling time. θ_{obs} is the angle of the obstacle with respect to the robot frame and d_{obs} is the distance between the robot and the obstacle. In case of moving obstacles, the information such as their position and velocity can be used to predict the information over the next T_p horizon and then the cost function can be computed. It has to be noted that we consider only convex polygonal obstacles.

4.4.3 Simulation Results

Our NMPC controller was first implemented in Matlab. Numerous simulations were performed to evaluate the performance of our system. All the elements of our NMPC framework were set as follows:

$$\begin{aligned} &Q = \text{diag}(0.2, 2, 0.01, 0.01), \quad R = \text{diag}(0.0001, 0.0001, 0.0001), \\ &P = \text{diag}(1, 1, 0.015, 0.015), \quad K_t = \text{diag}(1, 2, 0.01, 0.01), \\ &K_{obs} = 1.5,\ c_1 = 0.01,\ c_2 = 10,\ N_p = 3,\ T_c = T_p = 0.15 \text{ s}, \\ &\delta = 0.05 \text{ s},\ s(0) = 0 \text{ m},\ \alpha = 2.5,\ \beta = 1,\ \gamma = 1. \end{aligned} \tag{4.55}$$

The circle reference was given as

$$x_d(s(t)) = R\cos\tfrac{s(t)}{R} \quad y_d(s(t)) = R\sin\tfrac{s(t)}{R}$$

where $R = 1$ m and the desired translation and rotation were $v_d = 0.5$ m/s and $\omega_d = 0.5$ rad/s, respectively. The maximum and minimum control inputs were set to $v_{\max} = 2$ m/s, $v_{\min} = -2$ m/s, $\omega_{\max} = 2$ rad/s, $\omega_{\min} = -2$ rad/s, $a_{\max} = 2$ m/s^2 and $a_{\min} = -2$ m/s^2.

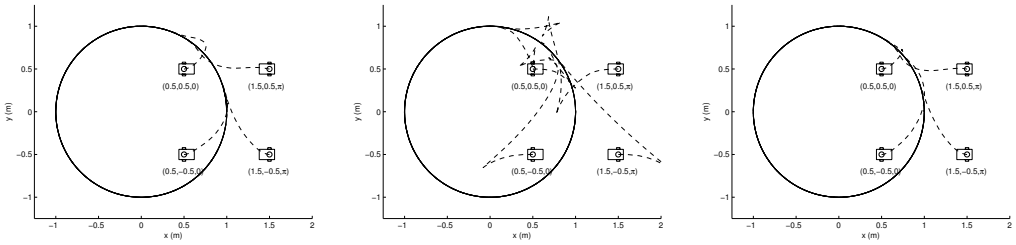
The performance achieved with pure path following, pure trajectory tracking (see [88] for details), and for combined trajectory tracking and path following is assessed. Figure 4.17(a) and Figure 4.17(b) show the simulation results of pure path following control and pure trajectory tracking control, respectively, with four different initial postures. The velocities and the posture errors of pure path following are depicted in Figure 4.18(a) and Figure 4.19(a), respectively, while those of pure trajectory tracking are plotted in Figure 4.18(b) and Figure 4.19(b), respectively when the initial posture of both cases was set to $(1.5, -0.5, \pi)$. As seen from the results, in case of path following control, the robot motions are less aggressive while the robot is approaching the reference path (see Figure 4.17(a)) and the control signals are less likely saturated. The reason is that the virtual vehicle slows down when the robot is far from the virtual vehicle, and consequently the predicted system states of the robot are able to easily reach the terminal region in finite time. However, in case of trajectory tracking where the reference point depends on time, the predicted system states are able to reach the terminal region in finite time with more effort than in case of path following control (see Figure 4.17(b)) because the conditions in (4.49), depending on time, need to be satisfied.

Figure 4.17(c) shows the simulation results of the combination of path following control and trajectory tracking control. The velocities and the posture errors are shown in Figure 4.18(c) and Figure 4.19(c), respectively when the initial posture was set to $(1.5, -0.5, \pi)$. This controller was able to achieve both reference convergence and time convergence with smooth motions. As seen in the results, the robot converges smoothly to the desired path and then it reacts to achieve zero trajectory tracking error.

Next, a convex polygonal obstacle was introduced in a position which prohibited path following. As it is shown in Figure 4.20, the controller deviated from the desired reference in order to safely avoid the obstacle and time convergence was still able to be achieved. In Figure 4.21, two moving obstacles were present. The velocity of the first obstacle was 0.2 m/s at -135°, while the velocity of the second obstacle was 0.6 m/s at 150°. In the simulation results, the robot moved backward to avoid the collision and waited until it was able to find a way to stay away from the obstacles and to follow the reference.

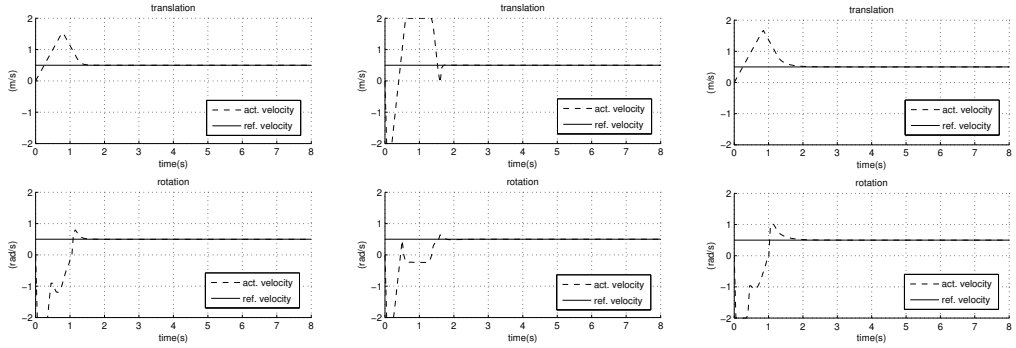
4.4.4 Experimental Results

In this section, a unicycle mobile robot, shown in Figure 3.6(a) was used in real-world experiments. The same reference used in simulation was employed in this experiment,



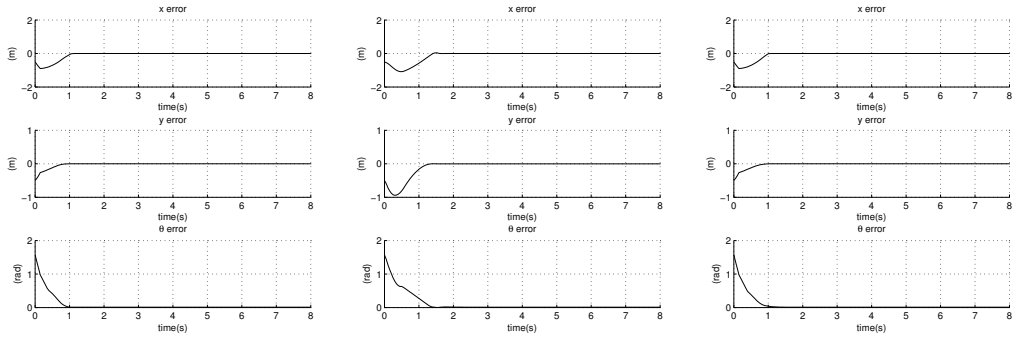
(a) (b) (c)

Figure 4.17: The simulation results with four different initial postures: (a) pure path following, (b) pure trajectory tracking, and (c) the combination of path following and trajectory tracking.



(a) (b) (c)

Figure 4.18: The robot velocities when the initial posture was set to $(1.5, -0.5, \pi)$: (a) pure path following, (b) pure trajectory tracking, and (c) the combination of path following and trajectory tracking.



(a) (b) (c)

Figure 4.19: The posture errors when the initial posture was set to $(1.5, -0.5, \pi)$: (a) pure path following, (b) pure trajectory tracking, and (c) the combination of path following and trajectory tracking.

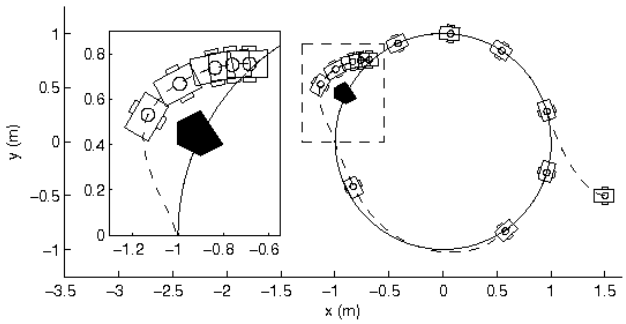


Figure 4.20: The simulation results when a static polygonal obstacle was present.

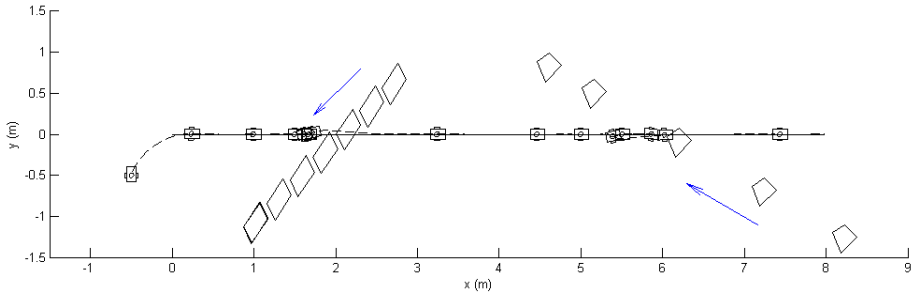
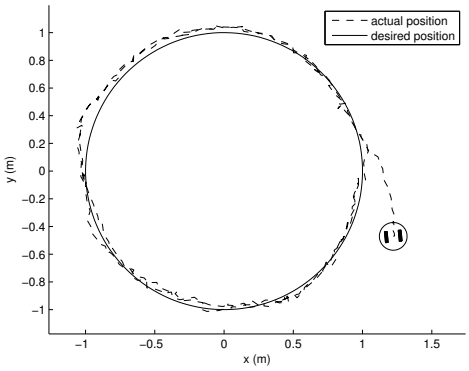


Figure 4.21: The simulation results when two moving polygonal obstacles were present.

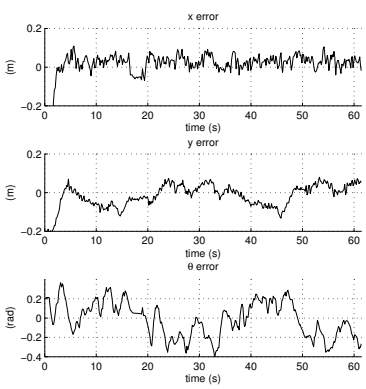
but with $v_d = 0.2$ m/s, $\omega_d = 0.2$ rad/s and $\delta = 0.1$ s. Incorporating the terminal penalty and the terminal-state constraints degrades the performance because of the high computational complexity. Therefore we did not include them in the cost function in the real-world experiments. However, we still penalized the cost function with the time-dependent penalty term in order to satisfy time constraints. Furthermore, obstacle avoidance will be considered in our future work because of the high computational demand under real-time constraints. The experimental results are shown in Figure 4.22.

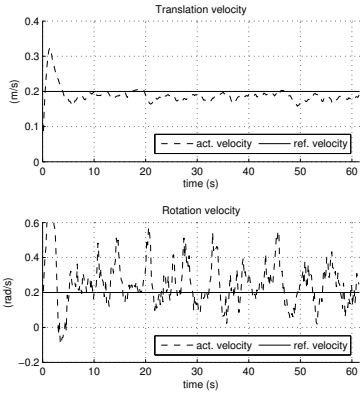
4.5 Discussions and Summary

In this chapter, we presented the solutions based on MPC schemes for the problem of path following control of both an omnidirectional mobile robot and a unicycle mobile robot. The comparison between path following and trajectory tracking of both kinds of mobile robots has been shown and discussed. The linear MPC version to solve the path following problem of an omnidirectional mobile robot with consideration of robot and path constraints has also been presented. The forward velocity assignment was investigated in order that the robot was able to travel safely. However, it is important to point



(a)





(b) (c)

Figure 4.22: The experimental results by using our NMPC law: (a) the robot positions and its reference, (b) the posture errors, and (c) the robot's velocities.

out that the linear MPC has the disadvantage that the linearized model is only valid for points near the reference path. As the state and input deviate from the current operating point, the model mismatch increases. This can generate large prediction errors with a consequent instability of the closed-loop system.

Basically, in trajectory tracking when the robot is far from the reference trajectory, the controllers are not suitable because of large control actions resulting from large error. A possible solution in such a situation is to use so-called landing curves, which allow the controller to drive the robot efficiently to the reference. However, in Section 4.4, we have shown that we can eliminate those problems using the combination of trajectory tracking and path following. Our approach can steer a unicycle mobile robot smoothly to a reference path with time and control input constraints. The computation limits the use of NMPC in real-time systems. The improvement of the computation efficiency and the analysis of feasibility are the topics of future work.

Chapter 5

Coordinated Path Following Control

Coordinated path following, which can be seen as a subtask of formation tracking, is investigated in this chapter. We first give a brief review on formation control strategies, that is, how a group of robots can be controlled to get into and to maintain a formation. In Section 5.3, nonlinear model predictive control (NMPC) based on the leader-following strategy is proposed for omnidirectional mobile robots, while the distributed version without any leader is proposed in Section 5.4. A decentralized approach based on a Lyapunov function and a second-order consensus protocol with a reference velocity for unicycle mobile robots is described in Section 5.5.

5.1 Review on Formation Control Strategies

In the literature, there is a wide spectrum of strategies, ranging from simple, behavior-based, purely local-communication ones to more involved ones relying, to varying extent, on global communication. The former category is characterized by minimalism and robustness but a lack of guarantees that the desired formation will actually emerge; the latter category is characterized by reliability and efficiency, but also a need for global knowledge and computation [77]. In the literature, the approaches to solve formation control problems are roughly categorized into three strategies: leader-following, behavior-based, and virtual-structure. Each approach has its own advantage and disadvantage, as discussed in the following subsections.

Before moving on to each strategy in detail, we point out some notable work concerning formation stability. Swaroop and Hedrick [218] proposed the notion of string stability for line formations and derived sufficient conditions for a formation to be string stable. Pant et al. [185] generalized string stability to formations in a planar mesh, through the concept of mesh stability. Tanner [219] concentrated on formations in acyclic graphs and studied the effect of feedback and feed-forward on the input-to-state stability of the formation. He also addressed issues related to safety and performance and investigated tools that allow to quantify, bound, and estimate the error amplitudes in the worst cases of different types of formation interconnection structures. Fax and Murray [72] analyzed the stability of formations in arbitrary graphs and proposed a Nyquist-like stability criterion that can be derived from the spectral properties of the graph Laplacian.

5.1.1 Behavior-based Approach

Behavior-based approaches start by designing simple and intuitive behaviors or motion primitives for each individual robot, e.g., formation keeping, trajectory tracking, goal seeking, and collision and obstacle avoidance. Then, more complex motion patterns can be generated by using a weighted sum of the relative importance of the simple primitives and the interaction of several robots. However, the group behavior cannot be explicitly defined and the theoretical formalization and mathematical analysis of this approach is difficult. Consequently, the convergence of the formation to a desired configuration cannot be guaranteed.

As multiple behaviors are allowed to affect the system simultaneously, one way of coordinating the different behaviors to achieve a satisfactory, global behavior is to use an arbitration mechanism, where behaviors with higher priorities simply overrule other behaviors or to determine the weights on each individual behavior in order to make the overall robot system behave in a satisfactory way. For instance, in a formation control mission, it is required that the robots maintain a given relative position while avoiding obstacles. However, it may not always be desirable to let different behaviors affect the system simultaneously. Obstacle-avoidance guarantees may no longer hold when this behavior is combined with other behaviors.

Nevertheless, the advantage of behavior-based schemes is that the group dynamics implicitly incorporates formation feedback by coupling the weights of the actions that depend on the relative coordinates of neighboring robots. Behavior-based approaches also give the system the autonomy to operate in unpredicted situations, using sensors to obtain information about the environment; thus they are useful in guiding a multi-robot system in an unknown or dynamically changing environment. Variations of this method have been used by researchers to achieve different formation configurations for mobile robots [12, 17], spacecraft [11], and underwater vehicles [159].

Balch and Arkin [17] followed motor scheme control. All the behaviors (i.e., goal seeking, collision avoidance and formation keeping) are summed together through suitable weight coefficients which set the relative priority between them. Antonelli et al. [12] proposed a decentralized scheme to solve the flocking problem by using the null-space-based behavior control. Simple behaviors are defined for each robot, and these behaviors are arranged in priority. Michaud et al. [164] employed a hybrid control architecture that combines a behavioral level with global level deliberation over the overall states of the group. All behaviors run in parallel and their resulting commands are prioritized and managed through a finite state machine to generate the control actions of the robot. Fredslund and Matarić [77] achieved formation control using only local sensing and minimal communication between robots. Their key idea is to have each robot keep another specific robot, called a friend, within its field of view using a panning camera. The robots are always organized in a chain of friendships by the order of their IDs. One robot is the conductor, who broadcasts a message indicating the current formation. However, the possible formations are limited to chain-shaped ones that do not make a backward curve.

5.1.2 Leader-following Approach

With the leader-following strategy, some robots are considered as leaders, while others act as followers. The primary advantage of using such a strategy is that maneuvers can be specified in terms of the leader's motion and the formation can still be maintained even if the leader is perturbed by some disturbances. This approach essentially reduces to a tracking problem where stability of the tracking error is shown through standard control-theoretic techniques: The leader is supposed to pursue some group objectives, such as navigation in obstacle environments or tracking a reference trajectory, while the following robots track transformed coordinates of the leader with some prescribed offsets. The internal formation stability is induced by the individual robot's control laws. However, the disadvantage is that the leader's motion is independent of the followers, i.e., there exists no explicit feedback from the followers to the leader. Hence, if the following robots are unable to maintain a small tracking error, the leader will not slow down and the formation will break. In addition, the formation does not tolerate leader faults.

The most common formation control strategies for leader-following are feedback linearization [51, 56, 74], dynamic feedback linearization [153], backstepping [143], model predictive control [89, 232], first-order sliding mode control [201], and second-order sliding mode control [53], to name a few. Important work is presented by [51, 56, 74] using a feedback linearization control method for the formation of nonholonomic mobile robots. They proposed two controllers: Separation-bearing control and separation-separation control. In separation-bearing control, robot j follows robot i at a desired separation l_{ij}^d and desired relative bearing ψ_{ij}^d, while in separation-separation control, robot k follows two leaders robot i and robot j at desired separations l_{ik}^d and l_{jk}^d respectively.

In the leader-following approach, the absolute velocity of the leader robot in the local coordinates of the follower robot is treated as a necessary exogenous input for the formation tracking controller, i.e., the formation controllers require reliable estimation of the linear velocity and angular velocity of the leader robot and relative orientation. In the absence of communication, this becomes quite challenging from a sensing viewpoint, because the motion of multiple moving objects needs to be estimated simultaneously. Das et al. [51] used an omnidirectional vision system to provide the range and the angle of the observed leader. This information is used by the velocity estimator that is based on an extended Kalman filter. Vidal et al. [227] used omnidirectional image based visual servoing in which motion segmentation techniques enable each follower to estimate the image plane position and velocity of other robots in the formation. Orqueda and Fierro [182] used a pan-controlled camera on each robot for decentralized stabilization of formations. A leader-follower formation controller was proposed utilizing only the relative positions of robots, in which the derivatives of the relative positions were estimated using a high-gain observer. Mariottini et al. [153] proposed a method to observe the centroid of the robot by feedback control via a dynamic extension. A monocular camera without dependence on special markings on the leaders was employed by Min et al. [166]. The

position and velocities of the leader and the pose of the follower were estimated by using an extended Kalman filter. However, the authors performed only line and diagonal formations.

Most of the approaches mentioned above rely on nonlinear observers and image information, e.g., on an extended Kalman filter [152], an unscented Kalman filter [153], or a high gain observer [182]. To use image information, either position-based visual servoing [51, 152, 153, 182] or image-based visual servoing [227] is integrated within a motion control loop. To eliminate the need for measurement or estimation of the absolute velocity of the leader, Defoort et al. [53] used a second-order sliding mode formation controller which is only based on the relative motion states is derived. Dierks and Jagannathan [60] developed a neural network tracking controller that considers the dynamics of the leader and the followers using backstepping with the robust integral of sign of the error (RISE) feedback.

There exist some algorithms employing a variant of the leader-follower type strategy. For instance, Consolini et al. [47] proposed a method that the position of the followers is not fixed with respect to the leader reference frame but varies in suitable cones, and only these cones remain stable with respect to the leader reference frame. Their approach generates smoother trajectories and allows a lower control effort, especially for large distances between the leader and the followers. Gamage et al. [79] developed high-level supervisory control of discrete event systems to take care of the coordination of low level continuous feedback-linearized controllers including formation keeping, obstacle avoidance, wall following, and goal navigation. The higher-level discrete event system manages the dynamic interaction of the robots with the external environment. Stipanovic et al. [215] employed a dynamic extension of the unmanned aerial vehicle (UAV) model. By locally linearizing this model, the formation model can be treated as an interconnect system of overlapping subsystems (the subsystems share common components). Ögren and Leonard [175] combined a formation-keeping control scheme with a dynamic window approach to obstacle avoidance in order to guarantee safety and stability of the formation as well as convergence to the goal position. In their approach, the leader's task is to determine which actions it can take and still be confident that none of the followers will collide while attempting to maintain the formation.

5.1.3 Virtual-structure Approach

Virtual structures consider the entire formation as a rigid body. The control law for a single vehicle is derived by defining the dynamics of the virtual structure and then translates the motion of the virtual structure into the desired motion of each vehicle. The main advantages of the virtual structure approach are that it is fairly easy to prescribe the coordinated behavior for the group, and that the formation can be maintained very well during the maneuvers, i.e., the virtual structure can evolve as a whole in a given direction with some given orientation and maintain a rigid geometric relationship among multiple vehicles. However, if the formation has to maintain the exact same virtual structure all

the times, the potential applications are limited, especially when the formation shape needs to be frequently reconfigured.

The virtual structure approach has been used for formation control of mobile robots [66, 141], spacecraft [20, 196], and marine vehicles [98]. In [141], Lewis and Tan assumed that all robots had global knowledge. Their algorithm iteratively fit the virtual structure to the robots' positions, displaced the virtual structure in some desired direction and updated the robots' positions. Their method included formation feedback, but they cannot guarantee that a formation converges to a final configuration. Beard et al. [20] achieved virtual structures for spacecraft interferometry by having all members of the formation track assigned nodes which move into a desired configuration with stability guarantees, but do not include formation feedback. Egerstedt and Hu [66] defined the formation through mathematical constraints that model the formation shape. The path for a virtual leader, including formation feedback is computed as a reference point for the robots to follow. Similarly, Young et al. [238] included a specific form of formation feedback in the coordination variable evolution. The virtual structure will slow down and stop as the robots get out of formation and it moves towards its final goal if the robots are maintaining formation. Formation of marine craft was proposed by Ihle et al. in [98], where the desired formation configuration is given as a set of constraint functions. The functions are treated analytically and by using feedback from the imposed constraint functions, constraint forces arise. These forces can be seen as control laws and they act so that the constraint functions are satisfied in order to keep the formation assembled during operation.

5.1.4 Other Control Strategies

Several methods have been identified to solve formation control problems using an optimization based approach and a graph-theory based approach.

One way to approach the formation control problem is to formulate it as an optimization problem. Model predictive control (MPC) is a well-known control strategy in which the current control action is computed by solving a finite horizon optimal control problem online. In general, the centralized implementation is not practical since the size of the state variables depends typically on the number of mobile robots. When the control horizon becomes larger, the number of variables, of which the robot has to find the value, increases rapidly. Also, the demands of computational power and memory are daunting for the real-time solution of systems with a large control horizon and a large number of mobile robots. Thus, the research has led to decomposing the centralized system into smaller subsystems, which are independently controlled in the MPC framework.

Jia and Krogh [107] proposed an approach that each controller views the signals from other subsystems as disturbance inputs in its local model. The distributed MPC controllers exchange predictions on the bounds of their state trajectories and incorporate this information into their local distributed MPC problems. They also impose their own predicted state bounds as constraints in subsequent distributed MPC iterations to guarantee

their subsystem satisfies the bounds broadcast to the other controllers. Each controller solves a local min-max problem on each iteration to optimize performance with respect to worst-case disturbances. Richards and How [199] presented a decentralized algorithm for systems with coupled constraints. Relevant plan data is exchanged between subsystems to ensure that all decisions are consistent with satisfaction of the coupled constraints, e.g., collision avoidance. The decentralized method employs at each time step a sequential solution procedure, in which the subsystems solve their planning problems one after the other. Keviczky et al. [121] presented decentralized MPC schemes for decoupled systems where cost function and constraints couple the dynamical behavior of the systems. Each MPC controller is associated to a different node and computes the local control inputs based only on its current states, its neighbors current states, its terminal region, its neighbors terminal regions and models and constraints of its neighbors. Based on such information each node computes its optimal input and its neighbors optimal inputs. The input to the neighbors will only be used to predict their trajectories and then discarded while the first component of its optimal input will be implemented. In [32, 106], a distributed MPC algorithm for unconstrained, linear time-invariant (LTI) systems in which the dynamics of the subsystems are influenced by the states of interacting subsystems was described. A contractive state constraint is employed in the MPC optimization of each subsystems and asymptotic stability is guaranteed if the systems satisfies a matrix stability condition. Dunbar and Murray [65] used the exchange of the most recent optimal control trajectory between coupled subsystems prior to each update. The stability analysis is more difficult in this case. The asymptotic stability of a multi-vehicle formation system without collision avoidance constraints is guaranteed by requiring that each distributed optimal control does not deviate too far from the previous optimal control.

There exist a large number of publications on formation control using the graph theory-based approach. We intentionally do not collect all published contributions to this approach. A non-exhaustive list of relevant research includes: Flocking behavior, which involves convergence of the velocity vectors and orientation of the agents to a common value at steady state ([103, 176, 220]), the rendezvous problem, in which agents must converge to the same point ([61, 131, 146]), decentralized formation tracking, where agents have to maintain a desired formation while following or tracking a reference ([63, 71, 131, 138, 194]), synchronization phenomena arising in systems of coupled nonlinear oscillators ([104, 186]), and cyclic pursuit ([154]). In particular, Ren [194] has shown that many existing leader-follower, behavioral, and virtual structure/virtual leader formation control approaches can be thought of as special cases of the consensus-based strategies.

An alternative approach to this problem is a decentralized navigation function proposed by DeGennaro and Jadbabaie [52]. This function is used to drive each agent of a group toward a desired final configuration which is expressed in terms of distances between the connected agents. The formation can be reached anywhere in the space and with any orientation.

5.2 Related Work on Coordinated Path Following Control

The difference between trajectory tracking and path following has already been explained in Chapter 4. Now we move on to the formation control problem of path following, also referred to as *coordinated path following problem*. The solution of this problem is divided into two basic tasks [84]: The geometric task ensures that the individual mobile robot converges to and stays at its designation in the formation. The dynamic task (also called the coordination task) ensures that the formation maintains a speed along the path according to the given speed assignment. Therefore, each member requires an individual parameterized reference path so that when all paths' parameters are synchronized each member will be in formation. However, practical constraints arise from the characteristics of the supporting inter-vehicle communication network. They have to exchange this parameter to each other via communication only. Thus, the quality of communication channels becomes a crucial part.

Ghabcheloo et al. [82] presented a solution to the problem of steering a group of wheeled mobile robots along given spatial paths, while holding a desired inter-vehicle formation pattern with bidirectional communication constraints. In [83], they focused further on coordinated path following with an extension on communication problems on autonomous underwater vehicles (AUVs), i.e., in the presence of communication losses and time delays. Skjetne et al. [208] used vectorial backstepping to solve an individual maneuvering problem for each marine vessel with an extension of a synchronization term in the resulting decentralized dynamic controllers to ensure that the vessels keep assembled in the desired formation. Ihle et al. [99] showed a passivity property for the path following control and then combined this with a passivity-based synchronization algorithm developed in [14]. Their solution can tolerate a time-varying formation configuration and allows communication dropouts.

In [66], Egerstedt and Hu proposed formation constraint functions to decouple the coordination and following problems, while maintaining the stability of the formation. Ghomman et al. [84] developed a control law based on the virtual structure approach for coordination of a group of nonholonomic mobile robots. The derivative of the path parameter is left as an additional control input to synchronize the formation motion. However, they assumed that each robot has to broadcast its state and reference to the rest of the team and it has to receive states and references from the other robots of the team. Do [62] designed a cooperative controller to force a group of mobile robots with limited sensing ranges to perform desired formation tracking, and to guarantee no collisions between robots using potential functions. The physical dimensions and dynamics of the robots are also considered. Recently, Xiang et al. [235] investigated decentralized speed adaptation with minimum communication. They showed that all speeds can finally converge to desired speed profiles using only exchange of the path's parameter.

In this chapter, we introduce two solutions for coordinated path following problems.

One employs nonlinear model predictive control (NMPC) approaches. The other is based on a Lyapunov function and a second-order consensus protocol with a reference velocity. We develop two strategies for the first solution: The leader-following strategy [111, 115] and the distributed version [110, 112]. Both strategies are proposed for omnidirectional mobile robots, given in Section 5.3 and Section 5.4, respectively. State variables and constraints are decoupled, while each robot's cost function is coupled with its neighbors. Unlike most NMPC controllers which have been employed to solve a trajectory tracking problem, our NMPC controller is used to solve a coordinated path following problem. Three key advantages of using NMPC in this chapter are (i) integrating the velocity of a virtual vehicle, $\dot{s}$, into the local cost function explicitly to solve the path following problem, (ii) controlling robot motions with input constraints, and (iii) utilizing future information of a reference path to produce an optimal predicted trajectory of a robot. In Section 5.5, a coordinated path following controller for unicycle mobile robots is developed. The idea of this solution is to synchronize the path derivative, based on a Lyapunov function and a second-order consensus protocol with a reference velocity [116].

5.3 Nonlinear MPC Using the Leader-following Strategy

The first solution to coordinated path following is based on the leader-following strategy, where the leader robot follows a given path and each follower robot tracks a trajectory, estimated by using the leader's information. To solve the path following problem for the leader robot, we propose to integrate the velocity of a virtual vehicle ($\dot{s}$) to be followed along that path into the local cost function of NMPC. After the open-loop optimization problem is solved, the optimal velocity at each time step in the future is obtained. This information and the leader's current state are broadcasted to all follower robots. With respect to a desired formation configuration and a reference path, each follower robot can estimate its own reference trajectory by using the leader's information and its time stamp. NMPC is also employed as a local control law to steer the follower robot to track that reference path.

5.3.1 Problem Formulation

In this section, we solve two problems, i.e., the path following problem and the formation keeping problem. In the path following problem, the path is parameterized by the path length s, instead of time, which is normally used in a trajectory tracking problem. To make a formation pattern, we employ an idea of formation configurations in a curvilinear coordinate system, proposed in [18]. When the formation is turning, the formation's shape can be slightly modified. Follower robots on the outside speed up and follower robots on the inside slow down, which allows the formation to be shape compliant on route (see Figure 5.1). In our method, only the path which the leader follows is gen-

erated, while each individual follower robot F_i in the group has a pre-specified offset $(p_i(s), q_i(s))$ in curvilinear coordinates relative to the reference point C, which the leader robot follows, as shown in Figure 5.1.

In some situations, the collision-free path does not always guarantee the safety for the whole formation. For example, the width of the path could be too narrow to allow for more than one robot to pass. Thus, the formation must be changed to a column (see Figure 5.2(b)). However, as stated in [18], the width of the formation (q-offset) can only be changed if the second derivative d^2q/ds^2 exists, i.e., offset q_i must be adequately smooth with respect to the corresponding progenitor path during the transient from one configuration to another. To solve this problem, we propose to use a fifth-order (quintic) polynomial to join two path segments with different offsets so that the position, first and second derivatives at the starting and end points, where two path segments are joined, match. A quintic is the minimum order polynomial which is able to give sufficient degrees of freedom and comply with the constraints on the slope. The general form of a quintic function is given by

$$q(s) = \sum_{j=0}^{5} (a_j s_d^j) \tag{5.1}$$

subject to constraints on the conditions of the starting and end points, and its slope:

$$\begin{aligned} q(s_{start}) = q_{start}, \quad q(s_{end}) = q_{end} \\ q'(s) = \tfrac{dq(s)}{ds}, \quad q'(s_{start}) = q'(s_{end}) = 0 \\ q''(s) = \tfrac{d^2q(s)}{ds^2}, \quad q''(s_{start}) = q''(s_{end}) = 0 \end{aligned} \tag{5.2}$$

a_j are the coefficients of the function, (s,q) is the position on the offset curve at the path length s, $s_d = \frac{s-s_{start}}{s_{end}-s_{start}}$, (s_{start}, q_{start}) is the starting point of the quintic curve, and (s_{end}, q_{end}) is the end point of the quintic curve.

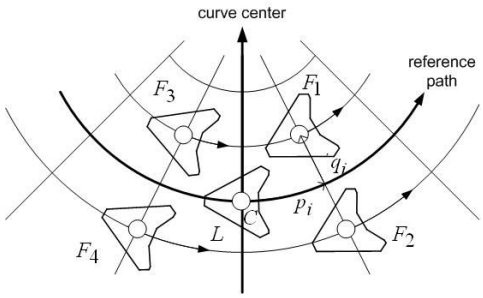


Figure 5.1: Graphical depiction of a mobile robot path and accompanying offset quantities [18] when the formation is turning. L denotes a leader robot and $F_1 - F_4$ denote follower robots.

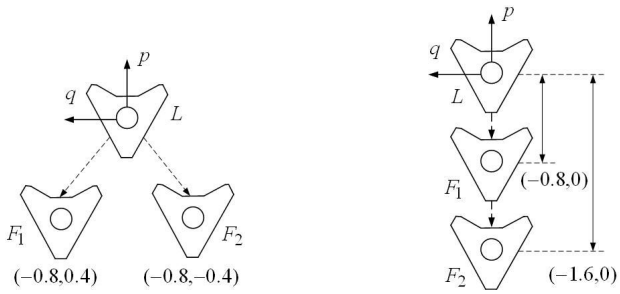


Figure 5.2: Graphical description of formation configurations: (a) a triangle, (b) a column. L, F_1, and F_2 denote the leader robot, the follower robot 1 and the follower robot 2, respectively. Units are given in meter.

Applying (5.2) to (5.1) yields $a_0 = q_{start}, a_1 = a_2 = 0, a_3 = 10, a_4 = -15$, and $a_5 = 6$. The quintic function becomes:

$$q(s) = q_{start} + (q_{end} - q_{start})(6s_d^5 - 15s_d^4 + 10s_d^3) \tag{5.3}$$

Using a curvilinear coordinate system allows us not only to have the offset-varying distance $q_i(s)$ but also to adjust the $p_i(s)$ coordinate. This is simply obtained by decreasing or increasing the velocity of a follower robot i in an appropriate manner. However, collision avoidance has to be taken into account. The timing of each follower robot has to be evaluated in order to ensure that no collision occurs during transition.

Let u_c be the translational velocity of point C, which the leader robot follows. In other words, u_c is the velocity of a virtual vehicle. Once the coordinates $(p_i(s), q_i(s))$ of a follower robot i have been determined, the path length of a follower robot s_i can be obtained by $s_i = s_c + p_i$, where s_c is the path length at point C. Then its velocity profile can be obtained by[1]

$$u_i = Hu_p \tag{5.4}$$

$$\omega_i = k_i u_i \tag{5.5}$$

where

$$k_i = \text{sign}(b)\frac{\sqrt{a^2+b^2}}{H^2} \qquad H = \sqrt{(1-k_p q)^2 + (\frac{dq}{ds})^2}$$
$$a = -2k_p\frac{dq}{ds} - q\frac{dk_p}{ds} - (1-k_p q)\frac{G}{H^2} \qquad b = k_p - k_p^2 q + \frac{d^2q}{ds^2} - \frac{dq}{ds}\frac{G}{H^2}$$
$$G = (1-k_p q)(-k_p\frac{dq}{ds} - q\frac{dk_p}{ds}) + \frac{dq}{ds}\frac{d^2q}{ds^2}$$

[1]The derivations can be found in Appendix A.

u_i, ω_i and k_i are the translational velocity, the rotational velocity and the curvature of the follower robot i, respectively, u_p is the translational velocity at s_i, which is usually equal to u_c. k_p is the curvature at s_i on the reference path, and $q_i(s)$ is the offset at s_i.

5.3.2 Controller Design

We begin with the leader robot and path following control. The main tasks for the leader robot are to steer itself to a given reference path, to produce an optimal predicted reference trajectory at each time instant, and to send out its information to all follower robots via broadcast communication. The first two tasks are simply handled by NMPC.

The path following problem is illustrated in Figure 5.3. The error kinematic model with respect to the path coordinate (4.4) is rewritten here

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & \kappa(s)\dot{s} & 0 \\ -\kappa(s)\dot{s} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} + \mathbf{u}_e \tag{5.6}$$

where $\mathbf{u}_e = \begin{bmatrix} -\dot{s} + u_R \cos\phi \\ u_R \sin\phi \\ \omega - \omega_b \end{bmatrix}$.

$\omega_b = \dot{\theta}_b, \mathbf{x}_e$ is the vector of the pose error with respect to the path coordinate, $\kappa(s)$ is the path curvature, and u_R is the desired translational velocity along the reference path. $\phi = \theta_t - \theta_d$ and $\theta_e = \theta - \theta_b$, where θ_t is the angle of the moving direction of the robot with respect to the world frame and θ_d is the orientation angle of the tangent to the reference curve. In this section, the desired orientation θ_b is simply equal to θ_d.

To drive the error $\mathbf{x}_e$ to zero, $\mathbf{x}$ and $\mathbf{u}$ in (2.18)-(2.20) are replaced by $\mathbf{x}_e$ and $\mathbf{u}_e$, respectively, and the terminal state feedback controller $\mathbf{u}^L = [u_1^L, u_2^L, u_3^L]^T$ is selected as follows:

$$u_1^L = -\alpha x_{eT} \quad u_2^L = -\beta y_{eT} \quad u_3^L = -\gamma \theta_{eT} \tag{5.7}$$

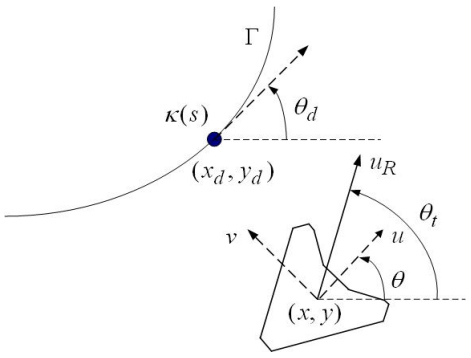


Figure 5.3: Illustration of the path following problem.

where $\mathbf{x}_e(t+T_p) = [x_{eT}, y_{eT}, \theta_{eT}]^T$, $\alpha \geq 0$, $\beta \geq 0$, and $\gamma \geq 0$. All weight parameters have to be selected such that (2.20) is satisfied (see (4.10) and (4.11)).

After the optimization problem at time t_k is solved, the current reference state ($s_{l,k}$), the optimal rate of progression at each time step in the future ($\dot{s}_{l,k|k}, \dot{s}_{l,k+1|k}, \ldots, \dot{s}_{l,k+T_p-1|k}$), and the sampling time $\delta_{l,k}$ are transmitted to all follower robots. Each data packet is time-stamped so that the *age* of the information can be extracted at a follower controller.

Next, we consider the trajectory tracking control in follower robots. The task for each follower robot is to track it own *estimated* reference trajectory, based on the leader robot's information, the predefined formation configuration, and the given reference path. In practice, some problems may arise, e.g., the information time delay is not zero, the sampling time of the follower robot can be different (asynchronous timing conditions) from that of the leader robot or the data packet can be lost. To overcome these problems, first we calculate the *age* of the received information and then estimate the robot's own reference trajectory with the velocity profiles, computed by using (5.4) and (5.5). In case of packet loss, the missing information can be filled in by using the previous information received from the leader robot.

The error kinematic model of each follower robot i with respect to the robot coordinate, adapted from [88] to omnidirectional mobile robots, can be given as

$$\dot{\mathbf{x}}_{i,e} = \begin{bmatrix} \dot{x}_{i,e} \\ \dot{y}_{i,e} \\ \dot{\theta}_{i,e} \end{bmatrix} = \begin{bmatrix} 0 & \omega_i & 0 \\ -\omega_i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{i,e} \\ y_{i,e} \\ \theta_{i,e} \end{bmatrix} + \mathbf{u}_{i,e} \tag{5.8}$$

where $\mathbf{u}_{i,e} = \begin{bmatrix} u_{i,R}\cos\theta_{i,e} - u_i \\ u_{i,R}\sin\theta_{i,e} - v_i \\ \omega_{i,R} - \omega_i \end{bmatrix}$.

$\mathbf{x}_{i,e}$ is the vector of the pose error with respect to the robot frame, $[u_i, v_i, \omega_i]^T$ is the vector of robot velocities. $u_{i,R}$ and $\omega_{i,R}$ are the desired translational and rotational velocities, respectively. Since $\mathbf{x}_{i,e}$ is required to converge to zero, $\mathbf{x}$ and $\mathbf{u}$ in (2.18)-(2.20) are replaced by $\mathbf{x}_{i,e}$ and $\mathbf{u}_{i,e}$, respectively. We select the terminal penalty and the feedback controller in the same way as we do for the leader robot.

5.3.3 Experimental Results

By our implementation, the internal clock of each robot has to be synchronized initially using clock synchronization. Furthermore, the sampling time is calculated based on the average past results [76]. Thus sampling time is varying. This leads us to work with asynchronous agents with different sampling time. This strategy allows agents to proceed at their own speed.

In this experiment, the leader robot was required to follow a lemniscate curve given by

$$x_d(t) = \frac{2.3\cos t}{1+(\sin t)^2} \quad y_d(t) = \frac{2.3\sin t\cos t}{1+(\sin t)^2}. \tag{5.9}$$

This curve was numerically parameterized by the path length s. All parameters used in our experiments for all robots are as follows: $Q = \mathrm{diag}\{1,1,1\}$, $R = \mathrm{diag}\{0.1,0.1,0.1\}$, $\delta = 0.07$ s, $N_p = T_p/\delta = 3$ steps, $\alpha = \beta = \gamma = 2$, where δ is a sampling time.

Figure 5.5 shows the superimposed snapshots of three mobile robots keeping and switching the formation, while the leader follows the reference path with the translational reference velocity u_R of 0.4 m/s and the rotational reference velocity ω_R of $\kappa_R u_R$, where κ_R is the curvature at the reference point. The formation is changed from a triangle (see Figure 5.2(a)) to a column (see Figure 5.2(b)) and then switched back to the triangle. The pose errors of the leader, of the follower 1, and of the follower 2 are shown in Figure 5.6(a), Figure 5.6(b), and Figure 5.6(c), respectively. The velocities of the leader, of the follower 1, and of the follower 2, compared with their reference velocities, are shown in Figure 5.7(a), Figure 5.7(b), and Figure 5.7(c), respectively. As seen in the results, the leader robot can follow the reference path with the desired translational velocity and the

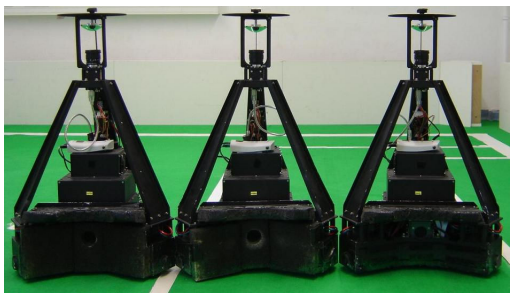


Figure 5.4: Omnidirectional mobile robots used in the formation control experiments.

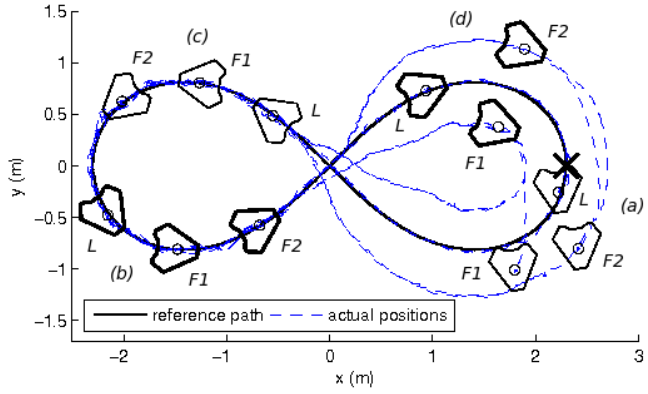
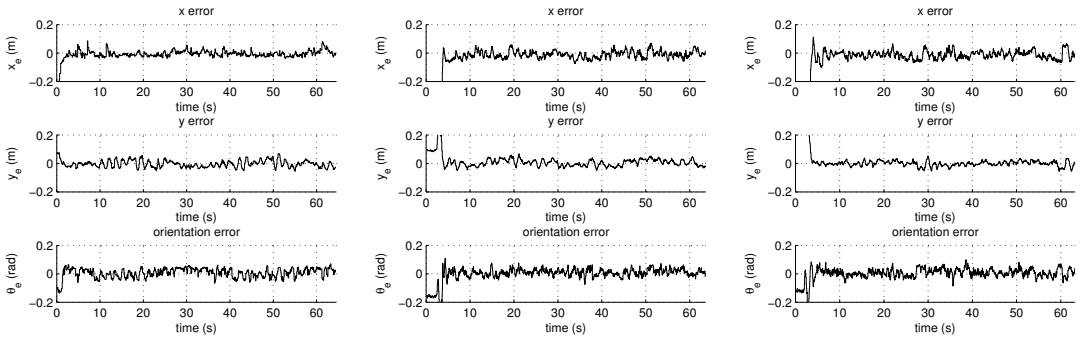
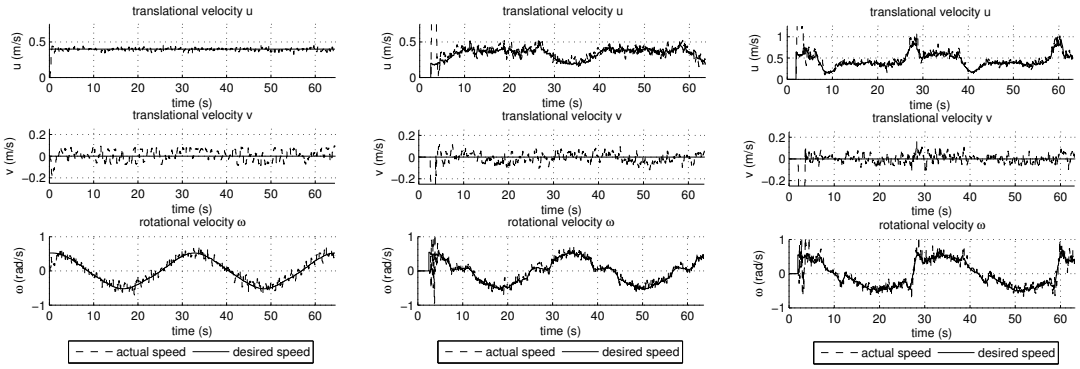


Figure 5.5: The snapshots are taken at the following time: (a) original configuration (thin line) at $t = 0$ s, (b) column configuration (thick line) at $t = 15.4$ s, (c) column formation (thin line) obtained at $t = 23.1$ s, and (d) triangle formation (thick line) obtained at $t = 39.5$ s. L denotes leader, $F1$ denotes the follower 1 and $F2$ denotes the follower 2. $\times$ denotes the starting position.



(a) (b) (c)

Figure 5.6: Pose errors of (a) the leader, (b) follower 1, and (c) follower 2.



(a) (b) (c)

Figure 5.7: Velocities of (a) the leader, (b) follower 1, and (c) follower 2.

follower robots can track the reference trajectory based on the leader robot's information. Also they can maintain the desired formation at any time.

5.4 Distributed MPC for Omnidirectional Mobile Robots

The solution in this section utilizes the distributed MPC scheme to formulate dynamically decoupled systems, in which the cost function of an optimization problem is coupled with neighbors. We adapt the strategies proposed in [65] to our coordinated path following problem. Each MPC controller associated with a different vehicle computes a solution to its local problem and exchanges the most recent information with its neighbors. In this case, there are two subproblems, i.e., (i) a path following problem: a robot is required to converge to a reference path, and (ii) a formation/coordination control problem: a group of robots is required to maintain a desired formation/coordination. To achieve these goals, both pose errors and formation errors are included into a local objective function,

which is minimized at each update time. In the formation control, the curvilinear abscissa s is used as a coupling term with neighboring robots.

We validate our distributed MPC on two scenarios. First, we want a group of omnidirectional mobile robots to move along only one reference path and also to keep a flexible formation. The flexible formation in this sense means that when a group of robots makes a turn an outer robot has to move faster while an inner robot has to move slower. Second, each member of the group is required to exchange information in order to maintain coordination, while following its own reference path. We formulate our problem according to the first scenario and then adapt it to the second scenario, as shown in the experimental results. Although very fast updates are currently not achieved with our implementation, experimental results show that our formation control strategy is promising to be further investigated.

5.4.1 Problem Formulation

We consider the problem of controlling N omnidirectional mobile robots, namely R_i where $i = 1,...,N$, to follow a reference path. In this work, only one reference path with a desired forward velocity u_o is prespecified. The reference path is parameterized by the curvilinear abscissa $s \in \mathbb{R}$. The individual coordinate of robot i can be defined as (s_i, q_i), where q_i is the offset distance perpendicular to s_i (see Figure 5.8). Thus, we can compute the desired pose of robot i by using the following equations:

$$\mathbf{x}_{i,r} = \begin{bmatrix} x_{i,r} \\ y_{i,r} \\ \theta_{i,r} \end{bmatrix} = \begin{bmatrix} x_{i,p} - q_i \sin\theta_{i,p} \\ y_{i,p} + q_i \cos\theta_{i,p} \\ \theta_{i,p} \end{bmatrix} \tag{5.10}$$

where $[x_{i,p},\ y_{i,p},\ \theta_{i,p}]$ is the state vector at s_i and $[x_{i,r},\ y_{i,r},\ \theta_{i,r}]$ is the vector of the reference pose at s_i with offset q_i (see Figure 5.9). Also, we can calculate the desired velocities by

$$u_{i,r} = u_o(1 - \kappa_{i,p}(s)q_i), \quad \omega_{i,r} = \kappa_i(l)u_{i,r} \tag{5.11}$$

where

$$\kappa_i(l) = \frac{\kappa_{i,p}(s)}{1 - q_i\kappa_{i,p}(s)} \tag{5.12}$$

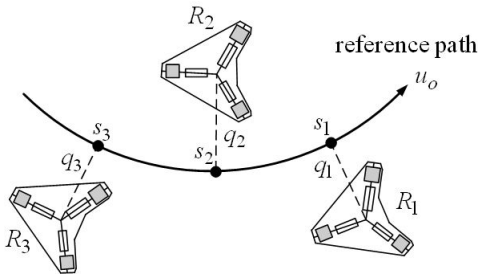


Figure 5.8: Example of a formation following a given reference path.

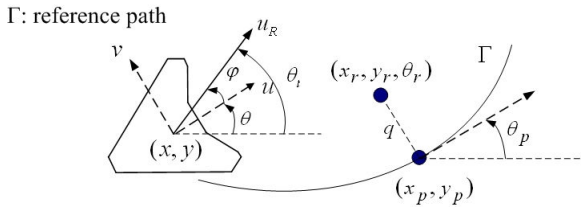


Figure 5.9: Illustration of the path following problem.

$u_{i,r}$ and $\omega_{i,r}$ are the desired translational and rotational velocities of robot i, respectively. $\kappa_i(l)$ and $\kappa_{i,p}(s)$ are the curvature of the robot's path and of the reference path, respectively. We locate a virtual vehicle at s_i, and $\dot{s}_i$ is defined as the velocity of a virtual vehicle moving along the reference path. However, in our case, robot i will follow this virtual vehicle with offset q_i. Then we have the relationship between the velocity of the virtual vehicle and the velocity of the robot's path as follows:

$$\dot{l}_i = \dot{s}_i(1 - \kappa_{i,p}(s)q_i) \tag{5.13}$$

It has to be noted that (5.10), (5.11) and (5.13) are not valid if q_i is not constant. Also, cusp or singularity at the robot's path is not considered in this work.

The formation graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, ..., N\}$ is the set of robots and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of relative vectors between robots. Two robots i and j are called neighbors if $(i, j) \in \mathcal{E}$, and the set of neighbors of robot i is denoted by $N_i \subseteq \mathcal{V}$. All graphs considered in this section are undirected and we assume that the undirected $\mathcal{G}$ is connected. The desired distance between two neighbors i and j can be determined by $p_{ij}(s)$, where $s_i + p_{ij} = s_j$ and $p_{ij} = -p_{ji}$. The formation error vector $E \in \mathbb{R}^M$, where M is the number of edges, has components $e_i \in \mathbb{R}$ defined as $e_i = s_i - s_j + p_{ij}$. Let E_i denote the vector of all components of E which have a coupled term with robot i. Then we have $E = C\mathbf{s} + \mathbf{p}$, where the vector $\mathbf{p} = [..., p_{ij}, ...]^T$, $s = [..., s_i, ...]^T$, and the matrix $C \in \mathbb{R}^{N \times N}$ is the incidence matrix [86].

This kinematic model can be formulated with respect to a Frenet frame moving along the reference path. Robot i will follow this virtual vehicle with offset q. Given the error state $\mathbf{x}_e$ between the robot state vector and the reference state vector and the kinematic model (3.1), the error state dynamic model expressed in the Frenet frame is derived as follows:

$$\begin{aligned} \dot{x}_e &= y_e \kappa(l)\dot{l} - \dot{l} + u_R \cos\phi \\ \dot{y}_e &= -x_e \kappa(l)\dot{l} + u_R \sin\phi \\ \dot{\phi} &= \dot{\theta}_t - \kappa(l)\dot{l} \end{aligned} \tag{5.14}$$

where $u_R = \sqrt{u^2 + v^2}$ is the forward speed, and $\varphi = \arctan\frac{v}{u} = \theta_t - \theta$ is the angle of the moving direction in the body frame.

Since translation and rotation of omnidirectional mobile robots can be controlled separately [31], we can drive the robot orientation θ to the desired orientation at the same

time. In this section, the desired orientation is the angle of the tangent direction to the reference path θ_r. The orientation error is defined as $\theta_e = \theta - \theta_r$. Then we have $\dot{\theta}_e = \omega - \omega_r$, where $\omega = \dot{\theta}$ and $\omega_r = \dot{\theta}_r$.

In the formation control problem, the forward speed of each robot has to be coordinated with its neighbors. Thus, we introduce a new state variable η, defined as $\eta = u_R - u_r$. Then we have $\dot{\eta} = \dot{u}_R - \dot{u}_r$. We define the control inputs of each robot as:

$$\mathbf{u}_e = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} -\dot{l} + (\eta + u_r)\cos\phi \\ \dot{\theta}_t - \kappa(l)\dot{l} \\ \omega - \omega_r \\ \dot{u}_R - \dot{u}_r \end{bmatrix} \tag{5.15}$$

Finally, the error state kinematic model becomes

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\phi} \\ \dot{\theta}_e \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} y_e \kappa_i(l)\dot{l} + u_1 \\ -x_e \kappa(l)\dot{l} + (\eta + u_r)\sin\phi \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{5.16}$$

5.4.2 Controller Design

The distributed cost function for each robot is defined as

$$L_i(\mathbf{x}_{i,e}, E_i, \mathbf{u}_{i,e}) = \mathbf{x}_{i,e}^T Q_i \mathbf{x}_{i,e} + \mathbf{u}_{i,e}^T R_i \mathbf{u}_{i,e} + \sum_{(i,j)\in\mathcal{E}} (W(s_i - s_j + p_{ij})^2) \tag{5.17}$$

where the term $(s_i - s_j + p_{ij})$ couples the states of neighboring robots and the deviation from the desired values is weighted by the positive definite matrices Q_i and R_i, and the positive constant W. Since each cost function depends on the neighbors' trajectories, each robot has to exchange an assumed trajectory with its neighbors at each update. Based on notations given in [65], over any prediction interval $\tau \in [t_k, t_k + T_p], k \in \mathbb{N}$, associated with current time t_k, for each robot we denote

$s_i^p(\tau; t_k)$: the predicted trajectory,
$s_i^*(\tau; t_k)$: the optimal predicted trajectory,
$\hat{s}_i(\tau; t_k)$: the assumed trajectory.

The corresponding robot state and control trajectories are denoted by $\mathbf{x}_{i,e}^p(\tau; t_k)$, $\mathbf{x}_{i,e}^*(\tau; t_k)$, $\hat{\mathbf{x}}_{i,e}(\tau; t_k)$ and $\mathbf{u}_{i,e}^p(\tau; t_k)$, $\mathbf{u}_{i,e}^*(\tau; t_k)$, $\hat{\mathbf{u}}_{i,e}(\tau; t_k)$, respectively. We concatenate the states and inputs into vectors as $\mathbf{x}_e = [\mathbf{x}_{1,e}, ..., \mathbf{x}_{N,e}]^T$ and $\mathbf{u}_e = [\mathbf{u}_{1,e}, ..., \mathbf{u}_{N,e}]^T$, respectively. The problem is to find

$$J^*(\mathbf{x}_{i,e}(t_k), E_i(t_k)) = \min_{u_i^p} J_i(\mathbf{x}_{i,e}(t_k), E_i(t_k), \mathbf{u}_{i,e}^p(\cdot; t_k)) \tag{5.18}$$

where

$$J_i(\mathbf{x}_{i,e}(t_k), E_i(t_k), \mathbf{u}^p_{i,e}(\cdot;t_k)) = V_i(\mathbf{x}^p_{i,e}(t_k+T_p;t_k)) + \int_{t_k}^{t_k+T_p} L_i(\mathbf{x}^p_{i,e}(\gamma;t_k), \hat{E}_i(\gamma;t_k), \mathbf{u}^p_{i,e}(\gamma;t_k))\, d\gamma$$

$$\begin{aligned}
\text{subject to: } & \dot{\mathbf{x}}^p_{i,e}(\tau;t_k) = \mathbf{f}_i(\mathbf{x}^p_{i,e}(\tau;t_k), \mathbf{u}^p_{i,e}(\tau;t_k)) \\
& \dot{\hat{\mathbf{x}}}_{i,e}(\tau;t_k) = \mathbf{f}_i(\hat{\mathbf{x}}_{i,e}(\tau;t_k), \hat{\mathbf{u}}_{i,e}(\tau;t_k)) \\
& \mathbf{u}^p_{i,e}(\tau;t_k) \in \mathcal{U}_i \;\; \forall \tau \in [t_k, t_k+T_c] \\
& \mathbf{x}^p_{i,e}(\tau;t_k) \in \mathcal{X}_i \;\; \forall \tau \in [t_k, t_k+T_p] \\
& \hat{\mathbf{x}}_{i,e}(\tau;t_k) \in \mathcal{X}_i \;\; \forall \tau \in [t_k, t_k+T_p] \\
& \mathbf{x}^p_{i,e}(t_k+T_p;t_k) \in \Omega_i \\
& \hat{\mathbf{x}}_{i,e}(t_k;t_k) = \mathbf{x}^p_{i,e}(t_k;t_k) = \mathbf{x}_{i,e}(t_k) \\
& |s^p_i(\tau;t_k) - \hat{s}_i(\tau;t_k)| \le \delta^2\mu \;\; \forall \tau \in [t_k, t_k+T_p]
\end{aligned}$$

$V_i(\mathbf{x}^p_{i,e}(t_k+T_p))$ is the decoupled terminal penalty and Ω_i is the terminal region of robot i, δ is the sampling time, and μ is a constant. T_c and T_p are the control horizon and the prediction horizon, respectively, with $T_c \le T_p$. $\mathcal{X}_i \subseteq \mathbb{R}^n$ and $\mathcal{U}_i \subseteq \mathbb{R}^m$ denote the set of feasible n dimensional states and m dimensional inputs of robot i, respectively.

The optimized trajectory s^p_i for robot i is constrained to be at most a distance of $\delta^2\mu$ from the assumed trajectory $\hat{s}_i$. The constraint is a means of enforcing a degree of consistency between what a robot plans to do and what neighbors believe that robot will plan to do. This constraint is called *compatibility constraint* proposed in [65]. Before giving any analysis, the following assumptions need to be made:

Assumption 1. Let $X_\Sigma \subset \mathbb{R}^{(n+1)N}$ denote the set of initial states $(\mathbf{x}_e(t), s(t_0))$, which can be steered to Ω by $\mathbf{u}^p_e(\tau;t) \in \mathcal{U}$, $\tau \in [t_0, t_0+T_p]$.

If Assumption 1 holds, the problem is feasible at initialization. The initial feasibility of the implementation implies subsequent feasibility, following the standard arguments in [41, 65, 163] by induction.

Assumption 2. There exists a constant $\rho_{\max} \in (0,\infty)$ such that $\|(\mathbf{x}^*_e(t;t_k), s^*(t;t_k)) - (\mathbf{x}^c_e, s^c)\| \le \rho_{\max}$ and $\|(\hat{\mathbf{x}}_e(t;t_k), \hat{s}(t;t_k)) - (\mathbf{x}^c_e, s^c)\| \le \rho_{\max}$, $\forall t \in [t_k, t_k+T_p]$, where $(\mathbf{x}^c_e, s^c)$ is the desired equilibrium state.

The symbol $\|\cdot\|$ denotes any vector norm in $\mathbb{R}^n$, and dimension n follows from the context. For any vector $x \in \mathbb{R}^n$, $\|x\|_P$ denotes the P-weighted 2-norm, defined by $\|x\|^2_P = x^T P x$, and P is any positive-definite real symmetric matrix.

If Assumption 2 holds, the optimal and assumed state trajectories remain bounded. Also, let $u_{\max}$ be the positive scalar constant $u_{\max} = \{\max \|v(t)\| \mid v(t) \in \mathcal{U}^N\}$. Then the implementation of the control algorithm can be given by Algorithm 5.1.

We note that Q_Σ and W_Σ, positive definite and symmetric, can be defined as follows: $Q_\Sigma = \text{diag}(Q_1, ..., Q_N)$, $W_\Sigma = [WC^TC]$, respectively. $\lambda_{\min}(Q_\Sigma)$ and $\lambda_{\max}(Q_\Sigma)$ denote the

Algorithm 5.1 The distributed MPC controller for any robot

Over any interval $[t_k, t_{k+1})$, $k \in \mathbb{N}$:
1. Apply $\mathbf{u}^*_{i,e}(\tau, t_k)$, $\tau \in [t_k, t_{k+1})$.
2. Compute $\hat{s}_i(\tau; t_{k+1}) = \hat{s}_i(\tau)$ as

$$\hat{s}_i(\tau) = \begin{cases} s^*_i(\tau; t_k), & \tau \in [t_{k+1}, t_k + T_p] \\ s_{i,eT}(\tau), & \tau \in [t_k + T_p, t_{k+1} + T_p] \end{cases}$$

   where $s_{i,eT}(\tau)$ is the solution from the terminal feedback controllers.
3. Transmit $\hat{s}_i(\cdot; t_{k+1})$ to every neighbor
4. Receive $\hat{s}_j(\tau; t_{k+1})$ from every neighbor j

At any time t_k, $k \in \mathbb{N}$:
1. Measure the current state $\mathbf{x}_{i,e}(t_k)$
2. Solve (5.18), yielding $\mathbf{u}^*_{i,e}(\tau, t_k)$, $\tau \in [t_k, t_k + T_p)$.

smallest and the largest eigenvalues of Q_Σ, respectively. Since we assume that our formation graph $\mathcal{G}$ is connected, the second smallest eigenvalue λ_2 of W_Σ (a weighted Laplacian of graph $\mathcal{G}$) is positive. The following characteristics of eigenvalues hold (see [86])

$$\lambda_2 = \min \frac{\mathbf{x}^T W_\Sigma \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \quad \lambda_{\max} = \max \frac{\mathbf{x}^T W_\Sigma \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

for any vector $\mathbf{x}$ and $\mathbf{x}^T W_\Sigma \mathbf{x} = \sum_{(i,j)\in\mathcal{E}} W(x_i - x_j)^2$.

Now we find the terminal penalty and terminal region, and then we will give the stability analysis of the distributed NMPC approach taken from [65] with slight adaptation to our path following problem.

Decoupled Terminal Controllers

In this section, a Lyapunov function for the decoupled terminal-state penalty is defined as follows:

$$V(\mathbf{x}_e(t_k + T_p)) = \tfrac{1}{2}\mathbf{x}_e(t_k + T_p)^T P \mathbf{x}_e(t_k + T_p) \tag{5.19}$$

where $P = \text{diag}(P_1, ..., P_N)$ is a positive definite matrix. Under the terminal-state controller $\mathbf{u}^L_e(t)$, the following condition is satisfied:

$$\dot{V}(\mathbf{x}_e(t)) + L(t, \mathbf{x}_e(t), \mathbf{u}_e(t)) \leq 0 \tag{5.20}$$

for any state $\mathbf{x}_e(t)$ belonging to the terminal region Ω. Then, the stability condition of each subsystem i becomes (from here in this subsection, we drop subscript i for conve-

nience)

$$\begin{aligned}
&\dot{V}(\mathbf{x}_e(t)+L(t,\mathbf{x}_e(t),\mathbf{u}_e(t))\\
&= p_{11}x_{eT}\dot{x}_{eT}+p_{22}y_{eT}\dot{y}_{eT}+p_{33}\varphi_{eT}\dot{\varphi}_{eT}+p_{44}\theta_{eT}\dot{\theta}_{eT}+p_{55}\eta_{eT}\dot{\eta}_{eT}+L(t,\mathbf{x}_e(t),\mathbf{u}_e(t))\\
&= p_{11}x_{eT}u_1^L+p_{22}y_{eT}(\eta_{eT}+u_r)\sin\varphi_{eT}+p_{33}\varphi_{eT}u_2^L+p_{44}\theta_{eT}u_3^L+p_{55}\eta_{eT}u_4^L+q_{11}x_{eT}^2\\
&\quad +q_{22}y_{eT}^2+q_{33}\varphi_{eT}^2+q_{44}\theta_{eT}^2+q_{55}\eta_{eT}^2+r_{11}u_1^{L^2}+r_{22}u_2^{L^2}+r_{33}u_3^{L^2}+r_{44}u_4^{L^2}\\
&\quad +w_T(s_{eT}-s^c)^2 \;\leq\; 0
\end{aligned}$$

where $w_T=\lambda_{\max}(W_\Sigma)$. By construction, $\lambda_{\max}(W_\Sigma)\mathbf{I}_{n\times n}\geq W_\Sigma$. The terminal state feedback controller $\mathbf{u}_e^L=[u_1^L,u_2^L,u_3^L,u_4^L]^T$ can be selected as follows: $u_1^L=-k_1x_{eT},\ u_2^L=-k_2\varphi_{eT},\ u_3^L=-k_3\theta_{eT},\ u_4^L=-k_4\eta_{eT}$, where $k_1,k_2,k_3,k_4\geq 0$. The stability condition is changed to

$$\begin{aligned}
&\dot{V}(\mathbf{x}_e(t))+L(t,\mathbf{x}_e(t),\mathbf{u}_e(t))\\
&= x_{eT}^2(-p_{11}k_1+q_{11}+k_1^2r_{11})+y_{eT}p_{22}(\eta_{eT}+u_r)\sin\varphi_{eT}+q_{22}y_{eT}^2\\
&\quad +\varphi_{eT}^2(-p_{33}k_2+q_{33}+k_2^2r_{22})+\theta_{eT}^2(-p_{44}k_3+q_{44}+k_3^2r_{33})\\
&\quad +\eta_{eT}^2(-p_{55}k_4+q_{55}+k_4^2r_{44})+w_T(s_{eT}-s^c)^2 \;\leq\; 0.
\end{aligned}$$

All weight parameters have to be selected such that (5.20) is satisfied. To have a negative derivative of the value function, the following requirement for the weight parameters is required:

$$\begin{aligned}
-p_{11}k_1+q_{11}+k_1^2r_{11}\leq 0 \qquad & -p_{33}k_2+q_{33}+k_2^2r_{22}\leq 0\\
-p_{44}k_3+q_{44}+k_3^2r_{33}\leq 0 \qquad & -p_{55}k_4+q_{55}+k_4^2r_{44}\leq 0
\end{aligned} \tag{5.21}$$

and the terminal-state region is defined as follows:

$$w_T(s_{eT}-s^c)^2+y_{eT}p_{22}(\eta_{eT}+u_r)\sin\varphi_{eT}+q_{22}y_{eT}^2<0 \tag{5.22}$$

Stability Analysis

The main idea, taken from [65] with slight changes, is to show that by applying Algorithm 5.1, the closed-loop state $(\mathbf{x}_e,s)$ converges to a neighborhood of the equilibrium state, with a sufficiently small upper bound on the update period $\delta_{\max}$. At any time t_k, the sum of the optimal distributed value functions is denoted

$$J_\Sigma^*(\mathbf{x}_e(t_k),s(t_k))=\sum_{i=1}^{N}J_i^*(\mathbf{x}_{i,e}(t_k),E_i(t_k)). \tag{5.23}$$

The following lemma gives a bounding result on the decrease in $J_\Sigma^*(\cdot)$ from one update to the next. The *compatibility constraints* are applied at each update times t_k with $k\geq 1$, thus the result holds for $k\in\{1,2,...\}$.

Lemma 5. (adapted from Lemma 4.3 of [65]) Suppose Assumptions 1 and 2 hold and $(\mathbf{x}_e(t_0),s(t_0))\in X_\Sigma$. Then, by the implementation of Algorithm 5.1 with the constant ξ defined by

$$\xi=W\mu(4\rho_{\max})2|\mathcal{E}|T_p \tag{5.24}$$

the function $J^*_\Sigma(\cdot)$ satisfies

$$J^*_\Sigma(\mathbf{x}_e(t_k+\delta),s(t_k+\delta))-J^*_\Sigma(\mathbf{x}_e(t_k),s(t_k))\leq-\int_{t_k}^{t_k+\delta}\sum_{i=1}^{N}L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k))d\gamma+\delta^2\xi.$$

Proof. For any $k\geq 1$,

$$J^*_\Sigma(\mathbf{x}_e(t_k),s(t_k))=V(\mathbf{x}^*_e(t_k+T_p;t_k))+\int_{t_k}^{t_k+T_p}\sum_{i=1}^{N}L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k),\mathbf{u}^*_{i,e}(\gamma;t_k))\,d\gamma$$

The optimal control is applied for $\delta\in(0,T_p]$ seconds, then at time $t_{k+1}=t_k+\delta$, we get a new state update $\mathbf{x}_{i,e}$ and s_i. A feasible control for (5.18) at time t_{k+1} is $\mathbf{u}^p_e(\cdot;t_{k+1})=\hat{\mathbf{u}}_e(\cdot;t_{k+1})$; therefore,

$$\begin{aligned}
J^*_\Sigma(\mathbf{x}_e(t_{k+1}),s(t_{k+1}))&\leq V(\hat{\mathbf{x}}_e(t_{k+1}+T_p;t_{k+1}))\\
&\quad+\int_{t_{k+1}}^{t_{k+1}+T_p}L(\hat{\mathbf{x}}_e(\gamma;t_{k+1}),\hat{E}(\gamma;t_{k+1}),\hat{\mathbf{u}}_e(\gamma;t_{k+1}))\,d\gamma\\
J^*_\Sigma(\mathbf{x}_e(t_{k+1}),s(t_{k+1}))-J^*_\Sigma(\mathbf{x}_e(t_k),s(t_k))&\leq\\
&\quad-\int_{t_k}^{t_{k+1}}\sum_{i=1}^{N}L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k),\mathbf{u}^*_{i,e}(\gamma;t_k))\,d\gamma\\
&\quad+\int_{t_{k+1}}^{t_k+T_p}\sum_{i=1}^{N}L_i(\hat{\mathbf{x}}_{i,e}(\gamma;t_{k+1}),\hat{E}_i(\gamma;t_{k+1}),\hat{\mathbf{u}}_{i,e}(\gamma;t_{k+1}))\,d\gamma\\
&\quad-\int_{t_{k+1}}^{t_k+T_p}\sum_{i=1}^{N}L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k),\mathbf{u}^*_{i,e}(\gamma;t_k))\,d\gamma\\
&\quad+\int_{t_k+T_p}^{t_{k+1}+T_p}\sum_{i=1}^{N}L_i(\hat{\mathbf{x}}_{i,e}(\gamma;t_{k+1}),\hat{E}_i(\gamma;t_{k+1}),\hat{\mathbf{u}}_{i,e}(\gamma;t_{k+1}))\,d\gamma\\
&\quad+V(\hat{\mathbf{x}}_e(t_{k+1}+T_p;t_{k+1}))-V(\mathbf{x}^*_e(t_k+T_p;t_k))
\end{aligned}$$

Because of $\hat{\mathbf{x}}_e(t_k+T_p;t_{k+1})=\mathbf{x}^*_e(t_k+T_p;t_k)$, $\hat{\mathbf{x}}_e(\tau;t_{k+1})$ obtained by the terminal feedback controllers for $\tau\in[t_k+T_p,t_{k+1}+T_p]$, (5.19) and (5.20), the sum of the last three terms in the inequality above is nonpositive, and thus the inequality holds after removing these three terms. Because of $L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k),\mathbf{u}^*_{i,e}(\gamma;t_k))\geq L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k))$, the lemma has been proven if we can prove that

$$\begin{aligned}
\int_{t_{k+1}}^{t_k+T_p}\sum_{i=1}^{N}\{&L_i(\hat{\mathbf{x}}_{i,e}(\gamma;t_{k+1}),\hat{E}_i(\gamma;t_{k+1}),\hat{\mathbf{u}}_{i,e}(\gamma;t_{k+1}))\\
&-L_i(\mathbf{x}^*_{i,e}(\gamma;t_k),\hat{E}_i(\gamma;t_k),\mathbf{u}^*_{i,e}(\gamma;t_k))\}\,d\gamma\leq\delta^2\xi.
\end{aligned}$$

Because of $\hat{\mathbf{x}}_{i,e}(\gamma;t_{k+1})=\mathbf{x}^*_{i,e}(\gamma;t_k)$ and $\hat{\mathbf{u}}_{i,e}(\gamma;t_{k+1})=\mathbf{u}^*_{i,e}(\gamma;t_k)$, for $\gamma\in[t_{k+1},t_k+T_p]$ the integrand above is equal to

$$\sum_{i=1}^{N}\sum_{j\in N_i}W_i\{(s^*_i(\gamma;t_k)-s^*_j(\gamma;t_k)+p_{ij})^2-(s^*_i(\gamma;t_k)-\hat{s}_j(\gamma;t_k)+p_{ij})^2\}$$

Using some algebraic calculation, we have

$$\begin{aligned}&(s_i^*(\gamma;t_k)-s_j^*(\gamma;t_k)+p_{ij})^2-(s_i^*(\gamma;t_k)-\hat{s}_j(\gamma;t_k)+p_{ij})^2\\&\quad=(s_j^*(\gamma;t_k)-\hat{s}_j(\gamma;t_k))(-2(s_i^*(\gamma;t_k)+p_{ij})+s_j^*(\gamma;t_k)+\hat{s}_j(\gamma;t_k))\\&\quad\leq\delta^2\mu(4\rho_{\max})\end{aligned}$$

where we use $p_{ij}=s_j^c-s_i^c$, Assumption 2, and the compatibility constraint. Then we have

$$\delta^2\mu W\int_{t_{k+1}}^{t_k+T_p}\sum_{i=1}^{N}\sum_{j\in N_i}(4\rho_{\max})d\gamma\leq\delta^2\xi$$

with the total number of pairwise neighbors $|\mathcal{E}|=\sum_{i=1}^{N}\sum_{j\in N_i}(1/2)$. This concludes the proof. ■

Now the result has been bounded by using Lemma 5. We still have to show that the closed-loop state trajectory converges to a closed neighborhood of the objective state. The neighborhood of convergence is a level set of the function $J_\Sigma^*(\mathbf{x}_e(t),s(t))$. We firstly define $z_i(t)=(\mathbf{x}_{i,e}(t),s_i(t))$ and we then have $\|z-z^c\|_G^2$, where

$$G=\begin{bmatrix}Q_\Sigma & 0\\ 0 & W_\Sigma\end{bmatrix}\tag{5.25}$$

and $z^c=(\mathbf{x}_e^c,s^c)$ is the desired equilibrium state.

According to [65], we define $\Omega_\beta=\{z\in\mathbb{R}^{(n+1)N}\mid J_\Sigma^*(\mathbf{x}_e(t),s(t))\leq\beta\}$ with constant $\beta\in(0,\infty)$ as the compact level set. The set Ω_β is in the interior of X_Σ if $\beta>0$ is sufficiently small. We can choose a constant $r=r(\beta)\in(0,\rho_{\max})$ with the following properties:

$$B(z^c;r)\subseteq\Omega_\beta/2\quad\text{and}\quad r^2\leq\tfrac{8\beta}{\lambda_{\min}(Q_\Sigma)}\tag{5.26}$$

where $B(z^c;r)$ denotes a closed ball in $\mathbb{R}^{(n+1)N}$ with center z^c and radius r. We require the following assumptions (see [65]).

Assumption 3. The following holds: (a) The update period is sufficiently small that the following first-order Taylor series approximation is valid:

$$\sum_{i=1}^{N}L_i(\mathbf{x}_{i,e}^*(\gamma;t_k),\hat{E}_i(\gamma;t_k))\approx\|z(t_k)-z^c\|_G^2+2(\gamma-t_k)(z(t_k)-z^c)^TGf(z(t_k),\mathbf{u}_e^*(t_k;t_k))$$

for all $\gamma\in[t_k,t_k+\delta]$ and any $k\in\mathbb{N}$; (b) there exists a Lipschitz constant $\mathcal{K}\in[1,\infty)$ such that for any $z,z'\in X_\Sigma,u,u'\in\mathcal{U}^N$,

$$\|f(z,u)-f(z',u')\|\leq\mathcal{K}(\|z-z'\|+\|u-u'\|)$$

Assumption 4. The following holds:

$$\begin{aligned}\lambda_{\min}(Q_\Sigma)&\leq\lambda_2(W_\Sigma)\\\lambda_{\max}(Q_\Sigma)&\leq\lambda_{\max}(W_\Sigma)\end{aligned}$$

Then, the theorem taken from [65] with slight changes can now be given.

Theorem 7. *(taken from Theorem 1 of [65]) Suppose Assumptions 1-4 hold, $z(t_0) \in X_\Sigma$ and for a given constant $\beta \in (0,\infty)$ with $\Omega_\beta \subset X_\Sigma$, the constant $r = r(\beta) \in (0, \rho_{max})$ is such that the properties in (5.26) are satisfied. Then by implementation of Algorithm 5.1 with*

$$\delta_{max} = \frac{(r/2)^2 \lambda_{min}(Q_\Sigma)}{\xi + \mathcal{K}\rho_{max}(\rho_{max}+u_{e,max})\lambda_{max}(W_\Sigma)} \tag{5.27}$$

and ξ given by (5.24), the closed-loop state trajectory enters $B(z^c;r)$ in finite time and remains in Ω_β for all future time.

Proof. Substituting the Taylor series expressions to:

$$J^*_\Sigma(\mathbf{x}_e(\tau), s(\tau)) - J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) \leq \int_{t_k}^{\tau} \sum_{i=1}^{N} L_i(\mathbf{x}^*_i(\gamma;t_k), \hat{E}_i(\gamma;t_k))\, d\gamma + \delta^2 \xi$$

$\forall \tau \in (t_k, t_k+\delta]$, for any constant $\delta \in (0, \delta_{\max}]$, we have

$$J^*_\Sigma(\mathbf{x}_e(\tau), s(\tau)) - J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) \leq -(\tau - t_k)\|z(t_k) - z^c\|^2_G + (\tau - t_k)^2 H$$

where $H = -(z(t_k) - z^c)^T G f(z(t_k), \mathbf{u}^*_e(t_k;t_k))$ has the upper bound
$H \leq \|z(t_k) - z^c\|\, \|f(z(t_k), \mathbf{u}^*_e(t_k;t_k))\| \lambda_{\max}(W_\Sigma) \leq \rho_{\max}\mathcal{K}(\rho_{\max} + u_{\max})\lambda_{\max}(W_\Sigma)$.
Because of $\tau - t_k \leq \delta \leq \delta_{\max}$, we have

$$\begin{aligned} J^*_\Sigma(\mathbf{x}_e(\tau), s(\tau)) - J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) &\leq -(\tau - t_k)\|z(t_k) - z^c\|^2_G + \delta\delta_{\max}(H + \xi) \\ &\leq -(\tau - t_k)\lambda_{\min}(Q_\Sigma)\|z(t_k) - z^c\|^2 + \delta\delta_{\max}(H + \xi) \\ &\leq -\lambda_{\min}(Q_\Sigma)\{(\tau - t_k)\|z(t_k) - z^c\|^2 - \delta(r/2)^2\} \\ &\leq -\delta\lambda_{\min}(Q_\Sigma)\{\|z(t_k) - z^c\|^2 - (r/2)^2\} \end{aligned}$$

with $\tau = t_k + \delta = t_{k+1}$. From this inequality, there exists a finite integer $l \geq 1$ such that $z(t_l) \in B(z^c;r)$. If this were not the case, the inequality implies $J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) \to -\infty$ as $k \to \infty$. Since the cost functions are nonnegative, $J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) \geq 0$ for any $z(t_k) \in X_\Sigma$. Therefore, by contradiction, there exists a finite integer $l \geq 1$ such that $z(t_l) \in B(z^c;r) \subseteq \Omega_{\beta/2}$. This verifies the first statement of the theorem. Now, there are two cases to be proved that $(\mathbf{x}_e(t), s(t)) \in \Omega_\beta$ for all time $t \geq t_l$. First, if $(z(t_k)) \in \Omega_{\beta/2} \setminus B(z^c; r/2)$, then $z(t) \in \Omega_\beta$ for all time $t \in [t_k, t_{k+1}]$ and $z(t_{k+1}) \in \Omega_{\beta/2}$. It is first shown that the upper bound becomes

$$J^*_\Sigma(\mathbf{x}_e(\tau), s(\tau)) - J^*_\Sigma(\mathbf{x}_e(t_k), s(t_k)) \leq \delta_{\max}\lambda_{\min}(Q_\Sigma)(r/2)^2$$

for all $\tau \in (t_k, t_{k+1}]$ and $\delta_{\max} < 1/4$, since

$$\delta_{\max} < \frac{(r/2)^2 \lambda_{\min}(Q_\Sigma)}{\rho_{\max}\mathcal{K}(\rho_{\max}+u_{\max})\lambda_{\max}(W_\Sigma)} \leq \frac{(r/2)^2}{\rho^2_{\max}} \quad .$$

Therefore, the bound on J_Σ^* becomes

$$J_\Sigma^*(\mathbf{x}_e(\tau), s(\tau)) \leq J_\Sigma^*(\mathbf{x}_e(t_k), s(t_k)) + \frac{\lambda_{\min}(Q_\Sigma)(r/2)^2}{4} \leq \beta$$

for all $\tau \in (t_k, t_{k+1}]$, using $J_\Sigma^*(\mathbf{x}_e(t_k), s(t_k)) \leq \beta/2$ and (5.26). Likewise, $z(t_k) \in \Omega_{\beta/2} \setminus B(z^c; r/2)$ and (5.26) imply that $J_\Sigma^*(\mathbf{x}_e(t_{k+1}), s(t_{k+1})) < J_\Sigma^*(\mathbf{x}_e(t_k), s(t_k))$ and so $z(t_{k+1}) \in \Omega_{\beta/2}$. In the second case, if $z(t_k) \in B(z^c; r/2)$, then $z(t) \in B(z^c; r) \subseteq \Omega_{\beta/2}$ for all time $t \in [t_k, t_{k+1}]$. From the bounding argument, we have

$$\begin{aligned}
\|z(t) - z^c\| &\leq \|z(t_k) - z^c\| + \| \int_{t_k}^{t} (f(z(\gamma), \mathbf{u}_e(\gamma)))\, d\gamma \| \\
&\leq r/2 + (t - t_k)\mathcal{K}(\rho_{\max} + u_{\max}) \\
&\leq r/2 + \delta_{\max}\mathcal{K}(\rho_{\max} + u_{\max})
\end{aligned}$$

for all time $t \in [t_k, t_{k+1}]$, and $\delta_{\max}\mathcal{K}(\rho_{\max} + u_{\max}) \leq \delta_{\max}\mathcal{K}(\rho_{\max} + u_{\max}) \leq \frac{(r/2)^2\lambda_{\min}(Q_\Sigma)}{\rho_{\max}(\lambda_{\max}(W_\Sigma))}$ $< \frac{r}{2}$. Combining two cases above, we have shown the following: There exists a finite update time t_l such that $z(t_l) \in B(z^c; r) \subset \Omega_\beta$; at any subsequent update time $t_k, k > l, z(t_k) \in \Omega_{\beta/2} \subset \Omega_\beta$; finally, for any two subsequent update times t_k and t_{k+1}, with $k \geq l$, $z(t) \in \Omega_\beta$ for all time $t \in [t_k, t_{k+1}]$. This concludes the proof. ■

The theorem guarantees that, by implementation of Algorithm 5.1 with $\delta_{\max}$ given by (5.27), the closed-loop state trajectory enters the closed ball $B(z^c; r)$ in finite time and remains in the level set Ω_β for all future time [65].

5.4.3 Experimental Results

In the first experiment, three omnidirectional mobile robots, shown in Figure 5.4, were required to follow an ellipse Γ: $x_p(t) = 1.5\cos(t)$, $y_p(t) = 1.0\sin(t)$ (it was rotated $\pi/6$ rad) with a desired forward speed $u_o = 0.4$ m/s and to keep a flexible triangle formation (see Figure 5.8). The formation error vector was given as $p_{12} = p_{23} = -0.4$ m, $p_{21} = p_{32} = 0.4$ m. The offset distances from the reference path were defined by $q_1 = -0.4$ m, $q_2 = 0.4$ m, and $q_3 = -0.4$ m.

However, using hard constraints can make the numerical solution difficult. To avoid this, we enforced the terminal constraints through the cost function as soft constraints. The average sampling time δ was able to be achieved at approximately 0.12 s. All parameters are listed as follows:

$Q_i = \text{diag}(0.05, 0.05, 0.0001, 0.0001, 0.05)$, $R_i = \text{diag}(10^{-5}, 10^{-5}, 10^{-5}, 5 \cdot 10^{-6})$,
$P_i = \text{diag}(0.12, 0.12, 0.02, 0.02, 0.12)$, $k_i = \text{diag}(0.5, 0.5, 0.5, 0.5)$,
$W_i = 0.05$, $\mu = 10$, $T_p = T_c = 3\delta$, $v_{\min} = -1.9$ m/s, $v_{\max} = 1.9$ m/s.

Figure 5.10 shows the superimposed snapshots of three omnidirectional mobile robots following the reference and keeping the flexible triangle formation. As seen from the results, the outer robot moves faster and the inner robot moves slower when the formation

makes a turn. Figure 5.11 shows the formation error from the experiments. The pose errors of all three robots are shown in Figure 5.12, while the velocities of all robots, compared with their reference velocities, are shown in Figure 5.13.

In the second experiment, we have a set of N omnidirectional mobile robots and a set of N spatial reference paths Γ_i, where $i = 1, ..., N$ and require that robot R_i follows path Γ_i. In order to maintain the motion coordination of the whole group, the speeds at which the mobile robots are required to travel can be handled in many ways. We propose to employ morphing. Morphing is a technique used to generate a sequence of configurations that transform a source configuration into a target configuration. In our problem, we assume that the reference path for each mobile robot is prespecified and collision avoidance is taken into account in the phase of generating these reference paths. Then, we determine the base reference path Γ_B, which can be taken either from one of reference paths or from a new developed reference path, and specify the base forward speed $u_{B,d}$, which is the base transition rate in the morphing technique. With the relation between the total length of the base reference path $P(\Gamma_B)$ and the total length of the reference path of mobile robot i, $P(\Gamma_i)$, the desired forward speed of mobile robot i is defined by

$$u_{i,d} = \frac{P(\Gamma_i)}{P(\Gamma_B)} u_{B,d} \tag{5.28}$$

where $u_{i,d}$ is the desired forward speed of mobile robot i on the reference path Γ_i. Let ψ_i and σ_i represent s_i and $u_{i,d}$ in the base reference coordinate, respectively. Both ψ_i and σ_i can be computed as follows:

$$\psi_i = \frac{P(\Gamma_B)}{P(\Gamma_i)} s_i \quad \sigma_i = \frac{P(\Gamma_B)}{P(\Gamma_i)} u_{i,d} \tag{5.29}$$

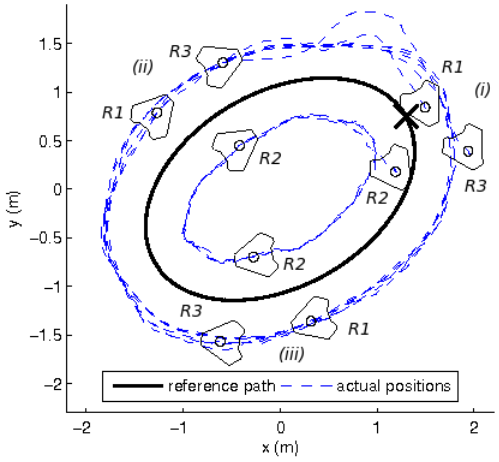


Figure 5.10: The snapshots are taken at the following time (i) initial position at t = 0 s, (ii) the formation obtained at 8.4 s, and (iii) the formation still maintained at t = 18.3 s. The solid-line ellipse is the given reference path and $\times$ denotes the starting point.

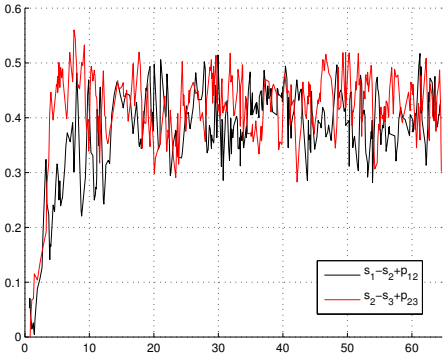
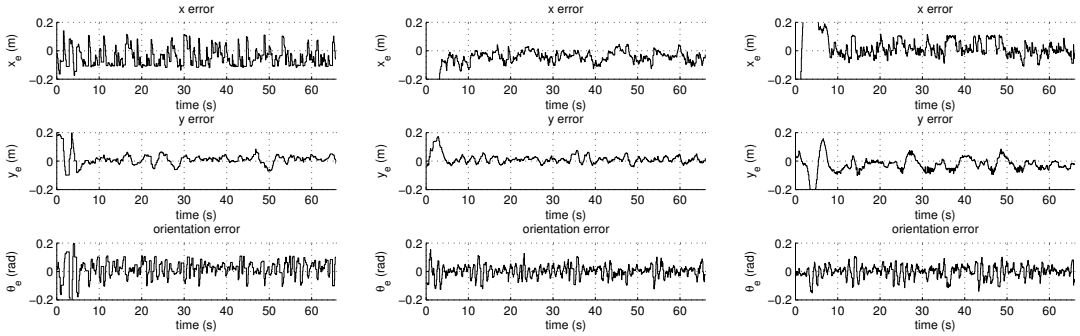
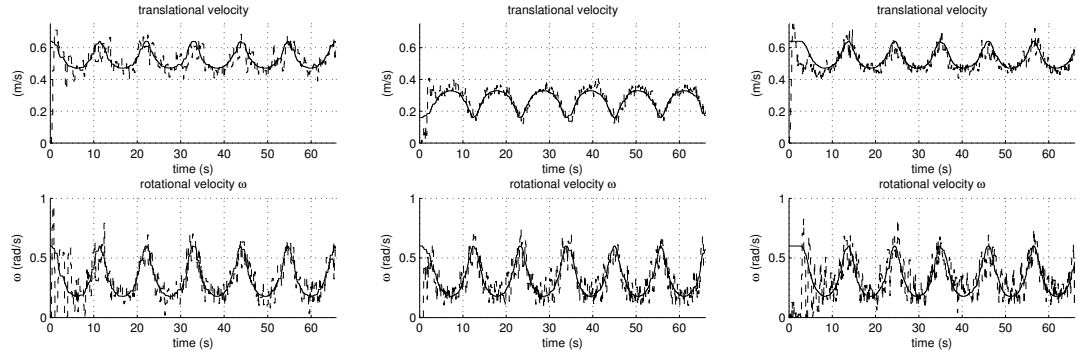


Figure 5.11: The formation errors of R_1 and R_2, and of R_2 and R_3.



(a) (b) (c)

Figure 5.12: Experiment 1: pose errors of (a) R_1, (b) R_2 and (c) R_3.



(a) (b) (c)

Figure 5.13: Experiment 1: velocities of (a) R_1, (b) R_2, and (c) R_3.

If σ_i is equal to σ_B, where $i = 1,...,N$, then the desired formation can be kept. Indeed, the base transition rate σ_B is not necessarily constant at each intermediate configuration. However, if it is kept constant, the path following problem increases the difficulty in following a sharp turning curve. Figure 5.14 illustrates superimposed snapshots, in which two omnidirectional mobile robots are required to follow their reference paths and also to keep inter-vehicle coordination. $P(\Gamma_i)$, specified as the total length of the ith reference path from the source configuration to the target configuration or as the path length corresponding to the path length of one period of the base reference path, can be computed by either using an analytical method or a numerical integration. Alternatively, paths Γ_B and Γ_i can be reparameterized so that $\frac{\partial s_i}{\partial \psi_i}$ can be obtained. The desired distance between two neighbors i and j can be determined by $p_{ij}(\psi)$, where $\psi_i + p_{ij} = \psi_j$ and $p_{ij} = -p_{ji}$. Then, an unweighted adjacency matrix A of graph $\mathcal{G}$ is used to describe the formation configuration.

In the experiment, three omnidirectional mobile robots were required to follow their own reference paths and to spend the same amount of time in each period. The reference paths were given by

$$\Gamma_1 : x_{1,r}(s_1) = 1.5\cos(s_1), \;\; y_{1,r}(s_1) = 1.0\sin(s_1) \tag{5.30}$$

$$\Gamma_2 : x_{2,r}(s_2) = 1.5\cos(s_2), \;\; y_{2,r}(s_2) = 1.0\sin(s_2) \tag{5.31}$$

$$\Gamma_3 : x_{3,r}(s_3) = 1.0\cos(s_3), \;\; y_{3,r}(s_3) = 1.0\sin(s_3) \tag{5.32}$$

The reference paths in (5.30), (5.31), and (5.32) were rotated $\pi/6$, $-\pi/6$, and $\pi/2$ rad, respectively. Γ_3 was selected as the base reference path Γ_B with the desired forward speed σ_B of 0.4 m/s. Adjacency Matrix A was set to $p_{12} = p_{21} = p_{23} = p_{32} = 0$. Figure 5.15 shows the superimposed snapshots of three omnidirectional mobile robots. Figure 5.16(a) shows the distance between two neighbors, expressed in the base reference

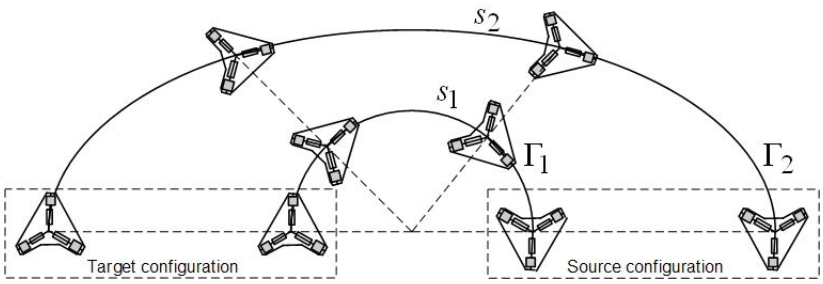
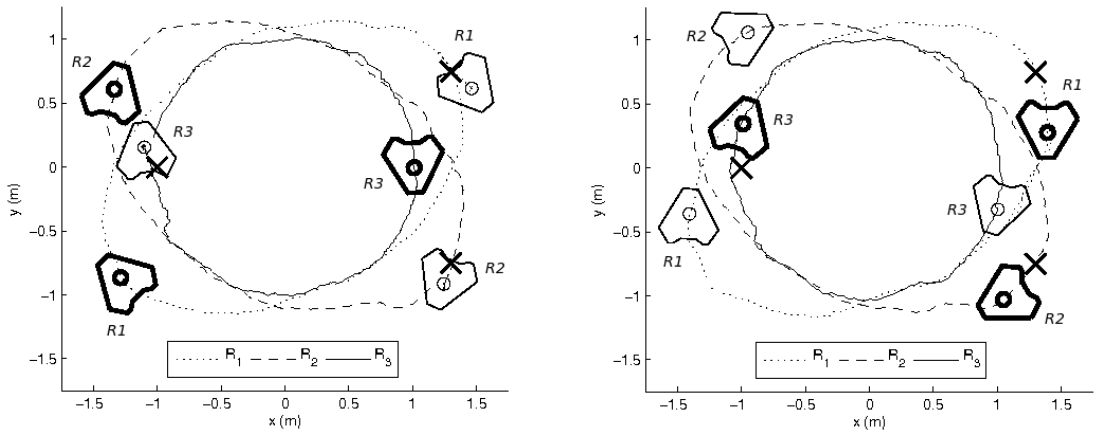


Figure 5.14: An example illustrates the transition from a source configuration to a target configuration of two omnidirectional mobile robots.



(a) (b)

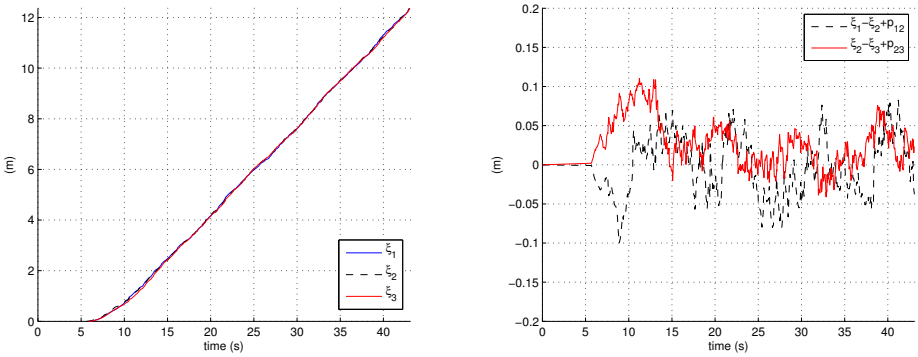
Figure 5.15: The snapshots of the intermediate configurations are taken at the following time: (a) thin-line robots obtained at t = 0 s, thick-line robots obtained at 15.4 s, and (b) thick-line robots obtained at t = 22.8 s, thin-line robots obtained at t = 31.7 s. $\times$ denotes the starting point.

coordinate and Figure 5.16(b) shows the coordination error, where the neighbor of R_1 is R_2, the neighbors of R_2 are R_1 and R_3, and the neighbor of R_3 is R_2. The pose errors of three mobile robots are shown in Figure 5.17, while their forward speeds and rotational speeds, compared with their reference speeds, are shown in Figure 5.18.

As seen from the results of both experiments, all team members were able to maintain motion coordination during following reference paths. The position errors were less than 0.15 m, while the orientation errors were less than 0.1 rad. To achieve the desired formation, the path variable was used as a coupling term between neighboring robots and the path derivative was explicitly controlled through the distributed MPC framework. The coordination errors were less than ± 0.1 m as seen in Figure 5.11 and Figure 5.16(b).

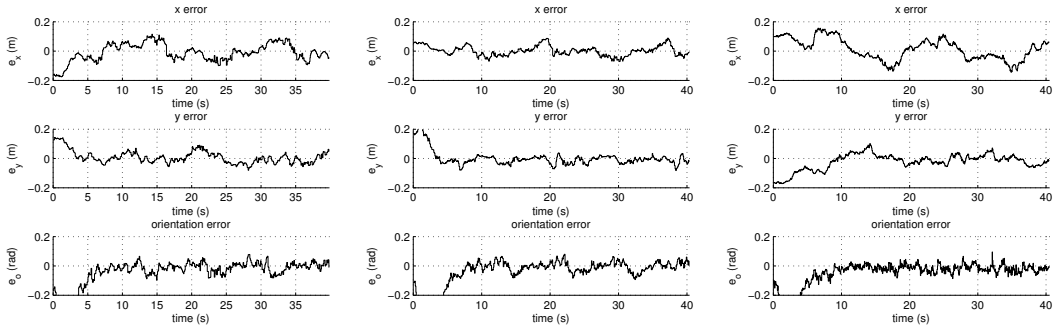
5.5 Coordinated Path Following for Unicycle Mobile Robots

In this section, we develop a control law based on a virtual vehicle approach for coordinated path following of a group of N unicycle mobile robots. The controller is designed in such a way that the derivative of the path parameter is used as an additional control input to synchronize the formation motion. This coordinated path-following problem can be divided into two subproblems, i.e., the path following control problem and the coordination problem. In this work, we derive our control law based on a Lyapunov function candidate and a consensus algorithm for a kinematic model of mobile robots.



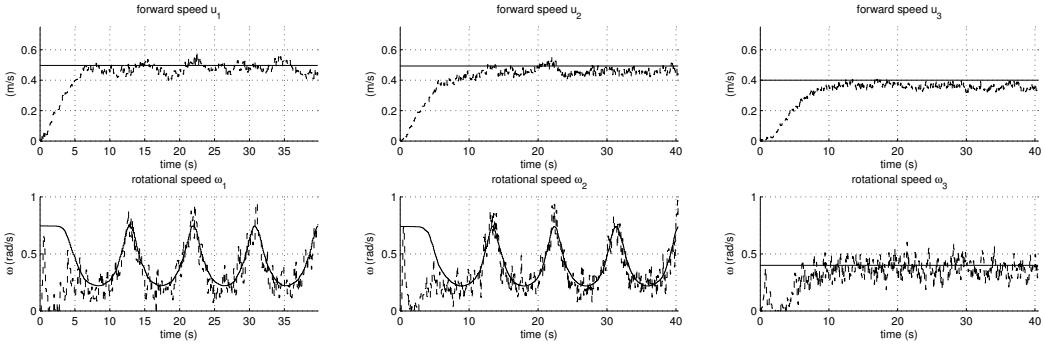
(a) (b)

Figure 5.16: (a) the distances between two neighbors expressed in the base reference coordinate system and (b) the coordination errors of R_1 and R_2 and of R_2 and R_3 expressed in the base reference coordinate system.



(a) (b) (c)

Figure 5.17: Experiment 2: pose errors of (a) R_1, (b) R_2 and (c) R_3.



(a) (b) (c)

Figure 5.18: Experiment 2: velocities of (a) R_1, (b) R_2, and (c) R_3.

Both path errors and coordination errors are considered in the Lyapunov function and the path parameter is used to synchronize coordination motions via a second-order consensus protocol with a reference velocity under undirected information exchange with connectivity assumption.

5.5.1 Problem Formulation

We consider a group of N mobile robots, each of which has the kinematic equations given by (3.3). We first consider path following for each mobile robot in the formation, i.e., we wish to find control law v_i and ω_i of robot i such that the robot follows a virtual vehicle with position $\mathbf{x}_{di} = [x_{di}, y_{di}, \theta_{di}]^T$ and inputs v_{di} and ω_{di}. A unicycle-type mobile robot is depicted in Figure 5.19, together with a spatial path Γ_i to be followed. The path error with respect to the robot frame is given by

$$\begin{bmatrix} x_{ei} \\ y_{ei} \\ \theta_{ei} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 \\ -\sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{di} - x_i \\ y_{di} - y_i \\ \theta_{di} - \theta_i \end{bmatrix} . \tag{5.33}$$

Then, the error dynamics are

$$\begin{aligned} \dot{x}_{ei} &= y_{ei}\omega_i - v_i + \dot{s}_i \cos\theta_{ei} \\ \dot{y}_{ei} &= -x_{ei}\omega_i + \dot{s}_i \sin\theta_{ei} \\ \dot{\theta}_{ei} &= \kappa_i \dot{s}_i - \omega_i \end{aligned} \tag{5.34}$$

where κ_i is the path curvature and $\dot{s}_i$ is the velocity of a virtual vehicle. It is bounded by $0 \leq \dot{s}_i \leq \dot{s}_{i,\max}$.

Next, we consider the coordination problem. To maintain the motion coordination of the whole group, each robot requires an individual parameterized path so that when all path parameters are synchronized, all robots will be in formation. The velocities at which the mobile robots are required to travel can be handled in many ways. In this section, there are three velocities to be synchronized, i.e., the velocity v_0 (or v_{di} in the robot frame), specifying how fast the whole group of robots should move, the velocity $\dot{s}_i$, denoting how fast an individual virtual vehicle moves along the path, and the velocity v_i, determining how fast an individual real mobile robot travels (see Figure 5.20).

5.5.2 Controller Design

Define the following variable

$$\dot{\tilde{s}}_i = \dot{s}_i - v_{di} \tag{5.35}$$

where $\dot{\tilde{s}}_i$ represents the formation speed tracking error of robot i. Let us choose

$$V = \frac{1}{2}\sum_{i=1}^{N}\left[x_{ei}^2 + y_{ei}^2 + \frac{1}{k_1}\left(\theta_{ei} - \delta_i(y_{ei}, v)\right)^2 + \dot{\tilde{s}}_i^2 + k_2\tilde{s}_i^2\right] \tag{5.36}$$

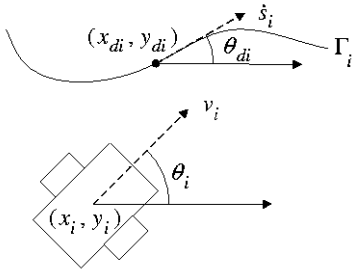


Figure 5.19: A graphical representation of a unicycle mobile robot and a path.

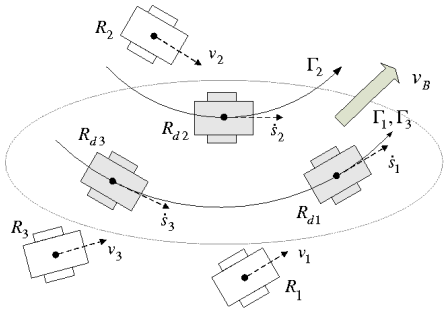


Figure 5.20: A graphical representation of coordinated path following.

as a candidate Lyapunov function, where k_1 and k_2 are positive gains. $\bar{s}_i = \sum_{j\in N_i}(s_i - s_j - s_{dij})$ is the coordination error of robot i and s_{dij} is the desired distance between two neighbors i and j. The function $\delta_i(y_{ei}, v)$ can be interpreted as the desired value for the orientation θ_{ei} during transients [161]. It is assumed that $\lim_{t\to\infty} v(t) \neq 0$, $\delta_i(0,v) = 0$, and $y_{ei} v \sin(\delta_i) \leq 0, \forall y_{ei} \forall v$. The function $\delta_i(y_{ei}, v)$ taken from [211] is $\delta_i(y_{ei}, v) = -\text{sign}(v_{di})\theta_a \tanh y_{ei}$ with $\theta_a = \frac{\pi}{4}$.

The derivative of V can be computed to give

$$\dot{V} = \sum_{i=1}^{N}\left[x_{ei}\dot{x}_{ei} + y_{ei}\dot{y}_{ei} + \frac{1}{k_1}(\theta_{ei} - \delta_i)\left(\dot{\theta}_{ei} - \dot{\delta}_i\right) + \dot{\tilde{s}}_i\ddot{\tilde{s}}_i + k_2\bar{s}_i\dot{\bar{s}}_i\right]. \tag{5.37}$$

We first design a controller to stabilize the x_{ei}, y_{ei}, and θ_{ei} dynamics. Substituting (5.35) into (5.34), adding $y_{ei}v_{di}\sin\delta_i - y_{ei}v_{di}\sin\delta_i$ to (5.37), the time derivative along the solutions of (5.34) yields

$$\begin{aligned}\dot{V} = \sum_{i=1}^{N}\Bigg[&x_{ei}\left(y_{ei}\omega_i - v_i + (\dot{\tilde{s}}_i + v_{di})\cos\theta_{ei}\right) + y_{ei}\left(-x_{ei}\omega_i + (\dot{\tilde{s}}_i + v_{di})\sin\theta_{ei}\right)\\ &+ \frac{1}{k_1}(\theta_{ei} - \delta_i)\left(\dot{\theta}_{ei} - \dot{\delta}_i\right) + y_{ei}v_{di}\sin\delta_i - y_{ei}v_{di}\sin\delta_i + \dot{\tilde{s}}_i\ddot{\tilde{s}}_i + k_2\bar{s}_i\dot{\bar{s}}_i\Bigg].\end{aligned} \tag{5.38}$$

Let the control laws for v_i and ω_i be defined as

$$v_i = k_4 x_{ei} + v_{di}\cos\theta_{ei} \tag{5.39}$$

$$\omega_i = k_5(\theta_{ei} - \delta_i) + \omega_{di} - \dot{\delta}_i + k_1 y_{ei} v_{di}\left[\frac{\sin\theta_{ei} - \sin\delta_i}{\theta_{ei} - \delta_i}\right] \tag{5.40}$$

where k_4 and k_5 are positive gains, and $\omega_{di} = \kappa_i v_{di}$. Then

$$\begin{aligned}\dot{V} = \sum_{i=1}^{N} \Bigg[& -k_4 x_{ei}^2 - \frac{k_5}{k_1}(\theta_{ei} - \delta_i)^2 + x_{ei}\dot{\bar{s}}_i \cos\theta_{ei} + y_{ei}\dot{\bar{s}}_i \sin\theta_{ei} \\ & + \frac{1}{k_1}(\theta_{ei} - \delta_i)\kappa_i \dot{\bar{s}}_i + y_{ei} v_{di} \sin\delta_i + \dot{\bar{s}}_i \ddot{\bar{s}}_i + k_2 \bar{s}_i \dot{\bar{s}}_i \Bigg]. \end{aligned} \tag{5.41}$$

To make the derivative of the Lyapunov function V negative, we choose the following consensus controller with a reference velocity

$$\begin{aligned}\ddot{s}_i = & \dot{v}_{di} - k_6(\dot{s}_i - v_{di}) - x_{ei}\cos\theta_{ei} - y_{ei}\sin\theta_{ei} - \frac{1}{k_1}(\theta_{ei} - \delta_i)\kappa_i \\ & - 2k_2 \sum_{j \in N_i} (s_i - s_j - s_{dij}) - k_3 \sum_{j \in N_i} (\dot{s}_i - \dot{s}_j) \end{aligned} \tag{5.42}$$

where $k_3, k_6 > 0$. Then we can achieve

$$\dot{V} = \sum_{i=1}^{N} \left[-k_4 x_{ei}^2 - \frac{k_5}{k_1}(\theta_{ei} - \delta_i)^2 - k_6(\dot{s}_i - v_{di})^2 + y_{ei} v_{di} \sin\delta_i \right] - k_3 \dot{s}^T L \dot{s} \leq 0 \tag{5.43}$$

where $\dot{s} \in \mathbb{R}^N$ is the stack vector of the robots' path derivative. We now state the main result of the coordinated path-following control for the mobile robots.

Theorem 8. *Assume that the undirected formation graph is connected. The control inputs v_i, ω_i, and $\ddot{s}_i$ given in (5.39), (5.40), and (5.42), respectively, for robot i solve the coordinated path-following objective.*

Proof. From (5.43), we have that $\dot{V} \leq 0$, which means that

$$V(t) \leq V(t_0), \quad \forall t \geq t_0. \tag{5.44}$$

From the definition of V, the right hand side of (5.44) is bounded by a positive constant depending on the initial conditions. Since the left hand side of (5.44) is bounded, it implies the boundedness of x_{ei}, y_{ei}, $(\theta_{ei} - \delta_i)$, $\dot{\bar{s}}_i$ and $\bar{s}_i$ for all $t \geq t_0 \geq 0$. We also assume boundedness of $\dot{s}_i$ and v_{di}, implying the boundedness of the overall closed-loop coordination system on the maximal interval of definition $[0, T)$. This rules out finite escape time so that $T = +\infty$.

From the above argument on the boundedness of x_{ei}, y_{ei}, $(\theta_{ei} - \delta_i)$, $\dot{\bar{s}}_i$ and $\bar{s}_i$, applying Barbalat's lemma [123] to (5.43) results in

$$\lim_{t \to \infty} (x_{ei}, \theta_{ei} - \delta_i, \dot{\bar{s}}_i, \dot{\bar{s}}_i) = 0. \tag{5.45}$$

To satisfy path-following tasks, we have to show that y_{ei} converges to zero as $t \to \infty$. In the closed loop of the θ_{ei} dynamics

$$\dot{\theta}_{ei} = \kappa_i \dot{\bar{s}}_i - k_5(\theta_{ei} - \delta_i) + \dot{\delta}_i - k_1 y_{ei} v_{di} \left[\frac{\sin\theta_{ei} - \sin\delta_i}{\theta_{ei} - \delta_i} \right]$$

we can conclude that $\lim_{t\to\infty}(y_{ei}) = 0$ since $\lim_{t\to\infty}(\theta_{ei} - \delta_i, \dot{\hat{s}}_i) = 0$ and v_{di} does not converge to zero.

Since the Laplacian matrix L is positive semidefinite, it follows that $L\dot{s} = 0$. L has a single zero eigenvalue with corresponding eigenvector $\overrightarrow{\mathbf{1}}$. It follows that $\dot{s}$ belongs to span$\{\overrightarrow{\mathbf{1}}\}$. Hence $\dot{s}_i = \dot{s}_j, \forall i, j \in N$, and $\dot{s}_i$ converges to v_{di}, which in turn, implies that $\ddot{s}_i = \dot{v}_{di}$. From this fact, we can get

$$\lim_{t\to\infty}(s_i - s_j - s_{dij}) = 0. \tag{5.46}$$

Define $s_{dij} = s_{di} - s_{dj}$, where s_{di} and s_{dj} are the path's desired parameters of robot i and robot j, respectively. We then have $s_i - s_j - s_{dij} = s_i - s_j - (s_{di} - s_{dj}) = (s_i - s_{di}) - (s_j - s_{dj}) = \hat{s}_i - \hat{s}_j$. Then we obtain $Ls + s_d = 0 \Rightarrow L\hat{s} = 0$. Thus, all $\hat{s}_i$ are equal to a common value, i.e., $s_i - s_j = s_{dij}, j \in N_i, \forall i, j$. We conclude that the robots converge to the desired configuration. ■

5.5.3 Simulation Results

Before testing on real robots, some simulations were carried out to evaluate the proposed controller as established in the previous subsection. Simulation allows to observe the controller behavior while varying some parameters, although it does not consider the team dynamics. Regarding more realistic situations in robot motions, we took into account the maximum velocities: $|v_i| \leq 0.5$ m/s, $|\omega_i| \leq 1.0$ rad/s. We performed a velocity scaling given in [181] so as to preserve the curvature radius corresponding to the nominal velocities. The control gains were set to $k_1 = 5.0$, $k_2 = 0.5$, $k_3 = 0.2$, $k_4 = 0.25$, $k_5 = 1.0$, $k_6 = 0.2$, and the desired speed for the whole group of robots was $v_0 = 0.2$ m/s.

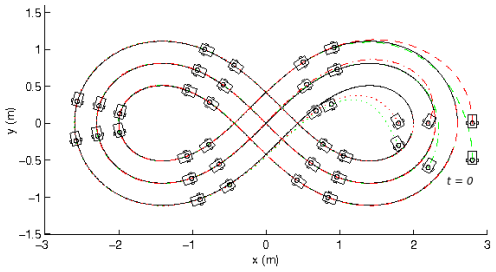


Figure 5.21: Simulation: the superimposed snapshots.

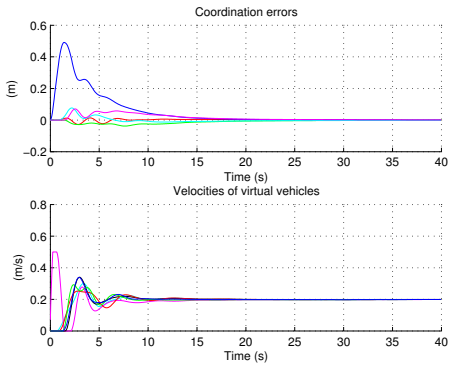


Figure 5.22: Simulation: the coordination errors and the velocities of virtual vehicles.

In this simulation, six mobile robots were required to follow a lemniscate curve given by

$$x_d(t) = \frac{2.3\cos\theta_d(t)}{1+\sin^2\theta_d(t)} \quad y_d(t) = \frac{2.3\sin\theta_d(t)\cos\theta_d(t)}{1+\sin^2\theta_d(t)}$$

and to maintain a desired formation described by the following elements of the adjacency matrix: $a_{14} = a_{13} = a_{36} = a_{56} = a_{25} = 1$. The superimposed snapshots are shown in Figure 5.21. The coordination errors converging to zero can be seen in Figure 5.22. The velocity tracking errors and the path errors of each robot also converge to zero, satisfying the path-following objective.

5.5.4 Experimental Results

In real-world experiments, the mobile robots shown in Figure 5.23 were used in this section. The lemniscate curve similar to the path in the simulation was employed in the first experiment. Each robot was required to maintain a column formation described by $s_{d12} = s_{d23} = 75$ cm. The elements $a_{12} = a_{21} = 1$, $a_{23} = a_{32} = 1$ in the adjacency matrix

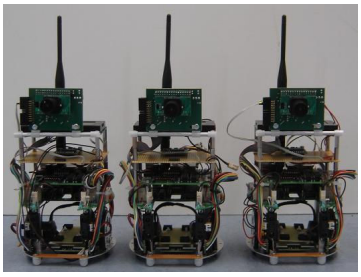


Figure 5.23: Three unicycle mobile robots used in experiments.

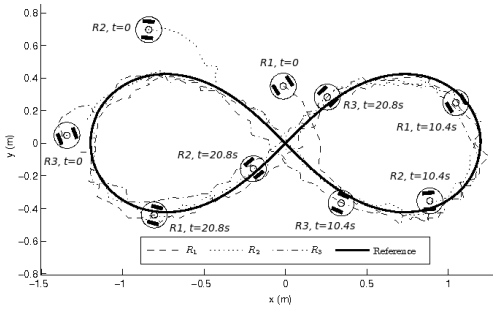


Figure 5.24: Exp. 1: the superimposed snapshots at t = 0 s, t = 10.4 s, and t = 20.8 s.

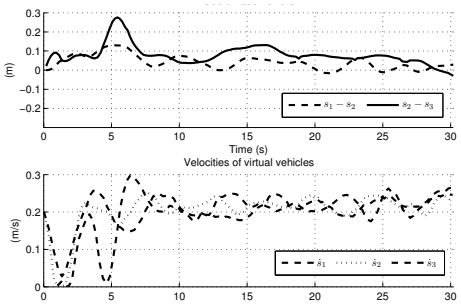


Figure 5.25: Exp. 1: the coordination errors and the velocities of virtual vehicles.

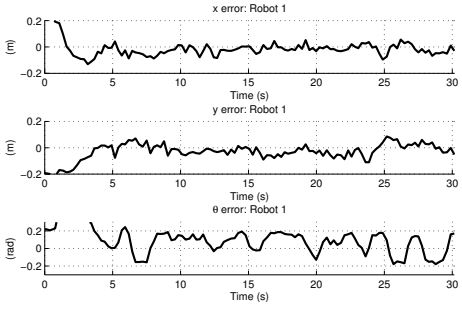


Figure 5.26: Exp. 1: the position errors of robot 1.

represented the information exchange in the formation graph. The experimental results are plotted in Figure 5.24. As seen in Figure 5.25, the coordination tasks are achieved. The coordination errors are less than 10 cm and the virtual vehicle of each robot can travel at the desired speed $v_0 = 0.2$ m/s. Likewise, the path-following tasks are attained as seen in Figure 5.26.

In the second experiment, each robot followed its own path, i.e., a sinusoidal curve for robot 1 and robot 3, and a straight line for robot 2. s_{dij} was set to 0 and the elements of the adjacency matrix were set to $a_{12} = a_{21} = 1$, $a_{23} = a_{32} = 1$. The results are depicted in Figure 5.27. The coordination errors and the velocity of each virtual vehicle can be seen in Figure 5.28. The experimental results show the effectiveness of our proposed control law: the group of robots can travel at the desired speed v_0 while keeping a desired formation. The main sources of disturbances during experiments include sensor distortion, vision-system delays, and communication delays.

5.5.5 Cooperation in Heterogeneous Robot Teams

This subsection presents a cooperative strategy for mobile robots based on nonlinear control techniques and omnidirectional vision. Such a strategy will be applied to a mobile robot team formed by small robots with a simple microcontroller and modest sensors such as wheel encodes, and a bigger leader robot with more computational power. The leader is responsible for group navigation and controls team formation. It has an omnidirectional camera and sees the other robots. Color segmentation and a Kalman filter is used to obtain the positions of the followers, related to the leader, while the orientation measured by a compass sensor is sent to the leader robot. The omnidirectional visual feedback has the advantage of allowing the leader to localize all the followers around itself by taking just one image. Those poses are then used for controlling team formation

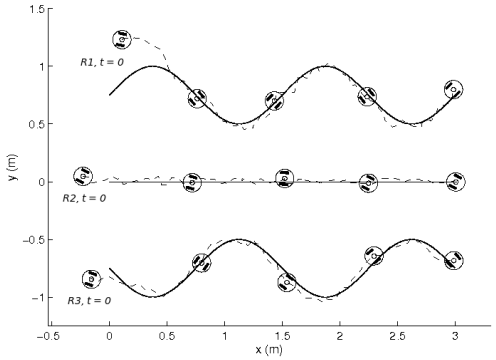


Figure 5.27: Exp. 2: the superimposed snapshots at t = 0 s, t = 5.2 s, t = 10.6 s, t = 16.2 s, and t = 21.6 s.

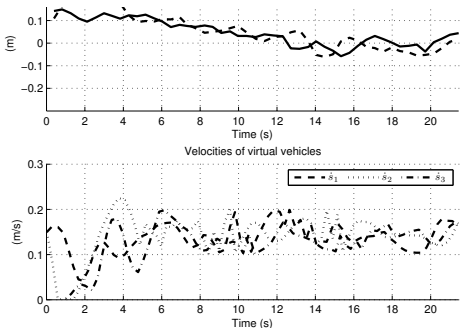


Figure 5.28: Exp. 2: the coordination errors and the velocities of virtual vehicles ($v_0 =$ 0.15 m/s).

through a nonlinear controller that defines the followers' linear and angular velocities to keep the desired formation.

In our implementation, each robot executes a formation control law in its own thread on the leader robot, which has greater computational power and owns the omnidirectional system. Although the formation control is centralized on the leader, the followers' velocity control is done in a decentralized manner. Once the followers receive their motor commands sent by the leader, they have their own controller for performing the linear and angular velocities. To make a mobile robot team (formed by one leader and n followers) navigate in an environment keeping a specific formation, we employ the navigator module from the CARMEN framework [167, 168] to generate collision-free way-points. The experiments were carried out with one omnidirectional mobile robot as the leader, and two unicycle mobile robots as the followers. The solid line in the middle in Figure 5.29 indicates the path of the leader. The distance s_d between the leader $R1$ to the first follower $R2$ was equal to 0.8 m, while the distance s_d between two followers, $R2$ and $R3$, was set to 0.5 m. The offset distances of $R2$ and $R3$ were 0.2 m and -0.2 m, respectively. The leader moved with linear velocity of 0.2 m/s. The dotted lines represent the actual trajectories for each robot. The filled circle and rectangle indicate the starting point and the end point, respectively. As seen in the results, the followers achieve their desired positions and successfully keep the formation. However, it is worth noting that collision avoidance is not included in this experiment and the restriction of this cooperative strategy is to maintain line-of-sight constraints.

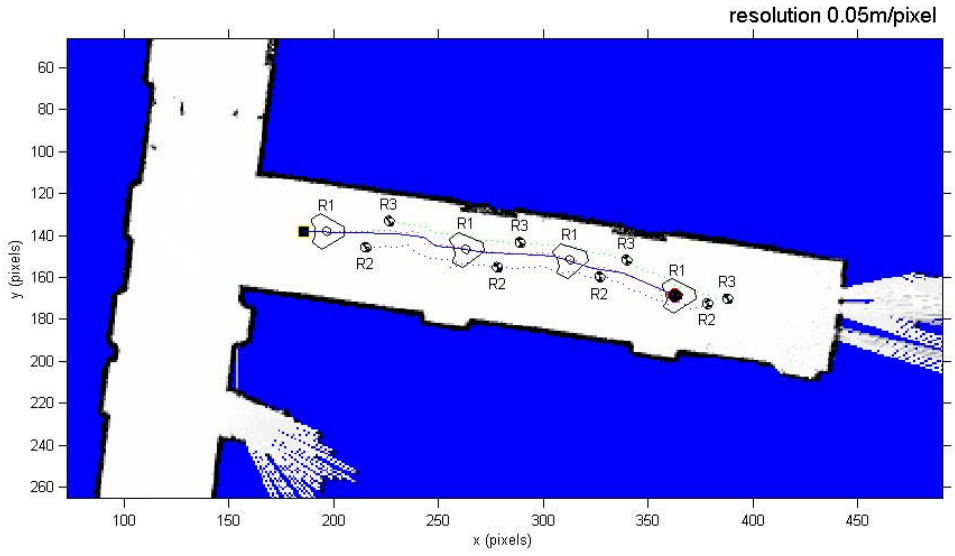


Figure 5.29: The superimposed snapshots of cooperative experiments. $R1$ denotes a big omnidirectional mobile robot, while $R2$ and $R3$ represent small unicycle mobile robots.

5.6 Discussions and Summary

In this study, controllers were implemented in a distributed fashion within network environments. These robots can cooperate and communicate with each other to achieve the objective of coordinated path following control. Two kinds of control laws have been proposed, i.e., one realized the attractive feature of NMPC based on two strategies: A leader-following strategy in Section 5.3 and a distributed version in Section 5.4. The other solution was to use a Lyapunov function and consensus protocols in Section 5.5.

The NMPC controller is an interesting control method as not only it can handle the state and input constraints but also utilize future information to generate a trajectory of optimal control input at each time step. However, the major disadvantages of NMPC are a control stability problem and high computational load. By our implementation, we have shown that NMPC is a promising control approach, applied to real-time applications of mobile robots as seen in our experimental results.

In Section 5.3, the two key points, which are employed to solve the path following problem and formation keeping problem, are that (i) the velocity of a virtual vehicle is integrated into the local cost function of the leader robot, and (ii) each follower robot computes its own reference trajectory with velocity profiles, estimated by using the leader's information, a given path, and a desired formation pattern. Velocity profiles for each follower robot can be computed by using (5.4) and (5.5) when $(p_i(s), q_i(s))$ coordinates have been determined. To guarantee stability of a local NMPC law, we employ a Lyapunov function for the terminal-state penalty and terminal constraints.

In Section 5.4, we presented the solution for the formation control and coordinated path following control using the distributed MPC, whose cost function is coupled with neighbors. Two experiments have been conducted. In the first experiment, a group of omnidirectional mobile robots was required to follow one reference path while keeping a desired formation in time. In the second one, each robot was required to follow its own predetermined reference path while maintaining a desired coordination with its neighbors in time. As stated in [65], the key requirements for stability are that each subsystem does not deviate too far from the previous open-loop state trajectory, and that the prediction horizon updates happen sufficiently fast. Even though very fast updates cannot be achieved, experimental results show the efficiency of our control algorithm. The less conservative method for stability proof should be further investigated.

In Section 5.5, we developed a new control law for coordinated path following of unicycle mobile robots. Each mobile robot was steered along a set of given spatial paths, while keeping a desired inter-vehicle coordination pattern. The solution adopted for coordinated path following built on Lyapunov function techniques and consensus algorithms. The desired formation pattern was achieved by controlling the path derivative such that the coordination error converges to zero.

Chapter 6

Role Assignment and Formation Switching

In this chapter, we study the problem of distributed role assignment and formation switching. This problem arises when a mobile robot in the team must decide what role to take on in a desired formation configuration. In some applications, in which the center and orientation of a desired formation are not predetermined, the rotation and translation of the formation can be computed by using average consensus protocols. However, a conflict arises when the same role is assigned to more than one robot. This problem is resolved by using a negotiation strategy, while each assigned robot is traveling to the target position. Furthermore, dynamic consensus is used to measure the degree of task completion in this work. Once the consensus reaches a preset threshold, this implies that robots should abandon the current task and start the next one.

First, we review some related work on these topics. Then our proposed algorithms are presented. We verify our proposed framework through real-world experiments on a team of nonholonomic mobile robots performing distributed formation switching.

6.1 Related Work

In this section, we review some methods concerning formation selection, role assignment, and formation switching. A good example on these topics is the RoboCup soccer domain [37, 216], in which a coordination protocol can be used by robots in order to select an appropriate formation according to the environment conditions and to make a decision on the roles assumed by the robots in the formation based on the concept of utility functions or rules. Since each robot's status does not necessarily coincide with those of the others, the robots may choose different formations. Therefore the formation selection algorithm is based on a voting scheme that allows for changing the formation only in presence of the absolute majority of votes.

6.1.1 Formation Selection and Formation Switching

In many applications, it is sometimes necessary to change the formation due to either a change in coordinated task specifications or a change in environmental conditions, e.g., the presence of uncertainty, adversarial vehicles, and narrow corridors. The questions on these research areas are: "Which formation should robots choose if they have a set of formations?", "Who or what mechanism decides the formation shape to be formed?", "Is there an optimal way for performing formation changes?", and "Under what conditions, switching between formation should occur?"

Switching between different rigid formations was studied by Das et al. [51], where the switching between simple decentralized controllers on follower robots allows formation switching, while follower robots are following a leader or performing a specific task. Desai [55] proposed a graph-theoretic approach for coordinating transitions between two formations. He also presented examples to demonstrate the general approach and showed how a team of robots can automatically change the shape of the formation and/or its control graph in order to circumvent obstacles. Fierro et al. [74] split the problem using a hybrid approach. They designed the continuous-state control algorithms based on input-output feedback linearization. Each robot can maintain a prescribed separation and bearing from its adjacent neighbors, while discrete-state formation control is used to achieve a desired formation by sequential composition of basic maneuvers. They used the sensor constraints and the presence of obstacles to determine the switching sequence in a finite state machine. However, the problem with the control strategies in [51, 55, 74] is that the control problems get more complicated as the number of robots in the formation increases.

The behavior-based formation switch proposed by Michaud et al. [164] involves the assignation of a new leader, which is influenced by the situation experienced by the robot, while in a method proposed by Fredslund and Matarić [77], special cases must be programmed to keep robots organized according to their ID numbers and switching must preserve the ordering of the robots based on their ID.

McClintock and Fierro [158] investigated when formation changes should occur. The robots change between formations in a given set by choosing the one which can operate in the current environment with minimum formation error. The path planner searches for an optimal path based on trade-offs between the total distance traveled and the distance traveled in less desirable formations. Zelinski et al. [241] implemented an optimized hybrid system approach on an aerial robot platform. They focused on determining a nominal state and input trajectory for each vehicle such that the group can start from a given initial configuration and reach its given final configuration at a specified time while satisfying a set of given inter- and intra-vehicle constraints.

The problem of selecting a suitable formation shape possibly depending on a dynamical context (e.g., environment modifications and dynamical tasks) has not been deeply investigated in the literature [91]. There is still no underlying theory that governs this problem. Haque and Egerstedt [91] modeled the bottlenose dolphins behavior: Agents

in hunting phase have to choose their layout between small or large circles using a hybrid control strategy and decentralized networked control in order to capture a prey. Recently, Di Rocco et al. [57] proposed an approach to select an optimal formation shape using the online selection of the optimal shape of the formation that maximizes some performance indices related to the task and to the environment.

6.1.2 Role Assignment in Formation

In general, an explicit assignment of the robots in a formation might be desired, however, for indistinguishable robots, this is not always easy. This problem can be seen as the combinatorial optimization, where n persons are optimally divided among n objects. It is also referred to as the *assignment problem*, or the minimum weight perfect matching problem in bipartite graphs. In 1955, Kuhn [129] developed the Hungarian method – the first polynomial solution for the assignment problem. Another approach is the *auction algorithm* [24, 25]. This problem might also be related to the multi-robot task allocation (MRTA) architecture [80], in which the question is encountered: "Which robot should execute which task?"

In a centralized fashion, a reasonable strategy would be to minimize the sum of the distances traveled by each robot to arrive at its target. However, with distributed decision-making and limited communication, the problem of deploying robots to form arbitrary target configurations is still open.

In [140], each robot builds a visibility table of all the robots within its sensing range and these tables are then cross-populated and shared among the group members. After initializing the table, each robot assigns itself as being the conductor of the desired formation and then searches the tree using a bounded depth-first search with pruning algorithm to find the best assignment of positions for other robots in the group. The best result obtained by each robot is broadcasted to the others, and the one with the minimum cost is selected as the conductor of the formation, with positions assigned accordingly to the other robots. Smith and Bullo [210] allowed the group of agents to divide the targets among themselves under limited communication. Their assignment-based algorithm has the following features: Initial assignments and robot motions follow greedy rules, and distributed refinement of the assignment exploits an implicit circular ordering of the targets. Lee et al. [139] considered a spacecraft formation configuration problem. They used a coupled combinatorial and continuous optimization framework, in which the inner loop consists of computing the costs associated with a particular assignment by using a discrete optimal control method. In the outer optimization loop, combinatorial techniques are used to determine the optimal assignments based on the costs computed in the inner loop.

Ji et al. [105] were interested in determining the rotation and translation of the target formation and in finding an appropriate permutation that assigns target positions to agents. They showed how the simultaneous rotation, translation, and assignment optimization problem can be casted as a parameterized assignment problem. However, the

solution results in a centralized off-line algorithm in the sense that the computation of the solutions requires complete information about all agents in the team. This algorithm has to be done before the team is deployed. Derenick and Spletzer [54] employed a formal definition from shape analysis for formation representation and used second-order cone programming techniques to find optimal solutions by minimizing either the total distance or the minimax distance the robots must travel. Their formation shape is invariant under the Euclidean similarity transformations of translation, rotation, and scaling. A distributed auction-based approach for a rotational and translational invariant configuration was proposed by Zavlanos and Pappas [240]. Distributed consensus algorithms are employed to guarantee that all agents agree on the rotation and translation of the final configuration, and then local market-based coordination protocols dynamically determine a permutation of the agent in the formation, while artificial potential fields are used to drive the group of agents to the desired formation. Michael et al. [162] extended this solution to the dynamic task allocation problem where the assignment of robots to tasks may need to be continuously adjusted depending on changes in the task environment or group performance.

6.2 Proposed Algorithms

In this chapter, we propose a novel distributed and online solution to the role assignment problem by using consensus protocols and negotiation algorithms. This problem becomes more challenging when the robots do not have complete information about other robots in the team and the number of roles may be more or less than the number of robots. This section is based on own work published in [118].

Given a team of N mobile robots, whose positions are $\mathbf{x}_i \in \mathbb{R}^2,\ i = 1,...,N$, and the target formation represented by the target positions $\mathbf{x}_k^d \in \mathbb{R}^2,\ k = 1,...,M$, assuming that all robots have knowledge of the target formation, we need to find an appropriate mapping $p : \{1,...,N\} \to \{1,...,M\}$ that assigns robot i, located at $\mathbf{x}_i$ to target position $\mathbf{x}_k^d$. Furthermore, we are also interested in how to determine the rotation angle θ_f and translation $v_f \in \mathbb{R}^2$ of the target formation in order to minimize the displacement of the whole group from the initial configuration to the target configuration. Thus, our proposed algorithm can be divided into three parts, i.e., determining the translation and the rotation of the target formation, conflict resolution, and navigating control with a collision-avoidance capability. In the first part, distributed consensus algorithms are employed to guarantee that all robots agree on the rotation and translation of the target formation. Then, local coordination protocols dynamically determine the role in the target formation. After the initial role is chosen, the robot will be driven to the target position. In addition, all robots can reconfigure themselves from one formation to another using dynamic consensus to detect the completeness of the current task.

6.2.1 Problem Formulation

The solution of the problem mentioned above greatly depends on notions from graph theory. Given a system of robots, we can define a dynamic graph $\mathcal{G}(t)$ as follows [162]:

Definition 10. (Dynamic Graph): We call $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$ a dynamic graph consisting of a set of vertices $\mathcal{V} = \{1, ..., N\}$, indexed by the set of robots and a time varying set of links $\mathcal{E}(t) = \{(i, j) \in \mathcal{V} \times \mathcal{V} | \ \|\mathbf{x}_i - \mathbf{x}_j\|_2 < r\}$, meaning that edges are established between robots i and j, if and only if the robots are within distance r of each other.

Dynamic graphs $\mathcal{G}(t)$ such that $(i, j) \in \mathcal{E}(t)$ if and only if $(j, i) \in \mathcal{E}(t)$ are called undirected. Moreover, for any pair of vertices i and j such that $(i, j) \in \mathcal{E}(t)$ we say that vertices i and j are adjacent, or neighbors, at time t and the set of neighbors of robot i is denoted by $N_i \subseteq \mathcal{V}$. A path between vertices (i, j) is a sequence of distinct vertices such that consecutive vertices are adjacent. A topological invariant of graphs that is of particular interest for the purposes of this work is graph connectivity.

Definition 11. (Graph Connectivity): We say that a dynamic graph $\mathcal{G}(t)$ is connected at time t if there exists a path between any two vertices in $\mathcal{G}(t)$.

Given any collection of m distinct instances of $\mathcal{G}(t)$, i.e., $\{\mathcal{G}(t_1), ..., \mathcal{G}(t_m)\}$, we say that the collection $\{\mathcal{G}(t_1), ..., \mathcal{G}(t_m)\}$ is *jointly connected* if the union of its members is a connected graph [103]. All graphs considered in this section are undirected and a connectivity on the underlying communication network is assumed.

Let $N \in \mathbb{N}$ be the number of the robots, which is unknown to any robot in the team, and $M \in \mathbb{N}$ be the number of the tasks (or the positions) in the formation. The assignment of the robots to the targets is described by $p \in P_N$, where P_N is the set of all possible permutations over N elements. As defined in [105], the centralized role assignment is given as follows:

$$\sum_c(\mathbf{x}, \mathbf{x}^d) : \min_{(v_f, \theta_f, p) \in \mathbb{R}^2 \times [0, 2\pi) \times P_N} J_c(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p) \tag{6.1}$$

where $J_c(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p)$ is the cost

$$J_c(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p) = \sum_{i=1}^{N} c(\mathbf{x}_i, R(\theta_f)(\mathbf{x}^d_{p(i)} + v_f)) \tag{6.2}$$

and $R(\theta_f)$ in (6.2) is the rotation matrix, i.e.,

$$R(\theta_f) = \begin{bmatrix} \cos\theta_f & \sin\theta_f \\ -\sin\theta_f & \cos\theta_f \end{bmatrix} \tag{6.3}$$

and c is a performance measure. The interpretation is that $c : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ gives the cost of assigning the robot i at $\mathbf{x}_i$ to the target located at $R(\theta_f)(\mathbf{x}^d_{p(i)} + v_f)$.

If c is the square of the l_2-norm (Euclidean norm) of the difference between $\mathbf{x}_i$ and $R(\theta_f)(\mathbf{x}^d_{p(i)} + v_f)$, we get

$$\sum_{l_2^2}(\mathbf{x}, \mathbf{x}^d) : \min_{(v_f, \theta_f, p) \in \mathbb{R}^2 \times [0, 2\pi) \times P_N} J_{l_2^2}(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p) \tag{6.4}$$

where

$$J_{l_2^2}(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p) = \sum_{i=1}^{N} \|\mathbf{x}_i - R(\theta_f)(\mathbf{x}^d_{p(i)} + v_f)\|_2^2. \tag{6.5}$$

When θ_f and v_f are provided, the optimal assignment satisfies

$$p^* = \operatorname*{argmin}_{p \in P_N} J_{l_2^2}(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p). \tag{6.6}$$

Moreover, the problem

$$\min_{p \in P_N} J_{l_2^2}(\mathbf{x}, \mathbf{x}^d, v_f, \theta_f, p) \tag{6.7}$$

corresponds to the well-known linear assignment problem. p^* is easily computed by using the Hungarian method, which is a polynomial time algorithm whose computational complexity is $O(N^3)$ [129]. In [54, 105], the authors are interested in determining the rotation and translation of the target formation and finding an appropriate permutation that assigns target positions to robots. However, the solution results in a centralized off-line algorithm in the sense that the computation of the solutions will require complete information about all robots in the team. This algorithm has to be done before the team is deployed. When computation and information are distributed among multiple robots and the number of robots in the team is unknown, the need for a fully distributed control framework to solve the role assignment problem becomes vital. We propose a distributed solution to this problem using consensus protocols and negotiation algorithms. It has to be noted that although no global cost is optimized in this work, robots can always achieve the role assignment task, as shown in our experimental results.

6.2.2 Distributed Role Assignment

The goal of this section is to develop a distributed control framework that is able to drive the robots to the target formation which is transformed through the common translation vector v_f and rotation matrix $R(\theta_f)$. The permutation P is determined dynamically by means of distributed negotiation strategies, using only local information. In some applications, in which the center and the orientation are not given, we employ consensus protocols to obtain v_f and θ_f that are agreed by all robots in the team. To drive the robots to the target positions safely, we integrate the navigating control law with the artificial potential field.

Consensus on the Rotation and Translation

To minimize the displacement of the whole group from the initial positions to the target positions, we extract the first principal components analysis (PCA) axis from the robot locations and use it as a rotation variable. This line goes through the centroid and also minimizes the square of the distance of each point to that line.

Inspired by [202], we estimate the eigenvectors of the covariance matrix of distributed positions in a decentralized fashion. The algorithm proposed in [202] is based on a combination of the so-called power method, that is used to compute the eigenvectors, and the average consensus protocol, that is utilized to structure the information exchange into an agreement protocol. The data covariance matrix encloses a linear representation of the data involved in its so-called principal components, that are the largest eigenvectors of the data covariance. The problem we are interested in is to extract the largest eigenvector in a decentralized fashion, assuming that the positions are measured by the localization module and the communication links are locally connected. The idea is based on the decomposition of the power method into a decentralized iterative protocol. The eigenvectors of the covariance matrix $\hat{C}$ can be derived by using a power method as follows

$$\mathbf{q}(n+1) = \frac{\hat{C}\mathbf{q}(n)}{\|\hat{C}\mathbf{q}(n)\|} \tag{6.8}$$

where $\hat{C} = (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, i = 1,...,N$ and $\mathbf{q}(0)$ is an initial random vector. $\bar{\mathbf{x}}$, the centroid, can also be computed by using the average consensus protocol. This method converges to the maximum eigenvector of $\hat{C}$ as long as the maximum eigenvalue of $\hat{C}$ is strictly greater than the other eigenvalues and the vector $\mathbf{q}(0)$ has a non-zero component in the direction of the eigenvector associated to the largest eigenvalue [202].

For simplicity, let us assume that $\bar{\mathbf{x}} = \mathbf{0}$. The recursive equation (6.8) can be expanded as follows:

$$\begin{aligned} \mathbf{q}_1(n+1) &= \frac{\hat{C}\mathbf{q}_1(n)}{\|\hat{C}\mathbf{q}_1(n)\|} \\ &= \frac{\sum_{i=1}^{N} \mathbf{x}_i(\mathbf{x}_i \cdot \mathbf{q}_1(n))}{\|\sum_{i=1}^{N} \mathbf{x}_i(\mathbf{x}_i \cdot \mathbf{q}_1(n))\|} \\ &= \frac{\overline{\mathbf{x}(\mathbf{x} \cdot \mathbf{q}_1(n))}}{\|\overline{\mathbf{x}(\mathbf{x} \cdot \mathbf{q}_1(n))}\|} \end{aligned} \tag{6.9}$$

Subscript 1 of $\mathbf{q}_1(n)$ denotes the first eigenvector. Unlike [202], we perform the average consensus protocols for $\mathbf{x}_i(\mathbf{x}_i \cdot \mathbf{q}_1(n)),\ i = 1,...,N$, instead of only the inner products and, in our algorithm, each robot does not need to know the number of robots in the team. Finally, we can obtain the estimate of $\mathbf{q}_1$. Algorithm 6.1 shows the derivation of the first eigenvector. The function *AC* implements the average consensus algorithm and ε is a small constant.

Algorithm 6.1 First eigenvector estimation (algorithm performed at robot i)

$\bar{\mathbf{x}} \leftarrow AC(\mathbf{x}_j),\ j = 1,...,N$
$\mathbf{x}_j \leftarrow \mathbf{x}_j - \bar{\mathbf{x}},\ j = 1,...,N$
Initial random vector: $\mathbf{q}_{1,i}(0)$
$n \leftarrow 0$
repeat
    $n \leftarrow n+1$
    $\mathbf{q}_{1,i}(n) \leftarrow AC(\mathbf{x}_j(\mathbf{x}_j \cdot \mathbf{q}_{1,j}(n-1))),\ j = 1,...,N$
    $\mathbf{q}_{1,i} \leftarrow \mathbf{q}_{1,i}/\|\mathbf{q}_{1,i}\|$ {normalization}
until $\|\mathbf{q}_{1,i}(n) - \mathbf{q}_{1,i}(n-1)\| \leq \varepsilon$

Negotiation Strategies

The notations borrowed from [162] are as follows. Let $\mathcal{I}_0 = 1,...,M$ denote the index set of all available positions in a formation. We say that a position $k \in \mathcal{I}_0$ is being *taken* if there is at least one robot i such that $p(i) = k$. Let $\mathcal{I}^a(t)$ denote the index set of all positions that are *available*, i.e., not *taken* at time t and $\mathcal{I}^t(t) = \mathcal{I}_0 \setminus \mathcal{I}^a(t)$ denote the index set of all *taken* positions. Similarly, let $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$ denote the index sets of available and taken positions from the perspective of robot i, respectively.

At first, robot i exchanges the position information with its neighbor j, where $(i,j) \in \mathcal{E}(t)$ and then each robot starts locally the Hungarian algorithm to obtain its favorite position. However, in case that $N > M$, the robot may not get any favorite position from the Hungarian algorithm. In this situation, we assign the nearest position to this robot in order to ensure that every robot receives one assigned position. Each robot then updates $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$. All assigned robots start moving to the target position using the navigating controller that is described in the next subsection. At each time step, every robot exchanges the information with its neighbors in order to update $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$. If two robots realize that they are assigned to the same position, they stop and each robot will add the index of the other into its queue. Then, the robot with the higher index starts the pairwise negotiation strategy, given in Algorithm 6.2, to resolve the conflict. Note that d_i of robot i is the Euclidean norm of the difference between $\mathbf{x}_i$ and its target position.

Correctness of the proposed algorithm depends on the assumption that every robot requesting to be assigned to a formation role will eventually be able to communicate and negotiate with all other robots requesting to be assigned to the same role. Only one robot will take the role when that conflict is resolved. The robots that do not get that position will choose the new position from $\mathcal{I}^a(t)$ in such a way that it is the nearest position and connectivity is still maintained.

In case that the number of robots is more than the number of roles, i.e., $N > M$, the unassigned robots have to perform the task of maintaining connectivity, e.g., [239] in order to keep the connectivity assumption satisfied, and they also have to move away from the *taken* positions. In this work, we simply apply artificial attractive and repulsive

Algorithm 6.2 Negotiation strategies for robot i

Require: $\mathcal{I}_i^a$, $\mathcal{I}_i^t$, $p(i)$, the descending index set of robots assigned to the same position, i.e., $\mathcal{A}_i = \{j \in N_i \mid p(j) = p(i)\}$.

$\mathcal{I}_i^a \leftarrow \mathcal{I}_i^a \setminus (\cup_{j\in N_i}\mathcal{I}_j^t)$ {exchange lists with neighbors}
$\mathcal{I}_i^t \leftarrow \mathcal{I}_i^t \cup (\cup_{j\in N_i}\mathcal{I}_j^t)$ {exchange lists with neighbors}
while $\mathcal{A}_i \neq$ **NULL do**
    $k \leftarrow \mathcal{A}_i$.dequeue
    if $d_i \leq d_k$ **then**
        robot i wins and robot k chooses a new position from $\mathcal{I}_k^a$
        update $\mathcal{I}_i^a$, $\mathcal{I}_i^t$, $\mathcal{I}_k^a$, $\mathcal{I}_k^t$
        dequeue robot k from $\mathcal{A}_i$ and empty $\mathcal{A}_k$
        robot k starts moving to the new assigned position
    else
        robot k wins and robot i chooses a new position from $\mathcal{I}_i^a$
        update $\mathcal{I}_i^a$, $\mathcal{I}_i^t$, $\mathcal{I}_k^a$, $\mathcal{I}_k^t$
        dequeue robot i from $\mathcal{A}_k$ and empty $\mathcal{A}_i$
        robot i starts moving to the new assigned position
    end if
end while

forces to meet these requirements. However, connectivity may be lost in case that $N < M$. This issue, solved by simultaneously maintaining connectivity and performing role assignment, is left to the future.

Navigating Control with Collision Avoidance

In fact, any controller with the capability to avoid inter-robot collisions and to navigate through waypoints can be used. In this work, we employed the controller from [35]. The control architecture combines two feedback loops: a motion control loop and a new-target control loop. The latter loop provides a modification of the target position when an obstacle appears on the path of the mobile robot. By representing the robot position in polar coordinates, and considering the error vector e, as well as by letting $\alpha = \phi - \theta$ be the angle measured between the distance vector e and the main robot axis, the kinematic equations (3.3) can be rewritten as

$$\dot{e} = -v\cos\alpha\ , \qquad \dot{\alpha} = -\omega + v\frac{\sin\alpha}{e}. \tag{6.10}$$

Then, the control laws for v and ω are given by

$$\begin{aligned} v &= \gamma \tanh e \cos\alpha \\ \omega &= k_c\alpha + \gamma\frac{\tanh e}{e}\sin\alpha\ \cos\alpha \quad \text{with } k_c > 0 \end{aligned} \tag{6.11}$$

where $\gamma = |v_{\max}|$ and $|\omega_{\max}| = k_c\pi + \gamma/2$. Since we do not have the prescribing orientation of the target position, these control laws can drive the distance vector e and α to zero asymptotically and θ becomes constant when $t \to \infty$.

To avoid obstacles, the concept of attractive and repulsive forces is applied to modify the target position when an obstacle suddenly appears on the path during navigating towards a target position. Similar to the method proposed by [224], we define the amplitude of the repulsive force as follows

$$F = k_r\left(\frac{1}{R_r} - \frac{1}{R_{\max}}\right)\frac{1}{R_r^2} \qquad \text{with } k_r > 0 \tag{6.12}$$

where R_r is the distance between the current robot position and the obstacle position and $R_{\max}$ represents a potential field's distance limit of influence. The projections of force F on the x and y axes are

$$F_x = |F\cos\beta| \qquad F_y = |F\sin\beta| \tag{6.13}$$

where β denotes the angle of the repulsive force direction. The coordinates of the vector of the repulsive force are

$$\begin{aligned} x_f &= x + \text{sign}(x - x_o)F_x \\ y_f &= y + \text{sign}(y - y_o)F_y \end{aligned} \tag{6.14}$$

where $[x_o, y_o]^T$ is the location of the obstacle. The resulting vector, which determines the new target position of the robot, is obtained as a sum of the vectors of repulsive and attractive forces. The new target coordinates of the robot are

$$\begin{aligned} x_n &= x_f + (x_d - x) \\ y_n &= y_f + (y_d - y) \end{aligned} \tag{6.15}$$

where $[x_d, y_d]^T$ is the target position. To avoid a possible standstill which may arise if the robot, the obstacle and the target are aligned, we simply change the target position in order to be able to drive the robot out of this situation.

Task Sequencing

The mission of this work is that a team of robots can reconfigure themselves from one formation to another. Global communication can simplify this problem because robots can communicate with all robots in the team. Once a robot detects the completion of the current task, it can broadcast this to the rest of the team in order to synchronize the whole system's transition to the next task. However, when global communication is not available or impossible, robots must measure the degree of consensus in their team. Once the consensus reaches a preset threshold, meaning that some proportion of the teammates agree that the current task is completed, they should induce their team to abandon the

current task and start the next one. This threshold highly depends on task requirements. If it is too high, the probability of a robot making an error and prematurely activating the switch to the next task may increase.

To attain this goal, we introduce a new state, $z_i(t)$ for robot i, to measure the degree of task completion of the current task. It becomes zero when all robots agree that the current task is completed. The following proportional-integral consensus estimator, based on [78], is used in this work:

$$\begin{aligned} \dot{z} &= -Lz + k_z(w - z) + L\eta \\ \dot{\eta} &= -Lz \end{aligned} \tag{6.16}$$

where L is the Laplacian matrix and η is an integrator variable. $w_i(t)$ is a dynamic input. It becomes 0, if robot i detects the completion of its current task, and 1, otherwise. k_z is the forgetting factor. Large values of k_z mean that we get rid of old information quickly. We initialize $z_i(0) = w_i(0)$ for each robot. Unlike static consensus, in which all robots must converge to the average of their initial states (i.e., $\frac{1}{N}\sum_i^N z_i$), $w_i(t)$ can be seen as a dynamic input in dynamic consensus. All robots must track the time-varying average of the w_i terms, i.e., they have to reach $\bar{w} = \frac{1}{N}\sum_i^N w_i$. Therefore, z_i can be considered to be a time-varying estimate of the instantaneous average value $\bar{w}$. We use z_i to measure the degree of task completion. When z_i reaches a preset threshold, this implies that robot i should abandon the current task and start the next one.

6.2.3 Simulation Results

Simulation results given in this subsection are based on own work in [114]. We considered a navigation task in $\mathbb{R}^2$, where $N = 16$ robots were required to reach the desired formation configurations. They started from randomly chosen initial configurations but they satisfied the connectivity assumption. We evaluated our algorithms with three examples, i.e., formation switching (reconfiguration in sequence, as shown in Figure 6.1), a circle formation, where $N > M$, and a lattice formation, where $N < M$. All parameters used in simulations were set as follows: $k_c = 1$, $\gamma = |u_{\max}| = 0.5$ m/s, $k_r = 0.2$, $R_{\max} = 0.4$ m, and the communication range r was 1.5 m. In consensus-based task sequencing, the threshold of the state z was 0.1 and the forgetting factor was 0.2.

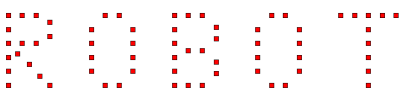


Figure 6.1: Five desired formation configurations: R O B O T. Squares represent the target positions in the formation. The number of positions M are equal to 16, 12, 16, 12, and 10, respectively.

As shown in Figure 6.2, all robots were able to successfully complete the formation switching task, in which the orientation of the target formation was predetermined. They first used the average consensus protocol to obtain an agreement on the translation and then they started negotiation to get the role, if a conflict arose. They were able to move to the target position without any collision. In case that $M < N$ (see Figure 6.2(c), Figure 6.2(e), and Figure 6.2(f)), some unassigned robots had to move around in order to maintain connectivity. Furthermore, robot 2 in Figure 6.2(c) and robot 7 in Figure 6.2(f) were uninformed that the position they were assigned to, had already been taken. Thus, they moved towards the target position and then they realized that those positions were no longer available. However, when the orientation of the target formation is not given, we can use the average consensus protocol to obtain an agreement on both translation and rotation. In this case, the translation and rotation may be changed from one formation to another, as seen in Figure 6.3, because there are some unassigned robots moving around. Moreover, our proposed framework can achieve the distributed role assignment in case that $N > M$ and $N < M$, as illustrated in Figure 6.4(a) and Figure 6.4(b), respectively.

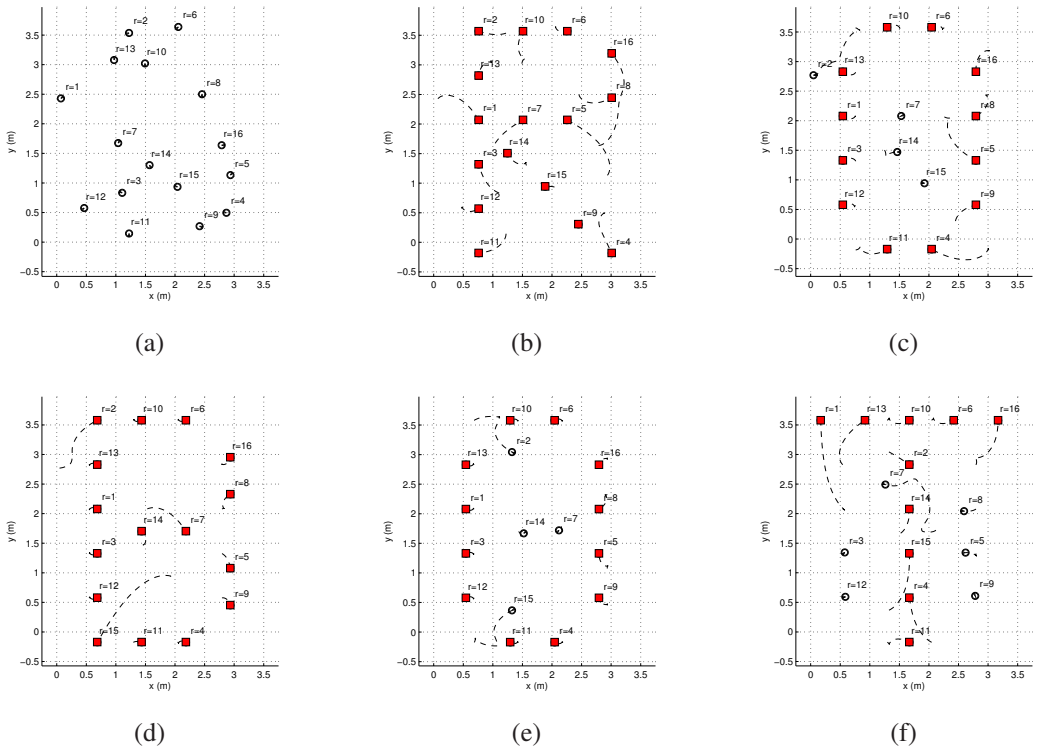
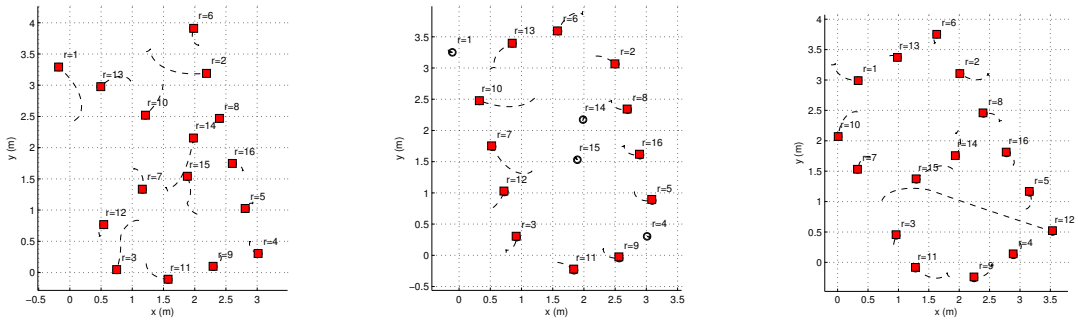
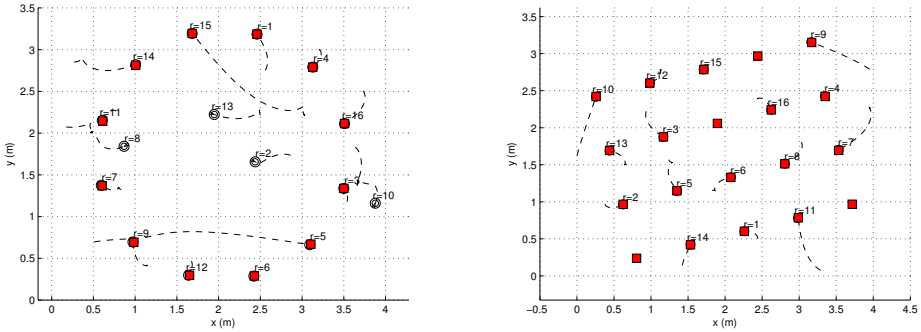


Figure 6.2: The simulation results on formation switching (a) each robot starts at a different random position, (b) each robot obtains an agreement on the translation and then moves to the target position - the letter *R*, (c) - (f) the robots can successfully reconfigure themselves without any collision.



(a) (b) (c)

Figure 6.3: The simulation results in case that consensus on rotation and translation is executed every time before role assignment is performed (a) - (c) the translation and the rotation are changed, compared to Figure 6.2(b), Figure 6.2(c), and Figure 6.2(d), respectively.



(a) (b)

Figure 6.4: The simulation results: (a) performing a circle formation where $N > M$ and (b) performing a lattice formation, where $N < M$.

6.2.4 Experimental Results

We evaluated our proposed framework through two experiments on a team of physical nonholonomic mobile robots shown in Figure 6.5: (i) the robots reconfigured themselves from one formation to another, and (ii) formation switching happened, while each robot was following a reference path. It is worth noting that selecting a particular formation shape, in general, relies on a dynamical context, e.g., environments and tasks. There is still no underlying theory that handles this problem. However, in this study, we assume to have such a mechanism that can choose a suitable formation (e.g., [57]).

The goal of the first experiment was to drive the robots to the target formation transformed through the common translation vector v_f and rotation matrix $R(\theta_f)$, while the

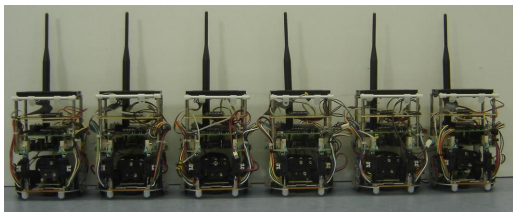


Figure 6.5: The unicycle mobile robots used in our experiments (see Chapter 3).

permutation P is determined dynamically by means of distributed negotiation strategies, using only local information. In some applications, in which the center and the orientation of the target formation are not given, we employ consensus protocols to obtain v_f and θ_f that are agreed by all robots in the team.

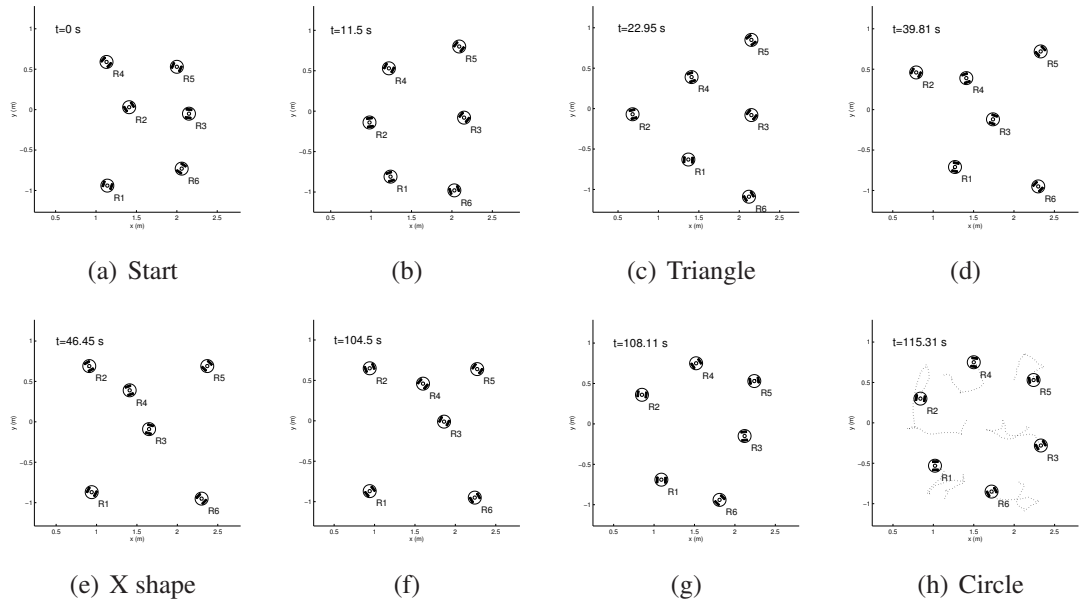
In the first experiment, we validated our algorithms with three different formations as shown in Figure 6.6. Six robots had to switch from one formation to another in sequence. The communication range r was set to 1.2 m. In task sequencing, the threshold of the state z was equal to 0.1 and the forgetting factor was 0.5.

Since the center and the orientation of the target formation were not predefined, all robots had to use the average consensus protocol to obtain an agreement on the translation as well as the rotation of the target formation. As depicted in Figure 6.7, all robots were able to successfully complete the formation switching task without any collision. Even though a conflict arises when at least two robots are assigned to the same role, the negotiation strategies were able to efficiently resolve this conflict. It is worth noting that the translation and rotation may be changed from one formation to another if there are some unassigned robots.

In the second experiment, the control strategy for coordinated path following in Section 5.5 is used to show the effectiveness of our framework. The C^2 reference path and the initial positions of three robots are plotted in Figure 6.9(a). The robots had to switch from one formation to another, see Figure 6.8. We assume that they have a mechanism that can make a decision on which formation they want to select according to the environment. To avoid collisions, the concept of attractive and repulsive forces was applied to modify the reference position. As seen in Figure 6.9(b) and Figure 6.9(c), the robots

(a) Triangle (b) X shape (c) Circle

Figure 6.6: Illustration of three different formation configurations. Squares represent the target positions in the formation. These formations must satisfy the connectivity assumption.



(a) Start (b) (c) Triangle (d)

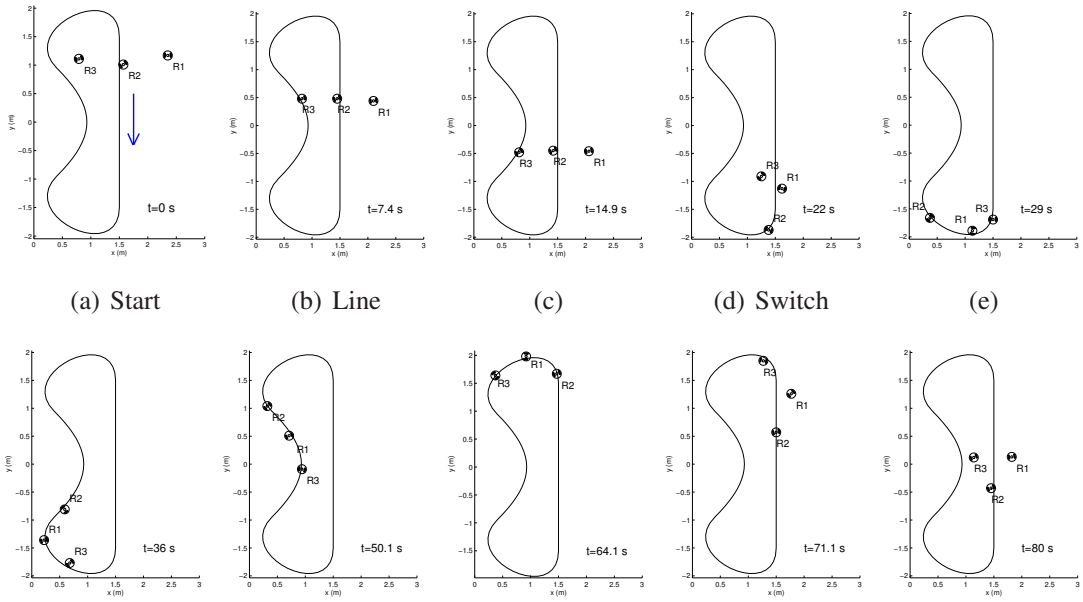
(e) X shape (f) (g) (h) Circle

Figure 6.7: The experimental results on formation switching. (a) each robot starts at a different random position, (c),(e) and (h) are the snapshots when the formation task is completed, corresponding to Figure 6.6(a), (b), and (c), respectively. Note that robot $R4$ in (e) is an unassigned robot because in the X shape formation the number of roles is less than the number of robots.

maintain a line formation, illustrated in Figure 6.8(a), while following a reference path. Figure 6.9(d) and Figure 6.9(e) show the snapshots where robots switch to a column formation, shown in Figure 6.8(b), and the robots then switch to a triangular formation, as depicted in Figure 6.9(i) and Figure 6.9(j). We can conclude that these robots can keep a desired formation, switch formations, and follow a reference successfully.

(a) Line (b) Column (c) Triangle

Figure 6.8: Illustration of three different formation configurations. Squares represent the target positions in the formation. These formations must satisfy the connectivity assumption.



(a) Start (b) Line (c) (d) Switch (e)

(f) Column (g) (h) (i) Switch (j) Triangle

Figure 6.9: The experimental results on coordinated path following and formation switching.

6.3 Discussions and Summary

In this chapter, we focused on formation control tasks that show what capabilities formation control can have. We presented a novel solution for the distributed role assignment in formation switching tasks. Our framework is mainly based on average consensus protocols and negotiation strategies. From the results of the first experiment, we can conclude that each robot was able to reconfigure itself from one formation to another successfully. It was also able to detect the completion of each task and automatically move onto the next task in its queue. In the second experiment, we were able to achieve coordinated path following and formation switching, i.e., each robot was able to be steered along a set of given spatial paths and to switch formations, while keeping a desired inter-vehicle coordination pattern.

Since no global cost is optimized in this work, the computational complexity and the theoretical proof need to be further investigated.

Chapter 7

Conclusions and Future Work

In conclusion, the focus of this dissertation was to develop control schemes that are suitable for path following control, coordinated path following control, and formation control of wheeled mobile robots including omnidirectional mobile robots and unicycle mobile robots. Successful real-time motion control of a group of mobile robots has been shown in our experimental results. In Chapter 4 and Chapter 5, the path variable and the path derivative played a major role through the MPC framework in the problem of path following control, while graph-theory consensus protocols were employed in Section 5.5 and Chapter 6. In the following section, the main contributions of this dissertation are summarized and suggestions for possible future work are provided.

7.1 Dissertation Summary

In Chapter 4, the main contribution to solve path following problems is to integrate the path derivative, also referred to as the velocity of a so-called virtual vehicle, into the local cost function of MPC schemes. This approach can overcome stringent initial condition constraints [211] in path following problems and it can take into account actuator saturation straightforwardly in optimization problems so that a robot can travel safely at a desired velocity. Furthermore, in Section 4.4, we achieved smooth convergence to the reference path with time constraints and also took into consideration obstacle avoidance. The proposed control law offers some advantages over other available trajectory tracking control laws. For example, it eliminates aggressiveness by forcing convergence to the desired path in a smooth way and its control signals are less likely pushed into saturation, while time constraints are satisfied.

In Chapter 5, we investigated coordinated path following problems. A challenge is the design of decentralized controllers, instead of centralized approaches. In this setting, each robot has only partial information about the state of other neighboring robots. To solve this problem, we provided two solutions. In the first solution, we incorporated the concept of a so-called virtual vehicle into the MPC framework. We proposed two strategies under this MPC framework, i.e., the leader-following method and the distributed method. Robots are presumed to have decoupled dynamics, modeled by nonlinear ordinary differential equations. In the distributed method, since the path's parameter was

used as a coupling variable between robots, it was included in a quadratic cost function of a local optimal control law. This coupling variable is the local information exchanged between neighboring robots to achieve formation keeping tasks. As stated in [65], the key requirements for stability are that each subsystem must not deviate too far from the previous open-loop state trajectory, and that the prediction horizon updates happen sufficiently fast. Even though very fast updates are currently not achieved in our study, experimental results show efficiency of our proposed algorithm. The second controller given in Section 5.5 under the assumption that the formation graph is bidirectional and connected is based on a Lyapunov function and consensus protocols. This solution has been shown to be computationally simple.

Based on these two proposed solutions, if the advantages of MPC schemes including constraint handling are not a dominating factor, the second solution might be more favored because of less computational demand. However, improvement of MPC schemes should be further carried out because a MPC controller is a promising control approach. For example, in some applications, a cooperative objective of multi-robot systems can be hard to mold into the framework of many other control approaches, even when constraints are not a dominating factor. MPC schemes can govern this problem easily.

In Chapter 6, we presented a novel algorithm for distributed role assignment in formation and formation switching. Consensus based protocols have been extensively used to obtain the common translation and rotation of the target formation and to detect the completion of formation tasks. Moreover, the number of robots participating in formation tasks need not to be known by team members. As shown in our experimental results, the proposed algorithms are completely distributed under the assumption that formation graphs are bidirectional and connected.

7.2 Future Research Directions

The following subsection outlines some possible extensions and future directions of the work in this dissertation.

7.2.1 A Unified Path Following Control Framework

This framework might be an extension to path following control in Chapter 4. It should provide a general algorithm for path following control in a dynamic environment, i.e., an environment characterized by moving obstacles. Thus, an interesting problem would be an integration of path following control, global path planning, velocity assignment, and obstacle avoidance so that a mobile robot can travel safely and effectively in a partially-known or unknown environment. Our preliminary results on this unified framework can be found in [119].

A common approach to obstacle avoidance is either to modify the entire pre-planned path at the navigation system level based on current range-sensor data [134] or to use

a reactive controller. The path replanning strategy requires the system to design a safe path that ensures the robot avoids obstacles and reaches a desired goal. Then control objective is just to follow this safe path. The advantage of this method is that, under the assumption that an accurate map of the environment is available, it designs a global solution that guarantees the system reaches the goal, if this solution exists. The drawback is the necessary computational time. On the other hand, the reactive path following controller, which acts directly at the control level, is generally considered as a behavior-based approach. Several behaviors are defined (avoid-obstacle, move-to-goal, etc.) and a switching scheme is designed to switch between these behaviors.

Besides steering the robot to the desired path, assigning a velocity profile to the robot can be a task, in which the forward velocity is used as an extra degree of freedom. The velocity profile should be shaped to comply with robot and environment constraints while the robot is following a path. For example, in case of sharp turns, the linear and angular robot velocities, must be constrained such that the turn is appropriately restrained and smooth. A large heading velocity together with a large angular velocity will jeopardize the stability and safety of the robot or cause saturation in the motors which will cause overshooting and long settling time. Thus, a method to generate an optimal velocity profile satisfying constraints has to be studied.

7.2.2 Communication Structures

Since communication networks play an important role in multi-robot systems, communication problems including quality, reliability and capability, have to be carefully considered in the controller design phase. For example, communication topologies between robots can be either static or switching with time, communication may introduce time delays in signal propagation among members, and data packets may be lost if communication capabilities are limited.

In this dissertation, information graphs were assumed to be bidirectional and connected. The problem of designing formation control systems becomes more challenging when connectivity has to be maintained over time and only unidirectional information exchange is allowed instead of bidirectional information exchange. This will be important in applications where bidirectional communication or sensing are not available. To handle asynchronous implementation problems is also an interesting topic in the future research.

7.2.3 Formation Control Subproblems

Formation control subproblems consist of formation shape generation, formation reconfiguration, formation tracking, and role assignment in formation. Most published contributions have been devoted to the area of formation tracking. Thus, we should pay more attention to the other subproblems in formation control. In particular, it would be inter-

esting to study an optimal way for formation reconfiguration, a mechanism of selecting a suitable formation, and optimization issues of role assignment.

Furthermore, to unify the formation control framework, which combines control theory, communication networks, and computer science, research in hybrid systems might be a step in the right direction since a logic-based supervisor at high level and continuous motion control at low level can be blended into one control structure.

For example, a set of formation shapes are generated at the beginning. Each robot then uses a decision making mechanism to select a suitable formation. They maintain a formation and avoid obstacles while navigating to the goal. If they are under the situation that a reconfiguration is required, they have to agree upon which formation will be selected and they have to generate an optimal way to reconfigure themselves.

7.2.4 A Real-time MPC Framework

Although there has been a considerable progress in the area of NMPC over the recent decades, there are still many problems that must be overcome in practice. Examples are the efficient and reliable online implementation, the development of robust NMPC approaches, the compensation of delays, and the design of output-feedback NMPC approaches [8].

As shown in our experimental results, the MPC framework can be used beyond process control. However, the main obstacle in applying the MPC technology in real-time applications such as wheeled mobile robots is that the optimization problem is computationally quite demanding, especially for nonlinear systems. In order to reduce the online computational requirements, there are a number of directions in which future research on this problem can proceed. The first direction is to apply function approximations, such as artificial neural networks, which can be trained off-line to represent the optimal control law. Second, explicit MPC techniques, such as multi-parametric quadratic programming (mp-QP) approaches, may be employed since they can handle constrained MIMO linear models as well as constrained MIMO piecewise linear models [23], where part of the computations are performed off-line.

An explicit MPC scheme is formulated and solved as mixed-integer linear programs (MILPs) which can be translated into equivalent piecewise affine state-feedback controllers requiring significantly lower computational effort that enables real-time implementation. Particularly, using the mp-QP solver from the Hybrid toolbox [21], we can obtain a representation of the MPC controller as a set of (possibly overlapping) piecewise affine controllers. During the online operation, at each step for each controller the value function is evaluated, and the input corresponding to the minimum cost is applied.

Furthermore, any relationship which can be expressed as proportional logic can be translated into this hybrid framework. In the area of control, by including integer variables representing logic propositions, it is possible to combine logic based control decisions within the MPC framework. This allows innovative control strategies which can be capable of prioritizing constraints as well as altering the control objective depend-

ing upon the positions of control inputs. By implementing such a strategy, controller performance can be improved. For example, for multi-variable systems wherein saturation of one of the manipulated variables prevents all objectives from being met, integer constraints can be used to improve performance and prioritize the objectives.

The third direction to reduce the online computational requirements relates to open-loop optimization solvers. We should improve and test several nonlinear optimization algorithms which can enlarge the range of conditions for which a nonlinear MPC controller becomes real-time implementable.

In the case of decentralized MPC, there are many open problems. For example, less conservative methods for stability and feasibility should be addressed. Reliable strategies for handling possible disruptions and delays in the communication of input trajectories among subsystems are required. To implement cooperative decentralized MPC for systems with fast sampling rates, one may require techniques that allow a quick evaluation of the MPC optimization problem. The possibility of employing explicit MPC techniques for decentralized MPC should be investigated.

Appendix A

Velocity Derivations of Offset-varying Curves

In this Appendix, we provide the derivations of (5.4) and (5.5) in Chapter 5.

Let the curve be parameterized by path length $s : \mathbf{r} = \mathbf{r}(s)$ and let l be the path length parameter of the offset-varying curve $\mathbf{o}(s) = \mathbf{r}(s) + \mathbf{n}(s)q(s)$, see Figure A.1. Let $\mathbf{t}_c(s)$ denote the unit tangent vector of the progenitor curve: $\mathbf{t}_c(s) = d\mathbf{r}/ds$ and the formulas of Frenet are given as follows:

$$\frac{d\mathbf{t}}{ds} = k\mathbf{n}, \quad \frac{d\mathbf{n}}{ds} = -k\mathbf{t} \tag{A.1}$$

where $\mathbf{t}$ is a unit vector tangent to the curve, $\mathbf{n}$ is a unit normal vector such that $(\mathbf{t}, \mathbf{n})$ forms the counter-clockwise oriented frame, k is the curvature at a point measuring the rate of curving as the point moves along the curve with unit speed.

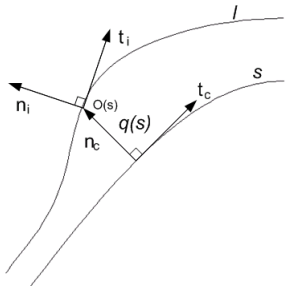


Figure A.1: The offset-varying curve with its progenitor curve.

Using the Frenet formulas we get (we omit 's' in order to make them easier to read):

$$\begin{aligned}\frac{d}{ds}\mathbf{o} &= \frac{d}{ds}(\mathbf{r}+\mathbf{n}q)\\ &= \mathbf{t}_c+(\mathbf{n}_c\frac{dq}{ds}+q\frac{d\mathbf{n}_c}{ds})\\ &= \mathbf{t}_c+\mathbf{n}_c\frac{dq}{ds}+q(-k_c\mathbf{t}_c)\\ &= (1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c\end{aligned}$$

Let $\mathbf{t}_i$ be a unit vector tangent to the offset-varying curve, i.e.,

$$\mathbf{t}_i=\frac{(1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c}{H} \tag{A.2}$$

where $H=\sqrt{(1-k_cq)^2+(\frac{dq}{ds})^2}$.

Then, we get

$$\frac{d}{ds}\mathbf{o}=\sqrt{(1-k_cq)^2+(\frac{dq}{ds})^2}\ \ \mathbf{t}_i. \tag{A.3}$$

This means

$$\frac{dl}{ds}=\sqrt{(1-k_cq)^2+(\frac{dq}{ds})^2}. \tag{A.4}$$

Based on the definition of the curvature: $\frac{d\mathbf{t}}{ds}=k\mathbf{n}$, that means we have to calculate $\frac{d\mathbf{t}_i}{dl}$ in order to get $k_i\mathbf{n}_i$. First we have to find $\frac{d\mathbf{t}_i}{ds}$ and then apply the chain rule: $\frac{d\mathbf{t}_i}{dl}=\frac{d\mathbf{t}_i}{ds}\frac{ds}{dl}$.

$$\begin{aligned}\frac{d\mathbf{t}_i}{ds} &= \frac{d}{ds}(\frac{(1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c}{H})\\ &= \frac{H\frac{d}{ds}((1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c)-((1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c)\frac{dH}{ds}}{H^2}\\ &= \frac{-k_c\frac{dq}{ds}\mathbf{t}_c-q\frac{dk_c}{ds}\mathbf{t}_c+(1-k_cq)(k_c\mathbf{n}_c)+\mathbf{n}_c\frac{d^2q}{ds^2}+\frac{dq}{ds}(-k_c\mathbf{t}_c)}{H}\\ &\quad -\frac{((1-k_cq)\mathbf{t}_c+\frac{dq}{ds}\mathbf{n}_c)((1-k_cq)(-k_c\frac{dq}{ds}-q\frac{dk_c}{ds})+\frac{dq}{ds}\frac{d^2q}{ds^2})}{H^3}\end{aligned}$$

Let $G=(1-k_cq)(-k_c\frac{dq}{ds}-q\frac{dk_c}{ds})+\frac{dq}{ds}\frac{d^2q}{ds^2})$, we get

$$\frac{d\mathbf{t}_i}{ds}=\frac{1}{H}(-2k_c\frac{dq}{ds}-q\frac{dk_c}{ds}-(1-k_cq)\frac{G}{H^2})\mathbf{t}_c+\frac{1}{H}(k_c-k_c^2q+\frac{d^2q}{ds^2}-\frac{dq}{ds}\frac{G}{H^2})\mathbf{n}_c. \tag{A.5}$$

Since $\frac{d\mathbf{t}_i}{dl} = \frac{d\mathbf{t}_i}{ds}\frac{ds}{dl} = k_i\mathbf{n}_i$, using (A.4) and (A.5), finally we obtain

$$k_i = \text{sign}(b)\frac{\sqrt{a^2+b^2}}{H^2} \tag{A.6}$$

where

$$\begin{aligned} a &= (-2k_c\frac{dq}{ds} - q\frac{dk_c}{ds} - (1-k_cq)\frac{G}{H^2}) \\ b &= (k_c - k_c^2q + \frac{d^2q}{ds^2} - \frac{dq}{ds}\frac{G}{H^2}). \end{aligned}$$

To calculate the translational velocity of the follower (u_i), the point on the offset-varying curve can be specified as $\mathbf{o}(t) = \mathbf{r}(t) + \mathbf{n}(t)q(t)$. The time derivative of point $\mathbf{o}(t)$ is obtained by

$$\begin{aligned} \frac{d}{dt}\mathbf{o}(t) &= \frac{d}{dt}(\mathbf{r}(t) + \mathbf{n}_c(t)q(t)) \\ &= \frac{d}{dt}\mathbf{r}(t) + \frac{d}{dt}(\mathbf{n}_c(t)q(t)) \\ &= \frac{ds}{dt}\mathbf{t}_c + (\mathbf{n}_c(t)\frac{dq}{ds}\frac{ds}{dt} + q(t)\frac{d\mathbf{n}_c(t)}{dt}) \\ &= \frac{ds}{dt}\mathbf{t}_c + \mathbf{n}_c(t)\frac{dq}{ds}\frac{ds}{dt} + q(t)(-k_c\mathbf{t}_c)\frac{ds}{dt} \\ &= \frac{ds}{dt}((1-k_cq(t))\mathbf{t}_c + \frac{dq}{ds}\mathbf{n}_c(t)) \end{aligned}$$

Let $\frac{ds}{dt} = u_c(t)$ and $\mathbf{t}_i$ be a unit vector tangent to the offset-varying curve, i.e.,

$$\mathbf{t}_i = \frac{(1-k_cq)\mathbf{t}_c + \frac{dq}{ds}\mathbf{n}_c}{H} \tag{A.7}$$

where $H = \sqrt{(1-k_cq)^2 + (\frac{dq}{ds})^2}$. Then we get

$$\begin{aligned} u_i(t)\mathbf{t}_i &= u_c(t)\sqrt{(1-k_cq(t))^2 + (\frac{dq}{ds})^2}\ \ \mathbf{t}_i \\ u_i(t) &= Hu_c(t). \end{aligned}$$

In summary, when we know the curvature and the translational velocity at the point on the reference path, we can calculate the velocity profiles at the point on the offset curve using the following equations:

$$u_i = Hu_c \tag{A.8}$$

$$\omega_i = k_iu_i \tag{A.9}$$

$$k_i = \text{sign}(b)\frac{\sqrt{a^2+b^2}}{H^2} \tag{A.10}$$

However, from (A.10), if $q(s) = 1/k_c(s)$ and $dq(s)/ds = 0$, then the offset-varying curve is not smooth; a singularity (so called cusp) appears. When the offset value increases, the singularities (cusps) might be developed.

In general, the curvature of the path is limited so we have additional constraints:

$$\mid k_c \mid \leq k_{c,max} \qquad \mid k_i \mid \leq k_{i,max} \tag{A.11}$$

Bibliography

[1] L. Acar. Some examples for the decentralized receding horizon control. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1356–1359, Tucson, AZ, December 1992.

[2] A. P. Aguiar, A. N. Atassi, and A. Pascoal. Regulation of a nonholonomic dynamic wheeled mobile robot with parametric modeling uncertainty using Lyapunov functions. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2995–3000, Sydney, Australia, December 2000.

[3] A. P. Aguiar, D. B. Dačić, J. P. Hespanha, and P. Kokotović. Path-following or reference-tracking? An answer relaxing the limits to performance. In *Proceedings of the IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[4] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino. Closed loop steering of unicycle-like vehicles via Lyapunov techniques. *IEEE Robotics and Automation Magazine*, 2(1):27–35, March 1995.

[5] B. M. Åkesson and H. T. Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16:937–946, 2006.

[6] S. A. Al-Hiddabi and N. H. McClamroch. Tracking and maneuver regulation control for nonlinear non-minimum phase systems: application to flight control. *IEEE Transactions on Control Systems Technology*, 10(6):780–792, 2002.

[7] J. Albuquerque, V. Gopal, G. Staus, L. Biegler, and E. Ydstie. Interior point SQP strategies for large-scale, structured process optimization problems. *Computers & Chemical Engineering*, 23(4):543–554, May 1999.

[8] F. Allgöwer, R. Findeisen, and Z. K. Nagy. Nonlinear model predictive control: from theory to application. *Journal of Chinese Institute of Chemical Engineers*, 35(3):299–315, 2004.

[9] C. Altafini. Following a path of varying curvature as an output regulation problem. *IEEE Transactions on Automatic Control*, 47(9):1551–1556, September 2002.

[10] O. Amidi. Integrated mobile robot control. Master's thesis, Dept. of Electrical and Computer Engineering, CMU, Pittsburgh, PA, 1990.

[11] M. R. Anderson and A. C. Robbins. Formation flight as a cooperative game. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pages 244–251, Boston, MA, August 1998.

[12] G. Antonelli, F. Arrichiello, and S. Chiaverini. Flocking for multi-robot systems via the null-space-based behavioral control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1409–1414, Nice, France, September 2008.

[13] T. Arai, E. Pagello, and L. E. Parker. Guest editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[14] M. Arcak. Passivity as a design tool for group coordination. *IEEE Transactions on Automatic Control*, 52(8):1380–1390, August 2007.

[15] H. Asama, M. Sato, L. Bogoni, H. Kaetsu, A. Mitsumoto, and I. Endo. Development of an omni-directional mobile robot with 3 DOF decoupling drive mechanism. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1925–1930, Nagoya, Japan, May 1995.

[16] M. Bak, N. K. Poulsen, and O. Ravn. Path following mobile robot in the presence of velocity constraints. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, 2001.

[17] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):1–15, 1998.

[18] T. D. Barfoot and C. M. Clark. Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46(2):65–78, 2004.

[19] R. A. Bartlett, A. Wächter, and L. T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, IL, June 2000.

[20] R. W. Beard, J. Lawton, and F. Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6): 777–790, November 2001.

[21] A. Bemporad. Hybrid toolbox - user's guide, February 2010. `http://www.dii.unisi.it/hybrid/toolbox`.

[22] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[23] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[24] D. P. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66, 1992.

[25] D. P. Bertsekas and D. A. Castañón. Parallel synchronous and asynchronous implementation of the auction algorithm. *Parallel Computing*, 17:707–732, 1991.

[26] L. T. Biegler and I. E. Grossmann. Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192, 2004.

[27] R. W. Brockett. *Differential Geometric Control Theory*, chapter Asymptotic stability and feedback stabilization, pages 181–191. Birkhauser, Boston, 1983.

[28] M. Broxvall, B. S. Seo, and W. Y. Kwon. The PEIS kernel: A middleware for ubiquitous robotics. In *Proceedings of the IROS-07 Workshop on Ubiquitous Robotics Space Design and Applications*, San Diego, CA, October 2007.

[29] B. J. Brunell, R. R. Bitmead, and A. J. Connolly. Nonlinear model predictive control of an aircraft gas turbine engine. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4649–4651, Las Vegas, Nevada, December 2002.

[30] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, June 2005.

[31] G. Campion, G. Bastin, and B. D'Andréa-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62, February 1996.

[32] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, February 2002.

[33] M. Cao, A. S. Morse, and B. D. O. Anderson. Coordination of an asynchronous multi-agent system via average. In *Proceedings of the IFAC world Congress*, Prague, Czech Republic, July 2005.

[34] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[35] R. Carelli, H. Secchi, and V. Mut. Algorithms for stable control of mobile robots with obstacle avoidance. *Latin American Applied Research*, 29(3/4):191–196, 1999.

[36] S. Carpin and L. Parker. Cooperative motion coordination amidst dynamic obstacles. In *Proceedings of the Symposium on Distributed Autonomous Robotic Systems*, pages 145–154, Fukuoka, Japan, 2002.

[37] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Coordination among heterogeneous robotic soccer players. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1385–1390, Takamatsu, Japan, October 2000.

[38] L. Cavagnari, L. Magni, and R. Scattolini. Neural network implementation of nonlinear receding-horizon control. *Neural Computing & Applications*, 8(1):86–92, 1999.

[39] A. Cervantes, A. Wachter, R. Tutuncu, and L. Biegler. A reduced space interior point strategy for optimization of differential algebraic systems. *Computers & Chemical Engineering*, 24(1):39–51, April 2000.

[40] C. C. Chen and L. Shaw. On receding horizon feedback control. *Automatica*, 18: 349–352, 1982.

[41] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[42] Y. Q. Chen and Z. M. Wang. Formation control: a review and a new consideration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3181–3186, Edmonton, Canada, August 2005.

[43] A. Cherubini, F. Chaumette, and G. Oriolo. A position-based visual servoing scheme for following paths with nonholonomic mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1648–1654, Nice, France, September 2008.

[44] S.-J. Chung and J. J. E. Slotine. Cooperative robot control and concurrent synchronization of Lagrangian systems. *IEEE Transactions on Robotics*, 25(3):686–700, June 2009.

[45] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control, Part I: The basic algorithm. *Automatica*, 23(2):137–148, 1987.

[46] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control, Part II: Extension and interpretations. *Automatica*, 23(2):149–160, 1987.

[47] L. Consolini, F. Morbidi, D. Prattichizzo, and M. Tosques. A geometric characterization of leader-follower formation control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2397–2402, Roma, Italy, April 2007.

[48] Intel Corporation. Opencv, February 2010. `http://opencv.willowgarage.com/wiki/`.

[49] J. B. Coulaud, G. Campion, G. Bastin, and M. De Wan. Stability analysis of a vision-based control design for an autonomous mobile robot. *IEEE Transactions on Robotics*, 22(5):1062–1069, October 2006.

[50] C.R. Cutler and B.L. Ramaker. Dynamic matrix control - A computer control algorithm. In *Proceedings of the Joint Automatic Control Conference*, San Francisco, CA, 1980.

[51] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, October 2002.

[52] M. C. De Gennaro and A. Jadbabaie. Formation control for a cooperative multi-agent system using decentralized navigation functions. In *Proceedings of the American Control Conference*, pages 1346–1351, Minneapolis, MN, June 2006.

[53] M. Defoort, T. Floquet, A. Kokosy, and W. Perruquetti. Sliding-mode formation control for cooperative autonomous mobile robots. *IEEE Transactions on Industrial Electronics*, 55(11):3944–3953, November 2008.

[54] J. C. Derenick and J. R. Spletzer. Convex optimization strategies for coordinating large-scale robot formations. *IEEE Transactions on Robotics*, 23(6):1252–1259, December 2007.

[55] J. P. Desai. A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525, June 2002.

[56] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, December 2001.

[57] M. Di Rocco, S. Panzieri, and A. Priolo. Formation control through environment pattern recognition for a multi-robot architecture. In *Proceedings of the European Conference on Mobile Robots*, pages 241–246, Mlini/Dubrovnik, Croatia, September 2009.

[58] F. Diaz del Rio, G. J. Moreno, J. L. S. Ramos, C. A. A. Rodriguez, and A. A. C. Balcells. A new method for tracking memorized paths: application to unicycle robots. In *Proceedings of the 10th IEEE Mediterranean Conference on Control and Automation*, Lisbon, Portugal, July 2002.

[59] S. L. Dickerson and B. D. Lapin. Control of an omni-directional robotic vehicle with Mecanum wheels. In *Proceedings of the National Telesystems Conference*, pages 323–328, Atlanta, USA, March 1991.

[60] T. Dierks and S. Jagannathan. Neural network control of mobile robot formations using RISE feedback. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 39(2):332–347, April 2009.

[61] D. V. Dimarogonas and K. J. Kyriakopoulos. On the rendezvous problems for multiple nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(5): 916–922, May 2007.

[62] K. D. Do. Formation tracking control of unicycle-type mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2391–2396, Roma, Italy, April 2007.

[63] W. Dong and J. A. Farrell. Consensus of multiple nonholonomic systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2270–2275, Cancun, Mexico, December 2008.

[64] G. Dudek, M. R. M. Jenkin, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3:375–397, 1996.

[65] W. B. Dunbar and R. M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549–558, 2006.

[66] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, December 2001.

[67] M. Egerstedt, X. Hu, and A. Stotsky. Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control*, 46(11):1777–1782, November 2001.

[68] P. Encarnação and A. Pascoal. 3D path following for autonomous underwater vehicle. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2977–2982, Sydney, Australia, December 2000.

[69] H. V. Essen and H. Nijmeijer. Non-linear model predictive control of constrained mobile robots. In *Proceedings of the European Control Conference*, pages 1157–1162, Porto, Portugal, September 2001.

[70] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[71] L. Fang and P. J. Antsaklis. Decentralized formation tracking of multi-vehicle systems with nonlinear dynamics. In *Proceedings of the Mediterranean Conference on Control and Automation*, Ancona, Italy, June 2006.

[72] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, September 2004.

[73] R. Fierro and F. L. Lewis. Control of a nonholonomic mobile robot using neural networks. *IEEE Transactions on Neural Networks*, 9(4):689–600, July 1998.

[74] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 21–26, Seoul, Korea, May 2001.

[75] R. Fletcher. *Practical methods of optimization*, volume 2nd ed. John Wiley & Sons Inc., 1987.

[76] R. Franz, M. Milam, and J. Hauser. Applied receding horizon control of the Caltech ducted fan. In *Proceedings of the American Control Conference*, pages 3735–3740, Anchorage AK, May 2002.

[77] J. Fredslund and M. Matarić. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[78] R. A. Freeman, P. Yang, and K. M. Lynch. Distributed estimation and control of swarm formation statistics. In *Proceedings of the American Control Conference*, Minneapolis, MN, June 2006.

[79] G. W. Gamage, G. K. I. Mann, and R. G. Gosine. Discrete event systems based formation control framework to coordinate multiple nonholonomic mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4831–4836, St. Louis, USA, October 2009.

[80] B. P. Gerkey and M. Matarić. A framework for studying multi-robot task allocation. In *Proceedings of the International Workshop on Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 15–26, Washinton, DC, May 2003.

[81] E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29(1):58–81, 2003.

[82] R. Ghabcheloo, A. Pascoal, C. Silvestre, and D. Carvalho. Coordinated motion control of multiple autonomous underwater vehicles. In *Proceedings of the International Workshop on Underwater Robotics*, pages 41–50, Genoa, Italy, November 2005.

[83] R. Ghabcheloo, A. Aguiar, A. Pascoal, C. Silvestre, I. Kaminer, and J. Hespanha. Coordinated path-following in the presence of communication losses and time delays. *SIAM - Journal on Control and Optimization*, 48(1):234–265, 2009.

[84] J. Ghomman, M. Saad, and F. Mnif. Formation path following control of unicycle-type mobile robots. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1966–1972, Pasadena, CA, May 2008.

[85] P. Gill, W. Murray, and M. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 1997.

[86] C. Godsil and G. Royle. *Algebraic Graph Theory*, volume 207. Springer-Verlag, New York, 2001.

[87] D. Gu and H. Hu. Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems*, 39(2):73–86, 2002.

[88] D. Gu and H. Hu. Receding horizon tracking control of wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(4):743–749, July 2006.

[89] D. Gu and H. Hu. A model predictive controller for robots to follow a virtual leader. *Robotica*, 27(6):905–913, October 2009.

[90] Y. Guo, D. Hill, and Y. Wang. Nonlinear decentralized control of large-scale power systems. Technical Report EE-98020, Electrical and Information Engineering School, University of Sydney, 2006.

[91] M. A. Haque and M. Egerstedt. Decentralized formation selection mechanisms inspired by foraging bottlenose dolphins. In *Proceedings of the Mathematical Theory of Networks and Systems*, Blacksburg, VA, July 2008.

[92] R. Hedjar, R. Toumi, P. Boucher, and D. Dumur. Finite horizon nonlinear predictive control by the Taylor approximation: application to robot tracking trajectory. *International Journal of Applied Mathematics and Computer Science*, 15(4):527–540, 2005.

[93] P. Heinemann. *Cooperative Multi-Robot Soccer in a Highly Dynamic Environment*. PhD dissertation, University of Tübingen, Faculty of Information and Cognition Science, 2007.

[94] P. Heinemann, J. Haase, and A. Zell. A combined Monte-Carlo localization and tracking algorithm for RoboCup. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1535–1540, Beijing, China, October 2006.

[95] P. Hines, D. Jia, and S. Talukdar. Distributed model predictive control for electric grids. In *Proceedings of the Carnegie Mellon Transmission Conference*, Pittsburgh, 2004.

[96] M. Hofmeister, M. Liebsch, and A. Zell. Visual self-localization for small mobile robots with weighted gradient orientation histograms. In *Proceedings of the 40th International Symposium on Robotics (ISR)*, pages 87–91, Barcelona, Spain, March 2009.

[97] A. Howard, L. E. Parker, and G. S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *International Journal Robotics Research*, 25(5-6):431–447, 2006.

[98] I.-A. F. Ihle, J. Jouffroy, and T. I. Fossen. Formation control of marine craft using constraint functions. In *Proceedings of the MTS/IEEE OCEANS*, pages 1023–1028, Washington D.C., September 2005.

[99] I.-A. F. Ihle, M. Arcak, and T. I. Fossen. Passivity-based designs for synchronized path-following. *Automatica*, 43(9):1508–1518, 2007.

[100] G. Indiveri. Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control. *IEEE Transactions on Robotics*, 25(1):164–171, February 2009.

[101] A. Jadbabaie, J. Yu, and J. Hauser. Stabilizing receding horizon control of nonlinear systems: A control Lyapunov function approach. In *Proceedings of the American Control Conference*, pages 1535–1539, San Diego, CA, June 1999.

[102] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[103] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48:988–1001, June 2003.

[104] A. Jadbabaie, N. Motee, and M. Barahona. On the stability of the Kuramoto model of coupled nonlinear oscillators. In *Proceedings of the American Control Conference*, pages 4296–4301, Boston, MA, June 2004.

[105] M. Ji, S. Azuma, and M. Egerstedt. Role assignment in multi-agent coordination. *International Journal of Assistive Robotics and Mechatronics*, 7(1):32–40, March 2006.

[106] D. Jia and B. Krogh. Distributed model predictive control. In *Proceedings of the American Control Conference*, pages 2767–2772, Arlington, VA, June 2001.

[107] D. Jia and B. Krogh. Min-max feedback model predictive control for distributed control with communication. In *Proceedings of the American Control Conference*, pages 4507–4512, Anchorage, Alaska, May 2002.

[108] Z.-P. Jiang, E. Lefeber, and H. Nijmeijer. Saturated stabilization and track control of a nonholonomic mobile robot. *Systems and Control Letters*, 42:327–332, 2001.

[109] T. Kalmár-Nagy, R. D'Andrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1):47–64, January 2004.

[110] K. Kanjanawanishkul. Formation control of omnidirectional mobile robot using distributed model predictive control. In *Proceedings of the 2nd International Conference on Robot Communication and Coordination (ROBOCOMM)*, pages 1–7, Odense, Denmark, March 2009.

[111] K. Kanjanawanishkul and A. Zell. A model-predictive approach to formation control of omnidirectional mobile robots. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, pages 2771–2776, Nice, France, September 2008.

[112] K. Kanjanawanishkul and A. Zell. Distributed model predictive control for coordinated path following control of omnidirectional mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2008)*, pages 3120–3125, Singapore, October 2008.

[113] K. Kanjanawanishkul and A. Zell. Path following for an omnidirectional mobile robot based on model predictive control. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009)*, pages 3341 – 3346, Kobe, Japan, May 2009.

[114] K. Kanjanawanishkul and A. Zell. Distributed role assignment in multi-robot formation. *Proceedings of the 7th Symposium on Intelligent Autonomous Vehicles (IAV)*, September 2010. Accepted for publication.

[115] K. Kanjanawanishkul, X. Li, and A. Zell. Nonlinear model predictive control of omnidirectional mobile robot formations. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS 2008)*, pages 41–48, Baden-Baden, Germany, July 2008.

[116] K. Kanjanawanishkul, M. Hofmeister, and A. Zell. Coordinated path following for mobile robots. In *Proceedings of Fachgespraech Autonome Mobile Systeme (AMS2009)*, pages 185–192, Karlsruhe, Germany, December 2009.

[117] K. Kanjanawanishkul, M. Hofmeister, and A. Zell. Smooth reference tracking of a mobile robot using nonlinear model predictive control. In *Proceedings of the 4th European Conference on Mobile Robots (ECMR)*, pages 161–166, Mlini/-Dubrovnik, Croatia, September 2009.

[118] K. Kanjanawanishkul, M. Hofmeister, and A. Zell. Experiments on formation switching for mobile robots. In *Proceedings of the 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2010)*, Montreal, Canada, July 2010. Accepted for publication.

[119] K. Kanjanawanishkul, M. Hofmeister, and A. Zell. Path following with an optimal forward velocity for a mobile robot. *Proceedings of the 7th Symposium on Intelligent Autonomous Vehicles (IAV)*, September 2010. Accepted for publication.

[120] S. S. Keerthi and E. G. Gilbert. Optimal, infinite horizon feedback laws for a general class of constrained discrete time systems: Stability and moving-horizon approximations. *Journal of Optimization Theory and Application*, 57:256–293, 1988.

[121] T. Keviczky, F. Borrelli, and G. J. Balas. A study on decentralized receding horizon control for decoupled systems. In *Proceedings of the American Control Conference*, pages 4921–4926, Boston, Massachusetts, June 2004.

[122] T. Keviczky, F. Borrelli, and G. J. Balas. Stability analysis of decentralized RHC for decoupled systems. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, pages 1689–1694, Seville, Spain, December 2005.

[123] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[124] W. K. Kim, B.-J. Yi, and D. J. Lim. Kinematic modeling of mobile robots by transfer method of augmented generalized coordinates. *Journal of Robotic Systems*, 21 (6):275–300, 2004.

[125] G. Klančar and I. Škrjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6):460–469, 2007.

[126] V. S. Kodogiannis, P. J. G. Lisboa, and J. Lucas. Neural network modeling and control for underwater vehicles. *Artificial Intelligence in Engineering*, 10(3):203–212, 1996.

[127] I. Kolmanovsky and N. H. McClamroch. Developments in nonholonomic control problems. *IEEE Control Systems Magazine*, 15(6):20–36, December 1995.

[128] S. L. D. Kothare and M. Morari. Contractive model predictive control for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 45(6): 1053–1071, June 2000.

[129] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2(1):83–97, 1955.

[130] W. H. Kwon and S. Han. *Receding Horizon Control: Model Predictive Control for State Models*. Springer-Verlag, London, 2005.

[131] G. Lafferriere, A. Williams, J. Caughman, and J. J. P. Veerman. Decentralized control of vehicle formations. *Systems and Control Letters*, 54(9):899–910, September 2005.

[132] W. F. Lages and J. A. V. Alves. Real-time control of a mobile robot using linearized model predictive control. In *Proceedings of the 4th IFAC Symposium on Mechatronic Systems*, pages 968–973, Heidelberg, Germany, September 2006.

[133] E. Lalish, K. A. Morgansen, and T. Tsukamaki. Formation tracking control using virtual structures and deconfliction. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5699–5705, San Diego, CA, December 2006.

[134] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, 2004.

[135] L. Lapierre, R. Zapata, and P. Lepinay. Combined path-following and obstacle avoidance control of a wheeled robot. *International Journal of Robotics Research*, 26(4):361–376, 2007.

[136] C. T. Lawrence and A. L. Tits. A computationally efficient feasible sequential quadratic programming algorithm. *SIAM Journal on Optimization*, 11:1092–1118, 2001.

[137] M. Lawrynczuk. A family of model predictive control algorithms with artificial neural networks. *International Journal of Applied Mathematics and Computer Science*, 17(2):217–232, 2007.

[138] J. Lawton, R. W. Beard, and B. Young. A decentralized approach to formation maneuvers. *IEEE Transactions on Robotics and Automation*, 19:933–941, December 2003.

[139] T. Lee, M. Leok, and N. H. McClamroch. A combinatorial optimal control problem for spacecraft formation reconfiguration. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5370–5375, New Orleans, LA, December 2007.

[140] M. Lemay, F. Michaud, D. Letourneau, and J.-M. Valin. Autonomous initialization of robot formation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3018–3023, New Orleans, LA, May 2004.

[141] M. A. Lewis and K. H. Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[142] X. Li and A. Zell. Motion control of an omnidirectional mobile robot. In *Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics*, pages 125–132, Angers, France, May 2007.

[143] X. Li, J. Xiao, and Z. Cai. Backstepping based multiple mobile robots formation control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 887– 892, Alberto, Canada, August 2005.

[144] X. Li, K. Kanjanawanishkul, and A. Zell. Nonlinear model predictive control of an omnidirectional mobile robot. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS 2008)*, pages 92–99, Baden-Baden, Germany, July 2008.

[145] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *IEEE Transactions on Automatic Control*, 49:622–629, 2004.

[146] Z. Lin, B. Francis, and M. Maggiore. Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE Transactions on Automatic Control*, 50(1):121–127, January 2005.

[147] Y. Liu, X. Wu, J. Jim Zhu, and J. Lew. Omni-directional mobile robot controller design by trajectory linearization. In *Proceedings of the American Control Conference*, pages 3423–3428, Denver, Colorado, June 2003.

[148] W. K. Loh, K. H. Low, and Y. P. Leow. Mechatronics design and kinematic modelling of a singularityless omni-directional wheeled mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3237–3242, Tapei, Taiwan, Sep. 2003.

[149] Y. Ma, J. Kosecka, and S. Sastry. Vision guided navigation for a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, 15(3):521–536, 1999.

[150] L. Magni and R. Scattolini. State-feedback MPC with piecewise constant control for continuous-time systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4625–4630, Las Vegas, USA, December 2002.

[151] L. Magni, D. M. Raimondo, and F. Allgöwer, editors. *Nonlinear Model Predictive Control – Towards New Challenging Applications*. Lecture Notes in Control and Information Sciences. Springer-Verlag, Berlin, 2009.

[152] G. L. Mariottini, G. J. Pappas, D. Prattichizzo, and K. Daniilidis. Vision-based localization of leader-follower formations. In *Proceedings of the IEEE Conference on Decision and Control*, pages 635–640, Seville, Spain, December 2005.

[153] G. L. Mariottini, G. Oriolo, and D. Prattichizzo. Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. *IEEE Transactions on Robotics*, 23(1):87–100, 2007.

[154] J. A. Marshall, M. E. Broucke, and B. A. Francis. Formations of vehicles in cyclic pursuit. *IEEE Transactions on Automatic Control*, 49(11):1964–1974, 2004.

[155] M. Matarić. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, December 1995.

[156] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.

[157] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.

[158] J. McClintock and R. Fierro. A hybrid system approach to formation reconfiguration in clutter environments. In *Proceedings of the Mediterranean Conference on Control and Automation*, pages 83–88, Ajaccio, France, June 2008.

[159] P. McDowell, J. Chen, and B. Bourgeois. UUV teams, control from a biological perspective. In *Proceedings of the Oceans 2002 Conference*, pages 331–337, Mississippi, MS, October 2002.

[160] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray. Asynchronous distributed averaging on communication networks. *IEEE/ACM Transactions on Networking*, 15(3):512–520, June 2007.

[161] A. Micaelli and C. Samson. Trajectory-tracking for unicycle-type and two-steering-wheels mobile robots. Technical Report 2097, INRIA Sophia-Antipolis, November 1993.

[162] N. Michael, M. Zavlanos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *Proceedings of the International Conference on Robotics and Automation*, pages 128–133, Pasadena, CA, May 2008.

[163] H. Michalska and D. Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, Nov: 1993.

[164] F. Michaud, D. Letourneau, M. Guilbert, and J.-M. Valin. Dynamic robot formations using directional visual perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2740–2745, Lausanne, Switzerland, October 2002.

[165] M. B. Milam, R. Franz, J. E. Hauser, and R. M. Murray. Receding horizon control of a vectored thrust flight experiment. *IEE Proceedings Control Theory and Applications*, 152(3):340–348, May 2003.

[166] H. J. Min, A. Drenner, and N. Papanikolopoulos. Vision-based leader-follower formations with limited information. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 351–356, Kobe, Japan, May 2009.

[167] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2436–2441, Las Vegas, NV, October 2003.

[168] M. Montemerlo, N. Roy, and S. Thrun. CARMEN, Carnegie Mellon Robot Navigation Toolkit, February 2010. `http://carmen.sourceforge.net`.

[169] M. Morari and J. Lee. Model predictive control: Past, present, and future. *Computers and Chemical Engineering*, 23(4/5):667–682, 1999.

[170] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(12):169–182, 2005.

[171] P. Morin and C. Samson. *Springer Handbook of Robotics*, chapter 34. Motion control of wheeled mobile robot, pages 799–826. Springer Berlin Heidelberg, 2008.

[172] R. M. Murray. Recent research in cooperative-control of multivehicle systems. *Journal of Dynamics, Systems, Measurement and Control*, 129(5):571–583, 2007.

[173] R. R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control: A survey. Technical Report 04-010, Delft Center for Systems and Control, Delft University of Technology, December 2004.

[174] J. E. Normey-Rico, J. Gomez-Ortega, and E. F. Camacho. A Smith-predictor-based generalized predictive controller for mobile robot path-tracking. *Control Engineering Practice*, 7(6):729–740, 1999.

[175] P. Ögren and N. E. Leonard. Obstacle avoidance in formation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2492–2497, Taipei, Taiwan, September 2003.

[176] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, March 2006.

[177] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49 (9):1520–1533, September 2004.

[178] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007.

[179] A. Ollero and O. Amidi. Predictive path tracking of mobile robots: Application to the CMU Navlab. In *Proceedings of the International Conference on Advanced Robotics*, pages 1081–1086, Pisa, Italy, June 1991.

[180] A. Ollero, A. Garciacerezo, and J. L. Martinez. Fuzzy supervisory path tracking of mobile robots. *Control Engineering Practice*, 2(2):313–319, 1994.

[181] G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: design, implementation and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, 2002.

[182] O. A. Orqueda and R. Fierro. Robust vision-based nonlinear formation control. In *Proceedings of the American Control Conference*, pages 1422–1427, Minneapolis, MN, June 2006.

[183] C. Ortiz, K. Konolige, R. Vincent, B. Morisset, A. Agno, M. Eriksen, D. Fox, B. Limketkai, J. Ko, B. Steward, and D. Schulz. Centibots: very large scale distributed robotic teams. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'04)*, pages 1022–1023, San Jose, California, 2004. AAAI Press / The MIT Press.

[184] M. Oubbati, M. Schanz, T. Buchheim, and P. Levi. Velocity control of an omnidirectional RoboCup player with recurrent neural networks. In *Proceedings of the International Symposium on RoboCup*, pages 691–701, Osaka,Japan, July 2005.

[185] A. Pant, P. Seiler, and K. Hedrick. Mesh stability of look-ahead interconnected systems. *IEEE Transactions on Automatic Control*, 47:403–407, February 2002.

[186] A. Papachristodoulou and A. Jadbabaie. Synchronization in oscillator networks: Switching topologies and non-homogeneous delays. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, pages 5692–5697, Seville, Spain, December 2005.

[187] T. Parisini and R. Zoppoli. A receding horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31:1443–1451, 1995.

[188] T. Parisini and R. Zoppoli. Neural approxinations for multistage optimal control of nonlinear stochastic systems. *IEEE Transactions on Automatic Control*, 41(6): 889–895, 1996.

[189] F. G. Pin and S. M. Killough. A family of omni-directional and holonomic wheeled platforms for mobile robots. *IEEE Transactions on Robotics and Automation*, 10(4):480–489, 1994.

[190] J. A. Primbs, V. Nevistic, and J. C. Doyle. A receding horizon generalization of pointwise min-norm controllers. *IEEE Transactions on Automatic Control*, 45(5): 898–909, May 2000.

[191] O. Purwin and R. D'Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1):13–32, January 2006.

[192] S. J. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, July 2003.

[193] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99 ():723–757, 1998.

[194] W. Ren. Consensus strategies for cooperative control of vehicle formations. *IET Control Theory & Applications*, 1(2):505–512, 2007.

[195] W. Ren and R. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, May 2005.

[196] W. Ren and R. W. Beard. A decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82, January 2004.

[197] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, April 2007.

[198] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of the Computer Graphics SIGGRAPH*, pages 25–34, Anaheim, CA, July 1987.

[199] A. G. Richards and J. P. How. A decentralized algorithm for robust constrained model predictive control. In *Proceedings of the American Control Conference*, pages 4261–4266, Boston, Massachusetts, June 2004.

[200] C. Samson. Control of chained systems: Application to path-following and time-varying point stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1):64–77, January 1995.

[201] J. Sanchez and R. Fierro. Sliding mode control for robot formations. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 438–443, Houston TX, October 2003.

[202] A. Scaglione, R. Pagliari, and H. Krim. The decentralized estimation of the sample covariance. In *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, pages 1722–1726, Pacific Grove, CA, October 2008.

[203] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44 (3):648–654, 1999.

[204] R. C. Scott and L. E. Pado. Active control of wind-tunnel model aeroelastic response using neural networks. *Journal of Guidance, Control, and Dynamics*, 23 (6):1100–1108, 2000.

[205] M. Seyr and S. Jakubek. Mobile robot predictive trajectory tracking. In *Proceedings of the International Conference on Informatics in Control*, pages 112–119, Barcelona, Spain, September 2005.

[206] D. H. Shim, H. J. Kim, and S. Sastry. Decentralized nonlinear model predictive control of multiple flying robots in dynamic environments. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3621–3626, Maui, Hawaii, December 2003.

[207] D. Siljak. Decentralized control and computations: status and prospects. *Annual Reviews in Control*, 20:131–141, 1996.

[208] R. Skjetne, I.-A. F. Ihle, and T. I. Fossen. Control by synchronizing multiple maneuvering systems. In *Proceedings of the IFAC Conference on Maneuvering and Control of Marine Crafts*, pages 280–285, Girona, Spain, September 2003.

[209] R. Skjetne, T.I. Fossen, and P. V. Kokotović. Robust output maneuvering for a class of nonlinear systems. *Automatica*, 40(3):373–383, 2004.

[210] S. L. Smith and F. Bullo. Target assignment for robotic networks: Asymptotic performance under limited communication. In *Proceedings of the American Control Conference*, pages 1155–1160, New York, July 2007.

[211] D. Soeanto, L. Lapierre, and A. Pascoal. Adaptive non-singular path-following, control of dynamic wheeled robots. In *Proceedings of the International Conference on Advanced Robotics*, pages 1387–1392, Coimbra, Portugal, June 2003.

[212] R. Solea and U. Nunes. Trajectory planning with velocity planner for fully-automated passenger vehicles. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 474–480, Toronto, Canada, September 2006.

[213] J.-B. Song and K.-S. Byun. Design and control of a four-wheeled omnidirectional mobile robot with steerable omnidirectional wheels. *Journal of Robotic Systems*, 21(4):193–208, 2004.

[214] P. Spellucci. An SQP method for general nonlinear programs using only equality constrained subproblems. *Mathematical Programming*, 82(3):413–448, 1998.

[215] D. M. Stipanovic, G. Inalhan, R. Teo, and C. J. Tomlin. Decentralized overlapping control of a formation of unmanned aerial vehicles. *Automatica*, 40(8):1285–1296, August 2004.

[216] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.

[217] D. Stonier, S.-H. Cho, S.-L. Choi, N. S. Kuppuswamy, and J.-H. Kim. Nonlinear slip dynamics for an omnidirectional mobile robot platform. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2367–2372, Roma, Italy, April 2007.

[218] D. Swaroop and J. K. Hedrick. String stability of interconnected systems. *IEEE Transactions on Automatic Control*, 41(3):349–357, 1996.

[219] H. G. Tanner, G. J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–454, June 2004.

[220] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, May 2007.

[221] P. Tatjewski and M. Lawrynczuk. Soft computing in model-based predictive control. *International Journal of Applied Mathematics and Computer Science*, 16(1): 7–26, 2006.

[222] M. Tenny, S. Wright, and J. Rawlings. Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming. *Computational Optimization and Applications*, 28(1):87–121, April 2004.

[223] B. Thuilot, J. Bom, F. Marmoiton, and P. Martinet. Accurate automatic guidance of an urban electric vehicle relying on a kinematic gps sensor. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

[224] J. Velagic, B. Lacevica, and B. Perunicica. A 3-level autonomous mobile robot navigation system designed by using reasoning/search approaches. *Robotics and Autonomous Systems*, 54(12):989–1004, December 2006.

[225] M. Velasco-Villa, B. del Muro-Cuellar, and A. Alvarez-Aguirre. Smith-predictor compensator for a delayed omnidirectional mobile robot. In *Proceedings of the Mediterranean Conference on Control and Automation*, pages 1–6, Athens, Greece, July 2007.

[226] A. N. Venkat, J. B. Rawlings, and S. J. Wright. Stability and optimality of distributed model predictive control. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, pages 6680–6685, Seville, Spain, December 2005.

[227] R. Vidal, O. Shakernia, and S. Sastry. Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 584–589, Taipei, Taiwan, September 2003.

[228] S. G. Vougioukas. Reactive trajectory tracking for mobile robots based on non linear model predictive control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3074–3079, Roma, Italy, April 2007.

[229] E. A. Wan and A. A. Bogdanov. Model predictive neural control with application to a 6 dof helicopter model. In *Proceedings of the American Control Conference*, pages 488–493, Arlington, VA, June 2001.

[230] K. Watanabe. Control of omnidirectional mobile robot. In *Proceedings of the International Conference on Knowledge-based Intelligent Electronic Systems*, pages 51–60, Adelaide, Australia, April 1998.

[231] J. Wen and J. Sooyong. Nonlinear model predictive control based on predicted state error convergence. In *Proceedings of the American Control Conference*, pages 2227–2232, Boston, MA, June 2004.

[232] K. Wesselowski and R. Fierro. A dual-mode model predictive controller for robot formations. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3615– 3620, Maui, Hawaii, December 2003.

[233] R. L. Williams II, B. E. Carter, P. Gallina, and G. Rosati. Dynamic model with slip for wheeled omni-directional robots. *IEEE Transactions on Robotics and Automation*, 18(3):285–293, June 2002.

[234] J. Witt, C. D. III Crane, and D. Armstrong. Autonomous ground vehicle path tracking. *Journal of Robotic Systems*, 21(8):439–449, 2004.

[235] X. Xiang, L. Lapierre, B. Jouvencel, and O. Parodi. Coordinated path following control of multiple wheeled mobile robots through decentralized speed adaptation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4547–4552, St. Louis, USA, October 2009.

[236] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53:65–78, 2004.

[237] X. Yang, K. He, M. Guo, and B. Zhang. An intelligent predictive control approach to path tracking problem of autonomous mobile robot. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 3301–3306, San Diego, CA, October 1998.

[238] B. Young, R. W. Beard, and J. Kelsey. A control scheme for improving multivehicle formation maneuvers. In *Proceedings of the American Control Conference*, pages 704–709, Arlington, VA, June 2001.

[239] M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, August 2007.

[240] M. Zavlanos and G. J. Pappas. Distributed formation control with permutation symmetries. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2894–2899, New Orleans, LA, December 2007.

[241] S. Zelinski, T. J. Koo, and S. Sastry. Hybrid system design for formations of autonomous vehicles. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1–6, Maui, Hawaii, December 2003.

[242] Y. Zhang, S. Velinsky, and X. Feng. On the tracking control of differentially steered wheeled mobile robots. *Journal of Dynamic Systems, Measurement, and Control*, 119(3):455–466, September 1997.