KTH Electrical Engineering

On Communication and Flocking in Multi-Robot Systems

MAGNUS LINDHÉ

Licentiate Thesis
Stockholm, Sweden 2007

KTH School of Electrical Engineering
Automatic Control Lab
SE-100 44 Stockholm
SWEDEN

TRITA-EE 2007:069
ISSN 1653-5146
ISBN 978-91-7178-795-8

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie licentiatexamen i reglerteknik fredagen den 30 november 2007, klockan 10:15 i sal D41, Kungliga Tekniska högskolan, Lindstedtsv 17, Stockholm.

Tryck: Universitetsservice AB

Abstract

Coordination of multi-robot systems to improve communication and achieve flocking is the topic of this thesis. Methods are proposed for mobile autonomous robots to follow trajectories in a way that improves communications with a base station. Further, a decentralized algorithm is presented that yields flocking with obstacle avoidance.

The communication-aware trajectory tracking is adapted to radio communication in indoor environments. Our experimental data show that the effect of multipath fading, well-known in the radio communication literature, causes significant variations in the signal strength between a mobile robot and a base station. A contribution of this thesis is to formulate a tradeoff between tracking a reference trajectory and maintaining communication, first for a stationary reference position and then for a general trajectory. For the general case, the robot and an onboard communication buffer are modelled as a hybrid system, switching between standing still to communicate at positions with good signal strength and driving to catch up with the reference. This problem is solved using relaxed dynamic programming. For the case of a stationary reference position, experimental validation shows that loss of communication is avoided and that the method yields a gain in signal strength.

The algorithm for flocking is based on Voronoi partitions. They can be approximated using only local information and allow the agents to avoid collisions. Our contribution is to add obstacle avoidance and movement towards a goal by using a navigation function—a scalar potential field with exactly one local minimum at the goal. To bound the inter-agent distances and thus avoid flock dispersion, any agent on the boundary of the flock uses a mirroring mechanism to create virtual neighbors that drive it inwards. We can prove collision safety and bounded group dispersion, and simulations show reliable goal convergence even in the presence of non-convex obstacles. A version of the algorithm with lower computational complexity is also presented. It can be used for formation control and it is proven to be locally asymptotically stable for a particular case. A hierarchical control structure is proposed for implementing the flocking on non-holonomic vehicles. It has been tested on a realistic car-like robot model in a flight dynamics simulator and the results confirm that the results on safety, group dispersion and goal convergence apply also in this case.

Acknowledgements

First I would like to thank my supervisor, Karl Henrik Johansson. Not only did he make me understand that I wanted to do a PhD, but since then he has also helped me to cut manageable slices of the enormous cake of exciting research problems out there. I would also like to thank Petter Ögren for his invaluable support when writing the first paper that this thesis is based on, and both him and Xiaoming Hu for volunteering as sounding boards in my reference group.

A special thank you goes to Antonio Bicchi for the invitation to his group in Pisa and to both him and Lucia Pallottino for taking time to provide valuable inputs to my research. From now on, I wouldn't dream of using a knife when eating spaghetti! I would also like to thank Richard Murray for inviting me to his group at Caltech, where I had the chance to get a lot of new impulses.

To all of my colleagues at the Automatic Control Lab at KTH, thank you for impressing me with your diverse knowledge, for helping me when LaTeX doesn't and for making it such a welcoming workplace!

The research reported here was partially supported by the European Commission through the RUNES and HYCON programmes, the Swedish Research Council, the Swedish Foundation for Strategic Research and the Swedish Defence Materiel Administration and VINNOVA through the TAIS project. Their contributions are gratefully acknowledged.

Last, but not least, thanks to Jenny for her patience when I'm overexcited about robotics and for giving me a perspective on what really matters.

Stockholm, October 2007

Magnus

Contents

Abstract **iii**

Acknowledgements **v**

Contents **vi**

1 Introduction **1**
1.1 Emerging Multi-Robot Applications 1
1.2 Problem Formulation 7
1.3 Contributions 8
1.4 Outline 8

2 Background **9**
2.1 Multi-Agent Robotics 9
2.2 Multi-Agent Coordination under Communication Constraints 11
2.3 Flocking and Formation Control 15
2.4 Radio Communication for Robotic Systems 20

3 Communication-Aware Trajectory Tracking **33**
3.1 Problem Formulation 33
3.2 Stationary Reference Position 34
3.3 Time-Varying Reference Trajectory 40
3.4 Experimental Results 53
3.5 Summary 54

4 Flocking and Formation Control Using Voronoi Partitions **57**
4.1 Agent and Communication Model 57
4.2 Distributed Coordination Components 58
4.3 Flocking 62
4.4 Formation Control 72
4.5 Experimental Results from Virtual Testbed 81
4.6 Summary 86

5 Conclusions and Future Work **89**
5.1 Conclusions 89
5.2 Future Work 90

Bibliography **91**

Chapter 1

Introduction

This chapter begins by an introduction to multi-robot systems. We present some motivating applications, chosen to elucidate some of the advantages that can be gained by using cooperating groups rather than one single robot. We then turn to the problem formulation for the research in this thesis. Finally, we state the specific contributions contained herein and give an outline of the thesis.

1.1 Emerging Multi-Robot Applications

As MP3 players, mobile phones and digital cameras become part of our everyday life, their basic components such as microprocessors, memory circuits, radio transceivers and tiny sensors become cheaper, smaller and better. The same components can be used in robots, which in less than a decade has broadened the range of applications for robotics. Capabilities earlier only found in expensive research robots are now possible to replicate in thousands of small units. This means that the four basic capabilities—computation, communication, sensing and locomotion—that define multi-robot systems, are already available. So why don't we see more multi-robot systems being used today? One reason is that much work remains to ensure that the systems are robust enough to be deployed in uncontrolled environments. System integration, in which control is an essential part, must also be improved to allow verification of whole systems. Further, because the robots are still very resource-constrained, distributed methods for control and sensing need to be developed. Some specific challenges ahead are to design suitable navigation algorithms, new wireless communication protocols and decentralized coordination. This thesis aims at providing a contribution to solving some of these problems.

In this section, we will introduce multi-robot systems and point out some advantages they have compared to single robots. This will be done by presenting some emerging applications, chosen to demonstrate these advantages and also to point out the challenges that follow.

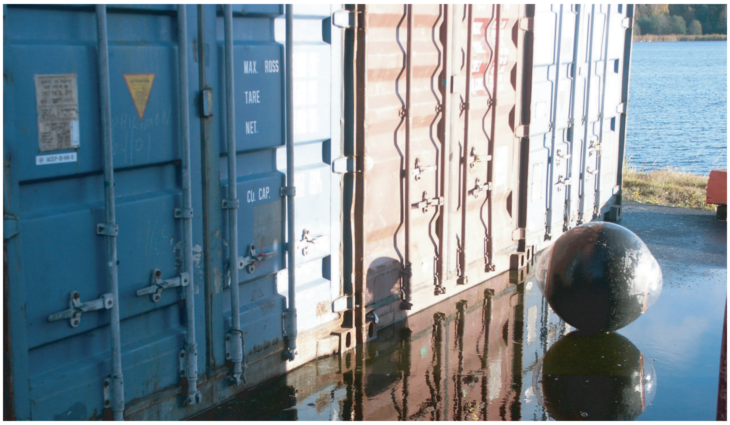

Figure 1.1: A spherical UGV patrolling a container storage. The transparent windows on each side house cameras, providing a $360°$ view around the robot. (Copyright: Rotundus AB)

1.1.1 Autonomous UGV Systems for Reconnaissance and Surveillance

For Swedish military units on international missions or at regiments at home, precious personnel resources are tied up in dull patrolling or stationary surveillance for securing camps, barracks and storage areas. Civilian security firms have the same problem, competing in a market where using the employees efficiently means the difference between success and extinction. By using unmanned ground vehicles (UGVs), human guards can be freed to respond to real alarms and actual incidents. One such UGV, the spherical GroundBot, is depicted in Figure 1.1 when patrolling a container storage.

Recognizing this potential for rationalization, the Swedish Armed Forces have initiated a project, aiming to develop, evaluate and demonstrate methods for increasing the level of autonomy in UGVs used for reconnaissance and surveillance. In military operations in urban terrain, UGVs are already used today, *e.g.*, looking around corners without exposing soldiers or checking buildings for hostile persons or mines, but with no or very little autonomy. Each robot is tele-operated by one soldier, usually by visual feedback from an onboard camera. This has several drawbacks (Lif et al., 2006). One example is that the operator cannot be used for other tasks when operating the robot and often also needs another soldier protecting him. Second, the TV link requires very good communications between robot and operator, which limits its useful range, especially indoors. As a final example, the limited field of view of an ordinary TV camera and the unusual ground-up perspective easily confuses the operator and slows the advancement. To alleviate these problems and also expand the spectrum of tasks where UGVs can be of assistance, the project aims at investigating some basic autonomous capabilities: The robots should be able to autonomously position themselves so that their cameras provide coverage of an area of interest. They should also be able to patrol an area to detect intruders or anomalies such as fire, open doors or broken windows. Further, they should be able to autonomously search an area, leaving no way for an intruder to sneak past the searchers undetected. Finally, if a target is detected, they should track it from a safe distance.

For searching an area, which is one of the capabilities above, it is important that the robots move in a coordinated fashion and have the possibility to send an alarm if an intruder is found. Therefore, as illustrated in Figure 1.2, each robot should maintain contact with the base station while performing the search. This problem of maintaining communication to robots in challenging environments, has been the focus of some of our research and the results are presented in Chapter 3.

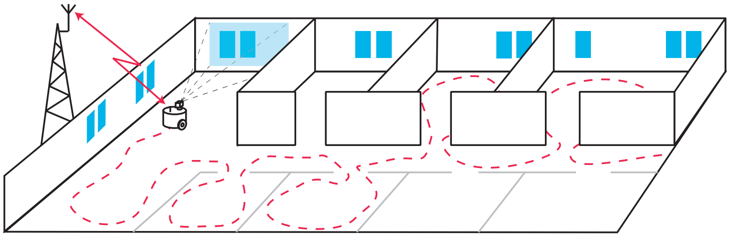

Figure 1.2: An example of a robot searching an area while maintaining communication with a base station.

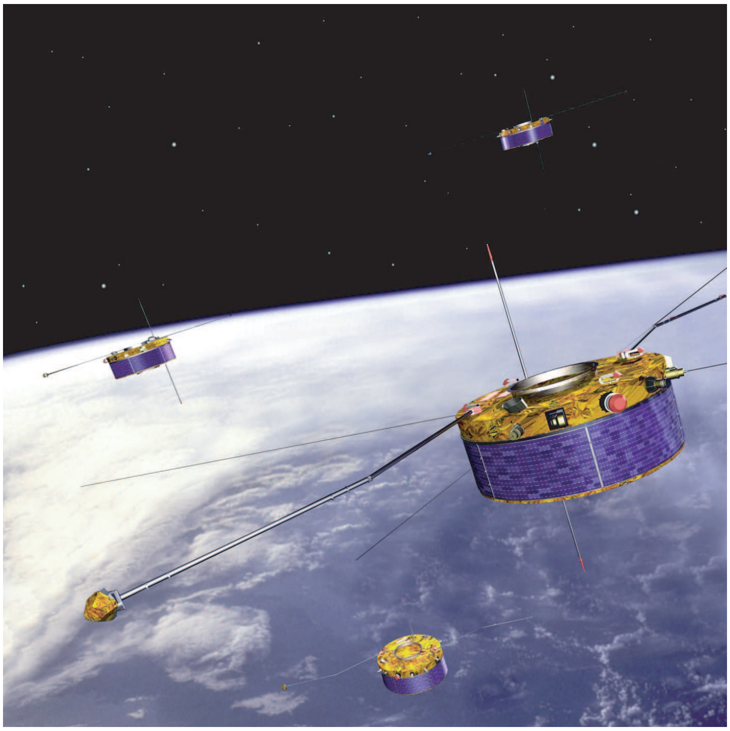

Figure 1.3: An artist's impression of the Cluster-II satellites in orbit around the Earth. (Copyright: European Space Agency)

1.1.2 ESA Cluster Mission

In the summer of 2000, two Russian Soyuz rockets were launched, carrying a total of four Cluster-II satellites, illustrated in Figure 1.3. The purpose of the Cluster mission, organized by the European Space Agency (ESA), was to investigate the small-scale structure of the Earth's plasma environment. The researchers wanted to make detailed maps of the magnetosphere that protects the Earth from the solar wind—mainly electrons and protons ejected from the Sun.

To allow estimation of the three-dimensional gradient of the magnetic field and to distinguish spatial and temporal variations, measurements must be taken at several distant points at the same time. Therefore, the four satellites are used to form a tetrahedron formation, making up a giant sensor. The spatial resolution of the sensor can be adjusted by changing the separation between the spacecraft. The distances were varied between 600 km and 200 000 km, and in June 2007 two of the satellites were brought within 17 km from each other as a test. This may not seem very close, but at the time, they were travelling at 6 km/s. (Cluster Project

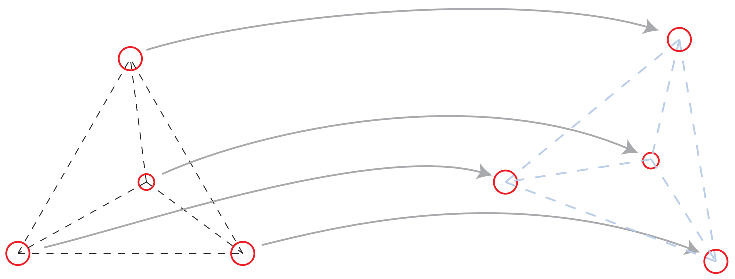

Figure 1.4: A tetrahedral formation of robots, moving while enforcing the dashed inter-vehicle distances.

Team, 2000; European Space Agency, 2007)

Such formation maneuvering can be achieved by designing controllers that run on board each spacecraft, coordinating the movement of each vehicle so that the distances between neighbors converge to a desired value. At the same time, the group as a whole could be translated or rotated as a rigid body. In the spacecraft example this enables the operators to move the whole sensor towards areas of interest, and in the case of ground robots moving in formation, the operator can steer the group clear of obstacles. An example of tetrahedral a formation is illustrated in Figure 1.4, where dashed lines show which inter-vehicle distances are enforced to maintain the formation. Part of our research has been on developing such formation control algorithms, suitable for large groups of vehicles, and the results can be found in Chapter 4.

1.1.3 Reconfigurable Ubiquitous Networked Embedded Systems

In the European Union alone, there exist more than 180 road tunnels that are over 1 km long and 64 more are being planned before 2010. Tunnel fires, such as in the Mont Blanc tunnel 1999, have caused about 200 fatalities in the last decade. One of the problems in the Mont Blanc fire was that the communication networks and sensors failed due to the heat, so rescue workers had very little information on where the fire was and how many people were trapped in the tunnel (Årzén et al., 2007; Hailes, 2007). Motivated by this, the European Union research project RUNES has chosen as its motivating application a network of sensors in a tunnel, designed to support rescue personnel with dynamic information about the accident site. The network can work wirelessly to increase its robustness to damage. Figure 1.5 illustrates a possible disaster scenario where the network delivers information to a rescue coordination center.

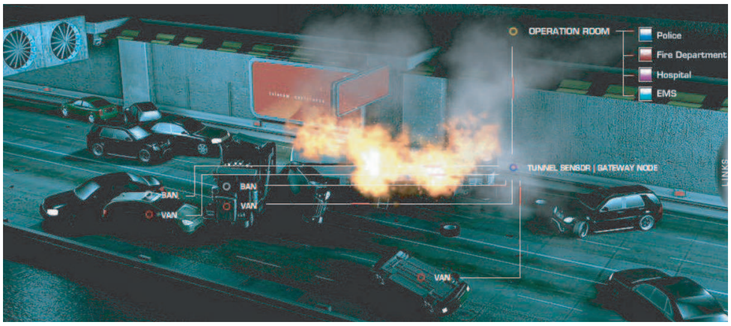

Figure 1.5: A tanker has caught fire in a road tunnel and an embedded sensor network supports the rescue personnel with dynamic information about the accident site. (Copyright: RUNES)

Using wireless communication makes the network independent of cables that may melt, but if some nodes are destroyed, parts of the network may become disconnected. To avoid this, the system can also use robots as mobile nodes to bridge the gap between disconnected subnets, as illustrated in Figure 1.6. Doing this requires dynamic routing protocols that can exploit the added nodes, control laws that guide the robots around obstacles or dangerous areas and communication models that can be used to determine where the robots need to position themselves. In this thesis we present such methods for positioning robots to ensure communication and this is reported in Chapter 3.

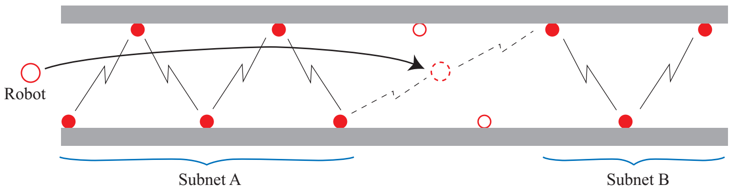

Figure 1.6: A tunnel where two sensor nodes fail, rendering subnets A and B disconnected. A robot is sent in to bridge the gap between the nets, restoring connectivity.

1.2 Problem Formulation

In this thesis, we will use the term *multi-agent system* as an abstraction of a multi-robot system. We define it as a cooperating group of autonomous mobile agents, agent i being at position q_i, capable of locomotion, communication, sensing and computation. These capabilities have constraints: The locomotion is subject to dynamics of the platform, non-holonomic constraints and limits on maximum and minimum velocities. Important constraints for the communication are maximum range, available bandwidth and available energy. Sensing is typically limited by range or field of view of the sensors, available resolution and sensor accuracy. Finally, the computational capacity is constrained by the processor frequency and the available memory. This is schematically illustrated in Figure 1.7.

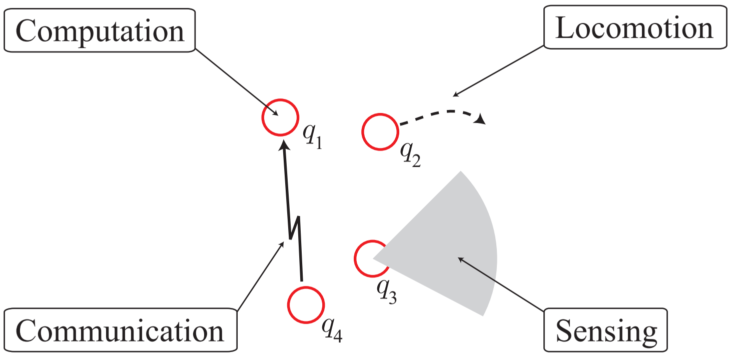

Figure 1.7: The main capabilities of agents in a multi-agent system.

Under these constraints, the task is to design local controllers that achieve some desired functionality of the group as a whole. The problem considered in this thesis is to produce two such functionalities: The first is communication-aware trajectory tracking, *i.e.*, the ability of an agent to follow a predetermined reference trajectory while maintaining communication with another agent or a base station. The second is flocking and formation control, which means that the group should move itself towards a goal with limited dispersion and no collisions.

1.3 Contributions

This thesis is based on the following publications:

- Chapter 3:

  M. Lindhé, K. H. Johansson and A. Bicchi An experimental study of exploiting multipath fading for robot communications. In *Proceedings of the Robotics: Science and Systems conference*, 2007.

  M. Lindhé and K. H. Johansson Communication-aware trajectory tracking, Submitted to *the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

- Chapter 4:

  M. Lindhé, P. Ögren and K. H. Johansson Flocking with obstacle avoidance: A new distributed coordination algorithm based on Voronoi partitions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

  M. Lindhé and K. H. Johansson *Taming heterogeneity and complexity of embedded control*, chapter A Formation Control Algorithm using Voronoi Regions, pages 419–434. ISTE Ltd, 2006.

1.4 Outline

The next chapter provides a background of the scientific challenges in the field in general, and more detailed overviews of the literature in the two areas of the contributions: coordination of multi-agent systems under communication constraints, followed by flocking and formation control. The last section in Chapter 2 provides a basic background on radio communication, especially focused on the issue of multipath fading.

In Chapter 3, we present two algorithms for agent coordination under communication constraints. We also report some initial experimental results that validate our approach. Then, in Chapter 4, we turn to the problem of transportation. Using the geometric construction of Voronoi regions, we present two closely related algorithms that yield flocking with obstacle avoidance and formation control, respectively. The chapter ends with results from a virtual testbed where we have tested the flocking algorithm.

The thesis ends with conclusions and suggestions for future work in Chapter 5.

Chapter 2

Background

This chapter provides a background on the science of multi-agent robotics. We start by presenting some fundamental challenges in the field and then move on to literature overviews on two specific problems considered in this thesis: agent coordination under communication constraints and flocking and formation control. The chapter ends with an introduction to radio communication in general and the effect of multi-path fading in particular, presented from a robotics perspective.

2.1 Multi-Agent Robotics

Multi-agent robotics is strongly inter-disciplinary, encompassing several traditional scientific areas such as control theory, computer science, communication theory, mechanical and electrical engineering and even biology. In this section, we identify some fundamental problems that are central to the field and point out open issues that are subject of ongoing research.

Distributed Coordination

Some objectives of a multi-agent system are global, such as coverage of a region, flocking or rendezvous. Yet, the controllers are run locally on each agent, often with access only to local information. In some cases the agents must also decide on a role, such as leader or follower, in a distributed way. Some methods to solve this are artificial potentials, adapting to the headings of one's nearest neighbors, optimization-based approaches and protocols for role assignment (Murray, 2006).

Communication Limitations

Traditional control theory was developed under the assumption that communication from the plant to the controller and to the actuators is instantaneous and reliable. In the case of multi-agent systems, that often communicate through wireless packet-switched networks, this is no longer the case. Wireless communication has limited range and even between agents that are within range, packets can be lost, delayed or unsynchronized. This can lead to loss of performance, or even instability, for the system as a whole.

To handle these imperfections of the wireless channel, there are intensive efforts to design protocols that reliably transfer data or handle data loss more gracefully. There is also development of methods to estimate the state of the system in ways that are robust to packet losses and to find bounds on the minimum bit-rate needed to stabilize unstable systems over a network (Antsaklis and Baillieul, 2004).

System Integration

For a multi-agent system to work well, the individual agents must perform several functions simultaneously. Examples include maintaining the communication network structure, adapting the radio transmission power, avoiding collisions and localizing themselves. The software to do this should preferably be reusable, both between different applications and different hardware platforms. One way to ensure this is to use a component-based structure, with well-defined interfaces between components so that they can be replaced by new or other components without need to redesign the overall system or verify all components again (Årzén et al., 2007).

Cooperative Sensing

Multi-agent systems can be used for cooperative sensing, where the accuracy can be improved by aggregating sensor data from all agents. Reaching a common state estimate by communicating with selected neighbors is referred to as *consensus*, and is an active area of research. Besides sensor fusion, the same mechanisms can also be used for rendezvous in a decentralized way or to achieve formations. An important tool for treating such problems is to describe the communication network as a graph and then use algebraic graph theory to study convergence towards consensus (Olfati-Saber et al., 2007).

Security

As mentioned above, multi-agent robot systems can often act as wireless sensor networks, which gives rise to some special security challenges. First, it is important to ensure *authentication* and *privacy*. This can be done through encryption, but it requires new encryption hardware that can support resource constrained processors, and new key distribution schemes are needed that work in decentralized settings. Second, as opposed to traditional networks, sensor nodes may have to be placed where they are accessible to potential attackers, which makes it difficult to physically protect them against *node capture* or *tampering*. Instead, algorithms must be developed that are robust to this and provide ways to detect malicious (or malfunctioning) nodes (Perrig et al., 2004).

We now present some previous work on the more specific problems of coordination under communication constraints, in Section 2.2, and flocking and formation control, in Section 2.3.

2.2 Multi-Agent Coordination under Communication Constraints

To exploit the full advantage of multi-agent systems, the agents often need to communicate to synchronize their movement and sensor usage. Since communications have limited range, an active strategy for not losing contact is needed. Performing tasks in a communication-aware manner becomes a crucial ability for each agent in the group. Since the reasons for interruptions of communication can vary, as well as the requirements on what bandwidth is needed and what delays are tolerated, we have chosen a wide definition of this problem:

> *Find control laws for agents in a multi-agent system, so that the system can perform a task under the constraint of maintaining inter-agent communication.*

By stating that the *system* should perform a task, we allow for some agents possibly being fully devoted to fulfilling the communication constraint. We also do not detail the constraint, since it can range from requiring high-bandwidth communications between all agents at all times to requiring sporadic contacts so that messages can be relayed between any two agents in the group in bounded time. Finally, we restrict the scope to control laws for the *movement* of agents. For given positions of agents, the questions of routing, media access control as well as other issues on higher protocol layers are relevant, but not treated here. The interaction between mobility and these higher layers is, however, an interesting direction of future research, as discussed in Chapter 5.

Most of the papers referenced below treat the subject of radio communication, since it is the dominant technology for inter-robot communication. But some also consider communication through infrared light (Kelly and Keating, 1996), sound (Karimian et al., 2006) or by extracting information from a camera image (Moshtagh et al., 2006). Although in practice they are different, the demands for maintaining contact are mainly the same. The signal quality decays with distance, and it is also attenuated by obstacles. For this reason, many of the strategies carry over to other domains than presented in the original papers.

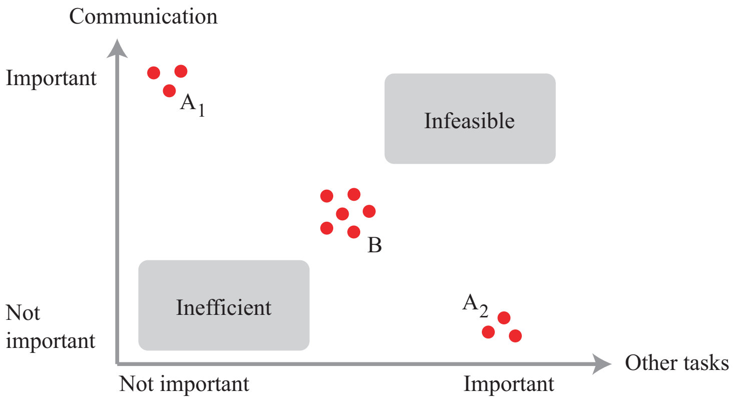

Figure 2.1: Typical role assignments for maintaining inter-agent communication. Either some agents (A_1) ensure communication and others (A_2) focus on other tasks needed for the mission, or all agents (B) make tradeoffs between communication and other tasks.

Existing methods of ensuring inter-agent communication while carrying out a mission can be categorized depending on how they assign roles to the different agents. As shown in Figure 2.1, there are two main approaches:

- One or more agents are dedicated relaying agents, which means that they only focus on communication. The rest of the agents perform other tasks needed to solve the mission, with little or no adaptation to the communication constraint. (A_1 and A_2 in Figure 2.1.)
- All agents make tradeoffs between communication and other tasks needed to solve the mission. (B in Figure 2.1.)

Approaches that fall under each of these categories will now be described in more detail. We will then end with the somewhat separate issue of disruption-tolerant networks.

Either communicating or performing other tasks

Nguyen et al. (2004) suggest a system of small and cheap relaying robots. The intended application is indoor exploration, and the relay robots follow the lead robot into a building until they detect that the signal strength from the base station falls below a given threshold. A relay robot is then left behind, and the link is routed over that robot. If a relay discovers that it is no longer needed (*e.g.*, because another relay further ahead achieves a direct link to the base), it will use a map sent by the lead robot to catch up with the group, making it available for redeployment. With the help of this internal map, the relays can also be recalled when the mission is over. Similarly, there is an example of small Scout robots (Drenner et al., 2002) using stationary relay stations to enhance the communication range and to allow sending video imagery back to a base station.

A somewhat different approach is presented by Årzén et al. (2007), using mobile relaying agents that can be sent into a tunnel to reconnect a multi-hop network of stationary sensors, if some sensors are damaged. (Also described in Section 1.1.3.) There is an elaborate power-control protocol to adapt the transmission power of the relay, so as to avoid bandwidth contention and save battery power.

Sometimes, the lead agent has both communication and other objectives to take into account, while other agents are dedicated to only communication. An example of this is a system (Sweeney et al., 2004), where a chain of relay agents has the task of relaying data from a leader to a fix base station. The lead agent moves towards a goal, under the constraint that it must not lose contact with the adjacent relay agent. Likewise, the relay agents try to maximize the signal strength to the agent ahead of them, under the constraint that they must not lose contact with the agent behind.

Tradeoff between communication and other tasks

An example of all agents helping to solve the mission is given by Arkin and Balch (2002). A group of agents starts from the entrance of an unknown building, and one of the agents starts exploring more or less randomly. When it loses the line of sight with the rest, it retraces its trajectory until the contact is reestablished. It then stops and becomes the *anchor* for the next robot, that can get further into the building before losing contact. The last robot of the group does not stop like this, but retraces in case of loss of contact, and tries another direction of movement.

In the above case, the agents at some point switch from sensing to relaying. Another way is to let the agents continuously solve, *e.g.*, a sensing task while making a tradeoff between sensing performance and communication. This can be formulated as gradient climbing in one or more scalar utility functions, describing sensing and communication performance, respectively (Chung et al., 2006; Popa and Helm, 2004).

Another mission, as opposed to the sensing described above, could be to move the group. Esposito and Dunbar (2006) have studied a method to move a formation of agents from one configuration to another (which could also mean the same formation but in another place). This is done without losing line of sight between specified agents, in an environment with obstacles. A more experimental approach to the same problem formulation is presented by Powers and Balch (2004), using ground robots with a behavioristic control law. A similar problem is solved by Pereira et al. (2003), by reducing it to formation control.

All papers presented so far use simplified models of radio propagation, assuming that the signal strength is a function of the distance and/or depends on a free line of sight. For radio signals, this overlooks the significant effect of multipath fading in urban or indoor environments. As shown in Chapter 3, established models for fading

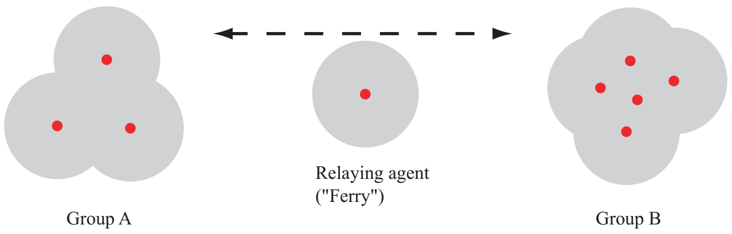

Figure 2.2: A schematic illustration of a disruption-tolerant network, where a relaying agent stores data and physically transports it between two connected sub-networks. The grey regions indicate the communication range of each agent.

from the literature of radio propagation can be used to predict how small corrections (in the order of centimeters) to the position of an agent can yield significant improvements in signal strength (Lindhé et al., 2007). When tracking a reference trajectory, multipath fading can also be exploited to improve communications with a base station by making a tradeoff between stopping to communicate from good positions or keeping up with the reference (Lindhé and Johansson, 2008).

Another way to take communication into account is to program all agents to perform the mission as usual, but impose a hard constraint that they must not continue if they lose communications. Hsieh et al. (2006) describe experiments with surveillance robots moving towards separate goals, while monitoring the achieved data rate to the base station. If the data rate falls below a given threshold, they stop and thus risk not to complete the mission. After a certain waiting time, if the data rate has not recovered, they retrace to improve communications. The waiting adds some robustness to short temporal dips in the signal strength.

Disruption-tolerant networks

A somewhat separate category of methods can be used if connectivity is not measured at each instant, but rather over a time interval. The constraint is that during some interval, there must have existed connections that together allowed a message to be relayed between any two agents. This is called *disruption-tolerant networks* (Burns et al., 2006), and is illustrated in Figure 2.2. Mobile agents with onboard memory could act as *ferries*, relaying messages by storing them and sending them at a later time, when they have contact with the intended receiver or another relaying agent. For applications where delays in the time-scale of agent movements are acceptable, this can be shown to overcome fundamental limitations on the possible throughput (Grossglauser and Tse, 2002).

Zhao et al. (2005) suggest four alternative ways of computing trajectories for dedicated ferries. A practical example of such an application is an Australian group (Dunbabin et al., 2006), using unmanned underwater vehicles as data mules, collecting data from stationary underwater sensors. An example of agents that do not adapt their trajectories is a network of public transportation buses, also carrying computers with wireless connections. By storing data on board, they can connect two disjoint wireless networks in different parts of a university campus (Burns et al., 2006).

2.2.1 Summary

Using relaying agents is a proven method that appears to be robust, at the expense of taking agents off the solving of the task. The relays could be positioned in a static position or constantly moving to maximize the end-to-end bandwidth. Agents can also switch from the role of "active" agents to relaying if the need arises. We have found no methods that explicitly minimize the number of relaying agents.

A more complex method is to let the agents make a tradeoff between communication and task completion, but this usually decreases the performance in solving the task. For tasks such as searching or of "lawn-mowing" type, this may not be acceptable. Instead, a system designer could trade agent mobility for longer end-to-end delays, if the network is disruption-tolerant. This means that messages are relayed when agents are close enough to communicate, and in some cases there are dedicated agents (called "ferries") that just have the task of physically transporting messages.

2.3 Flocking and Formation Control

In large multi-agent systems, it is unpractical for a user to control each agent separately. In many applications it is therefore of interest to abstract the control so that the group *as a whole* is controlled, and each member follows automatically. This can for example be used to transport a group of UGVs from one place to another, either in a completely autonomous fashion or under the supervision of an operator. For the purpose of this overview, we define this problem broadly as:

> *Find a control law for each agent in a group, such that the relative positions of the agents converge to some desired set, collisions are avoided and the group as a whole performs a desired motion.*

The desired result can be defined as just all agents moving in the same direction and thus maintaining an upper bound on the inter-agent distances, which is called *flocking*. Inspired by biology, a group performing flocking is sometimes called a *swarm*. On the other hand, if there are stricter requirements for the relative positions, such as prescribed inter-agent distances or wanting the agents to form a geometric shape, this is called a *formation*.

Collision avoidance can be defined as no agents colliding, or also avoiding collisions with obstacles. The latter is usually called *obstacle avoidance*. Finally, the group movement can be encoded as a simple desired velocity of each agent or a

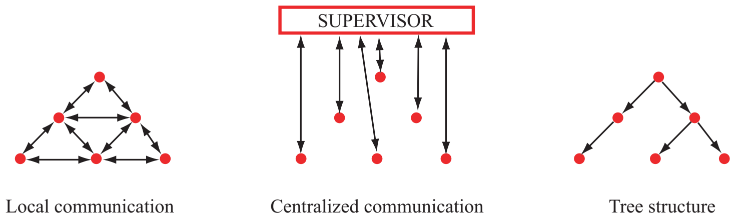

Figure 2.3: Three coarse categories of communication topologies for formation control. The arrows indicate information flow.

more elaborate path-following or even autonomous navigation for the group as a whole. We will now give an overview of some of the work that exists in this field.

To synchronize their movement, agents need to exchange information. An important difference between algorithms is the communication topology, *i.e.*, which agent communicates with which, and also in what direction information is sent. We will roughly divide the algorithms we present into three categories, depending on the communication topology they employ. As illustrated in Figure 2.3, the first is *local communication*, where all agents communicate only with their neighbors. The set of neighbors can be fixed or vary over time, *e.g.*, depending on physical proximity. Next, we will consider the case of *centralized communication*, where all agents communicate directly with a supervisor. Finally, we separately consider the more special case where the communication topology has a *tree* structure and communication only happens in the direction from the root. The root of the tree can be a physical agent or a virtual leader.

The choice of communication topology also affects the scalability of the formation. Algorithms that only require communication with neighbors scale well, since the demands on each agent are constant even if the group size increases. Further, Fax and Murray (2004) have shown how communication topology affects the stability margins for convergence towards the prescribed formation.

Local communication

One of the first papers on flocking came from the field of computer graphics and was written by Reynolds (1987). He introduced “boids”, a type of agents that could perform animal-like flocking and are still used in computer games and animated movies. The boids follow three rules:

- *Collision avoidance:* Avoid obstacles and other boids
- *Velocity matching:* Adapt the velocity to that of nearby flockmates
- *Flock centering:* Attempt to stay close to nearby flockmates

The control output from each rule was combined based on priority, and used to control the acceleration of the boid.

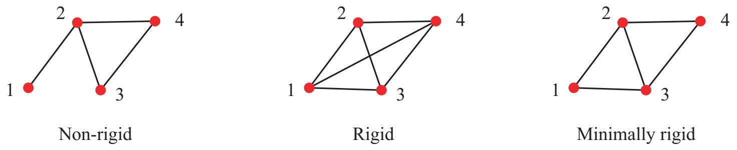

Figure 2.4: A formation with prescribed inter-agent distances indicated by lines. Depending on what distances are prescribed, the formation has different rigidity properties.

More recently, Tanner et al. (2007) have formulated the same rules and shown formally that they lead to velocity alignment and stable inter-agent distances. They describe flocking for kinematic single integrator agents, whose control law consists of two terms. One aims to align the velocities of all agents, and the other is the sum of the negative gradients of an artificial inter-agent potential function. The potential function is a function of only the distance between two agents, and it has a unique minimum at a desired inter-agent spacing. Two kinds of communication topologies are considered: First the communication graph is fixed, and then it is time-varying, by letting agents communicate with all neighbors within a given sensing radius. In the fixed case, the group will converge to an equilibrium where the sum of the potentials has a local minimum. A condition for this is that the communication graph is connected, *i.e.*, there is a path between any two agents. If the communication graph is a tree graph, this local minimum will also be the global one, with the desired distance between all connected agents. In the time-switching case, convergence to an equilibrium can also be shown, if the communication graph remains connected at all times.

Jadbabaie et al. (2002) have shown that a sufficient condition for flocking is that there exists a finite time T such that the flock is connected at least once during every time interval of length T.

An interesting issue in the context of potential methods, is what neighbor distances need to be prescribed for the formation to be well-defined. A formation is *rigid* if maintaining the inter-agent distances specified by the communication graph is a sufficient condition for *all* inter-agent distances to be constant (Roth, 1981). If the graph contains the minimal necessary number of edges to remain rigid, the formation is said to be *minimally rigid*. This is illustrated in Figure 2.4, where the leftmost formation is not rigid, since the distance between agents 1 and 3 can vary.

Another class of methods is based on *behaviors*. A behavior is a controller that, based on sensor information, outputs a recommended control. Each behavior typically performs a simple function, and many behaviors may run in parallel. The agent then combines the outputs from each behavior into an aggregated control signal. This arbitration can be done through a weighted vector sum, by voting or by subsumption. When using vector summation, each behavior is weighted according to its importance, and safety behaviors such as avoiding obstacles typically get a high weight. In voting, all behaviors get a number of votes corresponding to their weight, and cast them on one or several discrete candidate controls. The candidate command with the highest number of votes is then executed. In a subsumption architecture, the control signal of the highest prioritized active behavior is used (Arkin, 1998).

An application of behaviors to formation control is described by Balch and Arkin (1998). They suggest the following basic behaviors:

avoid-static-obstacle	Move away from obstacles
avoid-robot	Move away from other agents
move-to-goal	Move towards a waypoint for the group
noise	Move randomly, to dissolve deadlocks
maintain-formation	Approach the prescribed position in the formation

The three first behaviors are fairly intuitive, but the fourth is included to avoid getting stuck at saddle points of the vector fields surrounding obstacles. The fifth behavior steers the agent towards its position in the formation. This position can be relative to the leader, to the center of mass of the group or to a single designated neighbor. These alternatives also imply different communication topologies. The authors then propose arbitration by vector summation.

Raffard et al. (2004) have presented methods from convex optimization, deriving a method for controlling formations of aircraft by using dual decoupling. The formation problem is expressed as a convex optimization problem, where each aircraft has both individual objectives and formation objectives. Further, the dynamics of the aircraft are formulated as a set of constraints. By introducing slack variables and solving the dual problem, the solution can be found in a decentralized iterative manner, with only local communication.

As a final example, there are also geometric methods that have been applied to the problem of positioning mobile sensors to maximize sensor coverage. Cortés et al. (2004) formulate the problem of dispersing sensors in a convex polygonal environment so that the probability of sensing an event is maximized. This is solved iteratively, using the concept of Voronoi regions. As shown by Lindhé et al. (2005), the same concept can also be integrated with navigation functions to achieve simultaneous flocking and obstacle avoidance. A similar algorithm can also be used for formation control (Lindhé and Johansson, 2006). For more details on Voronoi regions and our contribution, see Chapter 4.

Centralized Communication

Ren and Beard (2004) have presented a centralized method for formation control, mainly intended for spacecraft. The method uses a *virtual structure*, which is an imaginary frame, to which each agent is attached. To move the formation, the position and orientation of the virtual structure is sent out from a central supervisor to every agent, that moves to maintain its relation to the structure. Simultaneously, the framework allows for feedback from each agent if it deviates too far from the structure. This can happen if the actuators of the agent saturate, or if there is some unforeseen problem. The structure then slows down, to minimize the formation error.

An example of a combination between centralized and local communication is given by Leonard and Fiorelli (2001). They suggest an algorithm based on artificial potentials, where each agent reacts to the positions of its neighbors. To be able to control the formations of the group, and to herd the agents in a desired direction, they then introduce *virtual leaders*. They are moving reference points, to which the agents react as if they were physical agents. Knowing the position of the leaders requires centralized communication with a supervisor.

Tree Communication

Methods based on tree-like communication topologies and with one-way communication, are often called leader following. With this communication topology, some agents, called *leaders*, are not affected by the movement of the others, so they are free to follow trajectories defined by an overall task. A *follower* is an agent that tries to maintain a given relative position to a leader or another follower. The leader can also be a virtual point, so that all physical vehicles are followers (Egerstedt and Hu, 2001).

A subclass of leader following systems is automated highway systems, where cars form *platoons* of vehicles that follow each other autonomously. Here the communication tree has no branches, but forms a single line. Stability of the velocities and inter-vehicle distances in such a topology is called *string stability* and has been thoroughly studied (Swaroop and Hedrick, 1996).

Connected to this is the more general *leader-to-formation stability* (LFS), proposed by Tanner et al. (2004). They consider a slightly broader class of systems than described above, where each follower may follow several leaders simultaneously. Roughly speaking, LFS is defined as the existence of an upper bound on the error in follower positions as a function of the perturbations to the leader positions. Tanner *et al.* show that LFS is invariant to general kinds of interconnections of groups and how LFS can be proven for a formation controller. They also show that LFS can be used as a design tool to compare different formation topologies and choose the one where errors are attenuated the most.

2.3.1 Summary

Using decentralized formation controllers gives advantages such as scalability of the group, lower requirements on communication range and no need for a central supervisor. The drawback is that it can be very difficult to encode a given desired formation as local controllers. A centralized topology, on the other hand, gives more freedom to explicitly prescribe the desired formation. The class of leader-follower methods is interesting because of their simple communication structure, but they are less robust to disturbances and agent failures.

Two other design considerations also deserve mentioning before we look at specific algorithms. The first is heterogeneity. Does one agent have to be designated as leader and, in the case of formation control, do the agents have rigidly prescribed roles in the formation or can they choose any position? This affects the robustness of the group to failures of single agents, and whether several groups can easily merge. The other consideration is what kind of agent dynamics are allowed. The models can range from single or double integrators, via unicycles to car-like platforms or even more complex dynamics, especially in the case of aircraft.

2.4 Radio Communication for Robotic Systems

In this section we review some basic results on radio communication, with a focus on issues that are important when communicating with mobile robots. This is intended as an introduction for readers who have no background in communication, and also to establish some notation for our contribution in Chapter 3. Unless stated separately, results in this section are derived from the book by Stüber (1996).

2.4.1 The Wireless Channel

We start by a simple model of a transmitter, the wireless channel and a receiver, as illustrated in Figure 2.5. The signal from the transmitter (Tx) is fed to an antenna, transmitted through the wireless channel and received together with interference and noise in the receiver antenna. The interference comes from other transmitters, such as radios or other electrical equipment that emits radio frequency energy. Noise is added in all stages of the transfer, from the atmosphere as well as from thermal and other effects inside the receiver, but we schematically represent it as being added before the antenna. Finally, the signal is band pass filtered to select only the frequencies of interest, before it reaches the receiver (Rx).

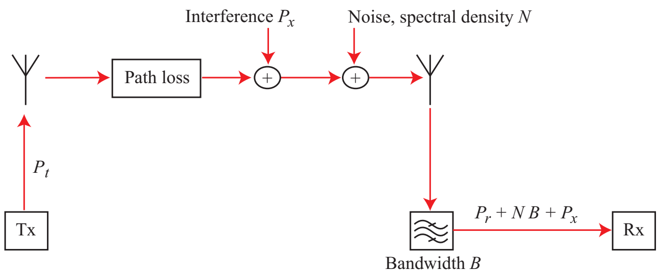

Figure 2.5: A simple model of a link, with a transmitter (Tx), the wireless channel with additive noise and interference, band pass filtering and finally the receiver (Rx).

Path Loss

Given a transmission power of P_t, and a distance d m to the receiver, the *nominal* power P_r of the useful signal in the receiver is

$$P_r(d) = P_t - PL_0 - 10\ n\ \log_{10} d\ [\text{dB}].$$

The signal is attenuated by distance, which is called *path loss*. The term PL_0 is the path loss at 1 m from the antenna and n is the path loss exponent. The path loss exponent n is 2 for free-space propagation and can reach about 4 in some environments. At 2.4 GHz and in office environments, values around 2.5 have been reported (Darbari et al., 2005).

Noise and Interference

The power of all interference in the pass band of the filter is denoted P_x and the bandwidth of the filter is B. For constant noise spectral density N, the input noise power for the receiver is NB. This is why, to reduce the noise in the receiver, the signal is band pass filtered to only contain the frequencies where the useful signal resides. The choice of filter bandwidth is an important tradeoff between noise immunity and communication data rate. A system with high data rate will generally have a higher signal bandwidth than one with low data rate. Therefore, as illustrated in Figure 2.6, high data rate requires a wide pass band in the filter, which also lets more noise into the receiver. This is why, at a constant noise spectral density, a radio link with low data rate is more immune to the noise. This is also why, below, we must fix a nominal data rate for a link to express how its capacity changes with changing channel conditions.

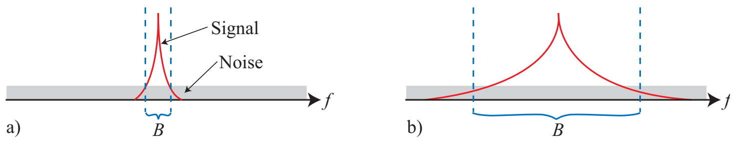

Figure 2.6: A signal with low data rate is narrow in the frequency domain (a). This allows a smaller bandwidth B in the band pass filter, which lets less noise through than for a high data rate signal (b).

Detecting the Useful Signal - Introducing S(I)NR

The receiver now has the job of discriminating the useful signal from the total received signal, and its performance is usually measured as the *bit error rate* (BER), which is a probability of the detector mistaking a 0 for a 1 or vice versa. For a given modulation method, the BER is usually stated as a function of the *signal-to-interference-and-noise-ratio* (SINR):

$$\text{SINR} = \frac{P_r}{NB + P_x}$$

In the sequel, we will neglect the interference (*i.e.*, assume $P_x = 0$) or consider it as noise. Then the SINR is called *signal-to-noise ratio* (SNR) instead.

To give an illustrative example of the impact of the SNR on the BER and eventually on the effective data rate of a link, we quote Zuniga and Krishnamachari (2004). They computed the BER for MICA2 wireless sensor motes, operating at a nominal data rate of 19.2 kbit/s and with a noise bandwidth of 30 kHz, as

$$\text{BER} = \frac{1}{2} e^{-\frac{\text{SNR}}{1.28}}.$$

Assuming that a packet consists of $f = 50$ bytes, and that a packet is discarded if not all bytes are correctly decoded, the packet reception ratio (PRR) can also be computed:

$$\text{PRR} = \left(1 - \frac{1}{2} e^{-\frac{\text{SNR}}{1.28}}\right)^{8f}$$

This result, along with the resulting data rate, is plotted in Figure 2.7. It also illustrates that the importance of gaining signal strength very much depends on the operating point. For a receiver operating at an SNR of 7 dB (corresponding to 350 bit/s), gaining 4 dB in signal strength would mean increasing the effective data rate by a factor of 54. If the SNR would have been 11 dB, the same signal gain would only have given a 1% improvement in data rate.

2.4.2 Shadowing and Multipath Fading

In an urban or indoor environment, the received signal strength will exhibit large fluctuations around the nominal level due to shadowing and multipath fading. Shadowing is caused by objects obstructing the signal path and varies over distances the same order as the obstructions. Multipath fading, on the other hand, is caused by destructive or constructive interference between the signal and its reflections and it varies over very short distances, in the order of a wavelength. Even a stationary receiver will experience varying multipath fading if the environment is changing, for example due to cars and people moving or doors opening and closing. If all signal components that reach the receiver are of equal strength, the multipath fading is called Rayleigh fading, while if there is a line-of-sight component that is significantly stronger, we have Ricean fading.

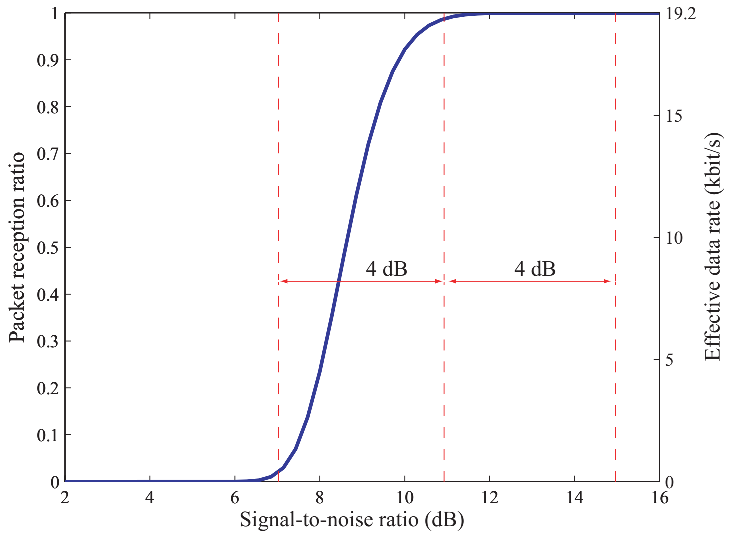

Figure 2.7: The packet reception ratio as a function of the signal-to-noise ratio for a MICA2 wireless sensor node. The scale on the right shows what the corresponding effective data rate would be.

It should be pointed out that *for given antenna positions*, the fading is reciprocal and thus affects the signal path equally in both directions. But its spatial properties are in general not the same at the receiver and the transmitter. Specifically, if a base station with very open surroundings is communicating with a robot in a cluttered environment, the multipath fading varies over much larger movements of the base station than of the robot.

Due to the difficulty of predicting the Rayleigh fading, it is usually modelled as a stochastic effect. The probability density function (pdf) of the received power in a Rayleigh fading environment is

$$f_P(x) = \frac{1}{P_r} \exp\left(\frac{-x}{P_r}\right). \tag{2.1}$$

The expected value is P_r, the nominal received power. (The reader may ask why this is not the Rayleigh distribution, but the signal *amplitude* is Rayleigh distributed, while the power is the square of the amplitude and follows an exponential distribution.)

For comparison with measurements later, we will also compute the pdf of the received power when expressed in dB. We do this via the cumulative distribution function (cdf), which describes the probability that the received power is less than x:

$$\text{Prob}(P < x) = \int_0^x f_P(\tau)\, d\tau = \int_0^x \frac{1}{P_r} \exp\left(\frac{-\tau}{P_r}\right) d\tau = 1 - e^{-x/P_r}$$

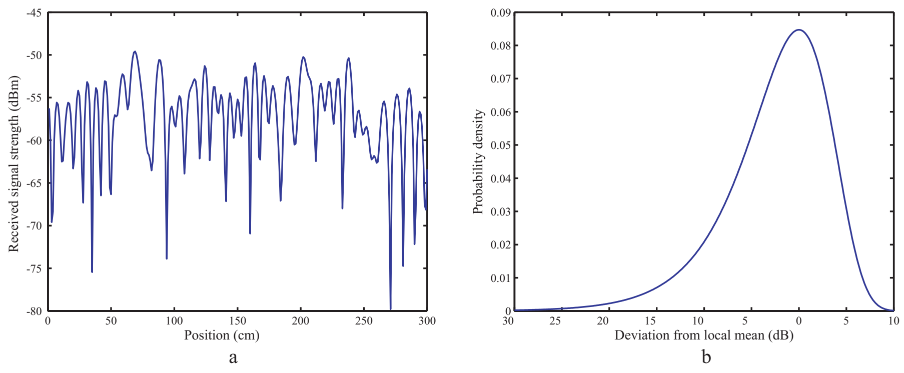

Figure 2.8: a: Simulated Rayleigh fading at 2.4 GHz as the receiver moves along a 3 m line, taking samples of the received signal strength at every cm. b: The probability density function of the fluctuations of the received power, measured in dB, in a Rayleigh fading environment.

A signal power of x corresponds to y dB, where $x = 10^{y/10}$, so the probability of the power being less than y dB is

$$\text{Prob}(P < 10^{y/10}) = \text{Prob}(P < e^{y\frac{\ln 10}{10}}) = 1 - \exp\left(-\frac{1}{P_r}\, e^{y\frac{\ln 10}{10}}\right).$$

By differentiating this cdf, we get the pdf for the power in dB:

$$\frac{d}{dx}\left(1 - \exp\left(-\frac{1}{P_r}e^{y\frac{\ln 10}{10}}\right)\right) = \frac{\ln 10}{10P_r}\exp\left(y\frac{\ln 10}{10} - \frac{1}{P_r}\, e^{y\frac{\ln 10}{10}}\right) \qquad (2.2)$$

This is illustrated in Figure 2.8, together with a plot of a simulated signal strength for a receiver moving 3 m along a straight line in an environment with static Rayleigh fading.

Finally, we will also need the spatial autocorrelation of the fluctuations due to multipath fading. The autocorrelation as a function of the distance δ between two samples is

$$R(\delta) = kJ_0^2(2\pi\delta/\lambda_c),$$

where k is a constant, J_0 is the zero-order Bessel function of the first kind and λ_c is the carrier wavelength. This is illustrated in Figure 2.9. It shows that two samples taken $0.38\lambda_c$ apart (4.75 cm at 2.4 GHz) should have zero correlation, and samples taken at a greater distance should always have small correlation. We call this distance the *decorrelation distance* Δ. In practice, samples taken more than about half a wavelength apart are considered to have independent Rayleigh fading. This is also why fading can be alleviated by antenna diversity, using several antennas with a separation of at least Δ. This increases the chance of at least one antenna having good reception.

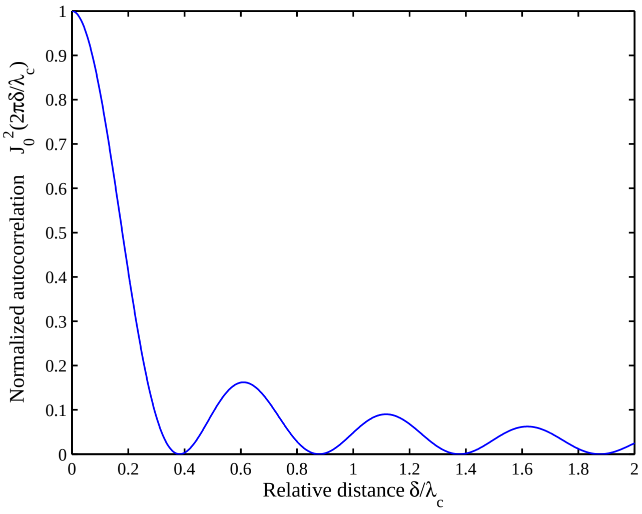

Figure 2.9: Normalized spatial autocorrelation of the received signal strength in a Rayleigh fading environment.

2.4.3 Experiments

We have performed experiments to validate the models of the fading, both for the probability density function and the autocorrelation. For this, we have chosen a basement corridor and a cluttered lab room as representative environments.

To automate the measurements, we have mounted a radio based on the CC2420 chip on a robot. It communicates with the same chip on a TMote Sky wireless sensor node, connected to a PC, see Figure 2.10. The CC2420 operates at 2.4 GHz with a maximal output power of 0 dBm and has a software-accessible received signal strength indicator (RSSI) (Chipcon AS, 2006). It is worth noting that the CC2420 uses direct sequence spread spectrum modulation. This is supposed to alleviate the effects of multipath fading, but as shown below and by Puccinelli and Haenggi (2006), the problem of points with strong negative interference, so called *deep fades*, remains.

The TMote has an antenna integrated on its circuit board, while the robot has a quarter-wave antenna on top. The integrated antenna is by design not omnidirectional, and measurements show that the antenna mounted on the robot also experiences some directional dependence, due to the influence from the rest of the robot. This means that for the robot to get the same measurement value when returning to a point, it must also return to the same orientation.

To estimate the spatial correlation in the different environments, we have driven the robot along straight 200 cm lines, stopping and sampling the RSSI each centime-

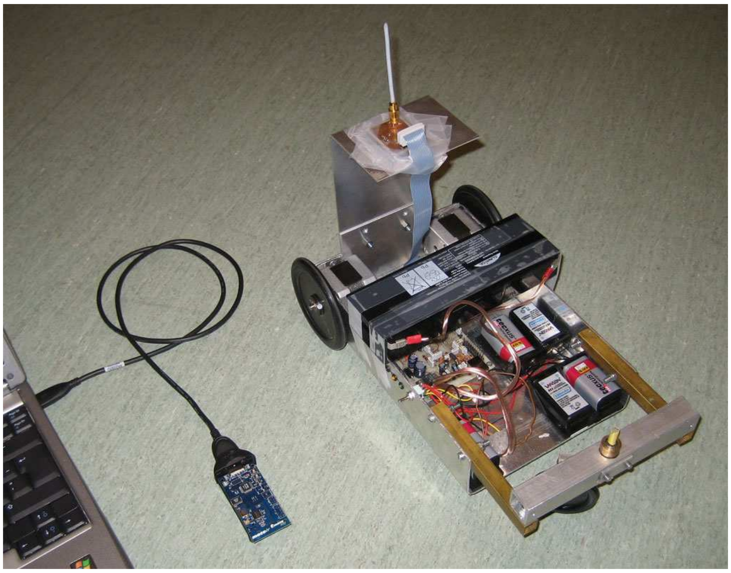

Figure 2.10: The measurement system, with the robot and the TMote connected to a PC. The robot has two driving wheels and a third caster wheel, and its antenna is positioned about 25 cm above ground.

ter. Each sample is formed by averaging the RSSI readings from four radio packets. For each sequence of $N = 200$ samples, we computed the unbiased estimate of the autocorrelation

$$\hat{R}(k) = \frac{1}{N - |k|} \sum_{m=0}^{N-k-1} [z(m) - \bar{z}][z(m+k) - \bar{z}],$$

where $z(m)$ is the signal envelope (in dBm) in sample m and $\bar{z}$ is the mean value.

The histogram of the signal strength samples was estimated using the same measurement setup, but driving a distance greater than Δ between samples. We then assumed that the samples could be regarded as independent. Since the nominal signal strength (*i.e.*, taking only path loss and shadowing into account) is difficult to calculate, we estimate it by the local average.

The lab room contains lots of computers, metal cabinets and other effective scatterers, so it is our belief that this environment approximately gives Rayleigh fading. This is also confirmed by the measurements. One representative measurement series is depicted in Figure 2.11, and the estimated autocorrelations for five measurements are superimposed in Figure 2.12. The autocorrelation decays rapidly, and reaches the noise floor at $\Delta = 6$ cm in all measurements in this environment. (This matches the predicted $\lambda_c/2$ =6.25 cm.)

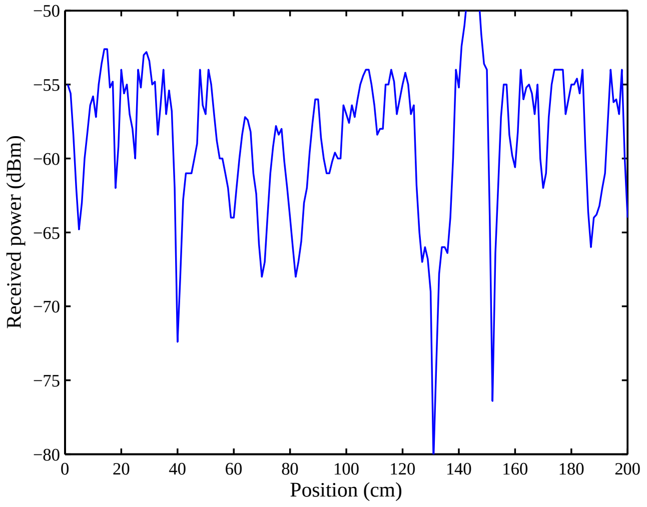

Figure 2.11: Measurement results from the lab room, where the signal strength varies over 30 dB. Note the deep fade at 130 cm, where the connection was temporarily lost.

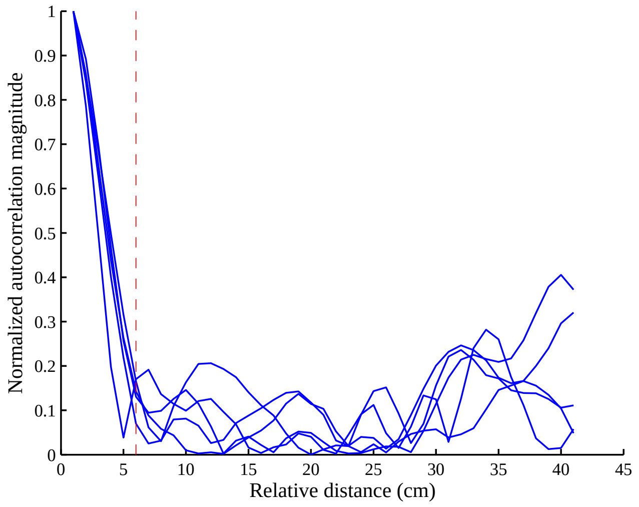

Figure 2.12: Autocorrelation estimates for five measurement series in the lab room. The dashed line is the estimated decorrelation distance Δ. The autocorrelation drops rapidly and the spurious values at 40 cm are probably due to the estimate being noise sensitive at high lags.

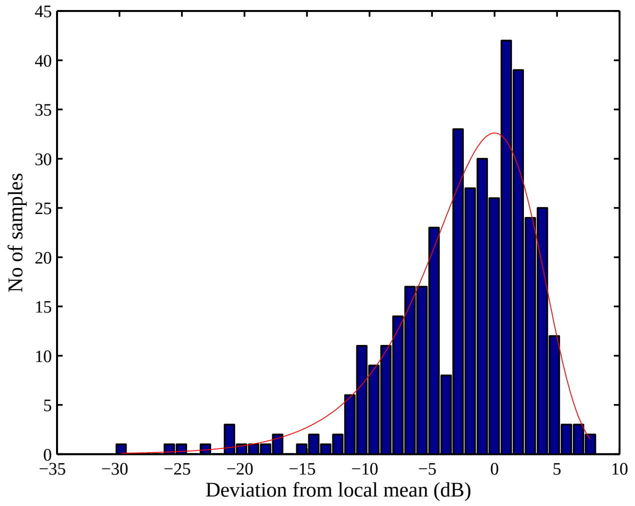

Figure 2.13: Histogram of 400 signal strength samples, taken in the lab room with sample spacing 10 cm. The distribution closely resembles the pdf of perfect Rayleigh fading, included as a reference.

As an estimate of the pdf, the measured signal strength is then plotted as a histogram in Figure 2.13, where the pdf of the Rayleigh fading (2.2) is included as a reference. The pdf is scaled to match the total number of samples and the number of bins in the histogram. The close match between the pdf and the histogram indicates that Rayleigh fading is a good approximation of the actual fading conditions.

The corridor has metal grates and cable ducts along its sides, and large metal cabinets at the entrance. This means that the radio signal may be efficiently reflected from a few large surfaces, so the robot does not receive signals of equal strength from all directions as required for Rayleigh fading. As shown by the measurements, this gives somewhat different spatial properties to the fading. The RSS fluctuates as in the lab room, but also over longer distances, much like shadowing. A representative measurement result is illustrated in Figure 2.14, and autocorrelation estimates for eight measurements are superimposed in Figure 2.15. The measurement in Figure 2.14 corresponds to the slowest decaying autocorrelation estimate. At $\Delta = 15$ cm, all autocorrelation estimates seem to have reached the noise floor for this environment.

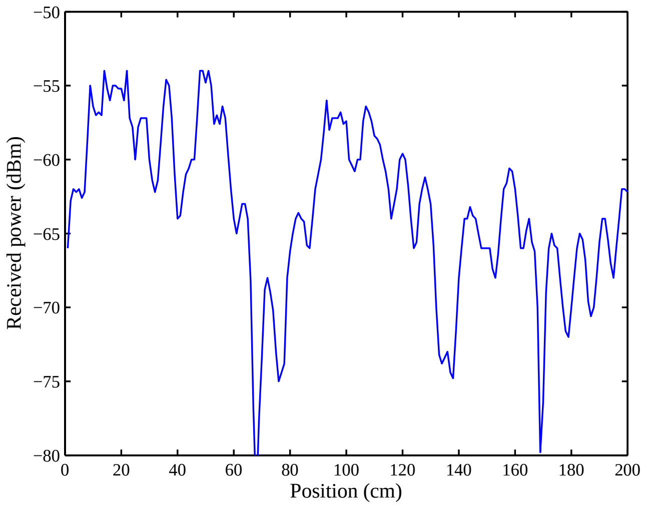

Figure 2.14: Measurement results from the corridor. The multipath fading is probably superimposed on a shadowing effect.

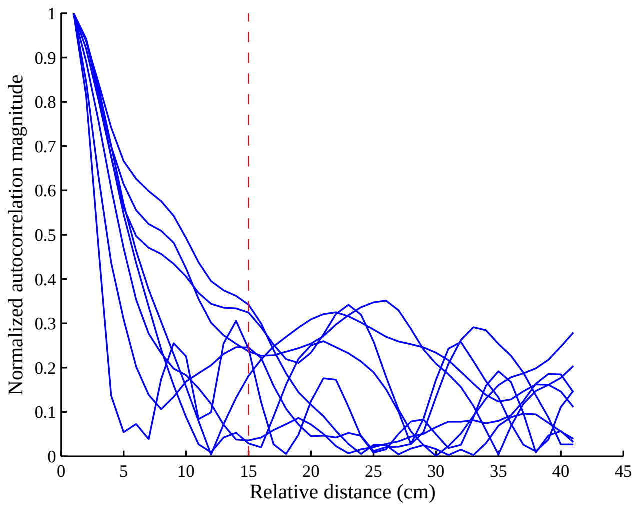

Figure 2.15: Autocorrelation estimates for eight measurement series in the corridor. The dashed line is the estimated decorrelation distance Δ. The autocorrelation decays slowly for some series (cf. Figure 2.12), probably due to shadowing effects.

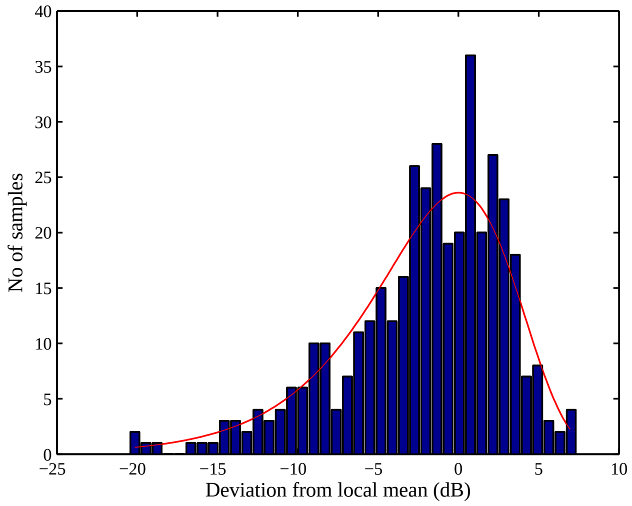

Figure 2.16: Histogram of 400 signal strength samples, taken in the corridor with sample spacing 15 cm. The distribution is similar to the pdf of perfect Rayleigh fading, included as a reference.

To estimate the pdf for the corridor, we took samples 15 cm apart and collected them in the histogram in Figure 2.16. Despite the difference in spatial properties, this distribution also resembles that of Rayleigh fading.

The CC2420 data sheet states an RSSI accuracy of ± 6 dB and linearity within ± 3 dB (Chipcon AS, 2006). Since we are not interested in the absolute signal power, we therefore consider the measurements to have an uncertainty of 3 dB. During our measurements in static environments, the typical standard deviation within 20 packets was less than 1 dB.

Since our motivating application is autonomous exploration or surveillance of indoor environments, we expect those environments to be static, *i.e.*, with no humans present and little or no movement except for that of the robots themselves. Therefore, the fast fading should not change over time, but only as a function of the position of the transmitter and receiver. We call this *static multipath fading*. To verify this, we made two measurements, first driving the robot forward 100 cm and then back again along the same line. As illustrated in Figure 2.17, the signal strength as a function of position is very similar between the measurements. The RMS deviation between the measurements is 1.2 dB, *i.e.*, well within the expected uncertainty.

We finally conjectured that the angle between the line of movement and the line connecting the transmitter and receiver did not affect the statistical properties of the fading. We verified this by doing measurements in several directions during the

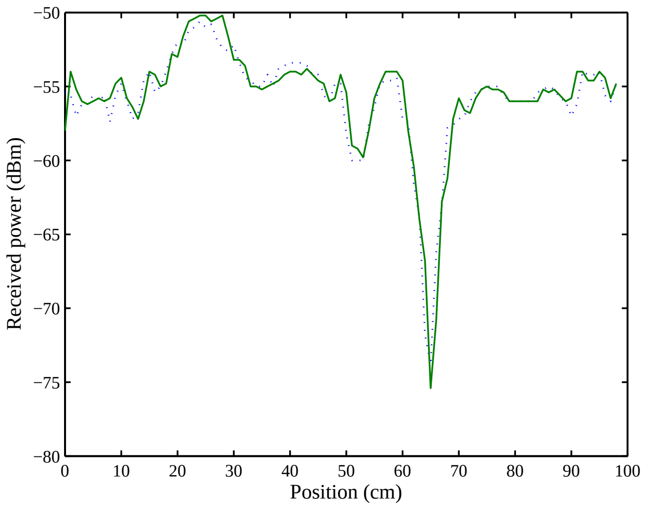

Figure 2.17: Two measurement series along the same trajectory. The signal strength is clearly a function of the position, and does not vary over time. The RMS deviation between the measurements is 1.2 dB, *i.e.*, within the measurement accuracy.

test in the lab room, and it did not affect the distribution or spatial correlation. Of course this only holds for movements much shorter than the distance to the transmitter, where the change in path loss in negligible.

This concludes the section on radio communication, the results from which will be employed in Chapter 3. There we propose methods to exploit the multipath fading for improving communication between robots.

Chapter 3

Communication-Aware Trajectory Tracking

To achieve the full advantage of using multi-agent systems, it is important that the agents can cooperate, and this often requires communication. In this chapter, we formulate the tradeoff between tracking a desired trajectory and at the same time maintaining communications. First we study the special case of stationary agents, *i.e.*, when the reference trajectory is reduced to a fixed position where the agent will stand still for a long time. Second, we study the more general case when the agent is given a time-varying reference trajectory to follow, and propose a way to trade tracking accuracy for better communication performance.

3.1 Problem Formulation

We assume that each agent generates some kind of information that must be sent to a base station or other agent. The overall performance of the system depends on making a proper tradeoff between maintaining good communication and following a desired trajectory. We also assume that the information can be buffered and delivered with some delay without losing its value. An example of such a scenario could be a surveillance robot equipped with a camera, sending pictures every second to a human operator.

For modelling each agent, we have concentrated on what we believe to be the most common robot types today—car-like robots and robots with differential drive. The kinematics of a robot with position $q \in \mathbb{R}^2$, bearing $\theta \in [0, 2\pi[$ and controls (v, ω) can be expressed as:

$$\begin{aligned} \dot{q}_1 &= v \ \cos\theta \\ \dot{q}_2 &= v \ \sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3.1}$$

For car-like robots, the angular velocity is not a free input, but determined by the steering angle α:

$$\omega = \frac{v}{L}\tan\alpha,$$

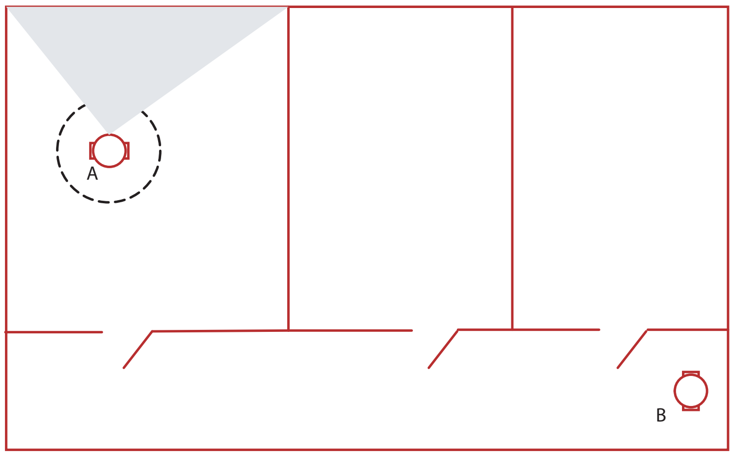

Figure 3.1: A scenario where a robot (A) enters a room to monitor part of it, and send the data back to another robot (B). The task can be fulfilled from anywhere inside the dashed circle, so the robot can move inside it to improve the communications with B.

with L being the distance between rear and front axels, and α limited by $|\alpha| \leq \alpha_{max}$.

For both stationary reference points and time-varying trajectory tracking, we assume that the received signal strength (RSS) varies due to static multipath fading, as described in Chapter 2. In both cases, the problem can be described as:

Track the desired reference position, while ensuring good communications to a fix base station.

This will be formulated more precisely for each case below.

3.2 Stationary Reference Position

In this section, we consider situations when a robot is to stand still at a specified position, but the application allows the robot to digress from it slightly to improve the communication. A typical scenario is demonstrated in Figure 3.1, where robot A monitors a room and needs to send data to robot B in the corridor outside. The task allows A to deviate within a given region in search of higher signal strength.

As described in the background, the RSS due to multipath fading is practically impossible to predict and has multiple local maxima, so finding the optimal position would require visiting all of the feasible region. We therefore suggest sampling the RSS in a finite number of points, and then going back to the best. This requires high-accuracy navigation, which is not always available. An alternative is to sample a few points to estimate the nominal RSS and then continue the sampling, stopping at a point that offers a given improvement over the nominal level. We can then statistically express how many points need to be sampled.

3.2.1 Problem Formulation

The method we propose gives rise to two separate problems: First we need to find the number of RSS samples we need to take to achieve a given improvement in signal strength over the nominal value. Our first problem is thus formulated as:

Problem A: Find the number N of independent samples that we need to get an improvement of G dB over the nominal RSS, with probability P.

Then we need to find a feasible trajectory that visits that number of sampling points without deviating too far from the desired static position:

Problem B: Find a trajectory that is simple to follow with a car-like or differential drive robot, and offers N sampling points, spaced at least Δ, without deviating more than R from the original position.

The general problem has thus been divided into two parts: First finding the number of samples N required to achieve the desired performance and, second, finding a suitable trajectory for the robot to visit that many sampling points. In the following, we provide solutions first in the theoretical case of Rayleigh fading, and then based on properties of real environments.

3.2.2 Solution for Perfect Rayleigh Fading

In this section we give a solution with provable properties in the case of perfect Rayleigh fading. We first give a proposition on the number of samples required to achieve a certain gain and then suggest two alternative strategies of fitting the required number of independent samples within the region where the robot is allowed to move.

Proposition 3.2.1 (Number of samples)**.** For a Rayleigh fading environment, the number of independent samples N needed to achieve a power gain of G dB with probability P compared to the nominal RSS is given by

$$N = \frac{\ln(1-P)}{\ln(1-\exp(-10^{G/10}))}.$$

Proof: From (2.1), we have the pdf of the signal power, which gives the cumulative distribution function (cdf)

$$C(P_n) := \text{Prob}(X < P_n) = 1 - e^{-P_n/P_r}$$

i.e., the probability that the power in a single sample is lower than the threshold P_n. Taking N independent samples, the probability that all of them are lower than P_n is $C(P_n)^N$. We note that at least one sample being greater than P_n is the complementary event to the above, and since $P_n/P_r = 10^{G/10}$, the probability of this is

$$\text{Prob}(G) = 1 - \left[1 - \exp(-10^{G/10})\right]^N.$$

Solving for N gives the proposition. □

As described in Section 2.4, samples taken at a distance of $0.38\lambda_c$ can be considered independent. This can be viewed as each sample being surrounded by a disc of radius $0.19\lambda_c$ where no more samples should be taken. So taking N independent samples inside the feasible region with radius R is essentially a two-dimensional sphere-packing problem.

We propose two possible sampling trajectories; driving in a circle and sweeping a hexagonal lattice. They represent different trade-offs between ease of navigation and maximizing the number of samples.

Proposition 3.2.2 (Circular sampling trajectory)**.** If N samples are taken on a circle, and the samples are at a distance not less than Δ, the radius of the circle must be

$$r \geq \frac{\Delta}{\sqrt{2}\sqrt{1-\cos(2\pi/N)}}.$$

This is illustrated in Figure 3.2a. Another possible sampling pattern is the intersection of a hexagonal lattice and a circle with radius r. A hexagonal lattice can be defined as

$$\left\{(x,y) = \Delta(k+a, \ell\sqrt{3}/2) : \ a = \frac{1}{2} \bmod (\ell, 2), \ k, \ell \in \mathbb{Z}\right\}$$

which was proven by Thue and Tóth (Hales, 1998) to be optimal for two dimensional sphere-packing. The distance Δ is the vertex distance. This arrangement of sampling points is also suitable for being covered by differential drive or car-like robots and with collision sensors also in the back, one could reverse along every second line to simplify maneuvering. Sensors such as a camera can be pointed in the interesting direction during the whole sampling procedure. If the robot detects an obstacle, it can simply turn earlier and start following the next line back. A

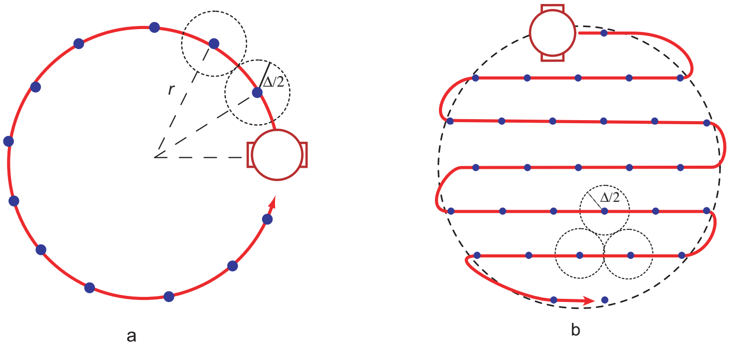

Figure 3.2: Two possible sampling trajectories: a circle (a) and a hexagonal lattice (b). In both cases, the distance between sampling points is at least Δ.

hexagonal lattice with sampling trajectory is depicted in Figure 3.2b. The required size of the sampling region is stated by the following proposition:

Proposition 3.2.3 (Hexagonal lattice of samples)**.** A hexagonal lattice with vertex distance Δ has at least N vertices within the distance

$$r = \left\lceil \sqrt{\frac{\sqrt{3}(N+1)}{2\pi}} + \frac{1}{\sqrt{3}} \right\rceil \Delta \tag{3.2}$$

from the origin.

Proof: Each vertex can be regarded as the center of a hexagon with area $\sqrt{3}\Delta^2/2$, as shown in Figure 3.3a. A circle of radius a has an area equal to or greater than the area covered by

$$N = \left\lfloor \frac{2\pi a^2}{\sqrt{3}\Delta^2} \right\rfloor \tag{3.3}$$

such hexagons. The hexagons can be tiled so that their centers all fit within a circle of radius $a + \Delta\sqrt{3}$, see Figure 3.3b. This can be proved as follows.

Assume that any hexagon is completely outside the circle. Since the remaining hexagons cannot fill the circle, there must be some free space partially inside, and since hexagons can be tiled with no gaps, this space must be on the perimeter. So the hexagon can be moved there instead. To complete the proof, we also note that no part of a hexagon is more than $\Delta/\sqrt{3}$ from its center, so since all hexagons have some part inside the circle of radius a, their centers must then fit inside a concentric circle of radius $a + \Delta/\sqrt{3}$.

Solving (3.3) for a, using that $N+1 \geq \lfloor N \rfloor$ and adding the margin $\Delta/\sqrt{3}$, gives the proposition. $\square$

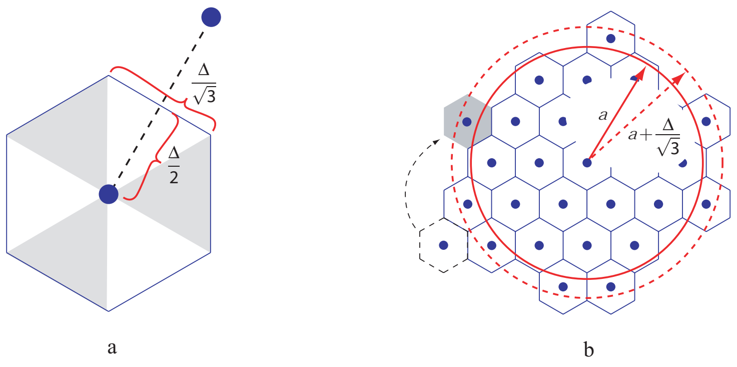

Figure 3.3: (a) The sampling pattern follows a hexagonal lattice. (b) If one of the N hexagons (dashed) is completely outside the circle of radius a, there must exist a free space partially inside the circle, where it can be moved (in gray).

Other trajectories than the two described here are of course also possible; the fewer samples needed, the greater the flexibility to choose a trajectory.

3.2.3 Application to Indoor Environment

Using the measurements from Chapter 2, we can provide solutions parallel to the above, but parameterized for real environments. We first compute the estimated cdf $\hat{C}(P_n)$. This yields a result similar to Proposition 3.2.1, but where the signal gain is expressed in relation to the local average: The probability of achieving gain G when taking N samples can be estimated as

$$\text{Prob}(G, N) = 1 - \hat{C}(G)^N.$$

Several curves of $\text{Prob}(G, N)$, for some values of G, are plotted for the lab environment as well as the corridor, in Figures 3.4 and 3.5, respectively. These figures summarize the answer to Problem A, showing how many samples are needed to reach a given gain with a specified probability.

In practice this means that if the robot can take approximately 9 samples in the lab room (or 14 in the corridor), it has a 95% chance of finding a position where the signal strength is 3 dB better than the local average. Under the same conditions, the probability of finding a point where the signal strength is at least equal to the local average (and thus avoiding any of the deep fades) is greater than 99.9%. Taking 9 samples in the lab room can be done by going in a circle of radius 8 cm. Conversely, the curves can be used as a guideline for an application designer, choosing the allowed deviation based on what signal gain is needed. The method is illustrated further in Section 3.4.

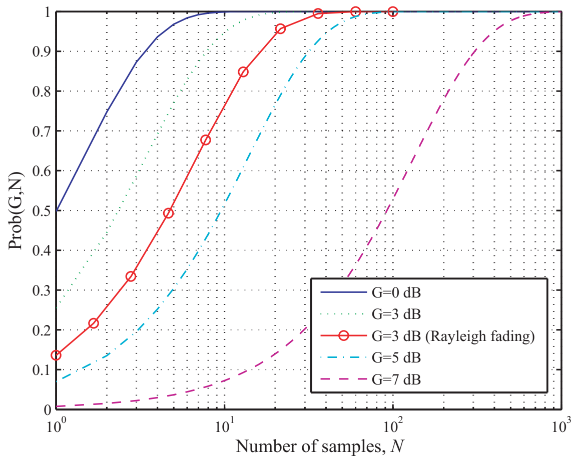

Figure 3.4: Results for the lab room: The required number of independent samples of the RSS to achieve a given gain (compared to the local average) with a certain confidence level. We have plotted curves for several gains in the interval 0-7 dB, as well as the corresponding 3 dB curve for the case of perfect Rayleigh fading.

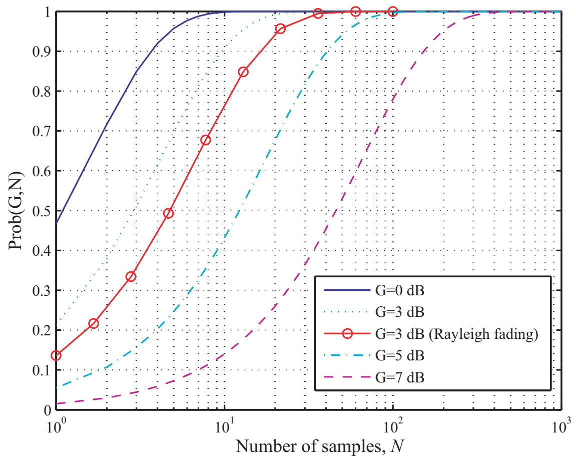

Figure 3.5: Results for the corridor: The required number of independent samples of the RSS to achieve a given gain (compared to the local average) with a certain confidence level. We have plotted curves for several gains in the interval 0-7 dB, as well as the corresponding 3 dB curve for the case of perfect Rayleigh fading.

3.2.4 Other Types of Antenna Diversity

As mentioned in the introduction, the problem of multipath fading can also be countered in other ways, for example by antenna diversity as used in WLAN base stations. Our approach can be seen as a sort of antenna diversity over time, in case the robot is too small to host two antennas separated by at least half a wavelength. (Due to the asymmetry pointed out earlier, in some cases antenna diversity at the base station does not give the same advantages.) Relating to Chapter 4, this also motivates moving groups of robots in hexagonal lattice formations with some decorrelation distance between agents. Such a lattice represents the tightest possible formation offering antenna diversity gain for the group as a whole: ensuring that at least some of the robots have good signal strength.

3.3 Time-Varying Reference Trajectory

In the previous section, we considered how to position a robot close enough to a stationary point, while maintaining good communications. We now consider the more general case of the robot tracking a *time-varying* reference trajectory in a multipath fading environment. How should the tradeoff between tracking and communication be done then?

To illustrate the problem, we consider the following scenario, depicted in Figure 3.6: During night, a robot is patrolling one floor of an office building, collecting infrared camera imagery of each room that is transmitted to an operator on the ground floor. The robot follows a predetermined pseudo-random trajectory, but can adjust its movement along the trajectory to ensure that the images are fed to the operator with minimum delay. Due to multipath fading, the link capacity from the robot to a base station varies when it drives, giving a certain average capacity. But the robot can also choose to stop at a point with stronger signal, thereby emptying its buffer faster, at the expense of lagging behind the reference and having to use more power to catch up. We then ask the question: What is the optimal way to switch between driving and stopping and how should it accelerate, to minimize the tracking error, transmission buffer length and power consumption in its motors? This problem lends itself well to a hybrid optimal control formulation. It is well known that solving such problems can be problematic due to the "curse of dimensionality", so we have applied the method of *relaxed dynamic programming*, proposed by Lincoln and Rantzer (Lincoln and Rantzer, 2006), that alleviate this problem by approximating the value function with bounded relative error.

3.3.1 Problem Formulation

We consider a robot with state x, input u, position $q(x)$ and general dynamics $\dot{x} = f(x, u)$, following a time-varying reference trajectory $q_{ref}(t)$. The robot collects information at a rate r and tries to transmit it to a base station over a multipath fading channel. Data are stored in a buffer of length $z \geq 0$ (with z being part of

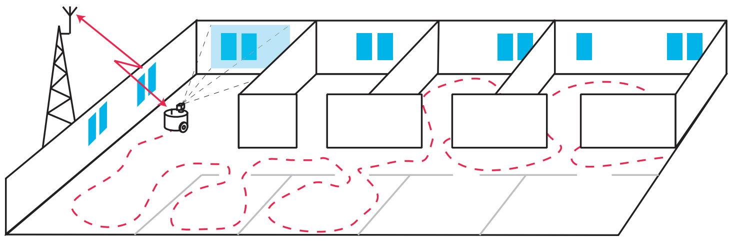

Figure 3.6: Illustration of the considered scenario: A robot follows a patrol trajectory in an office building, sending infrared camera imagery of each room to a base station. Since the signal strength fluctuates due to multipath fading, the robot has to make a tradeoff between stopping at good points to communicate and keeping up with the reference trajectory.

the state vector x) and we assume that the environment is static, which means that the fading is a function of the state of the robot. The buffer dynamics can thus be expressed as $\dot{z} = g(r, x)$.

We can then formulate our problem: *Find a control law $u = u(x)$ such that z is kept small (leading to low latency) as well as the deviation from the reference trajectory $q_{ref}(t)$.*

3.3.2 Robot and Communication Models

In this section, we reduce the model of the agent to only studying its one-dimensional movement along the reference trajectory. It is formulated as a hybrid model, switching between driving and standing still. We then present a model for the buffer and how the communication link capacity varies due to the discrete state of the robot.

Reduced Robot Model

If s represents time, the kinematics of the differential drive robot (3.1) can be rewritten as:

$$\frac{dq_1}{ds} = v(s)\ \sin\theta(s)$$
$$\frac{dq_2}{ds} = v(s)\ \cos\theta(s)$$
$$\frac{d\theta}{ds} = \omega(s)$$

We assume that the robot has a given reference trajectory $q_{ref}(s)$ and a controller capable of following it, *i.e.*, computing controls $v_{ref}(s)$ and $w_{ref}(s)$ so that $q(s) \equiv q_{ref}(s)$. To solve the problem of communication-aware trajectory tracking, we introduce another controller that varies the velocity along the trajectory. It can be viewed as a time scaling

$$\frac{ds}{dt} = \gamma(t) > 0, \; s(0) = 0, \tag{3.4}$$

where $\gamma(t)$ is determined by the controller. This allows us to speed up the system or slow it down, by applying the controls

$$\begin{aligned} v(t) &= \gamma(t) v_{ref}(s) \\ \omega(t) &= \gamma(t) \omega_{ref}(s), \end{aligned}$$

where s is governed by (3.4).

We define $\Delta \in \mathbb{R}$ as the one-dimensional position along the trajectory, relative to the reference. If $\Delta < 0$, this means that the robot is lagging behind. We also introduce the relative velocity $\varphi = \dot{\Delta}$. To achieve relative velocity φ, we simply set the scaling

$$\gamma(t) = 1 + \frac{\varphi}{v_{ref}(s)}. \tag{3.5}$$

To make the movement smoother, we model the one-dimensional reduced system as a double integrator. We further want to model the fact that many kinds of robots only consume negligible power when breaking, using disc breaks or by short-circuiting its electric motors. So we consider the robot to have two discrete modes: `drive` and `stop`, where it drives and stands still, respectively. In the `stop` mode, we let the relative velocity be self-stabilizing to the value of $-v_{ref}$ (which means standing still). The motion along the reference trajectory can thus be described as:

$$\texttt{stop}: \begin{cases} \dot{\Delta} = -\varphi \\ \dot{\varphi} = -k_v(\varphi - v_{ref}) \end{cases} \quad \texttt{drive}: \begin{cases} \dot{\Delta} = -\varphi \\ \dot{\varphi} = u \end{cases}.$$

The factor $k_v \gg 1$ is chosen to ensure fast convergence of φ to $-v_{ref}$ when stopping.

Data Buffer Model

We then turn our attention to the data buffer onboard the robot. It has size z, inflow rate r and outflow (or link capacity) c, so its dynamics are

$$\dot{z} = r - c.$$

For simplicity the inflow r is assumed constant, but the approach presented could be extended to adapt to changing inflows. Below, we will describe our model for how the outflow varies with the mode of the robot. But first we note that the buffer is modelled as lossless, which means that no packets are discarded.

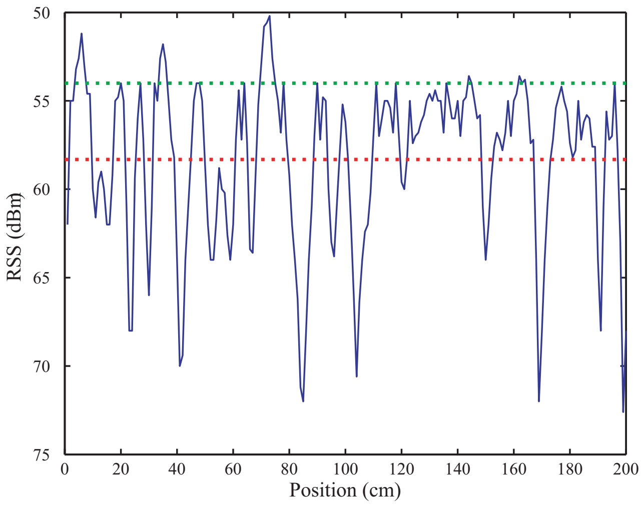

Figure 3.7: Measurement results from a lab room, where the RSS varies due to multipath fading. The lower dashed line denotes the average level, while the upper line is chosen by hand, so that a point with that RSS can be found within a few cm from almost any position.

Communication Model under Multipath Fading

As mentioned above, the attenuation (or gain) due to multipath fading varies over movements of fractions of a wavelength. In Figure 3.7, we illustrate a representative data set from measurements, described in more detail in Section 2.4.3. The graph shows the received signal strength (RSS) for a robot as it moves along a straight line in a lab room with computers and equipment reflecting the incoming signal. The average RSS of −58 dBm is marked in the figure, but almost everywhere, the robot is only a few cm away from the nearest peak at the higher level of −54 dBm that is also marked. A gain of 4 dB may seem small, but as described in Section 2.4, for a robot on the limit of losing contact it can mean a substantial increase in bandwidth.

Motivated by this, we make a simple model for the communication channel: During motion, the radio hardware will smooth the RSS variations, producing an average buffer outflow in the `drive` state, defined as c_d. But when the robot decides to stop, we assume that it can instantly find a point with a higher signal strength and a higher capacity defined as c_s. It is important to point out that this approach is most useful in the interval $c_d < r < c_s$, since if the inflow is lower than c_d the robot is not forced to stop, and if it is larger than c_s, some higher-level protocol must discard data to stop the buffer from overflowing.

3.3.3 Hybrid Optimal Control Solution

In this section, we formulate the problem as a hybrid optimal control problem. We then present relaxed dynamic programming as a way of approximating the solution with bounded suboptimality and state an algorithm that computes a value function that we can derive a control law from. Finally we show how to do this derivation and represent the control law in a compact way and we illustrate the result in a subset of the state space.

Switched Linear System

To describe the whole system, we collect both the robot and buffer states in the same state vector. We also include an integral state Δ_I to allow the controller to attenuate the static error in Δ. For a more compact representation, we finally add a constant element to the state vector so the system can be denoted as

$$\dot{x} = A_\sigma x + B_\sigma u, \; x = (\Delta, \varphi, \Delta_I, z, 1)^T \,,$$

where the controls are $u \in \mathbb{R}$ and $\sigma \in \{0, 1\}$, which correspond to `stop` and `drive`, respectively, and

$$\begin{aligned} A_0 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -k_v & 0 & 0 & -k_v v_{ref} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r - c_s \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; B_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ A_1 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r - c_d \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; B_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned}$$

Time Discretization

We will treat the system in discrete time with sampling time τ. Defining $x[n] = x(n\tau)$, we can express the discrete dynamics as

$$x[n+1] = f(x[n], u[n], \sigma[n]) = \Phi_{\sigma[n]} x[n] + \Gamma_{\sigma[n]} u[n],$$

where

$$\Phi_\sigma = e^{A_\sigma \tau} \text{ and } \Gamma_\sigma = \int_0^\tau e^{A_\sigma(\tau - s)} B_\sigma \, ds.$$

Cost Function

To maintain low latency and margin for unexpected buffer inflow, it is desirable to keep the buffer size low. At the same time we also want to stay close to the reference trajectory and, for both smoothness and power conservation, limit the control magnitude. Unfortunately, under the assumption that $c_d < r < c_s$, both Δ and z cannot simultaneously converge to zero, so we introduce a decay factor λ^n, with $\lambda < 1$, to get a finite cost even though we use an infinite horizon. Now, given an initial condition x_0, the optimal control problem can be defined as

$$\begin{aligned} \min_{\sigma[n],u[n]} \quad & \sum_{n=0}^{\infty} \left(x^T[n]Qx[n] + Ru^2[n] \right) \lambda^n \\ \text{s.t.} \quad & x[n+1] = f(x[n], u[n], \sigma[n]) \\ & x[0] = x_0 \\ & x_4 \geq 0, \end{aligned} \tag{3.6}$$

where $Q = Q^T$ is positive semidefinite and R is a positive constant. We will use a higher penalty on z to reduce its static error, which will be illustrated in Section 3.3.4.

Dynamic Programming

Relaxed dynamic programming is based on approximating the optimal value function (also called cost-to-go) at state x, defined as

$$V^*(x) = \min_{\sigma[n],u[n]} \sum_{n=0}^{\infty} \ell(x[n], u[n]) \lambda^n,$$

where

$$\ell(x, u) = x^T Q x + Ru^2,$$

under the same constraints as in (3.6), but with $x_0 = x$. Once we have $V^*(x)$, we can derive the optimal control law as

$$(u^*(x), \sigma^*(x)) = \text{argmin}_{u,\sigma} \left\{ V^*(f(x, u, \sigma)) + \ell(x, u) \right\}.$$

We introduce the value function $V_k(x) : \mathbb{R}^5 \to \mathbb{R}$ to approximate the optimal value function, and use value iteration to recursively refine the approximation:

$$V_{k+1}(x) = \min_{u,\sigma} \left\{ V_k(f(x, u, \sigma)) + \ell(x, u) \right\}. \tag{3.7}$$

We set $V_0 \equiv 0$, which is used to start the iteration. For a given k, the iterate $V_k(x)$ answers the question "what is the lowest possible cost for k time steps of the system trajectory, given that it starts in x?" Under mild conditions, it holds that (Lincoln and Rantzer, 2006)

$$\lim_{k \to \infty} V_k(x) = V^*(x).$$

The problem is that, if applied naively, value iteration requires that we consider *all* possible switching sequences of length k steps, so the complexity of the problem grows exponentially with our horizon length k. This "curse of dimensionality" is a well known drawback of dynamic programming. Before we present a way to avoid this, we will see how the optimal control u can be computed for a known switching sequence.

To facilitate the notation, we here let $\Phi = \Phi_{\sigma[n]}$ and $\Gamma = \Gamma_{\sigma[n]}$ for some given mode $\sigma[n]$. We also assume that, at time $n{+}1$, the value function can be written on a quadratic form $V(x[n{+}1]) = x^T[n{+}1]P[n{+}1]x[n{+}1]$, where $P[n{+}1]$ is a symmetric positive definite matrix. Then the optimal cost at time n is $x^T[n]P[n]x[n]$, where

$$\begin{aligned} P[n] = \Phi^T P[n+1]\Phi + Q - \Phi^T P[n+1]\Gamma \\ \times \left[\Gamma^T P[n+1]\Gamma + R\right]^{-1} \Gamma^T P[n+1]\Phi \end{aligned} \qquad (3.8)$$

and positive semidefinite (Åström and Wittenmark, 1997). Further, the optimal control signal is

$$u^*(x[n]) = -\left[R + \Gamma^T P[n+1]\Gamma\right]^{-1} \Gamma^T P[n+1]\Phi\ x[n]. \qquad (3.9)$$

Since the cost is again on quadratic form, for a known switching sequence, we can use that $P[k] \equiv 0$ to iteratively compute the optimal cost and control signal.

Relaxed Dynamic Programming

Let N_k be the number of candidates for the optimal switching sequence of length k. Then switching sequence number $\kappa \in \{1, \ldots, N_k\}$ is $\sigma_\kappa[n] : \{0, 1, \ldots, k\} \to \{0, 1\}$. Also let $\Pi_k = \{P_1, \ldots, P_{N_k}\}$ be the set of matrices P_κ such that the cost associated with $\sigma_\kappa[n]$ is $x^T[0]P_\kappa x[0]$. Using horizon length k and a sufficiently rich set Π_k, we can now parameterize the value function in a way that can be used to perform the iteration (3.7):

$$V_k(x) = \min_{P_\kappa \in \Pi_k} x^T P_\kappa x.$$

As mentioned above, the set Π_k quickly becomes prohibitively large if we do not discard some candidate switching sequences during the recursion. The method of relaxed dynamic programming, proposed by Lincoln and Rantzer (2006), does just that: at each iteration, it retains only the candidates P_κ that are needed to represent the value function with a given bounded relative error. If this bound is sufficiently large, the number of candidates will converge to a finite value as $k \to \infty$.

More formally, the idea is to find an approximation $V_k(x)$ of the optimal value function such that, for $\underline{\alpha} < 1 < \overline{\alpha}$,

$$\begin{aligned} \min_{u,\sigma}\{V_k(f(x,u,\sigma)) + \underline{\alpha}\ell(x,u)\} \leq V_k(x) \\ \leq \min_{u,\sigma}\{V_k(f(x,u,\sigma)) + \overline{\alpha}\ell(x,u)\}\ \forall\ x. \end{aligned} \qquad (3.10)$$

Algorithm 1 Relaxed Dynamic Programming

```
1: $k := 0$, $\Pi_0 = 0_{n\times n}$
2: while (3.10) is not fulfilled do
3:     $k := k+1$
4:     Form $\overline{\Pi}_k$ and $\underline{\Pi}_k$ by propagating the matrices in $\Pi_{k-1}$ one step backwards
       in time, both with $\sigma = 0$ and $\sigma = 1$, as defined in (3.8).
5:     Sort the sets $\overline{\Pi}_k = \{\overline{P}_1, \dots, \overline{P}_{N_k}\}$ and $\underline{\Pi}_k = \{\underline{P}_1, \dots, \underline{P}_{N_k}\}$ so that
       $\operatorname{tr} \overline{P}_1 \leq \dots \leq \operatorname{tr} \overline{P}_{N_k}$ and $\overline{P}_i \geq \underline{P}_i \ \forall\ i$.
6:     $\Pi_k := \emptyset$, $i := 0$
7:     while $i \leq N_k$ do
8:         if $\nexists$ a convex combination $P$ of matrices in $\Pi_k$ such that $P \leq \overline{P}_i$ then
9:             Add $\underline{P_i}$ to $\Pi_k$.
10:        end if
11:        $i := i+1$
12:    end while
13: end while
```

Using the appropriate "slack", the cost-to-go function can be parameterized by a much smaller set Π_k, and we can discard many candidate switching sequences at each iteration step. For the discarding procedure, we define $\underline{\Pi_k} = \{\underline{P}_1, \dots, \underline{P}_{N_k}\}$ as the set of matrices $\underline{P}_\kappa$ such that $\underline{\alpha}$ times the cost for the switching sequence $\sigma_\kappa(n)$ of length k is $x(0)^T \underline{P}_\kappa x(0)$. The set $\overline{\Pi}_k$ and the matrices $\overline{P}_\kappa$ are defined analogously, using $\overline{\alpha}$. The method to find $V_k(x)$ is presented in Algorithm 1.

Note that step 8 of the algorithm is an S-procedure test to see if there exists an x such that

$$x^T \overline{P}_i x < \min_{P \in \Pi_k} x^T P x.$$

If not, then P_i is not needed to represent the value function with sufficient accuracy. Also note that by ordering the matrices by trace, we ensure that smaller matrices are added first to Π_k, which in practice means that we will add fewer elements.

When $V_k(x)$ fulfills the stopping criterion (3.10) at, say, $k = \overline{k}$, it can be applied iteratively to yield that for all $k \geq \overline{k}$

$$\underline{\alpha} V^*(x) = \min_{\sigma[n],u[n]} \sum_{n=0}^{\infty} \underline{\alpha} \ell(x,\sigma,u) \leq V_k(x) \leq \min_{\sigma[n],u[n]} \sum_{n=0}^{\infty} \overline{\alpha} \ell(x,\sigma,u) = \overline{\alpha} V^*(x).$$

As an example, if $\overline{\alpha} = \underline{\alpha}^{-1} = 1.05$, this means that the computed $V_k(x)$ under- or overestimates the optimal cost-to-go by maximally a factor of 5%. This is illustrated in Figure 3.8, from Lincoln and Rantzer (2006).

When the stopping criterion is fulfilled, it means that no more candidate switching sequences need to be added to represent the value function. Then the number N_k of candidates stops growing, as depicted in Figure 3.9. For comparison we also included the number of candidates M_k that would have to be considered using

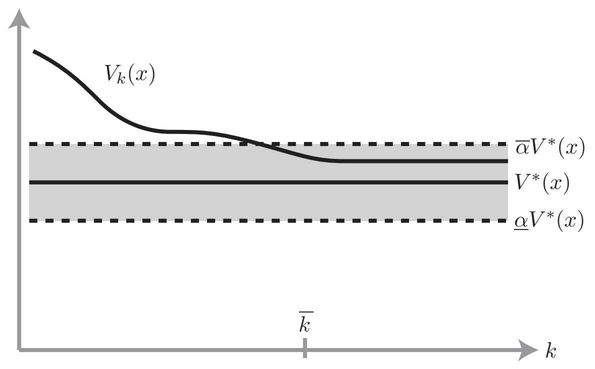

Figure 3.8: The value function $V_k(x)$ converging to the close-to-optimal set. At $k = \overline{k}$, the stopping criterion is fulfilled.

normal dynamic programming. Note that N_k does not converge to a number, but rather stops growing and then displays random variations due to numerical effects and small random perturbations in the sorting of $\overline{\Pi}_k$ to make the search more efficient. The figure shows the result for $\overline{\alpha} = \underline{\alpha}^{-1} = 2$ and $\lambda = 0.9$, which is also what we used to compute the controller used in all simulations. We used the result after 100 iterations, when N_k had clearly stopped growing.

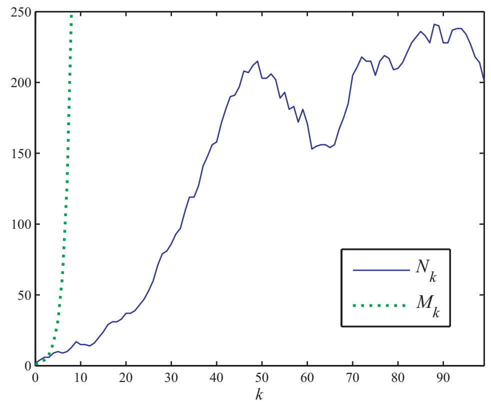

Figure 3.9: The number N_k of matrices in Π_k needed to represent the value function at each iteration step. N_k stops increasing, indicating that (3.10) is fulfilled, after about 50 iterations. Without discarding any candidates, the complexity would grow as $M_k = 2^k$, which is illustrated for comparison.

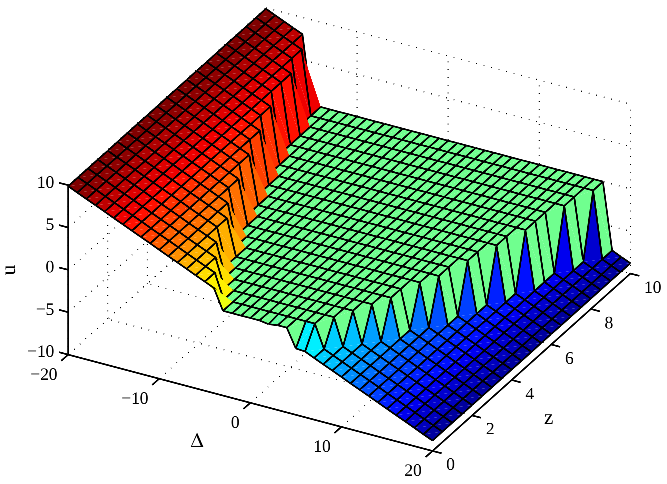

Figure 3.10: The resulting control law for the subset $\varphi = -v_{ref}$, $\Delta_I = 0$ (corresponding to standing still with an empty integral state in the controller). To also illustrate $\sigma(x)$, we have forced the control to $u = 0$ where $\sigma(x) = 0$. As one would expect, there is a `stop` region for small Δ. If Δ decreases, the controller accelerates the robot and if Δ becomes too large, it slows the robot down.

Resulting Controller

Using the approximation of the value function, we could find the optimal mode $\sigma^*(x)$ as the first mode in the switch sequence corresponding to the matrix

$$P^* = \operatorname{argmin}_{P \in \Pi_{100}} x^T P x.$$

The optimal continuous control $u^*(x)$ was then computed using (3.9), substituting P^* for $P[n+1]$. In a resource-constrained robot, this could also be precomputed and stored as a look-up table of feedback gains L_κ, each associated with a switching sequence. The control signal would then be $u^*(x) = -L_\kappa x$. The resulting control law is plotted in Figure 3.10, for the subset $\varphi = -v_{ref}$, $\Delta_I = 0$ of the state space.

3.3.4 Simulation Results

In this section, we first present an illustration of a system trajectory using the controller derived in the previous section. We then investigate the sensitivity of the closed-loop system to disturbances in buffer size, link capacity and reference trajectory velocity. In all simulations, we have used the sampling time $\tau = 0.1$ s for the controller, but the system dynamics are simulated with much higher resolution. We also set $k_v = 100$, $R = 1$ and $Q = \text{diag}(1, 0, 1, 5, 0)$.

Following a Curved Path

In Figure 3.11, we have simulated the system when following a curved reference trajectory corresponding to

$$\begin{aligned} v_{ref} &\equiv 1 \\ \omega_{ref} &= \sin 0.8t. \end{aligned}$$

This means that the reference is moving at constant velocity along the path, while the robot varies γ as in (3.5) to perform the "stop-and-go" motion dictated by its communication-aware controller. The figure consists of periodical samples of the states of the robot, where the height of the ball over the robot indicates the buffer size. Thus it is possible to see how it stops at some points to empty its buffer. We have used the exaggerated rates $r - c_s = 1$ and $r - c_d = -1$ to illustrate the behavior of the system more clearly.

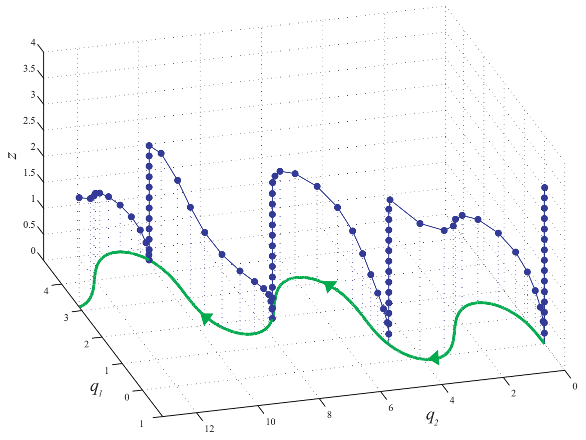

Figure 3.11: A trajectory of the system in the (q_1, q_2, z)-space, sampled with regular intervals. The robot follows the reference trajectory (thick solid line) while stopping from time to time to reduce the buffer size. The robot motion is from right to left.

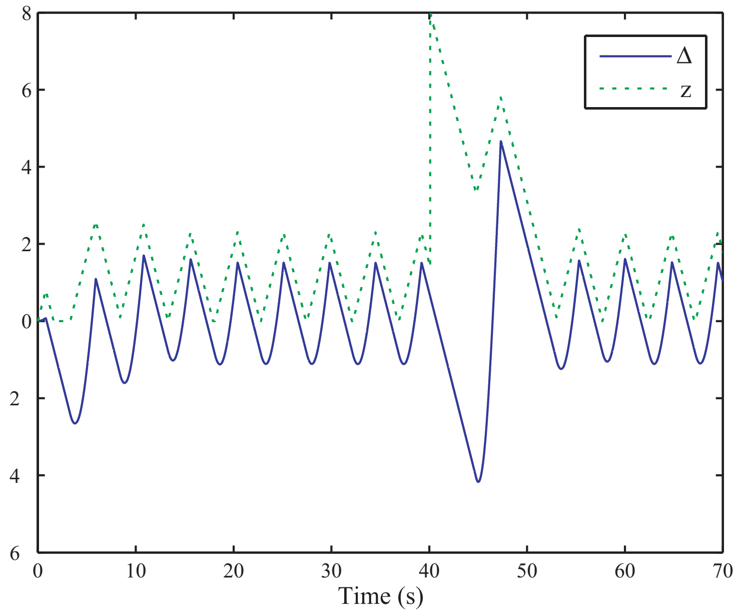

Figure 3.12: An example of a trajectory for the system, starting with an empty buffer and with perfect reference tracking. The robot approaches a limit cycle with a period time of 5.6 s, where it spends 50% of the time in the `drive` and `stop` modes, respectively. After 40 s, extra data is added to the buffer, and this disturbance is effectively attenuated.

Limit Cycle and Buffer Disturbance Rejection

We have also simulated the system starting with an empty buffer and perfect reference tracking. As seen in Figure 3.12, it approaches a limit cycle with a period time of 5.6 s, where is spends 50% of the time in each mode. The lag Δ oscillates around zero while there is still a small static error in z. However, at $t = 40$ s, extra data is added to the buffer, and this impulse disturbance is successfully attenuated. Here we also used $r - c_s = 1$ and $r - c_d = -1$.

Robustness to Capacity Variations

As indicated in the derivation of the communication model, the actual link capacity c_s at the position where the robot stops can vary from the predicted value. We have tested the robustness of the closed-loop system to this model error by adding zero-mean white gaussian noise with standard deviation 2 to c_s. With $c_s = 3$, $c_d = 1$ and $r = 2$, the simulations indicate that the system still oscillates around $\Delta = 0$ and maintains a bounded buffer size z. This is illustrated in Figure 3.13.

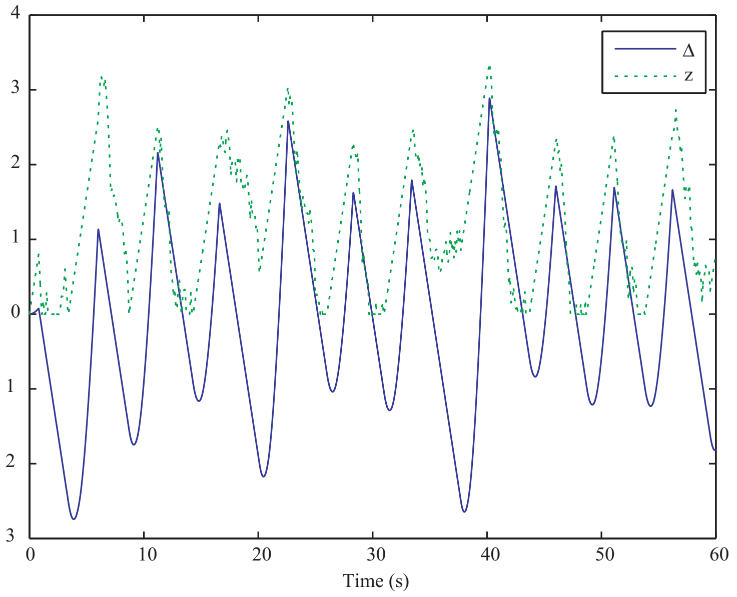

Figure 3.13: A test of how the system performs when zero-mean white gaussian noise with standard variation 1 is added to the link capacity c_s in the `stop` mode. The lag Δ oscillates around zero and the buffer size remains bounded.

3.4 Experimental Results

To demonstrate the method for stationary positioning of a robot, we have made a simple experiment. The robot is placed at random positions and orientations within a 1-by-1 m square in the lab room, as if a task such as patrolling or mapping had made it drive there. We then measured the signal strength between the robot and its base station (a TMote), in the other end of the room.

First we performed 20 such trials, *series 1*, allowing the robot to deviate slightly from the initial position, sampling the RSS in 9 points, separated by 5 cm. It then moved to the point with the best RSS before the measurement was made. Then we performed 20 more trials, *series 2*, where the robot was not allowed to deviate. The result of the two experiment series is plotted as two histograms in Figure 3.14. When just staying at the initial position, seven out of twenty trials yielded signal strengths worse than the local average, in one case by as much as 15 dB. When using our proposed method of adjusting the position, the theoretical analysis predicted a gain of at least 3 dB compared to the local average in 95% of the cases, but in practice this happened in 80% of the trials. It is worth noticing, however, that all trials avoided negative gains as predicted.

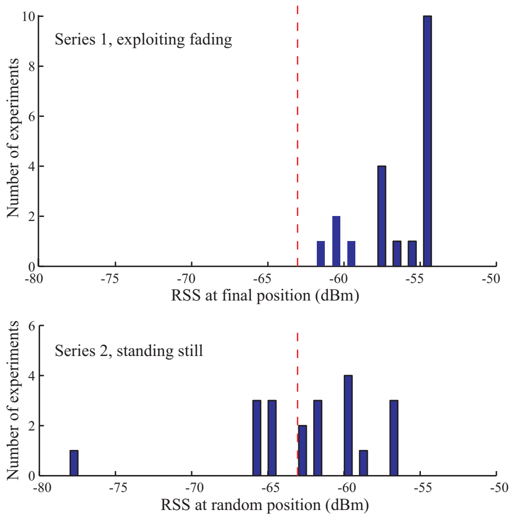

Figure 3.14: Experiment results, measuring the RSS at the final position of the robot. The upper histogram shows the result when exploiting the multipath fading, and the lower histogram shows what happened when the robot did not move from its (random) original position. The dashed line shows the local average RSS.

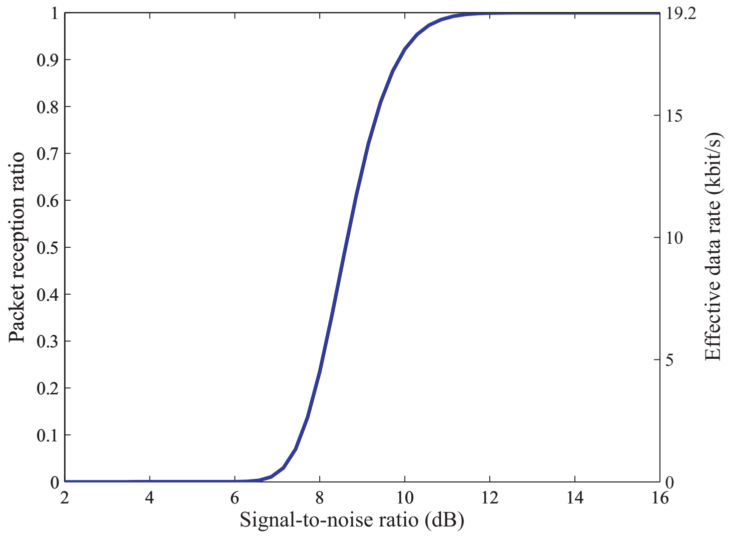

Figure 3.15: An illustration of the link capacity for a MICA2 sensor mote as a function of the signal-to-noise ratio. When the link is on the limit of losing contact, gaining just a few dB can have a large impact on bandwidth.

To illustrate the benefit of gaining 3 dB, Figure 3.15 repeats the results from Section 2.4, illustrating the link bandwidth for a MICA2 mote as a function of the signal-to-noise ratio (SNR). This clearly shows that when a node is on the limit of losing contact, even a small improvement in signal strength can significantly improve the bandwidth. As an example, a mote with bad signal quality (*e.g.*, SNR=7 dB) would receive 1.8% of the packets and thus have an effective bandwidth of 350 bits/s. Sending a 10 kbyte camera image would then take 3 min 49 s. Gaining 3 dB would raise the bandwidth to 17.7 kbit/s, reducing the transmission time to 4.5 seconds.

3.5 Summary

In this chapter, we have studied how to track a stationary or moving reference position in a communication-aware manner. This is done under the assumption of multipath fading, which measurements have shown is an important effect in indoor environments.

First we showed that if a robot is allowed to adjust its position before stopping to perform some static task, it can improve the capacity of its radio link to a base station or other robots. The problem was divided into two parts: first we provided an estimate of the number of points that it needs to sample to find a position that has a given signal strength. Then we suggested two possible sampling strategies to collect the required number of independent samples, without deviating too far from the original position.

Second, we studied the more general case when a robot is to track a time-varying reference position. Then the robot does not have to make detours to find new positions where the RSS may be higher, but rather use good positions it finds by stopping there to transmit. This makes it natural to pose the optimal control problem of deciding when to stand still and when to use extra power to catch up with the reference. We formulated the problem as a linear hybrid optimal control problem and computed an approximate solution by relaxed dynamic programming. Even though this is computationally intensive, the resulting controller can be stored in look-up tables and thus used also on resource-constrained robots. The closed-loop system was simulated under different conditions and maintains a bounded buffer size and zero-mean tracking error.

The approaches above are derived under the assumption of static multipath fading, but the principles carry over to more general situations as well. Making small adjustments to a fix position is relevant in any situation where the signal strength varies over short distances. As an example, it could be incorporated as a software module in laptops, advising the user to move the laptop in a given pattern if the WLAN signal strength is low. The method of stopping at points where communications are good could be used whenever the signal strength varies in space and it is simple to find good positions to stop at. Examples of this could be an underwater robot that can surface to communicate, or a robot searching office rooms, knowing that the signal strength is better in the corridor or near windows.

Chapter 4

Flocking and Formation Control Using Voronoi Partitions

Even small groups of agents quickly become difficult to control manually unless some degree of autonomy is added, as noted earlier in the introduction. In this chapter, we will describe two versions of an algorithm that provides this autonomy, using the geometric notion of Voronoi regions. The first version yields flocking, *i.e.*, keeps the group together, while autonomously navigating to a predefined goal. The other version yields a formation, with strictly defined inter-agent distances, that can be controlled as an entity by an operator. We demonstrate the result in Matlab simulations and finally also in a more realistic virtual testbed.

4.1 Agent and Communication Model

We consider a group of N agents, where $q_i = (x_i, y_i)^T$ is the position of agent $i \in \mathcal{I} = \{1, \ldots, N\}$. We describe their kinematics in discrete time as

$$\begin{aligned} q_i[k+1] &= q_i[k] + u_i[k] \\ ||u_i[k]|| &\leq u_{max}, \end{aligned}$$

where $u_i[k] \in \mathbb{R}^2$ is the control input. Every agent is assumed to be identical and anonymous, *i.e.*, the agents do not have any way to distinguish between different neighbors and they have no predetermined roles. This enhances the flexibility of the system, since group sizes and members can be chosen arbitrarily. Furthermore, we assume that each agent knows its own global position and can sense the relative position of all other agents within a given radius R_{max}. The set of sensed neighbors of agent i is denoted $\mathcal{N}_i$ and we will use the notation $|\mathcal{N}_i|$ for the number of agents in such a set. Each agent also has access to an accurate obstacle map of the workspace, containing the location of the global goal.

4.2 Distributed Coordination Components

In this section, we describe some common components of the algorithms and a suggested two-level architecture that allows such algorithms to control non-holonomic agents as well. This architecture is later used in the high-fidelity simulations reported in Section 4.5.

4.2.1 Voronoi Regions and Vertices

The algorithm relies on the geometric construction of Voronoi regions:

Definition 1 (Voronoi region and vertex)**.** The *Voronoi region* $V_i(\mathcal{N}) \subset \mathbb{R}^2$ consists of all points that are closer or at equal distance to agent i than any other agent in the set $\mathcal{N}$:

$$V_i(\mathcal{N}) = \left\{x \in \mathbb{R}^2 : ||x - q_i|| \leq ||x - q_j|| \; \forall \; j \neq i, j \in \mathcal{N}\right\}. \tag{4.1}$$

Sometimes we will use the simplified notation $V_i = V_i(\mathcal{N})$ when the set $\mathcal{N}$ is obvious from the context. The set of *Voronoi vertices* $X_i(\mathcal{N}) \subset \mathbb{R}^2$ is the set of points on the boundary of the Voronoi region that are centers of empty circles, touching at least three agents.

Figure 4.1a shows an example of the Voronoi region for an agent and its different neighbor sets. The Voronoi vertices are denoted by circles and the dashed lines indicate boundaries for adjacent Voronoi regions. In Figure 4.1b, we have illustrated what can happen if an agent has a finite sensing radius R_{max} and thus estimates its Voronoi region based only on a subset of all neighbors. Then the estimate may not be correct, but it follows from the definition that if a neighbor is farther away than R_{max}, no part of its Voronoi region can be closer to the agent than $R_{max}/2$. Thus, the Voronoi region estimate is correct up to the distance $R_{max}/2$. This makes Voronoi regions well suited for being computed locally, and this property will be used in the algorithms. We finally note that the union of all Voronoi regions covers all of $\mathbb{R}^2$, *i.e.*, there are no holes.

4.2.2 Neighbors

We also need to define the notion of Delaunay and Voronoi neighbors. The Delaunay graph is the dual of the Voronoi graph, connecting all agents whose Voronoi regions intersect.

Definition 2 (Delaunay neighbor)**.** The set $\mathcal{D}_i \subset \mathcal{N}_i$ represents all *Delaunay neighbors* of agent i, i.e., all agents that share at least one Voronoi vertex with it:

$$\mathcal{D}_i = \left\{j : \text{card}\left(X_j(\mathcal{N}_j) \bigcap X_i(\mathcal{N}_i)\right) \geq 1\right\}.$$

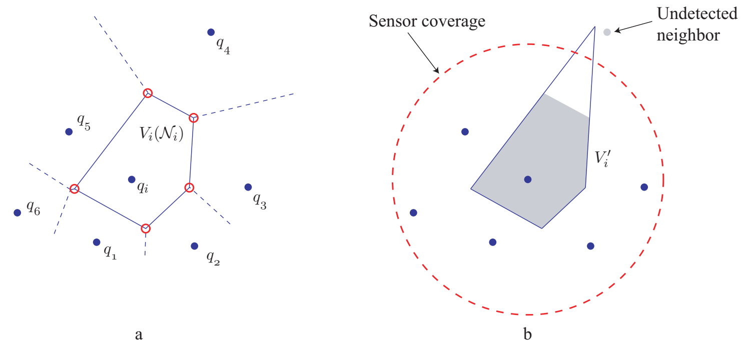

Figure 4.1: Figure a shows the Voronoi region for agent i, as well as boundaries of adjacent regions (dashed) and Voronoi vertices (circles). Here, $\mathcal{D}_i = \{1, \ldots, 6\}$ and $\mathcal{C}_i = \{1, \ldots, 5\}$. Figure b illustrates that if the agent has a finite sensing radius R_{max}, it may not detect all neighbors. The Voronoi region estimate V_i' is, however, correct up to the distance $R_{max}/2$ from the agent.

For some of those neighbors, the intersection is a line rather than just a point:

Definition 3 (Voronoi neighbor)**.** The set $\mathcal{C}_i \subseteq \mathcal{D}_i$ consists of all *Voronoi neighbors* of agent i, i.e., agents that share two vertices with it:

$$\mathcal{C}_i = \left\{ j : \text{card}\left(X_j(\mathcal{N}_j) \bigcap X_i(\mathcal{N}_i) \right) = 2 \right\}.$$

The algorithms also use virtual mirror neighbors, to influence the behavior of an agent. They are created by reflecting real neighbors in the position of the agent, but at a fixed distance d:

Definition 4 (Mirror operator)**.** The mirror operator $M_d(j, i)$ represents a new virtual agent, created as the mirror image of agent j in i, but at a fixed distance d, as illustrated in Figure 4.2. The position of $M_d(j, i)$ is denoted q_{ji}:

$$q_{ji} = q_i + \frac{q_i - q_j}{||q_i - q_j||}\, d. \tag{4.2}$$

We also adopt the notation

$$M_d(\mathcal{N}_i, i) = \bigcup_{j \in \mathcal{N}_i} M_d(j, i)$$

for the set of agents in $\mathcal{N}_i$, all mirrored in agent i.

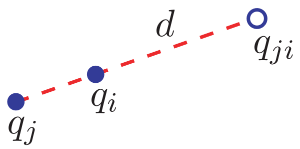

Figure 4.2: Illustration of the mirror operator, representing the image of agent j reflected in agent i but at the fixed distance d. This mirror agent is denoted $M_d(j,i)$, and its position is q_{ji}.

4.2.3 Navigation Function

The concept of a navigation function was introduced by Rimon and Koditschek (1992). It is an artificial potential $\text{NF}(x) : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ that maps all points in the connected part of the obstacle-free workspace Ω to a scalar value, such that there is only one local—and thus global—minimum, located at the goal. It was a clever mathematic construction that was continuously differentiable. It was, however, not very well suited for computation, so Ögren and Leonard (2005) have suggested a modified version that is only piecewise differentiable and has local maxima of measure zero. We make the construction as follows:

1. Let $V \subset \Omega$ be the set of points in a rectangular grid covering Ω, with grid size ξ. Let $E \subset V \times V$ be the set of edges in the grid such that that for two points $x_i, x_j \in V$,
$$(x_i, x_j) \in E \Leftrightarrow ||x_i - x_j|| = \xi.$$
2. Remove all edges in E that intersect obstacles, and then all points in V that no longer have any vertices associated with them.
3. Define a point $g \in V$ as the goal, with $\text{NF}(g) = 0$.
4. The value of NF for all other points $x \in V$ is computed recursively as
$$\text{NF}(x) = \xi + \min_{\{y \in V : (x,y) \in E\}} \text{NF}(y).$$
5. For points inside grids, $x \in \Omega \setminus V$, the value of NF is computed through linear interpolation.

The linear interpolation uses the values of NF in the four corners of the grid surrounding the point and it can be performed in a way that does not create any local minima. For details, we refer to Ögren and Leonard (2005). Figure 4.3 shows an illustration of NF(x), computed with $\xi = 1$ in a subset of Ω around the goal.

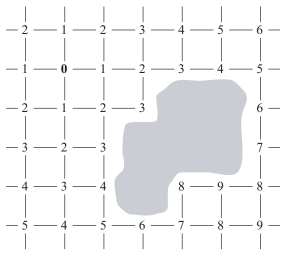

Figure 4.3: An example of a navigation function $\mathrm{NF}(x)$ in a neighborhood of the goal. The grid spacing is $\xi = 1$ and the goal point by definition has the value 0. Note that there is a local *maximum* between the two points of value 9, but that it has measure zero. Values of $\mathrm{NF}(x)$ between vertices are approximated by linear interpolation.

4.2.4 Controller Architecture

We propose two alternative algorithms, where each agent computes a subset of a Voronoi region and then moves towards the centroid of this subset. This lends itself well to a hierarchical implementation, where a higher-level controller computes regions and target waypoints for the simplified agent kinematics described above, while a lower-level controller has the task of driving a possibly non-holonomic platform such as a car or a differential drive robot towards the waypoint. Since Voronoi regions are disjoint by construction, the lower-level controller can be allowed to maneuver anywhere inside the region to reach the waypoint, without risking collisions with other agents. Such an architecture is illustrated in Figure 4.4.

In the following sections, descriptions of two different versions of the algorithm are provided.

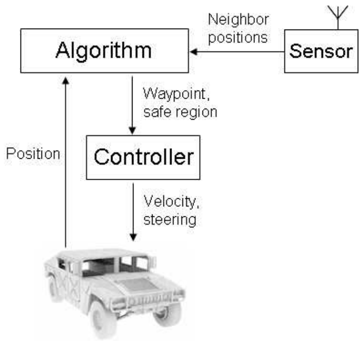

Figure 4.4: The proposed hierarchical architecture where the algorithm for holonomic agents feeds waypoints and safe regions to the lower-level controller that drives a non-holonomic car.

4.3 Flocking

4.3.1 Problem Formulation

The problem is to find a local control law for each agent i, based on its own position and that of its sensed neighbors:

$$u_i = f(q_i, Q),\ Q = \bigcup_{j \in \mathcal{N}_i} q_j$$

so that we achieve flocking while avoiding obstacles and converging to a global goal. As described in Section 2.3, flocking is sometimes defined as velocity alignment, which then implies bounded distances between all agents. Here, we have instead chosen to define flocking directly as maintaining bounded inter-agent distances, and we consider goal convergence in a loose sense, namely that all agents should reach within a given nonzero distance of the goal.

4.3.2 Algorithm

The proposed flocking algorithm is based on computing the Voronoi region of each agent, then moving towards the weighted centroid of the region. The centroid c of the region W, weighted by the navigation function, is computed as a surface integral:

$$c(W) = \frac{1}{m(W)} \int_W x\ \phi(x)\ dx, \tag{4.3}$$

where

$$m(W) = \int_W \phi(x)\ dx$$

and

$$\phi(x) = e^{-k_\phi \cdot \mathrm{NF}(x)}.$$

The exponential scaling of the navigation function is done to ensure that the velocity of the group is independent of the distance to the goal. The influence of the design parameter k_ϕ will be discussed in Section 4.3.4.

To ensure bounded size of the Voronoi region, an agent that is outside the convex hull of its sensed neighbors will mirror all of them in its own position, as described above. Further, we want to ensure that there are no collisions with other agents beyond sensor range or with obstacles. Therefore the Voronoi region is intersected with two other sets:

- L_i, consisting of all points $x \in \Omega$ such that the line between x and q_i does not intersect any obstacles. This means that there is a line of sight from agent i to all points in L_i.
- $S_i = \{r : ||r - q_i|| < R_{max}/2\}$, which contains all points seen by the onboard sensor.

Algorithm 2 Voronoi Flocking

```
1: Set $d$, $k_\phi$ and $\epsilon > 0$
2: loop
3:    Sense position $q_i$
4:    Sense neighbors $\mathcal{N}_i$
5:    Let $Q = \{q_j : j \in \mathcal{N}_i\}$
6:    if $q_i \in \text{convhull}(Q)$ then
7:       $V_i := V_i(\mathcal{N}_i)$
8:    else
9:       Mirror neighbors: $V_i := V_i(\mathcal{N}_i \bigcup M_d(\mathcal{N}_i, i))$
10:   end if
11:   Compute safe region: $W = V_i \bigcap L_i \bigcap S_i$
12:   Find centroid $c(W)$ as in (4.3).
13:   Compute $q_i'$ as
```

$$\min_{q_i'} \quad ||q_i' - c(W)||^2, \tag{4.4}$$

$$\text{s.t.} \quad \text{NF}(q_i') < \text{NF}(q_i) - \epsilon, \tag{4.5}$$

$$q_i' \in V_i \text{ (Voronoi region)} \tag{4.6}$$

$$q_i' \in L_i \text{ (Line of sight)} \tag{4.7}$$

$$q_i' \in S_i \text{ (Limited sensor range)} \tag{4.8}$$

$$||q_i' - q_i|| < u_{max}. \tag{4.9}$$

```
      If there is no feasible solution, set $q_i' = q_i$.
14:   Apply control $u_i = q_i' - q_i$
15: end loop
```

These properties will be considered in more depth below. Finally, we need to define the mirror distance d, the design parameter k_ϕ and a minimum step size $\epsilon > 0$. As will be discussed later, d should be chosen as the preferred inter-agent distance. The Voronoi Flocking Algorithm, as executed by agent i, is formally stated in Algorithm 2.

4.3.3 Properties

Here we give some theoretical results on the properties of the proposed algorithm. We consider the issues of safety, goal convergence and flocking. We also suggest some modifications that have not yet been integrated into the algorithm. As stated above, the highest priority has been given to safety so that neither the robots nor stationary objects in the surroundings will be damaged. This is guaranteed by the following proposition.

Proposition 4.3.1 (Safety). *If all agents follow the Voronoi Flocking Algorithm, there will be no collisions with obstacles or between agents.*

Proof. Agents closer than R_{max} will detect each other and not collide, since the interiors of their Voronoi regions are disjoint by construction. Two agents farther apart than R_{max} cannot collide since the constraint (4.8) restricts the step size to $R_{max}/2$. Because of constraint (4.7), there will be no collisions with obstacles. □

Ideally, we would like to prove that all robots will reach a set of equilibrium positions within a given distance of the goal. The following proposition contains a partial result.

Proposition 4.3.2 (Goal convergence). *Under the Voronoi Flocking Algorithm, agents will move towards the goal until they reach a configuration such that the optimization problem (4.4)–(4.9) has no feasible solution for any agent.*

Proof. This follows from condition (4.5) and the fact that $\mathrm{NF}(x) \geq 0$, with equality only at the goal. □

Remark: It is possible to construct situations when agents get stuck in dead-locks, which has never happened in simulations. This indicates that the dead-locks are not asymptotically stable, as discussed in more detail later.

Proposition 4.3.3 (Flocking). *Under the Voronoi Flocking Algorithm, for any two agents i and j and any time k_0 it holds that*

$$||q_i[k] - q_j[k]|| \leq \max_{n \in \mathcal{I}} 2NF(q_n[k_0]) \ \forall \ k \geq k_0.$$

Proof. Let g be the position of the goal. Then by definition of the navigation function, at time k_0 and for any agent i,

$$\max_{n \in \mathcal{I}} \mathrm{NF}(q_n[k_0]) \geq \mathrm{NF}(q_i[k_0]) \geq ||q_i[k_0] - g||.$$

Using the triangle inequality, we can show that for any two agents i and j,

$$||q_i[k_0] - q_j[k_0]|| \leq ||q_i[k_0] - g|| + ||q_j[k_0] - g||$$

and thus

$$||q_i[k_0] - q_j[k_0]|| \leq \max_{n \in \mathcal{I}} 2\mathrm{NF}(q_n[k_0]).$$

Because of constraint (4.5), NF is non-increasing so the inequality holds for any $k \geq k_0$. The result is not affected by the use of a grid to compute the navigation function, since NF is then based on the Manhattan distance, which is an overestimate of the Euclidean distance. □

Remark: The result can also be formulated as an invariant set

$$S[k] = \{x \in \mathbb{R}^2 : \mathrm{NF}(x) \leq \max_{i \in \mathcal{I}} \mathrm{NF}(q_i[k])\}$$

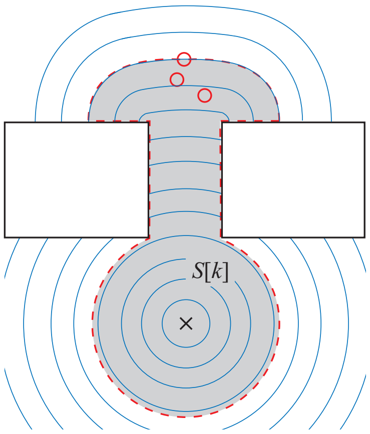

Figure 4.5: An example of how the invariant set is restricted if there are obstacles. Three agents (small circles) need to pass an opening to reach the goal (marked by a cross). The fine lines are level curves for NF, and the shaded area is the invariant set $S[k]$, that none of the agents will leave.

where all agents will remain. In an open field (for an infinitely dense grid), $S[k]$ will be circular around the goal and shrink over time. In an environment with obstacles, the actual invariant set will be more restricted since we can remove all parts of $S[k]$ that agents cannot reach without increasing their value of NF. An example of this is illustrated in Figure 4.5.

Constructing Dead-Locks

Under very special conditions, agents following the Voronoi Flocking Algorithm may find themselves in situations where they have not reached the goal but still cannot find feasible points to go to. Two examples of such situations are depicted in Figure 4.6. This has never led to a permanent blockings in the simulations, as round-off errors and other numerical effects have eventually moved the agents enough to resolve the blocking.

Even though the blockings do not seem to be asymptotically stable, they still delay the goal convergence considerably. This is most clearly seen in the testbed described in Section 4.5, where many agents are to pass a narrow passage. We have therefore devised a modification of the algorithm that will allow robots to detect a blocking and give way to others. The symmetry can easily be broken by allowing individual ID numbers, assigned to every agent, and communication between agents.

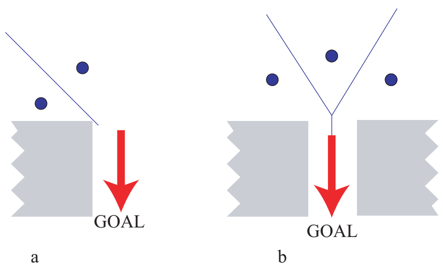

Figure 4.6: Two examples of situations when several agents following Algorithm 2 are blocking each other. The lines depict the boundaries of their Voronoi regions and the arrow denotes the direction towards the goal. In a, if both agents have a NF value that is less than ϵ greater than that at the corner, they will both stop moving. In b, the situation will be resolved eventually, but it makes goal convergence very slow.

The suggested modification can be described as:

- Remove constraint (4.5).
- If an agent takes a step in a direction where NF increases *and* there is at least one neighbor within its sensing radius that has a lower ID, it signals that it will stand still during a predefined period of time (typically one iteration).
- During this time, all surrounding agents can intrude on its space by calculating *asymmetric* Voronoi regions: Instead of making boundary lines half-way between the two agents, they can for example use 95% of the distance and leave 5% to the agent giving way.

The reason to take a step *away* from the goal is typically that there is a congestion ahead, so the above criterion should be able to detect that. Simulations have showed promising results, where a group of agents typically approach a narrow passage, take one step back and then stand still with the exception of the one with the lowest ID. Then all others start moving again, back away and stand still to let the agent with the next lowest ID through, and so on. We have made preliminary simulations of this algorithm under the same conditions as described in Figure 4.7 and it decreased the number of iterations required for goal convergence approximately from 500 to 200. Developing such methods of yielding further could be useful if one would like to implement the algorithm in environments where narrow passages are common.

4.3.4 Simulation

In this section we present some results from computer simulations, performed in Matlab. A group of 20 robots is simulated in two different settings. First we study an environment with irregular and non convex obstacles to underline the advantage of not having to make geometric assumptions about obstacle shape. We then test the same group in an open field to explore how the group attains a formation when moving over open fields.

Figure 4.7 shows four snapshots of a group of 20 agents, moving from the starting point in the lower right corner of the area to the goal in the upper right part, marked by an x. The agent positions are depicted by stars for the first and third snapshot and by dots for the second and fourth snapshot. The agents first perform a split/rejoin maneuver, then squeeze the formation when passing the corridor and finally gather around the goal. Due to the constraint $\mathrm{NF}(k+1) < \mathrm{NF}(k) - \epsilon$, they are distributed only in the third quadrant around the goal. In Figure 4.7 the preferred inter-agent distance is $d = 1$ and the maximum sensing radius is $R_{max} = 3$. The parameter k_ϕ is chosen to 1.

Figure 4.8 shows the same setting, but with $k_\phi = 0$. This removes the influence of NF on c and the only thing driving the agents towards the goal is constraint (4.5). This causes the agents to move much slower, but as can be seen in the figure, the spacing during the corridor traversal is somewhat wider. The snapshots are not necessarily taken at the same time instances in any of the Matlab plots.

To explore the flocking behavior in detail, two simulations are run in an open area with no obstacles. Figure 4.9 shows how the agents attain an almost perfect hexagonal lattice. The settings are $k_\phi = 1$, $d = 2$ and $R_{max} = 3$. This was expected because of the mirror neighbor mechanism, but not guaranteed. The distance between agents agrees well with the preferred distance, d.

Surprisingly enough, setting $k_\phi = 0$ gives less group cohesion, as seen in Figure 4.10. The transversal inter-agent distances are fairly correct, but the flock seems to have drifted apart in the direction of motion. The explanation is to be found in constraint (4.5). If k_ϕ is high enough, (4.5) will be satisfied by c itself, and the resulting behavior will be as if the constraint were not there. If, however k_ϕ is low enough, as in the $k_\phi = 0$ case, the agents would stand still in perfect formation if it were not for constraint (4.5). It forces the agents to move in a direction other than c, thus obstructing the formation maintenance.

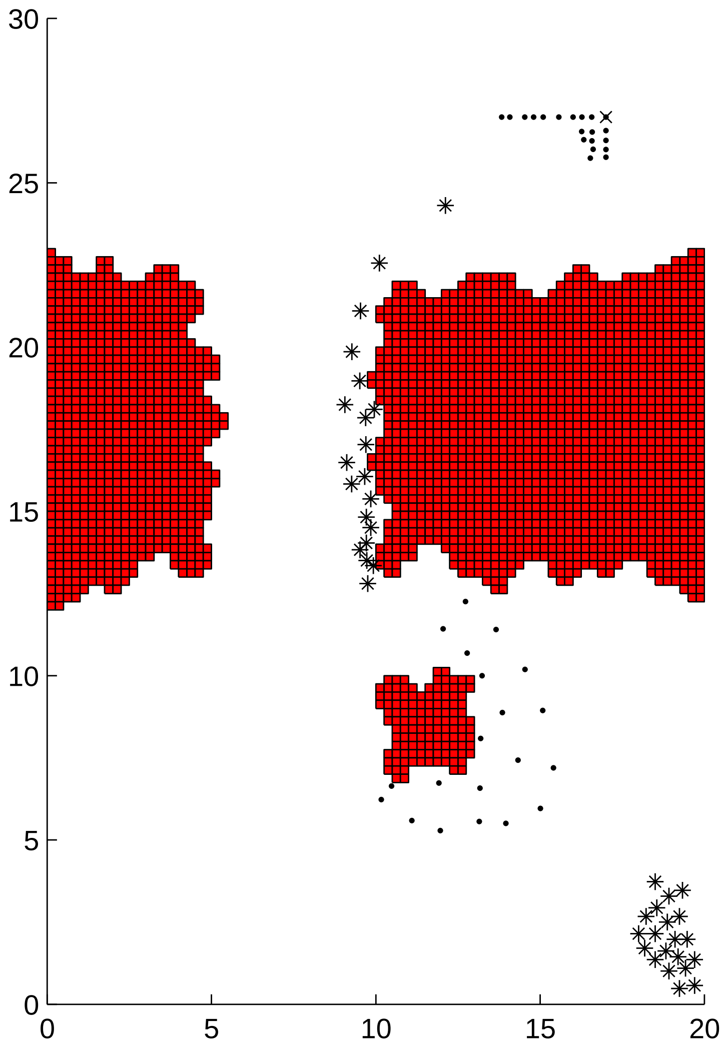

Figure 4.7: A group of 20 agents moving around irregular obstacles. The agents are depicted by stars for the first and third snapshot and dots for the second and fourth. The parameters are $R_{max} = 3$, $d = 1$ and $k_\phi = 1$. This gave the fastest goal convergence, at the expense of flock spacing in the corridor between the obstacles.

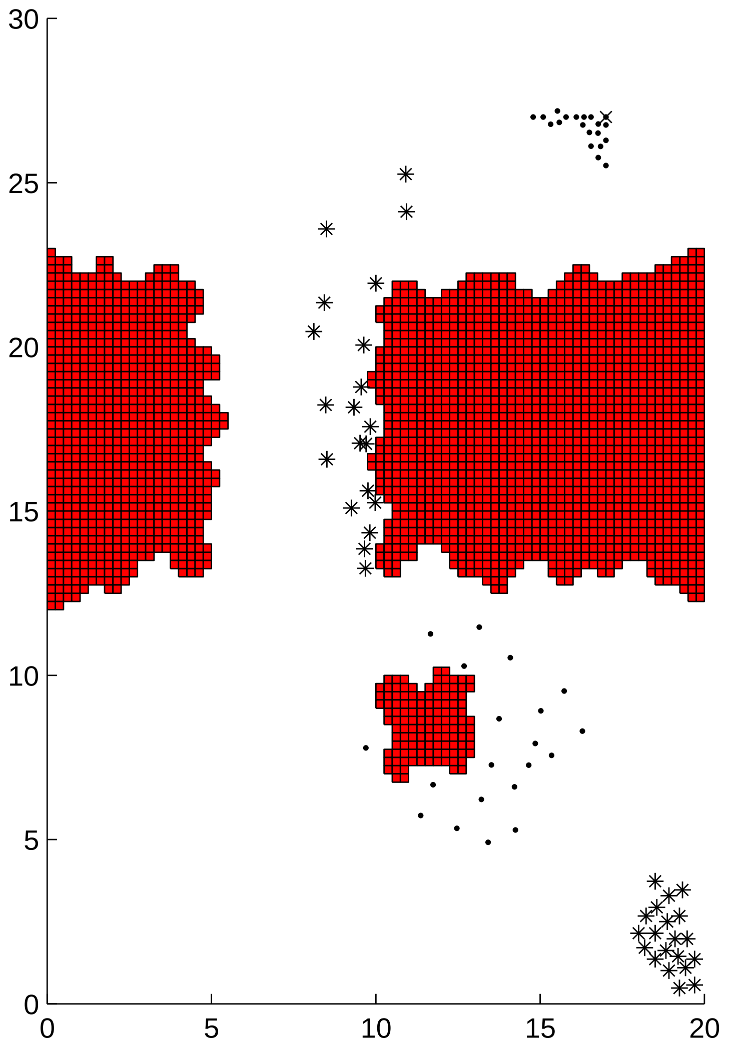

Figure 4.8: A group of 20 agents moving around irregular obstacles. The agents are depicted by stars for the first and third snapshot and dots for the second and fourth. The parameters are $R_{max} = 3$, $d = 1$ and $k_\phi = 0$. The group takes longer to reach the goal, but the inter-agent spacing is more appealing.

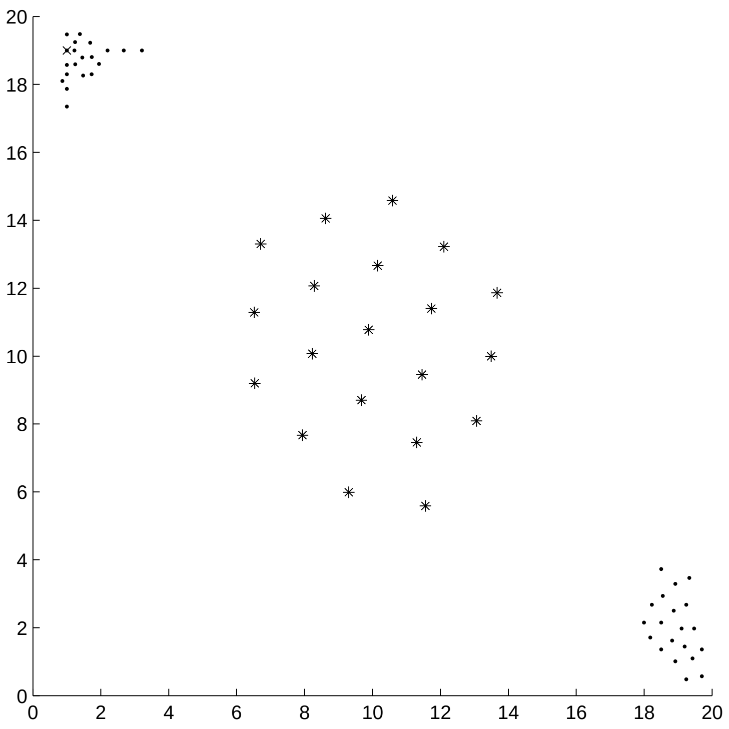

Figure 4.9: A group of 20 agents moving in a free field. The agents are depicted by dots for the first and third snapshots, and by stars for the second. The parameters are $R_{max} = 3$, $d = 2$ and $k_\phi = 1$. The group assumes an almost perfect hexagonal lattice formation with distance d between agents.

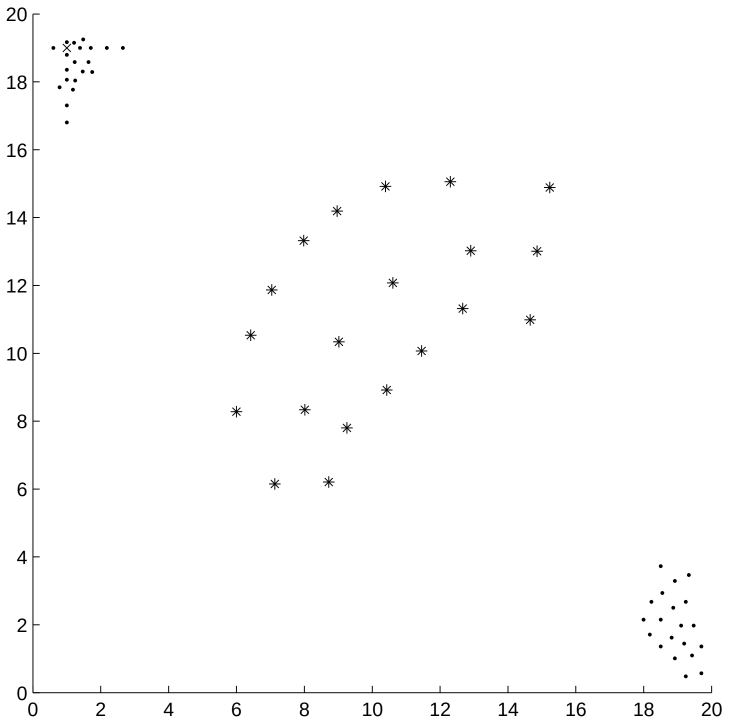

Figure 4.10: A group of 20 agents moving in a free field. The agents are depicted by dots for the first and third snapshots, and by stars for the second. The parameters are $R_{max} = 3$, $d = 2$ and $k_{\phi} = 0$. Due to the requirement that NF has to decrease monotonically, the hexagonal lattice formation is disturbed.

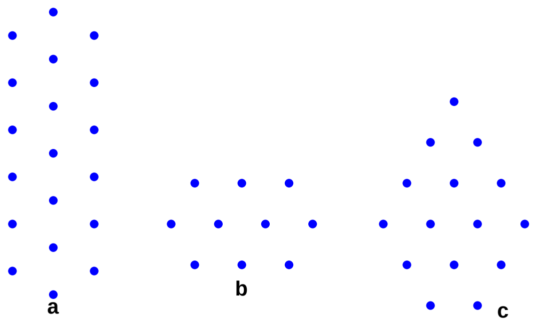

Figure 4.11: Formations a and b fulfill the definition of hexagonal lattice formations, while formation c does not, since the top agent has only two neighbors.

4.4 Formation Control

4.4.1 Problem Formulation

The previously presented Algorithm 2 was designed to achieve flocking combined with obstacle avoidance, but also displayed an appealing formation behavior when the agents were moving over open fields. We were, however, unable to formally prove that this formation was stable. We therefore formulated a formation control problem, to specifically study this emergent formation behavior. We first define the desired formation:

Definition 5 (Hexagonal lattice formation)**.** A hexagonal lattice formation is a set of agent positions such that each agent has 3, 4 or 6 Voronoi neighbors, all at the inter-agent distance d.

It is worth noting that this does not allow any subsets of an infinite hexagonal lattice, but also requires a certain compactness of the formation. Examples of configurations that do and do not fulfill the definition are given in Figure 4.11.

The problem is now to find a local control law for each agent i

$$u_i = f(q_i, Q), \; Q = \bigcup_{j \in \mathcal{N}_i} q_j$$

so that the agents form an asymptotically stable hexagonal lattice formation while moving along the *constant* negative gradient of a navigation function. This corresponds to how the navigation function behaves in an open field, far from the goal or any obstacles.

When considering stability, we do not care about translation or rotations of the group as a whole. Instead we seek to stabilize the relative positions between agents. To define this notion of stability, we first need the state vector (also called a formation)

$$\mathbf{x} = \begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & \dots & x_N & y_N \end{bmatrix}^T \in \mathbb{R}^{2N}.$$

Denote a ball around $\mathbf{x}$ as

$$B_\epsilon(\mathbf{x}) = \{\mathbf{x}' : \sqrt{(x_m - x'_m)^2 + (y_m - y'_m)^2} < \epsilon, m = 1, 2, \dots, N\}.$$

Also let

$$R(\varphi) = I_K \otimes \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$$

(where $\otimes$ denotes the Kronecker product) and $\mathbf{b} \in \mathbb{R}^2$ be an arbitrary translation. We can then define the stability we want to achieve:

Definition 6 (Formation stability)**.** A formation $\mathbf{x}$ is locally stable if

$$\begin{aligned} &\forall\, \epsilon > 0\ \exists\, \delta > 0 \text{ s.t. } \mathbf{x}(0) \in B_\delta(\mathbf{x}) \\ &\Rightarrow \forall\, t > 0\ \exists\, \mathbf{b}(t), \varphi(t) : \mathbf{x}(t) \in B_\epsilon(R(\varphi)\mathbf{x} + \mathbf{1} \otimes \mathbf{b}). \end{aligned} \tag{4.10}$$

It is locally *asymptotically* stable if it is locally stable and

$$\begin{aligned} &\exists\, \delta > 0 \text{ s.t. } \mathbf{x}(0) \in B_\delta(\mathbf{x}) \Rightarrow \\ &\Rightarrow \exists\, \mathbf{b}, \varphi : \lim_{t \to \infty} \mathbf{x}(t) = R(\varphi)\mathbf{x} + \mathbf{1} \otimes \mathbf{b}. \end{aligned}$$

4.4.2 Algorithm

To facilitate analysis, this approach differs from the Voronoi Flocking Algorithm in that each agent now moves to the centroid of *the vertices* of the Voronoi region, weighted by a scalar function ϕ. This can be computed as a sum instead of the integral (4.3):

$$c(X_i) = \frac{1}{6} \left(\sum_{j \in X_i} \phi(q_j) \right)^{-1} \sum_{j \in X_i} q_j\ \phi(q_j), \tag{4.11}$$

where

$$\phi(q_j) = e^{-k_\phi \cdot \mathrm{NF}(q_j)}.$$

Algorithm 3 Voronoi Formation Control

```
1: Set $d$ and $k_\phi$
2: loop
3:     Sense position $q_i$
4:     Sense neighbor sets $C_i$ and $D_i$
5:     if $|D_i| = 6$ then
6:         $X_i := X_i(D_i)$
7:     else if $|D_i| = 4$ then
8:         $X_i := X_i(D_i \bigcup M_d(C_i, i))$
9:     else if $|D_i| = 3$ then
10:        $X_i := X_i(D_i \bigcup M_d(D_i, i))$
11:    end if
12:    Let $q_i' := c(X_i)$
13:    Apply control $u_i = q_i' - q_i$
14: end loop
```

This simplification comes at a prize: To avoid unbalance in the number of vertices on each side of an agent, it must select more carefully what neighbors to mirror. The algorithm must be started in a configuration where each agent has four, five or six neighbors (but at varying distances), as described in the analysis section below. If an agent has four neighbors, it mirrors its Voronoi neighbors and if it has only three neighbors, it instead mirrors its Delaunay neighbors. This is formally stated in Algorithm 3.

4.4.3 Properties

In this section we show collision safety by using the convexity of the Voronoi regions. We then demonstrate that hexagonal lattice formations are asymptotically stable. Finally we present numerical arguments that indicate the size of the region of attraction around the equilibrium.

Safety

Theorem 4.4.1 (Collision safety)**.** An agent moving on a straight line towards the centroid of its Voronoi region will not collide with any other agents.

Proof. We show that the definition of Voronoi regions yields convex regions, and then show that the centroid will be inside this convex region. A straight line from the agent's present position to the centroid will then be contained in the region.

According to (4.1), x is inside $V_i(\mathcal{N}_i)$ if

$$|q_i - x| \leq |q_k - x| \;\forall\; k \in \mathcal{N}_i.$$

This is equivalent to the following condition

$$(q_k - q_i) \cdot x \leq \frac{q_k + q_i}{2} \cdot (q_k - q_i),$$

that can be written on the standard form for a (clearly convex) closed half-plane:

$$a^T x \leq b, \text{ where } a = (q_k - q_i) \text{ and } b = \frac{q_k + q_i}{2} \cdot a.$$

Now let $\{z_1, \ldots, z_m\}$ be an arbitrary set of points in $V_i(\mathcal{N}_i)$. Then any convex combination (*e.g.*, the centroid) of the points is also in the Voronoi region:

$$\sum_{n=1}^{m} \lambda_n = 1 \Rightarrow \sum_{n=1}^{m} \lambda_n z_n \ \in \ V_i(\mathcal{N}_i)$$

since

$$a^T \sum_{n=1}^{m} \lambda_n z_n = \sum_{n=1}^{m} \lambda_n a^T z_n \leq \sum_{n=1}^{m} \lambda_n b = b \sum_{n=1}^{m} \lambda_n = b.$$

The vertices $X_i(\mathcal{N}_i)$ are on the boundary of the Voronoi region. But except for in degenerate cases, at least one vertex, $v \in X_i(\mathcal{N}_i)$, is not on a line with all the others. The centroid will then be on the interior of a line between v and the centroid of all other vertices, which is in the interior of $V_i(\mathcal{N}_i)$.

The agent moves along the line between its previous position and the centroid, both of which are inside $V_i(\mathcal{N}_i)$. It then follows from the definition of convexity that the new point is inside the region. And since the interiors of Voronoi regions of different agents are disjoint by construction, no two agents will ever go to the same point. □

Remark 1: In practice, agents will have a limited sensor range R_{max}. But as in Algorithm 2, we can limit the step length to $R_{max}/2$, and so prevent collisions between agents that have not sensed each other.

Remark 2: The safety property also holds for an agent that does not go straight to the centroid, if it does not leave the Voronoi region or move farther than $R_{\max}/2$ in one iteration. This motivates the extension of our algorithm to produce waypoints for non-holonomic agents as described in Section 4.2.4.

Formation Stability

We next prove stability for a specific hexagonal lattice formation of agents. For this analysis we consider a group of 10 agents in a formation depicted in Figure 4.11b. This formation has been chosen because it is the smallest formation where all cardinalities of neighbor sets are represented. We assume $k_\phi = 0$, which corresponds to no net movement of the group.

In discrete time, under the Voronoi Formation Control Algorithm, the closed-loop dynamics of the multi-agent system are

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k]) = \frac{1}{6}\begin{bmatrix} v(1,2,5)+v(1,4,5)+v(1,2,\bar{4})+v(1,4,\bar{2})+v(1,\bar{2},\bar{5})+v(1,\bar{5},\bar{4}) \\ v(2,1,5)+v(2,5,6)+v(2,3,6)+v(2,1,\bar{6})+v(2,3,\bar{5})+v(2,\bar{5},\bar{6}) \\ v(3,2,6)+v(1,6,7)+v(3,2,\bar{7})+v(3,\bar{2},7)+v(3,\bar{6},\bar{7})+v(3,\bar{2},\bar{6}) \\ v(4,1,5)+v(4,5,8)+v(4,1,\bar{8})+v(4,\bar{1},8)+v(4,\bar{1},\bar{5})+v(4,\bar{5},\bar{8}) \\ v(5,1,2)+v(5,2,6)+v(5,6,9)+v(5,8,9)+v(5,4,8)+v(5,1,4) \\ v(6,2,3)+v(6,3,7)+v(6,7,10)+v(6,9,10)+v(6,5,9)+v(6,2,5) \\ v(7,3,6)+v(7,6,10)+v(7,3,1\bar{0})+v(7,\bar{3},10)+v(7,\bar{3},\bar{6})+v(7,\bar{6},1\bar{0}) \\ v(8,4,5)+v(8,5,9)+v(8,4,\bar{9})+v(8,\bar{4},9)+v(8,\bar{4},\bar{5},)+v(8,\bar{5},\bar{9}) \\ v(9,5,8)+v(9,5,6)+v(9,6,10)+v(9,\bar{5},10)+v(9,\bar{6},8)+v(9,\bar{5},\bar{6}) \\ v(10,6,9)+v(10,6,7)+v(10,7,\bar{9})+v(10,\bar{7},9)+v(10,\bar{6},\bar{9})+v(10,\bar{6},\bar{7}) \end{bmatrix}, \quad (4.12)$$

where we use the more compact notation $v(k,i,j)$ for the shared vertex of agents k, i and j:

$$v(k,i,j) = \begin{bmatrix} \frac{1}{2}\frac{(y_j-y_k)\left[x_i^2+y_i^2-(x_k^2+y_k^2)\right]+(y_k-y_i)\left[x_j^2+y_j^2-(x_k^2+y_k^2)\right]}{(x_i-x_k)(y_j-y_k)-(y_i-y_k)(x_j-x_k)} \\ \frac{1}{2}\frac{(x_k-x_j)\left[x_i^2+y_i^2-(x_k^2+y_k^2)\right]+(x_i-x_k)\left[x_j^2+y_j^2-(x_k^2+y_k^2)\right]}{(x_i-x_k)(y_j-y_k)-(y_i-y_k)(x_j-x_k)} \end{bmatrix} \quad (4.13)$$

A bar over the number of the second or third agent denotes the mirror operator, mirroring the agent in the position of the first agent. As an example,

$$v(k,\bar{i},j) = v(k, M_d(i,k), j),$$

using the mirror operator defined in (4.2).

Theorem 4.4.2 (Local asymptotic stability)**.** The formation depicted in Figure 4.11b is locally asymptotically stable under the Voronoi Formation Control Algorithm.

Proof. The agents are at positions

$$\mathbf{x}^* = \frac{1}{2}\left(1,\sqrt{3},3,\sqrt{3},5,\sqrt{3},0,0,2,0,4,0,6,0,1,-\sqrt{3},3,-\sqrt{3},5,-\sqrt{3}\right)^T.$$

We linearize the system (4.12) around the stationary point $\mathbf{x}^*$ and show that the eigenvalues of the Jacobian are all on or inside the unit circle. Finally we note that all eigenvectors corresponding to an eigenvalue on the unit circle describe rotations of the whole formation with preserved relative positions of the agents.

Using computer software for symbolic algebra, we find that the Jacobian

$$J = \frac{df}{d\mathbf{x}}(\mathbf{x}^*)$$

has two unit eigenvalues and that all others are inside the unit circle. The eigenvectors with unit eigenvalues correspond to rotations of the whole formation (clockwise and counterclockwise). Under these perturbations, all relative positions of the agents are preserved and, according to (4.10), this does not affect the asymptotic stability of the formation. □

Region of Attraction

To further test the stability properties of our proposed algorithm, we investigate what happens if we perturb the position of one single agent in a hexagonal lattice where all other agents are fix. We then let the agent move according to one iteration of the algorithm and plot the direction of its movement as a function of the perturbation. This yields a vector field that indicates what magnitude of perturbations the algorithm can handle.

We only study one iteration since in a real situation, where all agents move, in the next iteration all surrounding agents will have moved too, so the simplification does not hold anymore. But if the perturbed agent has then taken a step towards the origin, it is plausible that the whole formation converges (which is also confirmed by simulations in Section 4.4.4).

To simplify the calculations, we translate the agent (number 7) to the origin, surrounded by six neighbors:

$$\begin{aligned} q_1 = -q_4 &= (1,0)^T \\ q_2 = -q_5 &= \frac{1}{2}(1,\sqrt{3})^T \\ q_3 = -q_6 &= \frac{1}{2}(-1,\sqrt{3})^T \end{aligned}$$

The simplification of considering only six neighbors holds for perturbations of magnitude less than $1/\sqrt{3}$. This is because the Voronoi region of the center agent will not be affected by any of the other agents in the lattice. We also assume $k_\phi = 0$.

If the center agent is perturbed to the position $q_7 = (x_7, y_7)^T$, after one iteration of the algorithm, it will be at $q_7' = (x_7', y_7')^T$. The direction of the step $u_7 = q_7' - q_7$ as a function of q_7 can be plotted as a vector field, depicted in Figure 4.12. A circle with radius $1/\sqrt{3}$ shows the region where the assumption of six neighbors is valid. Within this circle, the vector field indicates that the algorithm should be convergent. So a conservative estimate of the region of attraction appears to be a circle of radius $1/\sqrt{3}$ centered around the perturbed agent, something that is also verified by simulations in the following section. This also means that one single agent may only be displaced by this much from an ideal hexagonal lattice formation when the algorithm is initiated. Otherwise the group may not converge to the desired formation.

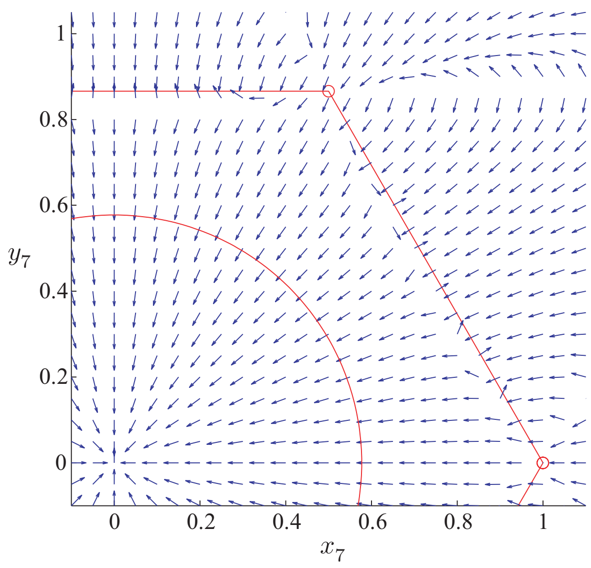

Figure 4.12: A vector field with normalized vector lengths, showing the direction of u_7 as a function of q_7. The circle shows the region where only the six closest neighbors affect the Voronoi region of the center agent. Within this circle, the vector field appears to be convergent. The smaller circles joined by lines are the neighboring agents.

4.4.4 Simulation

We simulate a larger group of agents that start in a hexagonal lattice formation and all evolve according to the algorithm. Three scenarios are studied: First one single agent in a stationary formation is perturbed, to test stability. Then we move the whole formation while adding noise to the control signal to each agent, to simulate the effects of uneven terrain and platform imperfections. Finally we apply the algorithm to a group of agents with car-like kinematics, to illustrate the usefulness of the hierarchical controller architecture.

By perturbing one of the agents in a stationary formation (where the weight ϕ is made constant by setting $k_\phi = 0$), we can study the asymptotic stability. Figure 4.13 depicts such a formation where the middle agent is perturbed by $0.5d$, which is attenuated in a few iteration steps. The original position of the agent is shown by a dashed circle. The maximum perturbation that can be attenuated even in the most sensitive direction is found to be $0.57d \approx 1/\sqrt{3}$. This concurs well with the results in Section 4.4.3.

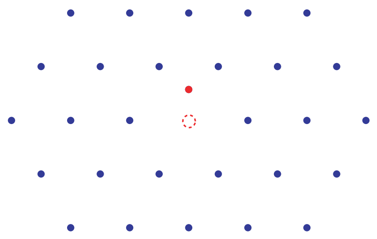

Figure 4.13: A formation where the middle agent has been displaced the distance $0.5d$ in the most sensitive direction. The dashed circle shows its original position. This perturbation is attenuated in a few steps.

In the second simulation we control the global movement of a formation by changing the slope of the weight function. We use the weight

$$\phi(x, y) = e^{x \cos \alpha + y \sin \alpha}, \tag{4.14}$$

where we can vary the angle α over time to make the group move in different directions. We have chosen the exponential function to get uniform speed of movement over the whole plane. The result in Figure 4.14 shows snapshots of the formation being steered around obstacles. We add noise to the control signal of each agent to simulate uneven terrain and other errors:

$$u_i = c - q_i + \begin{bmatrix} \xi_x \\ \xi_y \end{bmatrix} \tag{4.15}$$

The random variables ξ_x, ξ_y are uniformly distributed in the interval $[0, 0.15]$.

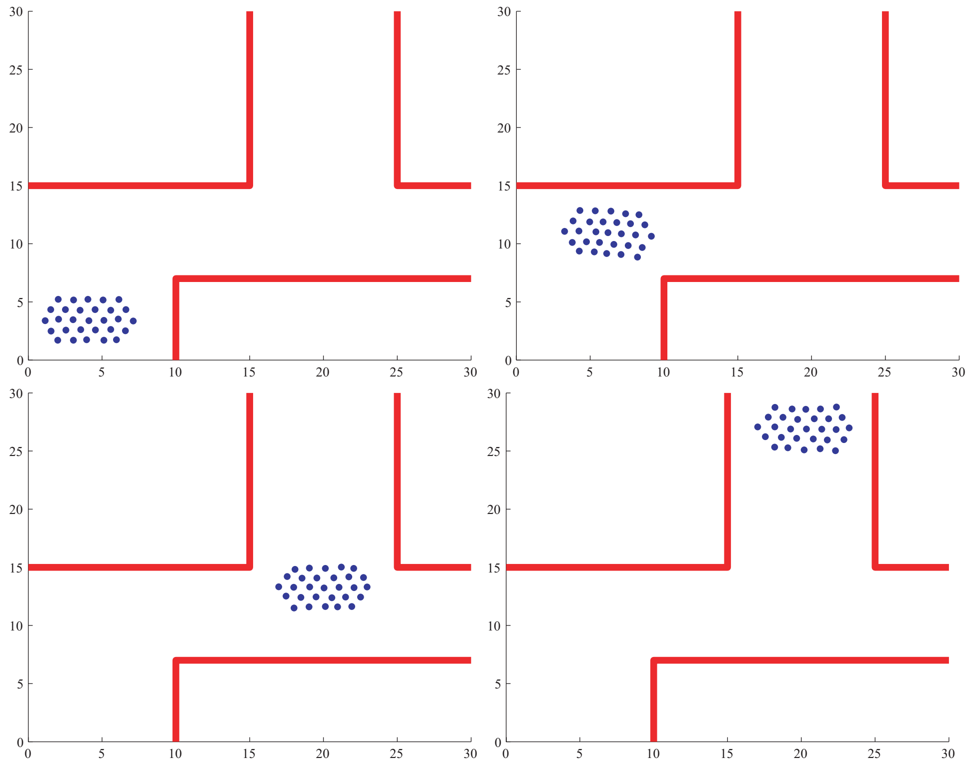

Figure 4.14: Snapshots of a group of 29 agents moving according to a linear weight function whose slope is changed to control the direction of movements. The whole sequence takes 160 iterations.

Finally we study a more realistic model, a kinematic car (Murray and Sastry, 1993), with the forward velocity v_i and steering angle δ_i as controls. The distance between back and front wheels is denoted L:

$$\begin{array}{rcl} \dot{x}_i & = & v_i \cos\theta_i \\ \dot{y}_i & = & v_i \sin\theta_i \\ \dot{\theta}_i & = & \frac{v_i}{L}\tan\delta_i \end{array} \qquad (4.16)$$

The model is depicted in Figure 4.15, that also shows the curvature r. The curvature is bounded below by

$$r_{min} = \frac{L}{\tan\delta_{max}}$$

where δ_{max} is the maximum steering angle. As described in Section 4.2.4, each car has a controller capable of driving the car between waypoints, while respecting the safe regions designated by our higher-level algorithm.

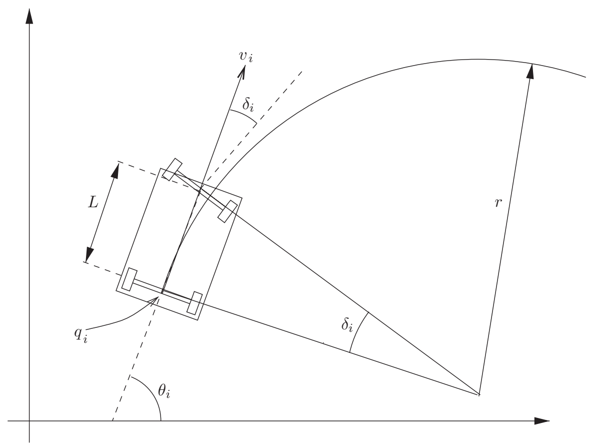

Figure 4.15: The kinematic car.

A formation of 13 kinematic cars is steered using the weight (4.14) in a labyrinth of 100×100 m^2, with the inter-agent distance d=5 m. Since each car must not leave its Voronoi region, they must sometimes perform parallel parking-like maneuvers to turn towards the next waypoint produced by our algorithm. Figure 4.16 shows snapshots of the group at four different time instances. In this case too, we add noise (4.15) to the waypoints given to the lower-level car controllers. With a maximum car speed of 1 m/s, the whole trajectory is completed in about 2.5 minutes. These simulations illustrate the validity of the stability analysis and the feasibility of structuring the control of non-holonomic agents in two levels, where the top level algorithm produces waypoints and associated safe regions for maneuvering.

4.5 Experimental Results from Virtual Testbed

As a complement to testing our algorithm in Matlab, we have performed more realistic simulations at the Swedish Defence Research Agency, FOI. FOI has a system of radio-controlled cars that have the same application interface as a simulated vehicle, running in the flight simulator Fenix (Swedish Defence Research Institute, 2007). The simulator has a realistic physics engine that can also be used to simulate the behavior of ground vehicles. This section provides a system overview, a description of the vehicles and their controllers and some results.

4.5.1 System Overview

The system is designed so that the interface between controller and vehicle is the same in Fenix as in a corresponding system of redesigned radio-controlled cars. One of the cars is depicted in Figure 4.17. Every car is equipped with a card-PC,

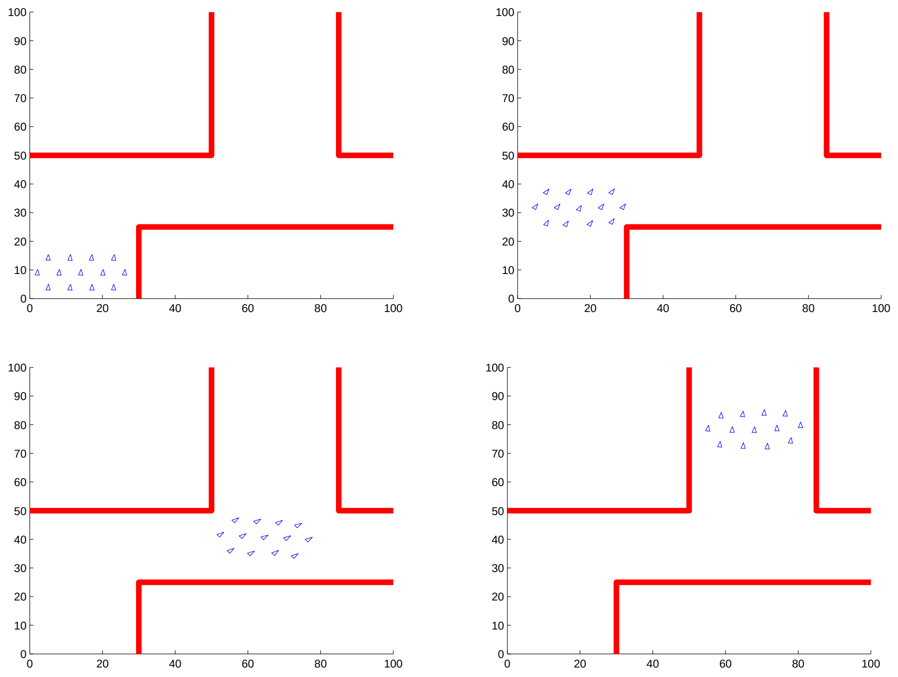

Figure 4.16: Snapshots of a formation of 13 robots with car-like kinematics moving through a simple labyrinth. The labyrinth is square, with side length 100 m.

actuators for thrust (and thus braking) and steering, a WLAN transceiver for short-range communication and a GPS receiver for navigation. A side-effect of having a GPS receiver is that it provides access to a very accurate time estimate. This is used to synchronize all cars and make them plan their next step at the same time. This is crucial for avoiding ambiguities in the Voronoi partitioning of the available space. At the time of the experiments only two cars were operational, and it was thus decided to use the simulator to demonstrate the group performance. In the simulator, each car is modelled as a US Army HMMWV jeep, with realistic suspension dynamics and possible slipping of the wheels.

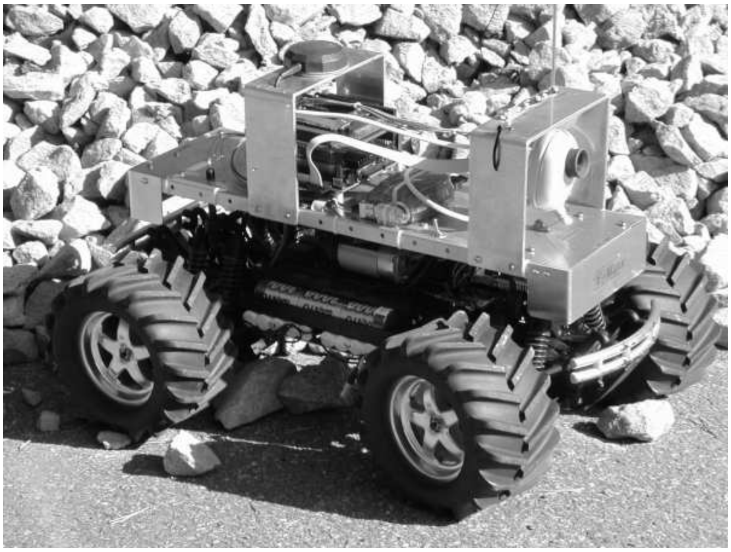

Figure 4.17: One of the radio-controlled cars owned by the Swedish Defence Research Agency.

4.5.2 Car Dynamics and Controller

The low-level controller has the following specifications:

- It should drive the car from an arbitrary position and orientation to the waypoint specified by Algorithm 2.
- The orientation of the car when reaching the waypoint is not important. [1]
- The straight line from the initial position to the waypoint is guaranteed to be obstacle-free, but if the car has to maneuver it must not leave its safe region W_i.

In these simulations we used a dynamic car model, which is the same as in (4.16), but where the velocity v_i cannot be controlled directly. Instead we can only control the acceleration: $a_i = \dot{v}_i$.

The shortest paths for a car with bounded curvature consists of circular arcs of minimum turning radius, connected with straight line segments (Dubins, 1957). Therefore, we used a simple controller that starts by turning towards the waypoint with a minimum turning radius and a constant speed, chosen not to give slipping,

[1] Since the controller does not know in what direction the car will have to go in the next iteration, we decided not to put any constraints on the final orientation of the car. Instead we prioritized to quickly reach the target.

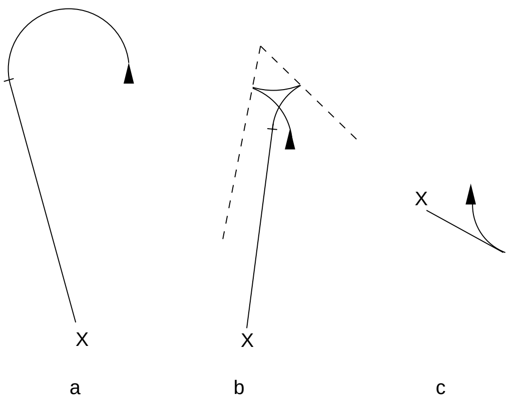

Figure 4.18: Three representative trajectories of the car controller. The triangle shows the initial position and orientation of the car, and the x is the waypoint location. In a, the car has sufficient space to turn, while in b, the dashed line shows the boundaries of the safe region that must not be passed. In c, the waypoint is too close, so the car reverses before driving straight ahead.

and then drives straight ahead at a higher speed until it reaches the waypoint. If the waypoint is too close, or if a path as described above would leave the safe region, the controller performs a simple parallel parking maneuver. Representative examples of paths are shown in Figure 4.18.

Figure 4.19 shows a map of the simulation environment, where the black areas are obstacles. We chose to use ditches, about 15 m deep, as obstacles instead of walls or hills for two reasons: First it allows overlooking the whole course and see all cars at once. Second, if a car hits a wall it might bounce off the obstacle without an observer noticing it, while if it falls into a ditch it stays there. The surface is modelled as uneven ground, which naturally adds some noise to the position and heading of each car.

4.5.3 Results

We have simulated two scenarios: First a group of ten cars was started at A, to investigate the obstacle avoidance properties. Then a larger group of 20 cars was started at B to study formation stability.

The group of ten cars reached the configuration depicted in Figure 4.20 after 2 min 45 s. There was a tendency of the cars blocking each other when driving onto the narrow bridge at C, but it never led to a locking situation with all cars standing still. There were no collisions even at the start, when the cars were lined up only ten meters from each other. The small-scale behavior of the controller worked well, carefully backing or turning away from the others. We tried starting the group from different positions around the world and the obstacle avoidance worked well, except

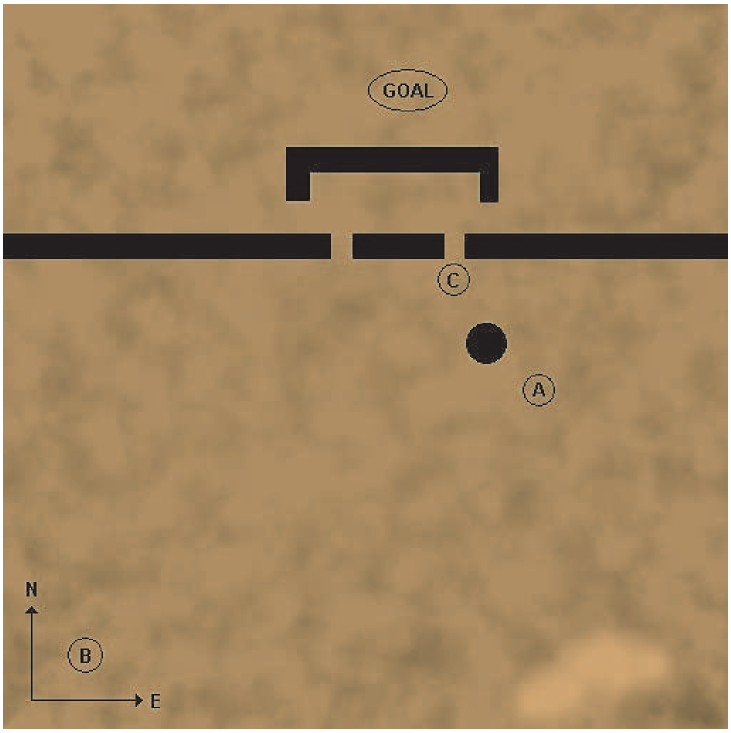

Figure 4.19: A height map of the simulated world that measures 2×2 km^2, with lighter areas being higher. The black areas are obstacles in the shape of ditches. The goal is indicated, as well as starting points A and B and the entrance of a narrow passage at C.

for some rare cases when the cars stood still for a long time waiting for their "turn" to pass a narrow bridge. When standing still the cars slide slowly to the side due to imperfections in the simulation engine, and this could lead to tipping over the edge into a ditch. This sliding mechanism has, on the other hand, also resolved some locking situations when several cars have blocked each other at narrow entrances.

When simulating the group of 20 cars, the computer could not run the simulation in real time due to the increased processing load. In a physical, truly distributed implementation this would not be a problem, as the only added complexity for the individual agent is that of computing more boundaries for the Voronoi region. It was discovered that the stability of the formations depended on the iteration interval of the algorithm. Long intervals gave the cars time to reach their waypoints between every iteration, so the algorithm worked much like in the simplified Matlab environment. As can be seen in Figure 4.21, the cars do not form a truly hexagonal lattice but stay well collected. The reasons that they do not form a lattice are prob-

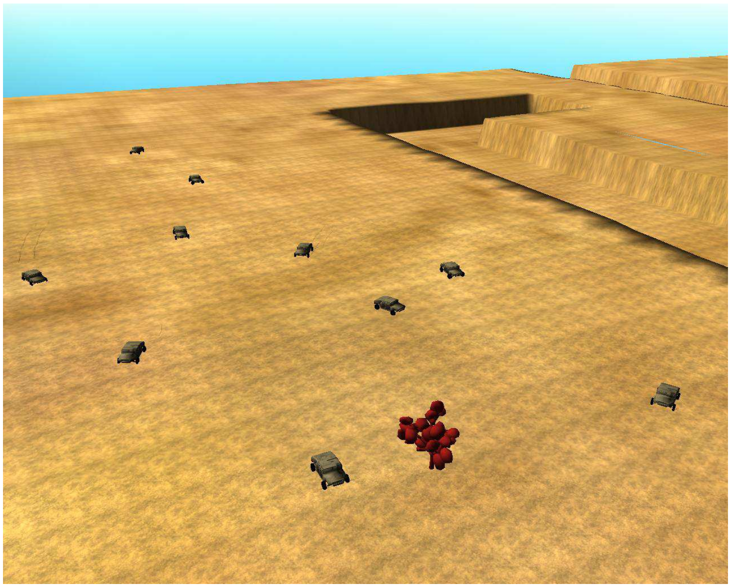

Figure 4.20: A screenshot from the simulation program showing ten cars at the goal, marked by a tree.

ably that the surface is uneven, which acts as a disturbance, and that robots on the side of the formation tend to zigzag ahead as described in Section 4.3.3. When the replanning interval was shortened the cars rarely made it to their waypoints before replanning. This smoothed the motion since they did not stop, but it disturbed the formations much like the case described in Section 4.3.4, as the cars did not really reach the centroids of their Voronoi regions.

4.6 Summary

As shown above, Voronoi regions offer a simple way to get collision safety. By weighting the regions with an appropriate scalar-valued function, a superimposed motion of the whole group can also be achieved. The navigation function is one such choice that yields attractive obstacle avoidance properties, regardless of obstacle geometry. The regions can be approximated using only local information and require only the position of neighbors, which can be found without explicit communication. A drawback is that while being geometrically simple, they yield nonlinear dynamics that are difficult to analyze for stability.

The main difference between the flocking and formation algorithms is that the former computes the centroid of the *surface* of the Voronoi region while the latter uses the *vertices* of the region, which replaces an integration with a summation

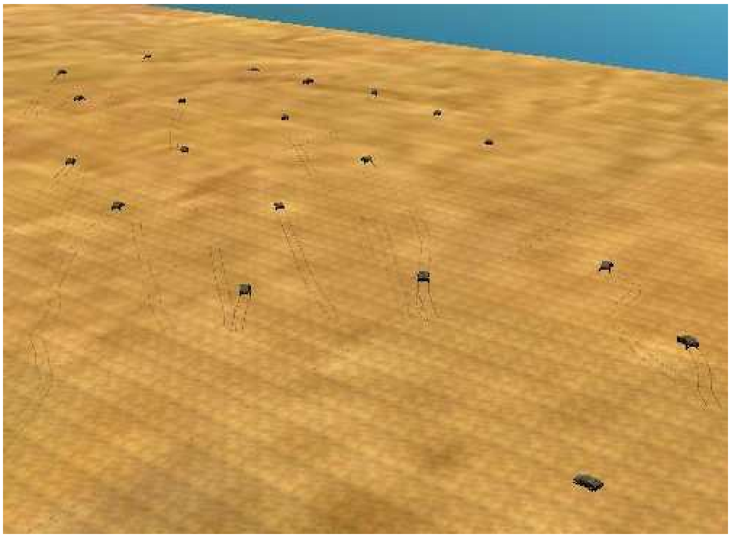

Figure 4.21: A screenshot from the simulation program, showing twenty cars moving in a loose formation. The goal is located outside the picture, far up to the left.

and thus reduces the computational load on each agent. It also facilitates finding a closed-form expression for the dynamics of the whole group. The drawback is that vertices can appear and disappear instantly due to very small agent movements, but the surface of the Voronoi region varies continuously with the movement of neighbors. This in turn affects the smoothness of the resulting trajectories. So the Voronoi Formation Control Algorithm requires a more careful selection of what neighbors will be used for the Voronoi region generation. It also requires that the agents are started in a configuration sufficiently close to the desired formation, as described in Section 4.4.3. If, due to disturbances or illegal initial conditions, any agents deviate outside the region of convergence, the whole formation diverges.

In summary, this means that the surface-based Voronoi Flocking Algorithm appears better suited for implementation, while the vertex-based algorithm is mostly of interest for analysis.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis investigates two problems that are relevant in the context of searching with multi-robot systems. The first problem is that of performing tasks under communication constraints, *i.e.*, while ensuring that the agents can still communicate within the group. The second problem is to transport a group of robots in an efficient manner by increasing the level of autonomy.

We have proposed two solutions for communication-aware task execution, both designed to exploit multipath fading. As pointed out earlier, they are useful in all environments where the signal strength varies over short distances, even if the reason is not multipath fading. It should also be clarified that both methods are most useful near the maximum range of the link, where the signal transitions from perfect to undetectable. The size of this region can be considerable and many network models tend to overlook it. Hopefully, our work can be one small contribution to finding efficient methods of dealing with this.

The proposed algorithms for transporting a group are based on Voronoi regions. They are well suited for decentralized computation, but even though the geometry is simple, the resulting system dynamics are difficult to analyze. Nevertheless, we have presented a version of the algorithm that is collision-free and yields flocking, *i.e.*, the group does not diverge. This is combined with a navigation function to make each agent move towards a common goal, avoiding obstacles. In open fields, the algorithm produces formations that appear to be asymptotically stable. To investigate this further, we formulated a second problem, using Voronoi vertices for formation control. This yielded asymptotically stable hexagonal lattice formations, and we have provided an estimate of the region of convergence for the equilibrium. The formation control algorithm is simpler to analyze, but due to its tendency to diverge for initial conditions that do not fall in the region of convergence, the flocking algorithm is probably better suited for practical implementation.

5.2 Future Work

In the future work, we plan to take a more holistic view on the integrated multi-agent coordination and inter-agent communication problem. By closer interaction between the mobility of the agents and the different layers in the communication stack, the agents can move in a communication-aware manner and the communication constraints can be adapted to facilitate planning. We have studied methods to exploit multipath fading for improving communication, but other effects such as shadowing, path loss and interference from others could also be incorporated to plan trajectories that allow communication. Uncertainties in the radio models could be compensated by applying feedback, which the approaches in Chapter 3 are examples of. In terms of the standard layered communication architecture, this can be regarded as closing the loop between robot mobility and the physical layer. It would be interesting to also interact with higher layers such as the network or link layers where, *e.g.*, a path planner could request a certain network routing to allow obstacle avoidance without loss of connectivity.

Coordination of multi-agent systems to avoid shadowing is essentially the same problem as maintaining a line of sight between selected agents, in the presence of obstacles. This, in turn, is very similar to the problem of visually tracking an evader or intruder, which is another form of communication, albeit involuntary from one of the parties. There is already some work done in this field, but not so much in combination with performing other tasks simultaneously. Formulating the line of sight constraint in a way that is compatible with standard path planning methods and studying feasibility over the whole execution of a task would be interesting directions of research. A starting point could be to investigate how to achieve a line of sight between selected pairs of agents in an environment where the obstacle positions are only partially known. This would allow the agents to plan approximate positions, which then need to be adjusted using feedback until all visibility constraints are fulfilled.

The future research will also contain experiments on physical robots, investigating the influence of the environment on the inter-agent communication. We plan to verify the hybrid control law derived in Chapter 3 in typical indoor environments. Further, we will implement part of the developed algorithms for coordination under communication constraints in a multi-robot demonstration as part of the project on autonomous UGVs, described in Section 1.1.1.

Bibliography

P. Antsaklis and J. Baillieul, editors. *IEEE Transaction on Automatic Control, special issue on networked control systems*, volume 49(9). 2004.

R. Arkin and T. Balch. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Proceedings of 7th International Workshop on Advanced Motion Control*, 2002.

R. C. Arkin. *Behavor-based robotics.* The MIT Press, 1998.

K. E. Årzén, A. Bicchi, G. Dini, S. Hailes, K. H. Johansson, J. Lygeros, and A. Tzes. A component-based approach to the design of networked control systems. *European Journal of Control*, 13(2-3), 2007.

K. J. Åström and B. Wittenmark. *Computer-controlled systems: theory and design.* Prentice-Hall, 1997.

T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

B. Burns, O. Brock, and B.N. Levine. Autonomous enhancement of disruption tolerant networks. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

Chipcon AS. CC2420 datasheet 1.4, 2006. URL `http://www.chipcon.com`. Last visited 2007-05-19.

T.H. Chung, J.W. Burdick, and R.M. Murray. A decentralized motion coordination strategy for dynamic target tracking. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

Cluster Project Team. The Cluster-II mission rising from the ashes. *ESA Bulletin*, 102:46–53, 2000.

J. Cortés, S. Martínez, T. Karataş, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

F. Darbari, I. McGregor, G. Whyte, R. W. Stewart, and I. Thayne. Channel estimation for short range wireless sensor network. In *Proceedings of the 2nd IEE/Eurasip Conference on DSP Enabled Radio*, 2005.

A. Drenner, I. Burtz, B. Kratochvil, B. J. Nelson, N. Papanikolopoulos, and K. B. Yesin. Communication and mobility enhancements to the scout robot. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and System*, 2002.

L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001.

J. M. Esposito and T. W. Dunbar. Maintaining wireless connectivity constraints for swarms in the presence of obstacles. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

European Space Agency. Formation flying at closest-ever separation, June 2007. URL `http://clusterlaunch.esa.int/science-e/www/object/index.cfm?fobjectid=41120`. Last visited 2007-10-01.

J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

P. Ögren and N. E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 21(2):188–195, 2005.

M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on networking*, 10(4):477–486, 2002.

S. Hailes. RUNES, middleware components and disaster relief scenario. <http://www.ist-runes.org/docs/presentations/ecc2.pdf>, July 2007. Presentation given at the European Control Conference.

T. Hales. An overview of the Kepler conjecture, 1998. URL `http://www.math.pitt.edu/~thales/kepler98/sphere0.ps`. Last visited 2007-10-10.

M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor. Towards the deployment of a mobile robot network with end-to-end performance guarantees. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.

A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

P. Karimian, R. Vaughan, and S. Brown. Sounds good: Simulation and evaluation of audio communication for multi-robot exploration. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2006.

I. D. Kelly and D. A. Keating. Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots. In *Proceedings of the Third International Conference on Mechatronics and Machine Vision in Practice*, 1996.

N. E. Leonard and E. Fiorelli. Virtual Leaders, Artificial Potentials and Coordinated Control of Groups. In *Proceedings of the IEEE Conference on Decision and Control*, 2001.

P. Lif, H. Jander, and J. Borgvall. Tactical evaluation of unmanned ground vehicle during a MOUT excersice (sic!). In *Proceedings of the Human factors and ergonomics society 50th annual meeting*, 2006.

B. Lincoln and A. Rantzer. Relaxing dynamic programming. *IEEE Transactions on Automatic Control*, 51(8):1249–1260, 2006.

M. Lindhé and K. H. Johansson. *Taming Heterogenity and Complexity of Embedded Control*, chapter A Formation Control Algorithm using Voronoi Regions, pages 419–434. ISTE Ltd, 2006.

M. Lindhé and K. H. Johansson. Communication-aware trajectory tracking. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008. Submitted.

M. Lindhé, P. Ögren, and K. H. Johansson. Flocking with obstacle avoidance: A new distributed coordination algorithm based on voronoi partitions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

M. Lindhé, K. H. Johansson, and A. Bicchi. An experimental study of exploiting multipath fading for robot communications. In *Proceedings of the Robotics: Science and Systems conference*, 2007.

N. Moshtagh, A. Jadbabaie, and K. Daniilidis. Vision-based Control Laws for Distributed Flocking of Nonholonomic Agents. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

R. Murray. Recent research in cooperative control of multi-vehicle systems. Submitted to ASME Journal of Dynamic Systems, Measurement and Control, 2006.

R. M. Murray and S. S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716, 1993.

H. G. Nguyen, N. Farrington, and N. Pezeshkian. Maintaining Communication Link for Tactical Ground Robots. In *AUVSI Unmanned Systems North America*, 2004.

R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95:215–233, 2007.

G.A.S. Pereira, A.K. Das, V. Kumar, and M.F.M. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. In *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II, Proceedings of the 2003 International Workshop on Multi-Robot Systems*, 2003.

A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47:53–57, 2004.

D. O. Popa and C. Helm. Robotic deployment of sensor networks using potential fields. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

M. Powers and T. Balch. Value Based Communication Preservation for Mobile Robots. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

D. Puccinelli and M. Haenggi. Multipath fading in wireless sensor networks: Measurements and interpretation. In *Proceedings of the International Wireless Communications and Mobile Computing Conference*, 2006.

R. L. Raffard, C. J. Tomlin, and S. P. Boyd. Distributed optimization for coopreative agents: Application to formation flight. In *Proceedings of the IEEE Conference on Decision and Control*, 2004.

W. Ren and R. W. Beard. Formation feedback control for multiple spacecraft via virtual structures. *IEE Proceedings - Control Theory and Applications*, 151(3): 357–368, May 2004.

C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

B. Roth. Rigid and flexible frameworks. *The American Mathematical Monthly*, 88: 6–21, 1981.

Gordon L. Stüber. *Principles of mobile communication*. Kluwer academic publishers, 1996.

D. Swaroop and J.K. Hedrick. String stability of interconnected systems. *IEEE Transactions on Automatic Control*, 41(3):349–357, 1996.

Swedish Defence Research Institute. Fenix flight dynamics simulator, 2007. URL `http://www.foi.se/FOI/Templates/ProjectPage____5147.aspx`. Last visited 2007-10-29.

J.D. Sweeney, R.A. Grupen, and P. Shenoy. Active QoS flow maintenance in controlled mobile networks. In *Proceedings of the Fourth International Symposium on Robotics and Automation*, 2004.

H. G. Tanner, G. J. Pappas, and V. Kumar. Leader-to-Formation Stability. *IEEE Transactions on robotics and automation*, 20(3), June 2004.

H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switched networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, 2007.

W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proceedings of the IEEE INFOCOM*, 2005.

M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Proceedings of the First IEEE International Conference on Sensor and Ad hoc Communications and Networks*, 2004.