KTH Electrical Engineering

Communication-Aware Motion Planning for Mobile Robots

MAGNUS MINNEMA LINDHÉ

PhD Thesis
Stockholm, Sweden 2012

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

TRITA-EE 2012:002
ISSN 1653-5146
ISBN 978-91-7501-234-6

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framläggs till offentlig granskning för avläggande av teknologie doktorsexamen i reglerteknik fredagen den 10 februari 2012, klockan 10:15 i hörsal F3, Kungliga Tekniska högskolan, Lindstedtsvägen 26, Stockholm.

Tryck: Universitetsservice US AB

Abstract

Mobile robots have found numerous applications in recent years, in areas such as consumer robotics, environmental monitoring, security and transportation. For information dissemination, multi-robot cooperation or operator intervention, reliable communications are important. The combination of communication constraints with other requirements in robotics, such as navigation and obstacle avoidance is called communication-aware motion planning. To facilitate integration, communication-aware methods should fit into traditional layered architectures of motion planning. This thesis contains two main contributions, applicable to such an architecture.

The first contribution is to develop strategies for exploiting multipath fading while following a reference trajectory. By deviating from the reference, a robot can stop and communicate at positions with high signal strength, trading tracking performance for link quality. We formulate this problem in three different ways: First we maximize the link quality, subject to deterministic bounds on the tracking error. We control the velocity based on the position and channel quality. Second, we consider probabilistic tracking error bounds and develop a cascaded control architecture that performs time-triggered stopping while regulating the tracking error. Third, we formulate a hybrid optimal control problem, switching between standing still to communicate and driving to improve tracking. The resulting channel quality is analyzed and we perform extensive experiments to validate the communication model and compare the proposed methods to the nominal case of driving at constant velocity. The results show good agreement with the model and improvements of over 100% in the throughput when the channel quality is low.

The second contribution is to plan velocities for a group of N robots, moving along pre-determined paths through an obstacle field. Robots can only communicate if they have an unobstructed line of sight, and the problem is to maintain connectivity while traversing the paths. This is mapped to motion planning in an N-dimensional configuration space. We propose and investigate two solutions, using a rapidly exploring random tree (RRT) and an exact method inspired by cell decomposition. The RRT method scales better with the problem size than the exact method, which has a worst-case time complexity that is exponential in the number of obstacles. But the randomization in the RRT method makes it difficult to set a timeout for the solver, which runs forever if a problem instance is unsolvable. The exact method, on the other hand, detects unsolvable problem instances in finite time.

The thesis demonstrates, both in theory and experiments, that mobile robots can improve communications by planning trajectories that maintain visual connectivity, or by exploiting multipath fading when there is no line of sight. The proposed methods are well suited for integration in a layered motion planning architecture.

Acknowledgements

The first acknowledgement goes to my advisor, Karl Henrik Johansson, for convincing me to start as a PhD student, for his never-ending enthusiasm and encouragement along the way and for teaching me the power of abstraction. Thank you also to Petter Ögren and Xiaoming Hu for being valuable sounding boards in my reference group and to Petter for luring me into research after a very stimulating Master Thesis work. Further, I would like to thank Bo Wahlberg and Carlo Fischione for being co-advisors, and Dimos Dimarogonas for proofreading parts of the thesis.

During my period as a PhD student, I have had the chance to visit some prominent research groups, which has provided new perspectives. I would like to thank Antonio Bicchi at the University of Pisa, Richard Murray at Caltech and Tamás Keviczky at TU Delft for their generous hospitality. Thank you also to Jean-Paul Laumond at LAAS-CNRS and Roland Geraerts at the Utrecht University for making time when I needed some expert input on motion planning.

Thank you also to Henrik Sandberg for your insight in everything regarding automatic control, Cristian Rojas for helping me with statistics, Bo Lincoln and Anders Rantzer for kindly providing numerical routines for relaxed dynamic programming, Simon Berg for building the first measurement robot, Fotios Katsilieris for all your hard work in the AURES project and Muhammad Altamash Ahmed Khan for staying up late to measure fading all over campus.

Interesting research aside, what really makes this job special are all the wonderful colleagues I've had the privilege to get to know over the years. Thank you to all my colleagues at the Automatic Control Lab, David, Johan, Maja and Tove at Opt&Syst. and Andrea & Co. in Delft. And a special thanks goes to Karin for making sure that things run smoothly and the lab gets decorated at Christmas. Get well soon!

The work reported in this thesis was funded by the Swedish Defence Materiel Administration through the TAIS project, the European Commission through the RUNES and HYCON projects, the Swedish Research Council and the Swedish Foundation for Strategic Research. Their support is gratefully acknowledged.

Finally, I want to thank my wife Jenny for showing me what really matters in life. Archimedes supposedly said "Give me a lever long enough and a fulcrum on which to place it, and I shall move the world." Whatever I've managed to move is thanks to you, Viggo and Kasper for providing me with that firm foothold!

Contents

Chapter 1

Introduction

Mobile robots have found use in numerous areas in recent years, including consumer robotics, environmental monitoring, surveillance and transportation. Many of the applications require that the robots communicate, either with other robots or with a base station or operator. Communication can offer many benefits, such as enabling coordination, collaborative processing or dissemination of sensor information. But these benefits come at a cost. The need to ensure reliable communications adds to the complexity of controlling a robot. This is the subject of this thesis: How can robotic motion planners be modified such that the robots fulfill an additional requirement of maintaining communications with each other or a base station? This is an emerging area of robotics, sometimes referred to as *communication-aware* motion planning. It is different from traditional motion planning, which is typically only concerned with avoiding collisions with obstacles or other robots while trying to reach a configuration that is either given beforehand or computed online to optimize some criterion such as coverage or formation keeping.

In this chapter, we first present some motivating applications where communication is a key requirement, ranging from ground robots to formations of satellites in space. Then we describe how communication-aware motion planning can be represented as interconnections between a typical architecture of robot motion planning and the standard layered model of communication. Finally, we outline the thesis and the contributions of each chapter and list what publications they are based on.

1.1 Motivating Applications

We present three applications where communication-awareness is a key criterion for success. They are chosen to demonstrate various application domains and communication constraints. There are requirements on maintaining clear lines of sight, bounding the inter-agent distance or overcoming multipath fading. For the robots to fulfill their mission, they need reliable communications with a base station or within the group.

Figure 1.1: A GroundBot rolling past a tent at the final demonstration of the AURES project (Ögren et al., 2009). The project demonstrated autonomous capabilities such as positioning for camera surveillance, minimum-time patrolling and searching and securing designated areas.

Robots for Reconnaissance and Surveillance

Both during international missions and at its secure facilities in Sweden, the Swedish Armed Forces see a need to enhance security by providing sentries with assistance from unmanned ground vehicles (UGVs). UGVs can help in providing complete coverage of sites, reduce the risks for sentries when responding to alarms and improve sensing by using infrared vision. With this in mind, the project Autonomous UGV-system for Reconnaissance and Surveillance (AURES) was pursued by the Swedish Defence Research Agency, KTH and Saab (Ögren et al., 2009). The goal of the project was to develop, demonstrate and evaluate algorithms which would provide various levels of autonomy to UGVs, making them a useful tool for surveillance personnel. The project resulted in new algorithms for autonomously choosing robot positions and camera angles to provide coverage of designated buildings. Algorithms were developed for patrolling a given area in minimum time and for searching and securing a given area so that no intruder could return to the searched regions without being detected. The project ended with a demonstration, demonstrating the developed algorithms on the GroundBot robot, developed by Rotundus AB. Figure 1.1 shows a photo of the demonstration site and one of the robots.

To maintain the connection to the operator, either directly or through relaying within the group, the robots need to move in a communication-aware manner. In typical applications, the main limiting factor for the communication range will be shadowing and multipath fading caused by obstacles. If the robots can maintain clear lines of sight between each other and use small-scale motion to avoid the negative effects of fading, this increases their operational range. In this thesis, we present algorithms for this.

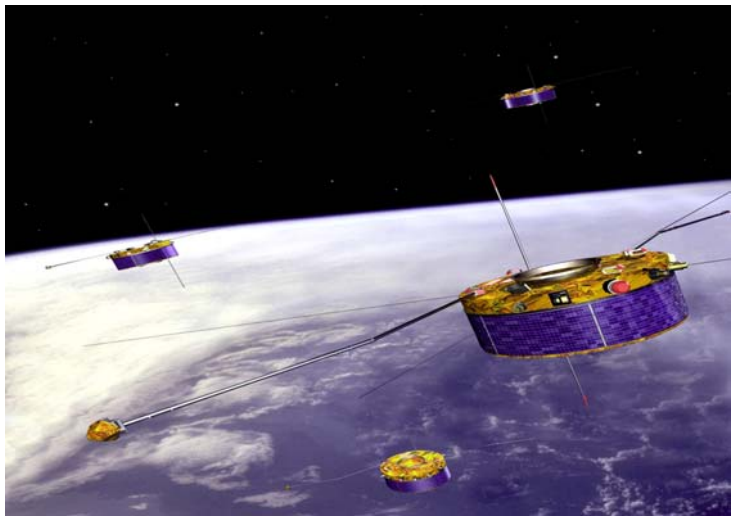


Figure 1.2: An artist's impression of the Cluster-II satellites in orbit around the Earth. (Copyright: European Space Agency)

Satellite Clusters

In the summer of 2000, two Russian Soyuz rockets were launched, carrying a total of four Cluster-II satellites, illustrated in Figure 1.2. The purpose of the mission, organized by the European Space Agency, 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. 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 were used to form a tetrahedron formation, making up a giant three-dimensional sensor array. The spatial resolution of the resulting synthetic sensor could 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 traveling at 6 km/s (Cluster Project Team, 2000; European Space Agency, 2007).

To precisely control their relative distances and orientations, the satellites must maintain communications with each other. In space, one can expect to operate in open sight, so the main limiting factor is the inter-agent distance. It is therefore important to include constraints on maintaining connectivity in the formation control laws, also during reconfiguration and initial deployment. In this thesis, we present methods to control groups of robots subject to such connectivity constraints.

Figure 1.3: An example of a small military robot, suitable for use as a LANdroid. It uses tracks for locomotion and has two powered paddles for passing obstacles or righting itself if it lands upside down. (Copyright: iRobot Corp.)

LANDroids

Modern military operations in urban terrain rely on high-bandwidth communications for transmitting voice data, real-time maps and video images. In such scenarios, radio signals are absorbed or reflected by buildings and other obstacles, making it hard to achieve satisfactory coverage for moving soldiers. Therefore, the American Defense Advanced Research Projects Agency has launched a project aimed at developing LANdroids, portable and disposable mobile base stations for an ad-hoc communication network (McClure et al., 2009). The robots can be deployed manually, and then move autonomously to improve the performance of the network. An example mission for such a system would be to create coverage in part of a city as preparation for a military campaign, or to provide rescue workers with communications after a natural disaster has damaged the existing infrastructure.

The robots must plan their motion to avoid obstacles, maintain connectivity with other LANdroids and maximize coverage. Even small movement, in the order of a few wavelengths, can have a big impact on the range of a single node. Figure 1.3 shows an example of a rugged miniature robot for military use, which could provide motion capabilities to each node. Algorithms for making such small motions are developed and analyzed in this thesis.

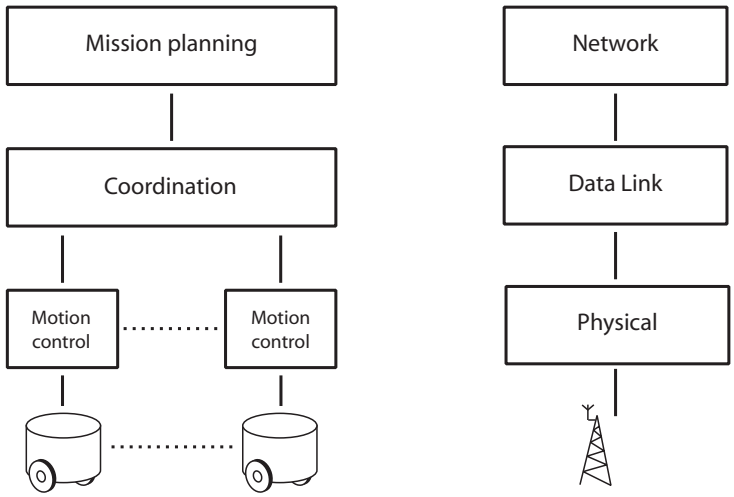


Figure 1.4: A typical layered architecture for robot motion planning, compared to the lower layers of the OSI stack for communication. Communication-aware motion planning can be described as interconnecting one or more of the planning layers with a layer in the communication stack.

1.2 Layered Motion Planning

To handle the complexity of robot motion planning, systems are often organized in a layered architecture (Varaiya, 2000). This is illustrated in Figure 1.4, along with the standard Open Systems Interconnection (OSI) reference model for communication networks, standardized in ISO/IEC 7498-1:1994. We will briefly explain the role of each layer and then argue how communication-aware motion planning can be represented as introducing connections between layers in the motion and communication stacks.

In the motion planning stack, the lowest layer is the *motion control* layer, which employs feedback from sensors to motors and other actuators on the robot to make it track some given reference trajectory. This controller handles the platform-specific dynamics and acts as a homogeneous abstraction to higher layers. The *coordination* layer produces continuous reference trajectories for the robots to track such that they do not collide with obstacles or other robots. Every trajectory could be associated with a specified tracking accuracy, both in space and time, to guarantee safety. The highest level is the *mission planning* layer, which allocates the required number of robots, assigns the communication topology and computes time-stamped waypoints for each robot such that the overall mission is fulfilled. Examples of missions are sensor deployment, searching or transportation.

Wireless communication systems are also structured in layers, usually following the OSI model (Goldsmith, 2005). The three lower layers in this model are shown

in Figure 1.4. The lowest level is the *physical* layer, which interfaces the antenna. Specifications of the physical layer include carrier frequency, modulation format and output power. It provides an unreliable link for sending individual bits. The next layer is the *data link* layer, which, among other functions, contains the medium access control (MAC). The MAC controls access to the shared medium to avoid collisions that incur losses of data. In wireless systems, this includes scheduling transmissions in frequency and time, and handling conflicts. The data link layer implements a reliable link for packets between two directly connected nodes. The third level is the *network* layer, which uses the node-to-node links to provide multi-hop connectivity for the network. This includes logical addressing and routing and offers a reliable end-to-end packet link. Layer four is the *transport* layer, which decomposes data into packets and reassembles it. The fifth layer is the *session* layer, which handles setting up and recovering communication sessions. Layer six is the *presentation* layer, which handles syntax conversion, compression and encryption. Finally, the seventh layer is the *application* layer, which implements specific services such as file transfer or terminal emulation.

Introducing communication awareness can be represented as implementing one or more of the motion planning layers with a component that interacts with a layer in the communication stack. To maintain the benefits of the layered structure, this should not change the interface to the other layers in the motion stack. We now give some examples of the benefits that such components could offer.

A natural part of mission planning is to specify the communication topology, *i.e.*, which robots should talk to which. If the mission planner communicates this suggested topology to the network layer, it can compute a routing table for the network. Depending on the mission requirements, the planner could decide on a maximum number of hops or some lower bound on robustness to link failures and query the network layer to see if the suggested topology allows it. If not, it could update the topology or change the routing parameters in the network layer.

For the coordination layer, there are potential benefits in interconnecting it with the MAC of the communication stack. The coordination layer ensures that no robots come close enough to collide, but this could be extended. Given some performance requirement such as bandwidth or maximum latency, the MAC could respond with a minimum number of frequency bands needed. If frequency reuse is needed to fulfill this, the coordination layer could enforce the required separation between robots to avoid interfering transmissions. This may affect what corridors or narrow passages the robots can traverse, so there will be a tradeoff between traversability and communication. Another possibility, which we investigate in this thesis, is to modify the reference trajectories so that the robots maintain visual contact with each other while moving between waypoints. In indoor or urban environments, this avoids the shadowing of radio signals by the obstacles. This represents an interconnection with the network layer in the communication stack, which can determine if the network is connected or not.

The motion of each platform, as dictated by the motion controller, will affect the multipath fading that the radio receiver experiences. As we show in this thesis,

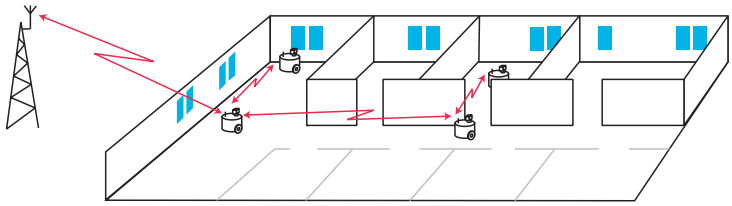


Figure 1.5: A motivating scenario for this thesis. A group of robots is patrolling an office, sending video streams to an operator. The robot communicating with the remote base station should adapt its motion to multipath fading, and all robots should maintain visual connectivity to avoid that obstacles block the wireless links.

interconnecting the physical layer of the communication stack with the motion controller, allows significant improvements of the link capacity. While still tracking the reference position, the robot can actively seek to spend more time at positions where the multipath fading is favorable.

Having motivated how communication-aware motion planning would fit into an existing architecture of a robot, we now formulate the high-level problem formulation that has guided the research presented in the thesis.

1.3 Problem Formulation

We consider a scenario of the type illustrated in Figure 1.5. A number of robots are patrolling an office floor, while sending video streams to an operator and exchanging information to coordinate within the group. This suggests a layered motion planning architecture, like that in Figure 1.4. A mission planner allocates rooms to robots, a coordination layer plans trajectories between rooms and a motion controller ensures that each robot tracks its given reference trajectory. While doing this, the robots must ensure that the group stays connected and that the wireless link to the base station maintains high channel quality. This problem can be formulated as follows: How can the coordination or motion control layers be modified so the motion planning requirements of the original architecture are fulfilled, as well as additional communication constraints? This thesis considers two examples of such constraints. The first example is that the robot communicating with the base station must adapt its motion to multipath fading, while not deviating too far from its reference trajectory. The second example is that the robots should maintain visual connectivity within the group, to avoid that obstacles block the wireless links. They should do so without leaving the obstacle-free paths computed by the coordination layer.

1.4 Outline and Contributions

To aid the reader, we now describe the outline of the thesis. For chapters that contain novel contributions, we highlight the contributions and also list what publications the chapter is based on. We end by a list of other publications by the author, which have inspired the thesis but are not treated further here.

Chapter 1

As seen above, this chapter contains some motivating applications and we also give our view of how communication-aware motion planning relates to traditional motion planning. One of the applications is searching and securing using unmanned ground vehicles. We have proposed a novel algorithm to do this, using triangulation of free areas to construct a graph representation of which areas are connected. We have presented two search methods that represent different tradeoffs between the completion time and the number of robots needed. Finally, we have reported from an experimental demonstration of the system. This is described in

- F. Katsilieris, M. Lindhé, D. V. Dimarogonas, P. Ögren and K. H. Johansson. Demonstration of Multi-Robot Search and Secure. *Workshop on Search and Pursuit/Evasion at the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010.

Chapter 2

This chapter contains brief overviews of some important background theory used in the thesis. First we introduce the model of a single-integrator robot, which will be used in the thesis. We show how the model can be reduced to a one-dimensional model if the robot is constrained to follow a given path and comment on how this applies to various commercially available robot types. Then we present some methods for robot motion planning, both exact and sampling-based methods. After that, we describe propagation of wireless signals and how diversity can help reduce problems caused by unfavorable propagation conditions. With the theoretical background in place, we finally review some related work in the intersection of the above research fields: communication-aware motion planning.

Chapter 3

This is the first of three chapters that all consider how a robot can exploit multipath fading when following a pre-planned reference trajectory: Since multipath fading causes fast variations in the channel quality, the robot could improve communication performance by deviating slightly from the trajectory to spend more time at positions where the channel is better. In this chapter, we study this problem under the constraint that the tracking error, *i.e.*, deviation from the reference, must never exceed a given limit.

This is a novel problem formulation, which represents a new type of diversity. It is similar to using multiple antennas and selecting the best one, but here we instead leverage the mobility of the robot to achieve a similar effect at no extra cost in radio hardware. We study five cases, differing in the assumptions on how densely the robot can sample the channel and what prior channel information the robot has. For each case, we derive the motion strategy that maximizes the expected link quality and compute the distribution of the resulting signal-to-noise ratio. This can, in turn, be used to compute the expected link capacity or throughput. We also report from extensive experiments, where we validated the model of static multipath fading and compared implementations of two of the motion strategies. This is reported in

- M. Lindhé and K. H. Johansson. Exploiting Multipath Fading with a Mobile Robot. Submitted to the *International Journal of Robotics Research*.
- M. Lindhé, K. H. Johansson and A. Bicchi. An Experimental Study of Exploiting Multipath Fading for Robot Communications. In *Proceedings of Robotics: Science and Systems*, Atlanta, Georgia, USA, June 2007.

The second paper is an initial investigation of a special case, corresponding to sparse sampling, full prior knowledge of the channel and a stationary reference position.

Chapter 4

Here we also exploit multipath fading, but the novel contribution compared to the above is to consider probabilistic constraints on the tracking error and a different sensing model. We assume that the robot can only sample the channel when standing still. This motivates a strategy where the robot drives for a constant time and then stops and samples the channel. The length of the stop, before it resumes driving, is a function of the sampled channel quality. By designing this function, called a stop-length policy, we can maximize the throughput while fulfilling the tracking constraints. The stop-length policy is embedded in a feedback architecture which regulates the tracking error and adapts to varying channel conditions. The resulting performance is illustrated by experiments. This chapter is based on

- M. Lindhé and K. H. Johansson. Adaptive Exploitation of Multipath Fading for Mobile Sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010.
- M. Lindhé and K. H. Johansson. Using Robot Mobility to Exploit Multipath Fading. *IEEE Wireless Communications, Special Issue on "Wireless Communications in Networked Robotics"*, February, 2009.

The second paper is an overview, which also reports on the results from the next chapter.

Chapter 5

This is the last chapter on exploiting multipath fading, and the problem formulation considered here does not place any constraints on the tracking error. Instead, we include both tracking and communication in a cost function and optimize the motion with respect to it. The robot and its on-board data buffer are modeled as a switched linear system, such that the robot can choose to stop to communicate or drive to catch up with the reference. The hybrid optimal control problem is solved by relaxed dynamic programming and we illustrate the result by simulations. This chapter is based on

- M. Lindhé and K. H. Johansson. Communication-Aware Trajectory Tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, California, USA, May 2008.

Chapter 6

This chapter considers visibility-constrained communication for a group of robots moving along pre-planned paths among obstacles. We compute the velocities for all robots along the paths so they reach their goals, under the additional constraint that the group must maintain visual connectivity at all times. We first describe how the problem can be mapped to a classic motion planning setting, using a configuration space representation. Then we present two alternative solution methods. First we use an RRT method, whose computation time scales favorably with the number of robots. But it is only probabilistically complete, so if there is no solution, the solver will run forever. We then propose an exact solution method, using a simplified scenario with parallel paths and a constrained network topology. The method is based on exact cell decomposition, but in contrast to most cell decompositions, we use overlapping cells. The cells are convex and simple to construct, but instead it is computationally expensive to test for cell adjacency. Using the decomposition, we abstract the problem into A* searching on a graph, which in finite time either finds a solution or correctly concludes that the problem is unsolvable. The worst-case time complexity of the method is exponential in the number of obstacles. We end by illustrating both methods with simulations. The RRT method is described in the following paper:

- M. Lindhé, T. Keviczky and K. H. Johansson. Multi-Robot Path Following with Visual Connectivity. *Proceedings of the The Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, USA, November 2011.

Chapter 7

In this chapter, we summarize the thesis and draw conclusions. We also point out some possible directions of future research.

Other Publications

The author has previously worked on the problem of flocking, particularly using Voronoi partitioning as a tool for decentralized collision avoidance. This has motivated part of the thesis, since successful flocking requires communication between the agents, but is not treated here. See

- M. Lindhé. On Communication and Flocking in Multi-Robot Systems. Licentiate thesis, KTH Royal Institute of Technology, 2007. ISBN 978-91-7178-795-8.
- M. Lindhé and K. H. Johansson. A Formation Control Algorithm Using Voronoi Regions. In *Taming Heterogeneity and Complexity of Embedded Control*, pages 419-434, edited by F. Lamnabhi-Lagarrigue, S. Laghrouche, A. Loria and E. Panteley, ISTE Ltd, 2006. (Awarded "Best PhD Student Presentation" at the *CTS-HYCON Workshop on Nonlinear and Hybrid Control*, Paris, France, 2006.)
- 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 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

Chapter 2

Background

This thesis builds on work in robotic motion planning and wireless communications. So before giving an overview of related research in communication-aware motion planning, we will provide a broad overview of relevant background material in motion planning and communications. But first, we define the model of robot dynamics that we consider in the thesis, show how it can be reduced to one-dimensional motion along a given path and how this applies to commercially available robots.

2.1 Robot Dynamics

Here we present the model of a single integrator robot, which will be used in this thesis. We further define a reference trajectory and the one-dimensional position along the corresponding path. We then relate this to existing robot platforms, which often have additional constraints compared to our simplified models.

Single-Integrator Model

We consider robots with single integrator dynamics. The robot has position $q \in \mathbb{R}^2$ and control $u \in \mathbb{R}^2$, with dynamics $\dot{q} = u$. We assume that a coordination layer, as discussed in Chapter 1, produces a reference *trajectory* $q_{\text{ref}} : [0, T] \to \mathbb{R}^2$ with constant reference speed $|\dot{q}_{\text{ref}}(t)| \equiv v_{\text{ref}}$ that takes the robot from one waypoint to another, in a given time T. The corresponding *path* is the set $\bigcup_{t\in[0,T]} q_{\text{ref}}(t) \subset \mathbb{R}^2$ of all points that the trajectory passes. We further assume that $q(0) = q_{\text{ref}}(0)$, and that there is a control $u_{\text{ref}} : [0, T] \to \mathbb{R}^2$ such that

$$q_{\text{ref}}(t) = q_{\text{ref}}(0) + \int_0^t u_{\text{ref}}(\tau)\mathrm{d}\tau.$$

Motion Along the Path

We denote the robot position along the path as $x(t) \in [0, Tv_{\text{ref}}]$, with constraint $x(0) = 0$. We further introduce the velocity along the path $v(t)$, which controls $x(t)$ as

$$\dot{x}(t) = v(t). \tag{2.1}$$

Now, if we apply the control $u(t) = \frac{v(t)}{v_{\text{ref}}} u_{\text{ref}}\left(\frac{x(t)}{v_{\text{ref}}}\right)$ to the robot, its position at time t will be

$$q(t) = q(0) + \int_0^t \frac{v(\tau)}{v_{\text{ref}}} u_{\text{ref}}\left(\frac{x(\tau)}{v_{\text{ref}}}\right) \mathrm{d}\tau = q_{\text{ref}}(0) + \int_0^{\frac{x(t)}{v_{\text{ref}}}} u_{\text{ref}}(s)\mathrm{d}s = q_{\text{ref}}\left(\frac{x(t)}{v_{\text{ref}}}\right).$$

In the second equality, we use the substitution $s = \frac{x(\tau)}{v_{\text{ref}}}$ and the identity $q(0) = q_{\text{ref}}(0)$. We also see that

$$\dot{q}(t) = \frac{v(t)}{v_{\text{ref}}} u_{\text{ref}}\left(\frac{x(t)}{v_{\text{ref}}}\right) = v(t)\hat{u}_{\text{ref}}\left(\frac{x(t)}{v_{\text{ref}}}\right),$$

where $\hat{u}_{\text{ref}}\left(\frac{x(t)}{v_{\text{ref}}}\right)$ is a unit vector in the direction of the reference path at $x(t)$. We end by defining the reference position along the path at time t as $x_{\text{ref}}(t) \triangleq tv_{\text{ref}}$.

Differential-Drive Robots

In practice, most robots do not have the simple single-integrator dynamics described above. In our experiments we have used a common type of robot, with differential drive. As illustrated in Figure 2.1, it has two parallel, individually powered wheels with wheel base L and radius r. Often, there are also one or more caster wheels to maintain the balance. The controls are the angular velocities $\omega_{\text{l}}, \omega_{\text{r}}$ of the left and right wheel, respectively. The states of the robot are the position $q = (q_1, q_2)$ and orientation θ, relative to the positive q_1 axis. Its dynamics are

$$\begin{aligned}
\dot{q}_1 &= \frac{r}{2}(\omega_{\text{r}} + \omega_{\text{l}})\cos\theta \\
\dot{q}_2 &= \frac{r}{2}(\omega_{\text{r}} + \omega_{\text{l}})\sin\theta \\
\dot{\theta} &= \frac{r}{L}(\omega_{\text{r}} - \omega_{\text{l}}).
\end{aligned}$$

We note two important things: First, the state of the robot not only contains the position q, but also the orientation, θ. The full state (q_1, q_2, θ) is called the *pose* of the robot. Second, the robot can only move along the direction of orientation, as opposed to the single-integrator model which can move freely in any direction. To emulate single-integrator dynamics, we can define a point q' that is offset from the wheel center point by a distance $d > 0$:

$$q' = q + d\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

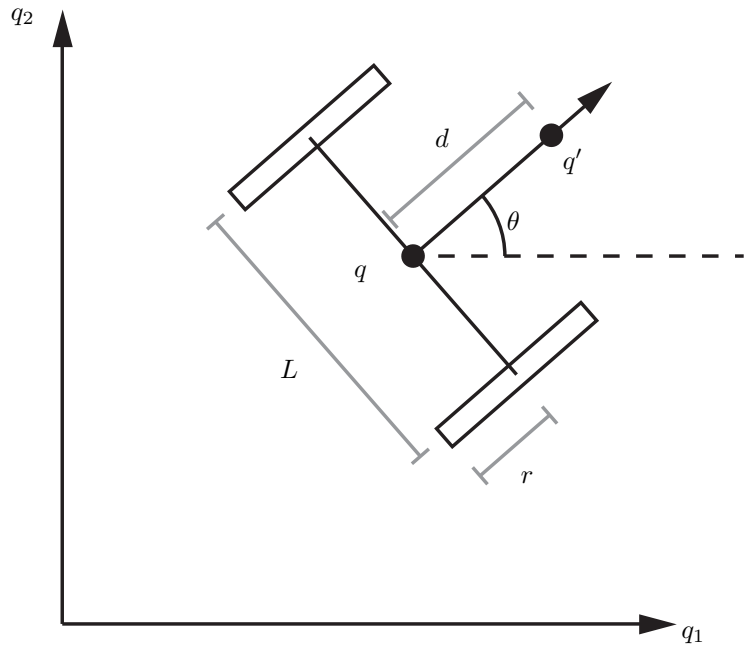


Figure 2.1: A differential drive robot with two wheels, with radius r and wheel base L. It has orientation θ and the point q cannot move in any direction, but q' can.

It has dynamics

$$\dot{q}' = \dot{q} + d \begin{bmatrix} -\dot{\theta}\sin\theta \\ \dot{\theta}\cos\theta \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{R(\theta)} \underbrace{\begin{bmatrix} r/2 & r/2 \\ dr/L & -dr/L \end{bmatrix}}_{A} \begin{bmatrix} \omega_{\mathrm{r}} \\ \omega_{\mathrm{l}} \end{bmatrix}.$$

The rotation matrix $R(\theta)$ is invertible, as is A, so to get $\dot{q}' = u$, we can choose the state transformation

$$\begin{bmatrix} \omega_{\mathrm{r}} \\ \omega_{\mathrm{l}} \end{bmatrix} = A^{-1}R^{-1}(\theta) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Hence, the position of q' can be controlled to have single-integrator dynamics (d'Andréa Novel et al., 1995).

Other Robot Dynamics

A differential drive robot can emulate the single-integrator model, as shown above, assuming that it can instantly determine the wheel velocities, which is a good approximation for ground robots with high gear ratios and high-traction wheels. Other robots have car-like dynamics, which makes them unable to turn on the spot or move sideways. Such robots can also be controlled to track the position of a single-integrator robot, under some smoothness constraints on the path (Latombe, 1991). Flying robots such as quadrotors or helicopters, or underwater rovers, typically have higher-order dynamics where the acceleration is controlled, rather than the direct velocities. With suitable controllers and strong enough control signals, they can track the position of a single-integrator model, albeit in $\mathbb{R}^3$. Aircraft with airplane-like dynamics are less well modeled by the single-integrator model, since they must maintain a minimum forward velocity to stay airborne (LaValle, 2006). This prevents them from stopping, which is required in all the motion strategies presented in this thesis.

Summary

To summarize, given a constant-velocity reference trajectory and corresponding control sequence for a single-integrator model, we can use the one-dimensional velocity $v(t)$ to control its motion along the corresponding path. It is then possible to solve for the required wheel velocities, $\omega_\mathrm{l}, \omega_\mathrm{r}$, to follow it with a differential drive robot. With this in mind, in the following chapters, we will only consider the dynamics (2.1) of the one-dimensional motion along the path.

2.2 Motion Planning

In the previous section, we discussed how to follow a given reference trajectory. It should typically be computed to avoid collisions with obstacles or other robots. In this section, we give a brief overview of methods to find such trajectories, which also serves as an introduction to Chapter 6. This section follows the excellent book by LaValle (2006).

Configuration Space

Motion planning techniques can be used to solve diverse problems, such as mounting a mechanical part inside a car with a robotic arm, making a humanoid robot walk or move a piano through a cluttered house. All these problems can be formulated in a common way, using the fundamental notion of the configuration space, $\mathcal{C}$. It is equal to the state space of the system, so each state, or configuration, of the system corresponds to a point in $\mathcal{C}$. Sets of configurations that violate the problem-specific constraints are called $\mathcal{C}$-obstacles, and the union of all $\mathcal{C}$-obstacles forms the obstacle region, $\mathcal{C}_\mathrm{obs}$. Constraints arise from limited joint angles, collisions be-

tween objects or, in the humanoid case, losing balance. All other configurations are called free space, denoted $\mathcal{C}_{\text{free}}$. Given an initial configuration $x_{\text{I}} \in \mathcal{C}_{\text{free}}$ and a goal configuration $x_{\text{G}} \in \mathcal{C}_{\text{free}}$, the general motion planning problem can be defined as

Definition 2.2.1 (General Motion Planning Problem)**.** *Find a continuous path $\tau : [0, 1] \to \mathcal{C}_{free}$ such that $\tau(0) = x_I$ and $\tau(1) = x_G$, or report that such a path does not exist.*

It can be shown that this general problem is NP-hard, mainly because the dimensionality of $\mathcal{C}$ is unbounded. Note that a solution to the motion planning problem is a path, which can be traversed at any velocity to yield a robot trajectory. The field of motion planning is therefore often referred to as path planning. We also note that, because of the geometry of $\mathcal{C}$, there may be no admissible path. We will see in the following that some solution methods can detect this in finite time, while others run forever in this case. We now describe some classes of solution methods.

Completeness

Motion planning methods can be broadly categorized in two groups: Sampling-based or combinatorial. Sampling-based methods probe $\mathcal{C}$ according to some sampling scheme. Sampling makes the computational complexity scale well with the dimensionality of the configuration space. The drawback is that if the sampling is not dense enough, the solver may fail to detect solutions. The methods thus do not offer completeness, *i.e.*, a guarantee to either find a solution or report that there is none. Instead, sampling-based methods that sample $\mathcal{C}$ in a deterministic way may be resolution complete, which means that they are guaranteed to find a solution if it exists, in the limit as the sampling becomes sufficiently dense. If there is no solution, they may run forever. Similarly, randomized methods may be probabilistically complete, meaning that the probability of finding a solution if it exists converges to one as time approaches infinity. In contrast, combinatorial methods find a path through $\mathcal{C}$ without using approximations and are therefore also called exact. They are typically also complete. For low-dimensional problems, *e.g.*, where $\mathcal{C} = \mathbb{R}^2$ or $\mathcal{C} = \mathbb{R}^3$, there are efficient exact methods that offer good performance in practice. For higher-dimensional configuration spaces, however, exact methods have high computational complexity, making them impractical to use.

Sampling-Based Methods

Sampling-based methods can be divided into single-query or multi-query methods. A single-query method solves the motion planning problem once, while a multi-query method solves it multiple times. Each query may have new initial and goal configurations, but the configuration space stays constant. This is applicable, for example, for a robotic arm in a robot cell where only the position of the workpiece changes between queries. Multi-query methods typically invest significant time for pre-processing the problem, to reduce the solution time for each query. For both

classes of methods, the only problem-specific component is the collision checker, which provides information about the geometry of $\mathcal{C}$. From the point of view of the algorithm, it can be regarded as a black box that answers if a configuration $x \in \mathcal{C}$ lies in $\mathcal{C}_{\text{free}}$ or not. Later, we will propose a slightly more advanced collision detector, which returns a straight-line path between two points if it exists.

A popular class of single-query algorithms builds a rapidly exploring dense tree inside $\mathcal{C}$, starting at x_{I}. It is iteratively constructed by adding samples that can be connected to the existing tree by free paths. The tree will cover all reachable parts of $\mathcal{C}_{\text{free}}$ as time goes to infinity. If the samples are chosen randomly, the tree is called a rapidly exploring random tree (RRT). At regular intervals, the algorithm tests if x_{G} can be connected to the tree, in which case the problem is solved. The RRT algorithm is probabilistically complete and is presented in more detail in Chapter 6. For some applications, the algorithm is modified so multiple trees are grown, starting at intermediate configurations. This may help to faster cover $\mathcal{C}_{\text{free}}$, but checking for connectivity between the different trees increases the complexity of the algorithm.

For multi-query problems, a popular solution is to build a sampling-based roadmap, which is a general graph with vertices representing points in $\mathcal{C}_{\text{free}}$ and edges representing free paths between points. If the roadmap is constructed by random sampling, it is called a probabilistic roadmap (PRM). Figure 2.2 shows an example of such a roadmap. Queries are solved by finding paths from x_{I} and x_{G} to any vertices in the graph, and then using discrete planning over the graph to connect them. One such solution is illustrated by the thick lines in the figure. Desirable properties for a roadmap are that it should be reachable from any point in $\mathcal{C}_{\text{free}}$, capture the connectivity of $\mathcal{C}_{\text{free}}$ and contain few vertices so it can be searched quickly.

Combinatorial Methods

Cell decomposition is an important class of combinatorial, or exact, motion planning methods. The main idea is that $\mathcal{C}_{\text{free}}$ is partitioned into cells. Cells should be defined so path planning inside a cell is simple, and it should be efficient to determine if two cells are adjacent, so there is a free path between them. The adjacency of cells can be represented in a graph, and a path planning query is solved by first finding the cells that contain the initial and goal configurations. Then a path is planned between them in the adjacency graph, and finally the discrete path in the graph is translated to a continuous path through $\mathcal{C}_{\text{free}}$. Note that the cell adjacency graph can be interpreted as a roadmap, which is guaranteed to be reachable from all of $\mathcal{C}_{\text{free}}$. For high-dimensional configuration spaces, computing the cell decomposition is in general computationally expensive. The fastest known method to date executes in time that is exponential in the dimensionality of $\mathcal{C}$, and it is so complicated that no general implementation of it exists.

Other combinatorial methods construct a roadmap directly, without using cells. One popular such method for the case $\mathcal{C} = \mathbb{R}^2$ is the visibility graph, or shortest-

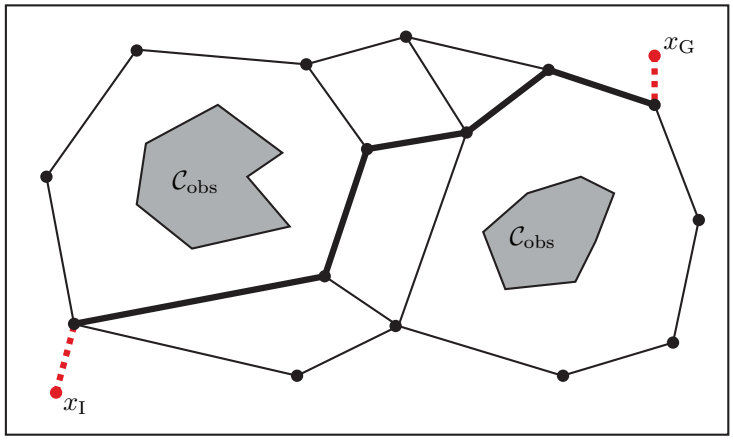


Figure 2.2: A sampling-based roadmap is a graph with vertices representing points in $\mathcal{C}_{free}$ and edges representing free paths between points. To solve the query (x_I, x_G), one first finds paths from the points to the roadmap, then connects them by a discrete search in the graph. The resulting path is drawn with thick lines.

path roadmap. Its vertices correspond to all corners of obstacles, and the edges correspond to corners whose connecting line does not intersect $\mathcal{C}_{obs}$. As the name suggests, this graph is guaranteed to contain the shortest path between any two obstacle corners. Another method instead finds the maximum-clearance roadmap, *i.e.*, the roadmap that contains all paths with maximum distance to any obstacle. This increases robustness to errors in the map or robot navigation, at the expense of path length.

As a final example of exact methods, we will mention potential-based algorithms. They define a scalar potential $\phi : \mathcal{C} \to [0, \infty]$, with a minimum at x_G. To reach the goal from any point in $\mathcal{C}$, the system should move in the direction of $-\nabla\phi$, *i.e.*, the negative gradient of ϕ. If ϕ has a single minimum at x_G and is infinite in all points from which the goal cannot be reached, it is called a navigation function (Rimon and Koditschek, 1992). Navigation functions can be computed on grids, but as the dimensionality of the state space increases, the number of grid elements increases exponentially for a given resolution. Alternatively, they can be computed directly from the configuration space representation, but the methods to avoid introducing local minima are difficult to apply to general high-dimensional problems.

Multi-Robot Motion Planning

The transportation problem for N robots, *i.e.*, reaching the goal configuration of each robot while avoiding collisions with obstacles or other robots, could be naively represented as any other motion planning problem. If each robot has three states, the resulting configuration space would have $3N$ dimensions. Configurations inside $\mathcal{C}_{\text{obs}}$ would correspond to collisions between a robot and an obstacle, or between two robots. The problem could be solved with the methods above, resulting in a path through $\mathcal{C}$, which can be translated to a trajectory for each robot. Such methods are called centralized and they are typically complete, but the time complexity scales badly with N (Hopcroft et al., 1984). Thus, in practice, they can only be applied to small groups (Barraquand and Latombe, 1991; Schwartz and Sharir, 1983; Svestka and Overmars, 1995).

As an alternative to centralized methods, various so called decoupled methods have been proposed. An early approach was to assign priorities to the robots and plan for one robot at a time, treating robots with higher priorities as moving obstacles (Erdmann and Lozano-Pérez, 1987). This is a fast method, but it may fail to find a solution even if it exists. Prioritization can also be combined with potential-based methods to resolve conflicts (Warren, 1990). Another way to decouple the problem is the path–velocity decomposition: First plan an obstacle-free path for each robot individually, then tune the velocities along the path to avoid inter-agent collisions (Kant and Zucker, 1986). The velocity tuning is referred to as the path coordination problem and was introduced by O'Donnell and Lozano-Perez (1989). This has inspired the problem formulation in Chapter 6. As a final example, there are also methods that can be applied in both a centralized and decoupled framework (LaValle and Hutchinson, 1998). The methods above avoid planning in a high-dimensional configuration space, but still assume a central planner that has access to information about all robots. We now turn to completely decentralized methods.

To achieve good scalability and avoid having a single point of failure, there has recently been great interest in decentralized methods for multi-robot motion planning. Each robot then computes its own trajectory, without using global information. There are decentralized methods for flocking (Olfati-Saber, 2006), coverage (Cortés et al., 2004), target tracking (Martínez and Bullo, 2006) and formation control (Ji and Egerstedt, 2007), to mention some applications. The specific problem of transportation can be solved by potential-based methods (Roussos et al., 2010), using reachability of hybrid systems (Tomlin et al., 1998; Pallottino et al., 2007) or local negotiations (Zhang and Vaughan, 2006).

2.3 Radio Communication

As a background primarily for Chapters 3–5, we here provide a short overview of important concepts in radio communication. Since the thesis focuses on exploiting multipath fading, the focus is on describing it and existing diversity methods for mitigating it. The discussion on channel capacity follows Proakis and Salehi (2002) and the discussion on wireless propagation follows Molisch (2005).

Channel Capacity and Throughput

For a radio receiver to successfully decode a received signal, it is important that the signal is sufficiently strong compared to other signals in the same frequency band, such as noise and interference. In this thesis, we do not consider interference, *i.e.*, intentional transmissions from other systems. Noise can arise from thermal or quantum physical effects inside the receiver, atmospheric effects or unintentional emissions from, *e.g.*, fluorescent lights or car ignition systems. It is often described as additive white Gaussian noise with spectral density N_0, measured in W/Hz. For a receiver with a bandwidth of B and a received useful signal power of P_{RX}, the signal-to-noise ratio (SNR) is

$$\gamma \triangleq \frac{P_{\mathrm{RX}}}{BN_0}.$$

We now define two quality metrics for the wireless link, which we will use in the thesis. They will be denoted as utilities, $U(\gamma)$. It would be straightforward to apply the same analysis to other utilities, as long as $U(\gamma)$ is strictly increasing with γ so the inverse U^{-1} exists. The expectation of a given utility depends on the SNR distribution f_γ as

$$E[U(\gamma)] = \int_0^\infty U(\gamma) f_\gamma(\gamma) \mathrm{d}\gamma. \tag{2.2}$$

For a wireless channel with additive white Gaussian noise, the maximal capacity (measured in bits/s) depends on the SNR and bandwidth as

$$U_{\mathrm{C}}(\gamma) \triangleq B \log_2 (1+\gamma).$$

We will use the channel capacity as our first channel utility. It has the advantage of being independent of the type of modulation, but on the other hand it cannot be directly measured in experiments. We therefore continue by introducing a more application-oriented utility.

For a given modulation method, it is possible to derive the bit error rate (BER), $P_b(\gamma)$. It gives the probability of incorrectly decoding a bit. Since the noise is white, we assume that the bit errors are independent. Thus, the probability of correctly decoding a sequence of N_{P} bytes is $(1-P_b(\gamma))^{8N_{\mathrm{P}}}$. We use this to define our second utility: throughput. For a packet transmission rate of R_0, measured in packets/s, and a packet length of N_{P} bytes, we define the throughput as

$$U_{\mathrm{T}}(\gamma) \triangleq R_0 \left(1 - P_b(\gamma)\right)^{8N_{\mathrm{P}}}. \tag{2.3}$$

We now present some basics in wireless propagation, which links the received signal power to the influence of the environment and the output power of the transmitter.

Wireless Propagation

Path Loss and Shadowing

Consider a mobile robot receiving radio signals from a base station. The distance between the robot and base station is d and there is a free line of sight. The base station emits a signal with power P_{TX} to its antenna, which has an antenna gain of G_{TX}, compared to an isotropic antenna. The radio signal, with wavelength λ, reaches the robot antenna, which has gain G_{RX}. The received power is then

$$P_{\mathrm{RX}}|_{\mathrm{dBm}} = P_{\mathrm{TX}}|_{\mathrm{dBm}} + G_{\mathrm{TX}}|_{\mathrm{dB}} + G_{\mathrm{RX}}|_{\mathrm{dB}} + 20\log\frac{\lambda}{4\pi} - n\log d,$$

where $|_{\mathrm{dB}}$ denotes a value in dB and $|_{\mathrm{dBm}}$ denotes a value in dB, relative to 1 mW. The constant n models how fast the signal decays with distance, and values in the range $1.5 < n < 5.5$ have been reported in the literature. This distance-dependent attenuation of a wireless signal is called *path loss*. We will alternatively denote the received power as received signal strength (RSS). Note that, for constant spectral density of the noise, the SNR in the receiver is related to the RSS by a constant offset:

$$\gamma|_{\mathrm{dB}} = P_{\mathrm{RX}}|_{\mathrm{dBm}} - BN_0|_{\mathrm{dBm}}$$

If obstacles obstruct the line of sight, the signal is typically attenuated more than predicted above, and this effect is called *shadowing* or large-scale fading. Shadowing varies over distances in the same order as the size of the obstacles, and many experiments have shown that it can be well approximated by a lognormal distribution. This means that when plotted on a logarithmic scale, the shadowing attenuation approaches the normal distribution.

Multipath Fading

Figure 2.3 illustrates another effect, which arises if the robot is surrounded by scatterers that reflect the incoming signal. Because of negative or positive interference, the received signal strength will vary. This is called *multipath fading* or fast fading and will cause variations in the signal strength as the robot moves only fractions of a wavelength. The variations depend in a complex way on the environment, and can be modeled using ray-tracing techniques (Fugen et al., 2006). This is, however, computationally demanding and requires accurate maps of the environment and the type of construction materials. As an alternative, multipath fading is often modeled stochastically. This is the approach used in this thesis. If the reflections are equally strong from all directions, this is called Rayleigh fading and the resulting SNR in the receiver is exponentially distributed, with probability density function (PDF)

$$f_\gamma^R(\gamma) = \frac{1}{\Omega}\exp\left(\frac{-\gamma}{\Omega}\right), \tag{2.4}$$

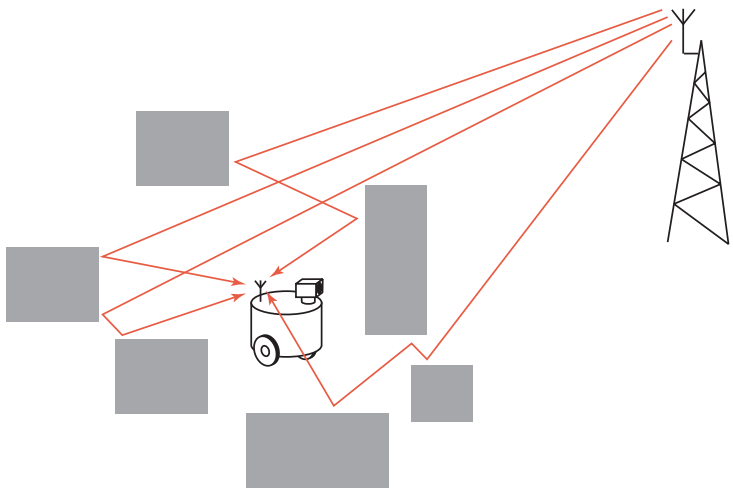


Figure 2.3: A mobile robot will experience multipath fading if multiple reflections of the transmitted signal interfere at its antenna. This typically happens in urban or indoor environments, where the robot is surrounded by scattering objects. If the environment is static, the fading is only a function of the position of the robot.

where $\Omega \triangleq E[\gamma]$ is the mean. If there is a dominant component, *e.g.*, because there is a free line of sight to the transmitter, the fading can be described as *Nakagami fading*, with distribution

$$f_\gamma^N(\gamma) = \frac{m}{\Omega\Gamma(m)} \left(\frac{m\gamma}{\Omega}\right)^{m-1} \exp\left(\frac{-m\gamma}{\Omega}\right), \tag{2.5}$$

where $\Gamma(\cdot)$ is the Gamma function and m is the Nakagami parameter

$$m \triangleq \frac{\Omega^2}{\mathrm{E}[(\gamma - \Omega)^2]}, \tag{2.6}$$

which describes the severity of the fading. The case $m = 1$ is Rayleigh fading, and for $m \to \infty$, the distribution becomes an impulse, corresponding to no fading. Note that, for constant Ω, m is inversely proportional to the variance of γ. The cumulative distribution function (CDF) for γ can be obtained by integration of (2.5) as

$$F_\gamma^N(z) \triangleq P(\gamma < z) = \frac{\gamma_{\mathrm{inc}}(m, \frac{mz}{\Omega})}{\Gamma(m)}, \tag{2.7}$$

where $\gamma_{\mathrm{inc}}(s, w) = \int_0^w e^{-t} t^{s-1} \mathrm{d}t$ is the incomplete gamma function. Another popular model for multipath fading with a dominant component is Rice fading, which we will not consider in this thesis. We just note that they have similar shapes and that they can be used to closely approximate each other.

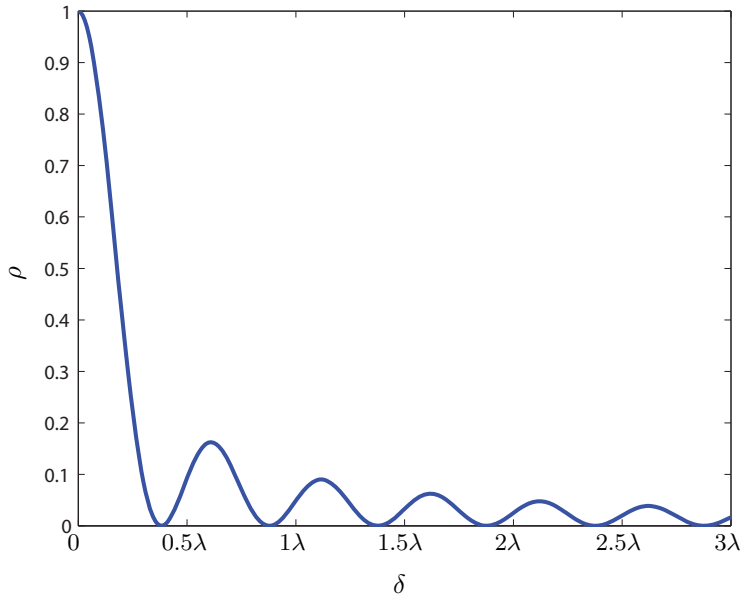


Figure 2.4: Power correlation coefficient ρ for two SNR samples, taken a distance δ apart.

Correlation in Space and Time

Under multipath fading with isotropic scattering, the power correlation coefficient for two SNR samples γ_1 and γ_2 with mean Ω, taken at a distance of δ apart is

$$\rho \triangleq \frac{\mathrm{E}[(\gamma_1 - \Omega)(\gamma_2 - \Omega)]}{\sqrt{\mathrm{E}[(\gamma_1 - \Omega)^2]\,\mathrm{E}[(\gamma_2 - \Omega)^2]}} = J_0^2(2\pi\delta/\lambda), \tag{2.8}$$

where J_0 is the zeroth-order Bessel function of the first kind. This is illustrated in Figure 2.4, which shows that the correlation has a zero at $\delta = 0.38\lambda$ (*i.e.*, 4.75 cm at 2.4 GHz). As a rule of thumb, two samples are often considered independent if the inter-sample distance is greater than half a wavelength (Jakes, 1974). Another common assumption is to neglect the correlation already at a separation of $\lambda/4$, when the amplitude correlation coefficient has decreased below 0.5, which leads to $\rho = 0.22$ (Molisch, 2005). This allows denser sampling without the increased complexity of considering correlation, which we will use in the experiments in Chapter 3. If the scattering is not isotropic, the correlation between two samples depends also on the relative incidence angle of the dominant component (Stüber, 1996). Note that if the receiver moves at constant velocity, which is a common assumption in cellular phone systems, the correlation above can equivalently be considered as temporal correlation.

In this thesis, we make heavy use of the assumption of *static fading*. This means that the robot is the only moving object in the environment, so the multipath fading is a function only of the pose of the robot. (We remind the reader that the pose includes the orientation of the robot, so this applies even if the antenna is not omnidirectional.) Particularly, it will be constant if the robot stands still. This assumption applies to scenarios such as nighttime surveillance, rescue missions inside collapsed buildings and bomb disposal. Static fading has been investigated before (Puccinelli and Haenggi, 2006; Smith et al., 2009), and in Chapter 3, we validate it experimentally. We also investigate what happens to the performance of the proposed motion strategy if the assumption is violated.

Reciprocity

The wireless channel is reciprocal, in the sense that for a given configuration of base station and robot, the path loss, shadowing and fading affects signals in both directions equally. The spatial correlation, however, can differ significantly. A common example of this is when a cell phone experiences multipath fading, because it is surrounded by scatterers. Moving just a fraction of a wavelength will then change the fading, as predicted by (2.8). But the base station is typically mounted in a more open location, surrounded by fewer and more distant scatterers. Two antennas at the base station would therefore need much larger separation before they could be considered to experience independent multipath fading. This is the reason why antenna diversity, as explained below, is normally easier to achieve at the mobile station than at the base station.

Finally, we consider transceivers with narrow bandwidth, so the fading is assumed flat. This means that all frequency components of the signal are affected equally, so the fading causes only attenuation, not distortion. Further, the transceivers are assumed to move slowly enough for the fading to be slow, *i.e.*, the channel is constant over the duration of the transmission of one bit.

Diversity

Multipath fading varies, as shown above, with the exact position of the receiver and hence also over time if the receiver is moving. But it also depends on the frequency, the polarization and the angle of the incoming signals. Exploiting signals that reach the receiver over channels that differ in some of the above is called *diversity*. Of course, diversity can be employed on a larger spatial or temporal scale to also mitigate shadowing or even path loss, but that will not be treated here.

The most widely used diversity is spatial diversity, where multiple antennas are used to receive the same signal. As remarked above, the antennas should be separated by about half a wavelength for the channels to be considered independent. An example of this is the wireless access point in Figure 2.5, using three antennas.

Once the receiver gets the same signal over different channels, or branches, they need to be combined to yield a quality gain. A simple way of doing this is switched

Figure 2.5: A wireless access point using spatial diversity to overcome multipath fading. The frequency is 2.4 GHz, so the antennas are spaced about half a wavelength, or 6 cm. (Copyright: D-Link Corp.)

diversity, where the receiver monitors the signal quality at the chosen branch and switches to another only if some switching criterion is fulfilled. If higher performance is needed, the receiver can monitor all branches and always switch to the best one. This is called selection diversity, and it is a simple and cost-effective method since simpler circuits than a full receiver can be used to monitor the branches. More advanced methods combine all branches, which uses all the information available but also needs more hardware. Hybrid methods select a subset of all branches and then combine them.

In the context of diversity, the problem arises on how to compute the SNR distribution of the resulting signal. If the branches are correlated, this becomes more complicated than for independent branches. There are exact expressions for the joint PDF of correlated Nakagami fading branches, but in practice they are computationally intractable for $N > 4$ (Zhang and Lu, 2002). Alternative approaches are to approximate the correlation matrix with a Green's matrix (Karagiannidis et al., 2003) or using Householder matrices (Alexandropoulos et al., 2009) or assume exponential correlation (Aalo, 1995). One can also estimate an equivalent number of independent channels that yield the same performance, and use it for the analysis (Muharemovic et al., 2008; Smith et al., 2010). We have chosen to use the approximation of exponential correlation, since it yields results that agree best with simulations.

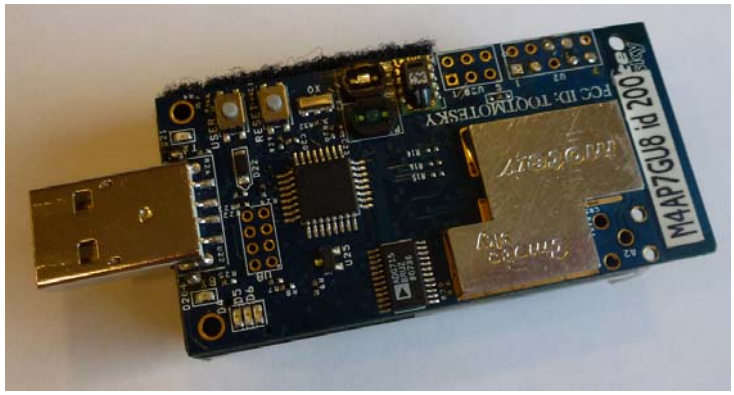


Figure 2.6: A Tmote Sky transceiver, used for the experiments in the thesis.

We end by noting that diversity can also be used to improve the network-wide performance, rather than that of a single point-to-point link as discussed above. In a network where all links are affected by time-varying fading, the total throughput can be improved by monitoring the channels and transmitting to the receiver which has the best channel at each moment. This is called opportunistic communication (Viswanath et al., 2002).

Tmote Sky Specifications

For the convenience of the reader, we end this section by a summary of the specifications of the specific transceiver board we have used for the experiments in this thesis. It is called Tmote Sky (Moteiv Corporation, 2006) and is depicted in Figure 2.6. The Tmote Sky is based on an MSP430 microcontroller and a CC2420 single-chip radio transceiver. The board features an internal antenna with approximately omnidirectional radiation pattern, but it can also be reconfigured to use an external antenna, connected to an SMA connector. The CC2420 transceiver complies with the IEEE 802.15.4 standard, works at 2.4 GHz, has a maximum output power of 0 dBm and a bit rate of 250 kbit/s. It can detect the signal strength of received packets, or measure the ambient noise level in a given channel. The CC2420 uses QPSK modulation, for which the BER is (Proakis and Salehi, 2002)

$$P_b(\gamma) = Q(\sqrt{\gamma}). \tag{2.9}$$

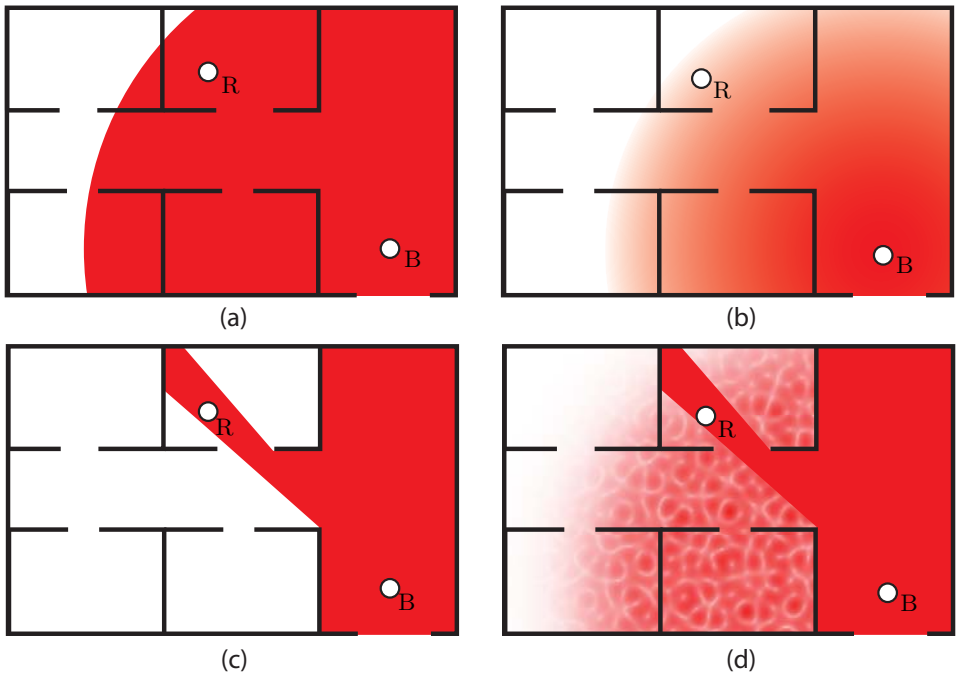


Figure 2.7: Some popular communication models, which emphasize various wireless propagation phenomena. In (a), the robot (R) and base station (B) can communicate perfectly within a certain distance, and not at all outside it. In (b), the channel quality decays gradually with the distance. Case (c) is a pure visibility model, where communication can only happen if there is a free line of sight. Finally, (d) shows that when the robot has no line of sight to the base station, signal strength variations due to multipath fading are considered. This case is the most realistic and the one considered in this thesis.

2.4 Communication-Aware Motion Planning

This section presents some related work in communication-aware motion planning, which lies in the intersection of the research fields presented above. One way to categorize the literature on communication-aware motion planning is by the communication model used. There are several established such models, each representing different tradeoffs between accuracy and simplicity of analysis. Figure 2.7 schematically illustrates four communication models for the case of a single robot communicating with a base station in an indoor environment. Below, we have roughly divided the related work into path-loss models, shadowing and multipath fading. Note that some papers, particularly in the multipath fading category, consider also other propagation effects, but we have tried to isolate where the main contribution lies.

Path loss is a deterministic effect that is well suited for analysis. A simple model is to assume perfect communication within a certain distance, and no connection outside it. This leads to a disc-shaped coverage region, useful for the study of multi-agent coordination under communication constraints, such as flocking (Zavlanos et al., 2009), coverage (Spanos and Murray, 2004), relaying (Tekdas et al., 2010*a*) or formation control (Ji and Egerstedt, 2007). Instead of such a sharp threshold, one can let the throughput decay or the magnitude of some additive noise increase with the distance. This allows joint optimization of communication and sensing objectives (Chung et al., 2006; Hsieh et al., 2008; Stachura and Frew, 2011) or motion planning for chains of routers (Stump et al., 2008). A related approach is to only assume that the signal strength is a sufficiently smooth function of the position, and then do gradient estimation to try to improve the signal strength (Dixon and Frew, 2009). Similarly, there is work on robots that perform random motion inside a bounded workspace if they are not sufficiently connected. This represents a model-free algorithm, but it is analyzed using a distance-based model (Correll et al., 2009). Models taking only path loss into account work well in space and open areas, but less so in settings with obstacles.

Shadowing is an important effect for microwave signals, such as in the popular 2.4 GHz band, which propagate similar to light around obstacles. This suggests that it is reasonable to assume that two robots can communicate if and only if there is a clear line of sight between them. Such visibility-based models are of course also relevant for optical communication. There are examples of rendezvous (Ganguli et al., 2009), transportation (Esposito and Dunbar, 2006; Lindhé et al., 2011), sensor deployment (Ganguli et al., 2007) and searching (Sweeney et al., 2004) problems, solved under these constraints. Another example is autonomous router placement (Tekdas et al., 2010*b*; Stump and Sadler, 2010), where robots have different roles: A few robots explore while others only focus on communication, acting as relays to ensure that the exploring robots stay connected. Connectivity can also be required only at certain time instances (Anisi et al., 2010; Hollinger and Singh, 2010). Line of sight models typically incorporate discontinuities that complicate analysis, and they are conservative since reflections or diffraction may make it possible to communicate even when there is no direct visibility. An alternative to the geometric modeling of shadowing is to model it probabilistically, which helps to improve communication while solving sensing tasks (Ghaffarkhah and Mostofi, 2011).

Visibility constraints are interesting not only for communication in the classical sense, but also for sensing the surroundings or non-cooperative evaders with a camera or other optical device. This can be used to formulate problems in searching (Tovar and LaValle, 2006; Park et al., 2001; Tovar et al., 2004), stationary sensing (Shermer, 1992; Nilsson et al., 2008) or tracking (Murrieta-Cid et al., 2007; Fabiani et al., 2002). The latter is an example of visual servoing (Chaumette and Hutchinson, 2006) for mobile robotics.

Multipath fading is mostly pronounced in indoor or urban settings, where radio signals are scattered by objects. Even small movements of a sensor node will change the fading, which can be exploited to improve the signal strength for a stationary sensor node (McClure et al., 2009; Smith et al., 2009; Lindhé et al., 2007). If nodes are connected in a network, such small movements will change the capacities of the different links, which may be used for load balancing (Puccinelli et al., 2007) or for improving the network throughput (Vieira et al., 2011). Estimating the spatial correlation of the fading can also give important information. Mobile sensors can adapt the step size of their movement so as to faster escape from deep fades (Mostofi, 2009). The stochastic model of multipath fading makes planning difficult and the effect is less significant if there is a strong direct component in the signal, compared to the reflections.

Chapter 3

Motion Planning under Multipath Fading: Deterministic Tracking Constraints

In the following three chapters, we develop and analyze motion controllers for a robot tracking a reference trajectory through areas which exhibit multipath fading. As described in Chapter 2, multipath fading is mostly pronounced in urban and indoor settings, particularly if there is no direct line of sight between the robot and its base station. Also, if the environment does not change over time, the fading is static, *i.e.*, a function only of the position of the robot. In such a scenario, we propose exploiting the fading by letting the robot deviate some from its reference trajectory. By stopping at positions where the SNR is high and then driving again to catch up with the reference position, the robot can improve the average channel utility compared to the nominal case of perfect reference tracking. This proposed *stop-and-go* motion is illustrated in Figure 3.1. In many applications, such small deviations from the reference trajectory do not affect the main task. Also, by deviating only along the reference trajectory, the robot is guaranteed to avoid obstacles if the original trajectory was obstacle-free.

Stop-and-go motion is a tradeoff between reference tracking and communication. The more the robot is allowed to deviate from the reference, the longer it can stay at positions with high SNR and the greater the improvement of the average channel utility. The optimal tradeoff depends on the application requirements and also the nominal channel utility, *i.e.*, that experienced when driving at constant velocity. If the nominal channel utility is low, the robot may need more freedom to deviate to maintain a given average utility, and if the nominal channel utility is high, stop-and-go motion may not be needed. The following three chapters present three alternative ways of quantifying this tradeoff, which lead to different methods of deriving motion controllers. In this chapter, we consider the problem under deterministic constraints on the maximum tracking error. Searching for the best SNR without violating the constraints is then solved as an optimal stopping problem. In Chapter 4, we instead formulate the constraints in a probabilistic way, which leads to a cascaded control architecture: An inner loop ensures that the expected velocity is equal to that of

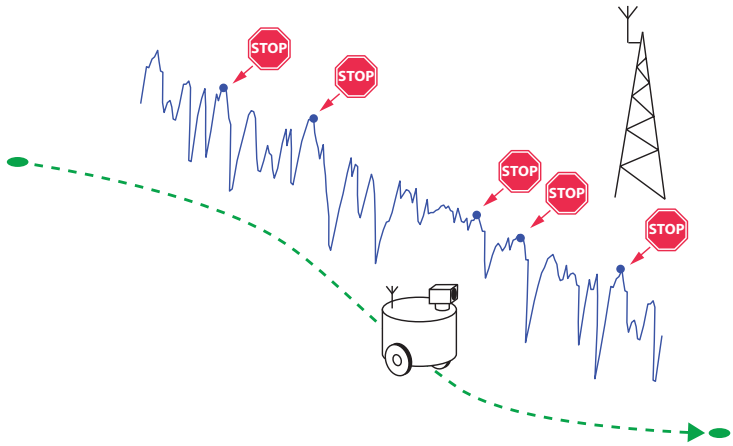


Figure 3.1: A robot follows a reference trajectory while communicating with a base station, subject to multipath fading. By making short stops where the SNR is high and then driving to catch up with the reference position, the robot can improve the average link utility.

the reference, to avoid drift, and an outer loop regulates the tracking error. Finally, in Chapter 5, we investigate a hybrid systems formulation: The robot is assumed to have an onboard data buffer that fills up at a constant rate and is unloaded through the wireless link. The robot switches between standing still to communicate at a high utility or driving to reduce the tracking error. The tradeoff can then be quantified by incorporating both tracking error and buffer size in a cost function. A motion controller is derived by hybrid optimal control.

The proposed stop-and-go motion is a reactive approach, in the sense that it does not in general require prediction of the fading. As discussed in Chapter 2, predicting fading is difficult because of its complex dependence on the geometry of the environment. Instead, the robot can stop when it finds a position where the SNR is high. This suggests using feedback, which is implemented slightly differently in all chapters. In this chapter, the robot is assumed to measure the SNR while moving, so it can stop when it finds a good enough value. In Chapter 4, we relax the requirements some and only assume that the robot can measure the SNR when standing still. This leaves more time for a slow sensor, or allows averaging to avoid noise. The robot then stops at regular intervals, but adapts the time it spends at each position to the local SNR. In Chapter 5, the hybrid optimal control approach requires that the robot knows what channel utility it can expect when stopping. This can be achieved by letting the robot do a quick local search for the highest SNR whenever it decides to stop. The resulting utility will in general vary from the expected, but since the optimal controller uses feedback, it adapts the stop time to the actual outflow of the buffer, providing robustness to this uncertainty.

The assumption on static fading is important for successful reactive stop-and-go motion. If the robot stops at a position with high SNR, it should experience high channel utility until it resumes driving again. As discussed earlier, this is the case if nothing in the environment is moving, which applies to scenarios like surveillance in empty buildings or exploration of contaminated or restricted areas. We will validate this model by measurements in locations that are more or less static and experimentally test the robustness of the developed motion controllers to violations of this assumption.

This chapter proposes and analyzes two novel strategies for communication-aware trajectory tracking under deterministic tracking error bounds. The first proposed strategy applies if only the SNR distribution is known. The second strategy applies if the fading is known beforehand, such as if the robot has recently traversed the trajectory. This represents an ideal case, which serves as an upper bound for the achievable performance. Both strategies are developed and analyzed for sparse SNR sampling, when the samples are assumed independent, and dense sampling, when the samples are assumed correlated.

In the following section, we present models of the tracking error dynamics and how the robot samples the SNR. We then formally define the problem, under five different assumptions on the available channel information. In Section 3.2, we propose a motion strategy to maximize the channel utility for each of the five cases. In Section 3.3, we analytically derive the resulting SNR distribution of each strategy, which can be used to compute the expectation of any channel utility. To illustrate this, Section 3.4 shows numerical results on how the expected link capacity and throughput vary with the allowed deviation from the reference trajectory. We also compare the analytical results with simulations. Then Section 3.5 presents measurements to validate the model of static Nakagami fading. We also report from experiments that compared the strategy for known SNR distribution to the nominal strategy. We end by a summary in Section 3.6.

3.1 Preliminaries

In this section, we present a model of the dynamics of the tracking error. It will also be used in Chapters 4 and 5. We then state a model of how the robot samples the SNR. Finally, we formulate the problem of this chapter.

Reference Tracking Error

In the following, we study the reference tracking error, $\Delta(t) \triangleq x(t) - x_{\text{ref}}(t)$. We remind the reader that $x(t)$ and $x_{\text{ref}}(t)$ are the position of the robot and the reference, respectively, along the reference path. The tracking error has the following dynamics

$$\dot{\Delta}(t) = v(t) - v_{\text{ref}}. \tag{3.1}$$

As defined in Chapter 2, $v(t)$ is the robot velocity along the path and v_{ref} is the reference velocity, which is assumed to be constant. We define the maximal velocity of the robot as $v_{\max}$ and assume that the reference path is smooth enough for the robot to be able to follow it at any velocity $v(t) \in [0, v_{\max}]$. The velocity is restricted to be non-negative to avoid wasting energy by covering the same path segment multiple times.

SNR Sampling

The robot is assumed to be able to measure the link SNR, either by comparing the received signal strength of received packets with the background noise level, or by getting reports back from the base station. We will assume that it can sample the SNR at equidistant positions

$$x_i \triangleq \delta i,\ i = 0, 1, \ldots, \left\lfloor \frac{T v_{\text{ref}}}{\delta} \right\rfloor,$$

along the path, where δ is defined as the distance between samples. We also define the notation $\gamma_i \triangleq \gamma(x_i)$. In the following, we will distinguish between *sparse sampling*, when $\delta \geq \lambda/2$ so the SNR samples can be regarded as independent, and *dense sampling*, when $\delta < \lambda/2$ so the samples are considered correlated.

Problem Formulation

As described above, the average utility of the wireless channel could be increased by giving a robot freedom to deviate some from the reference position, so it can spend more time at positions where the SNR is high. The robot can only get information about the SNR at the sampling points, x_i, so to maximize the time available to spend at points with high SNR, we assume that it always moves at $v_{\max}$ between sampling points. The problem is then how long to stop at each sampling point, or equivalently, to control the velocity $v(t)$:

Table 3.1: Available channel information.

A priori information	Sparse sampling ($\delta \geq \lambda/2$)	Dense sampling ($\delta < \lambda/2$)
None	Case 0	
Known distribution of $\gamma(x_i)$	Case 1A	Case 1B
Full knowledge of $\gamma(x_i)$	Case 2A	Case 2B

We consider the tracking error dynamics (3.1) for a robot moving along a path in an environment with static Nakagami fading, where the SNR is distributed as in (2.5), with spatial correlation (2.8). The position $x(t)$ and tracking error $\Delta(t)$ are known, and the robot has access to channel information as specified in Table 3.1. Find a *stopping strategy*, *i.e.*, a rule that determines how long to stop at the current sampling point, that yields high expected channel utility (2.2) while maintaining a bounded tracking error $|\Delta(t)| \leq \Delta_{\max} \forall\, t \in [0, T]$. We assume that $\Delta_{\max} > \delta/2$, so the robot can always reach at least one sampling position.

We will study five different cases of available channel information. They differ in the amount of *a priori* information available and the SNR sampling density, as defined in Table 3.1. Having no *a priori* information (Case 0) is the nominal case, which we will use as a reference. In Cases 1A and 1B, the joint distribution of all SNR samples is assumed known. In practice, this could be predicted from a map or estimated online, based on previous SNR samples. Finally, if the robot has already traversed the path, it could store the full SNR as function of position, which corresponds to Cases 2A or 2B. Note that this also requires accurate navigation, as even small deviations from the previous path will reduce the correlation between the current SNR and old measurements. For each case, we will derive the resulting SNR distribution f_γ^j, $j \in \{0, 1\text{A}, 1\text{B}, 2\text{A}, 2\text{B}\}$. This distribution can be used to compute the expectation of any utility.

3.2 Stopping Strategies

In this section, we describe stopping strategies for each case above. We first derive the decision horizon for the robot at each decision instance. We define the *tracking window*, $W(t) \subset \mathbb{R}$, as $W(t) \triangleq \{x : |x - x_{\text{ref}}(t)| \leq \Delta_{\max}\}$, *i.e.*, the x-interval where the tracking constraint is satisfied. Note that the window moves along the path at velocity v_{ref}. As stated above, the robot always drives at $v_{\max}$ between the sampling points, so it only needs to make stop-or-go decisions at sampling points. We denote the current sampling point as x_j, so at a decision time t, $j = x(t)/\delta$. From x_j, if the robot drives at $v_{\max}$ without stopping, it can reach the positions $x_{j+1}, \ldots, x_{j+N}$, where

$$N = \left\lfloor \frac{\Delta_{\max} - \Delta(t)}{\delta(1 - v_{\text{ref}}/v_{\max})} \right\rfloor . \tag{3.2}$$

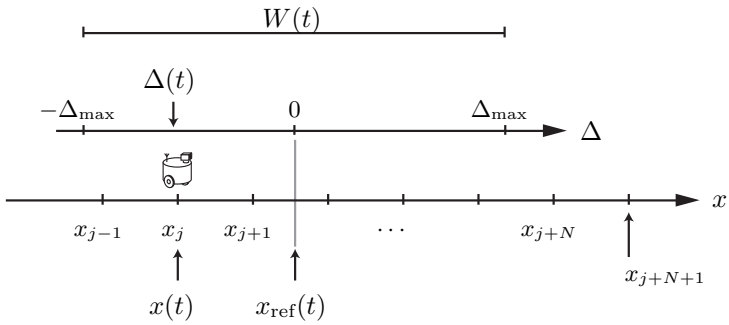


Figure 3.2: Illustration of the tracking window, $W(t)$, *i.e.*, the interval where $|\Delta(t)| \leq \Delta_{\max}$ so reference tracking is maintained. From the current sample position x_j, the robot can reach $x_{j+1}, \ldots, x_{j+N}$ by driving at $v_{\max}$, without stopping. It cannot reach x_{j+N+1} without stopping first, since then it would go outside of $W(t)$.

This is illustrated in Figure 3.2. It cannot reach x_{j+N+1} without stopping, since then it would violate the tracking constraint. Thus, the robot must stop at one of the $N+1$ positions $x_j, \ldots, x_{j+N}$ for some time. Then the sample x_{j+N+1} becomes reachable, and a new decision should be made.

The above discussion shows that when making a stop-or-go decision, the stopping strategy only needs to consider a horizon of N future SNR samples. Note that the driving time between samples does not need to be considered, since the robot needs to traverse the whole path in a constant time T, regardless of the stopping strategy. It will thus spend the time $Tv_{\text{ref}}/v_{\max}$ driving, passing each part of the trajectory once. Hence, the utility during this time is independent of the stopping strategy. With this background, we now present stopping strategies for each case of channel information.

No Channel Information (Case 0)

Without knowing the channel statistics, the robot cannot determine if a given SNR sample is worth stopping at or not. Thus, we propose a stopping strategy that maintains a low tracking error, Δ:

Uninformed Strategy: Stand still at x_j until $\Delta(t) < -\delta/2$, then go to x_{j+1}.

This strategy means that the robot stays close to the center of the tracking window at all times. Note that the robot could still sample the SNR at all stops and quickly have enough information for estimating f_γ^N. Then it could switch to the Partially Informed Strategy, described below, if it needs to improve the channel utility.

Known SNR Distribution

Knowing the SNR distribution, the robot can compare the current utility with what can be expected further ahead. The decision to stop at a position where the utility is high or continue exploring forward can then be formulated as an optimal stopping problem (Chow, 1971). We now derive the optimal stopping rules for this problem in the cases of sparse and dense sampling, respectively, and then summarize them in a proposed *partially informed* strategy.

After sampling γ_j inside $W(t)$, the robot can either stop there, which yields the utility $U(\gamma_j)$, or move forward to the next sample. If the robot rejects γ_j, there are N samples left to choose from, so if it reaches γ_{j+N}, it must stop. This is an instance of Moser's problem of optimal stopping, which can be solved by induction (Moser, 1956). The result is a threshold function $\bar{\gamma}(N)$ such that the robot should stop at x_j if and only if $\gamma_j > \bar{\gamma}(N)$. Note that the threshold function will also depend on the sampling density, δ. Below, we derive the threshold functions for the case of sparse and dense sampling, respectively.

Sparse Sampling (Case 1A)

Define the maximal expected utility, when starting from sample γ_j, as

$$V_j(\gamma_j) \triangleq \max\{U(\gamma_j), \mathrm{E}[V_{j+1}(\gamma_{j+1})]\}.$$

If the robot arrives at x_{j+N}, it has to stop, so $V_{j+N}(\gamma_{j+N}) = U(\gamma_{j+N})$. We define $A_{j+k} \triangleq \mathrm{E}[V_{j+k}(\gamma_{j+k})]$, where $k \in \{1, \ldots, N\}$ and $A_{j+N+1} \triangleq 0$. We compute A_{j+k} through the induction

$$\begin{aligned} A_{j+k} &= \mathrm{E}[\max\{U(\gamma_{j+k}), A_{j+k+1}\}] \\ &= \int_0^{U^{-1}(A_{j+k+1})} A_{j+k+1} f_\gamma^N(\gamma)\mathrm{d}\gamma + \int_{U^{-1}(A_{j+k+1})}^{\infty} U(\gamma) f_\gamma^N(\gamma)\mathrm{d}\gamma. \end{aligned}$$

The robot should stop if the current utility is higher than the maximal expected utility ahead, so the stop threshold for γ_j is $\bar{\gamma}(N) = U^{-1}(A_{j+1})$. We remind the reader that, through the induction above, A_{j+1} depends on N, which depends on $\Delta(t)$. As an example, if $\Delta_{\max} = \lambda$, $N \leq 5$, and the thresholds $\bar{\gamma}(N)$ for $N = 1, \ldots, 5$ are depicted in Fig. 3.3. In the figure, we have assumed $U = U_{\mathrm{C}}$, $m = 1$ and $\Omega = 0$ dB.

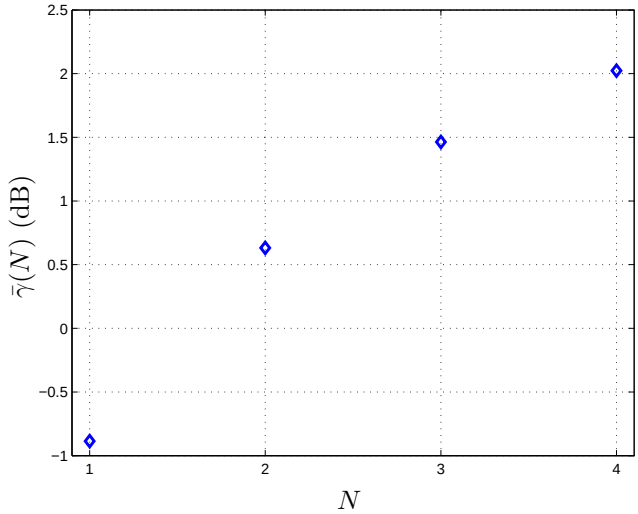


Figure 3.3: Stop thresholds for the Partially Informed Strategy, maximizing the capacity and assuming independent SNR samples (Case 1A). If the robot has N future samples to select from, it should stop at x_j if $\gamma_j > \bar{\gamma}(N)$. Note that $\bar{\gamma}(0)|_{\mathrm{dB}} = -\infty$, so it is not plotted.

Dense Sampling (Case 1B)

The correlation coefficient for any two samples γ_i and γ_j can be computed by (2.8), with separation $\delta|i-j|$. We define the maximal expected utility that can be found after sampling until γ_j as

$$\tilde{V}_j(\gamma_1, \ldots, \gamma_j) \triangleq \max\{U(\gamma_j), \mathrm{E}[\tilde{V}_{j+1}(\gamma_1, \ldots, \gamma_{j+1})|\gamma_1, \ldots, \gamma_j]\}.$$

Of course, $\tilde{V}_{j+N}(\gamma_1, \ldots, \gamma_{j+N}) = U(\gamma_{j+N})$. We define

$$\tilde{A}_{j+k}(\gamma_1, \ldots, \gamma_{j+k-1}) \triangleq \mathrm{E}[\tilde{V}_{j+k}(\gamma_1, \ldots, \gamma_{j+k})|\gamma_1, \ldots, \gamma_{j+k-1}].$$

As above, we can let $\tilde{A}_{j+N+1}(\gamma_1, \ldots, \gamma_N) = 0$ and inductively compute

$$\begin{aligned}&\tilde{A}_{j+k}(\gamma_1, \ldots, \gamma_{j+k-1}) \\ &= \int_0^\infty \tilde{V}_{j+k}(\gamma_1, \ldots, \gamma_{j+k}) f^N_{\gamma_{j+k}|\gamma_1,\ldots,\gamma_{j+k-1}}(\gamma_{j+k}|\gamma_1, \ldots, \gamma_{j+k-1}) \mathrm{d}\gamma_{j+k},\end{aligned}$$

where $f^N_{\gamma_{j+k}|\gamma_1,\ldots,\gamma_{j+k-1}}$ can be computed from the multivariate Nakagami PDF $f^N_{\gamma_1,\ldots,\gamma_{j+k}}$ (Aalo, 1995). Finding $\tilde{A}_j(\gamma_1, \ldots, \gamma_{j-1})$ requires computing an N-dimensional integral, which is computationally intractable as N grows.

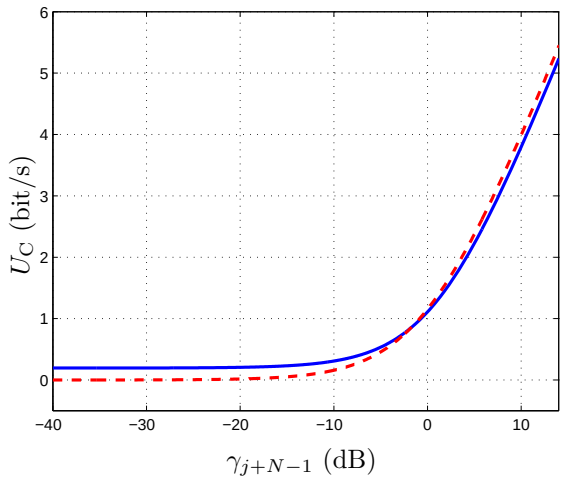


Figure 3.4: The first step in the induction to find stop thresholds for Case 1B. The expected channel capacity at the last sample, $\tilde{A}_{j+N}(\gamma_{j+N-1})$, is plotted as a blue line. If it is less than the capacity at the current sample, $U_C(\gamma_{j+N-1})$, (red dashed line), the robot should stop. In this case, with $\rho = 0.87$, $m = 1$ and $\Omega = 0$ dB, the threshold is -1.7 dB, where the curves intersect.

To make the problem tractable, we propose a Markov assumption: Since the correlation decays quickly with distance, we only consider the correlation between adjacent samples. This yields the simplified induction

$$\begin{aligned}\tilde{A}_{j+k}(\gamma_1, \ldots, \gamma_{j+k-1}) &\approx \tilde{A}_{j+k}(\gamma_{j+k-1}) \\ &= \int_0^\infty \tilde{V}_{j+k}(\gamma_{j+k-1}, \gamma_{j+k}) f^N_{\gamma_{j+k}|\gamma_{j+k-1}}(\gamma_{j+k}|\gamma_{j+k-1}) \mathrm{d}\gamma_{j+k},\end{aligned}$$

which can be computed as a two-dimensional integral. To illustrate the first step of the induction, Figure 3.4 shows $\tilde{A}_{j+N}(\gamma_{j+N-1})$, for $U = U_C$, $m = 1$ and $\Omega = 0$ dB. We assume a sample spacing of $\delta = \lambda/12$, which gives a pairwise correlation coefficient of $\rho = 0.87$. We have assumed $B = 1.15$ Hz so the expected capacity for $f^N_\gamma(\gamma)$ is 1 bit/s. The blue solid line depicts $\tilde{A}_{j+N}(\gamma_{j+N-1}) = \mathrm{E}[\tilde{V}_{j+N}(\gamma_{j+N})|\gamma_{j+N-1}]$. After sampling γ_{j+N-1}, the robot will stop if $U(\gamma_{j+N-1})$ (red dashed line) is higher than $\tilde{A}_{j+N}(\gamma_{j+N-1})$, or equivalently if $\gamma_{j+N-1} > U^{-1}(\tilde{A}_{j+N}(\gamma_{j+N-1}))$. Otherwise it moves on to x_{j+N}. The threshold for stopping at x_j is thus the intersection of the graphs, *i.e.*, $\bar{\gamma}(N) = \gamma : U(\gamma) = \tilde{A}_{j+1}(\gamma)$. The graphs are flat at the intersection, but we have not experienced any accuracy problems in our simulations. As an example, if $\Delta_{\max} = \lambda$, $N \leq 30$ in this case, and the thresholds for $N = 1, \ldots, 30$ and $U = U_C$ are shown in Figure 3.5.

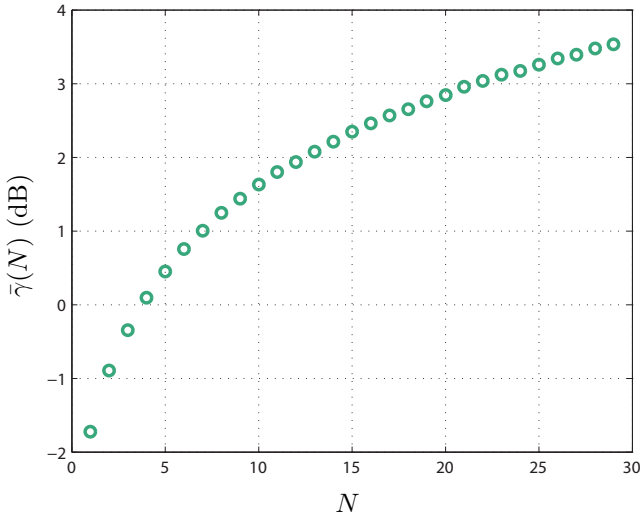


Figure 3.5: Stop thresholds for the Partially Informed Strategy, maximizing the capacity and assuming correlated SNR samples (Case 1B). If the robot has N future samples to select from, it should stop at x_i if $\gamma_i > \bar{\gamma}(N)$. Note that $\bar{\gamma}(0)|_{\text{dB}} = -\infty$, so it is not plotted.

To summarize this section, we have now computed the optimal stopping thresholds for the following proposed strategy for Cases 1A and 1B:

Partially Informed Strategy: Stand still at x_j while $\Delta(t) > -\Delta_{\max}$ and $\gamma_j > \bar{\gamma}(N)$, where N is computed from $\Delta(t)$ as in (3.2). Otherwise, continue to x_{j+1}.

Note that the robot will never violate the tracking constraint by exceeding $\Delta_{\max}$, since the stop threshold for $N = 0$ is defined as 0. We now continue to the cases when the robot has full information of the SNR as function of the position.

Full Channel Knowledge (Cases 2A and 2B)

If the robot knows the SNR along the whole path, we can expect higher resulting channel utility than when only the distribution is known. With full channel knowledge, the optimal strategy does not depend on the choice of utility:

Fully Informed Strategy: Stand still at x_j while $\Delta(t) > -\Delta_{\max}$ and $\gamma_j = \max\{\gamma_j, \ldots, \gamma_{j+N}\}$, where N is computed from $\Delta(t)$ as in (3.2). Otherwise, continue to x_{j+1}.

As above, the robot will never violate the tracking constraint by exceeding $\Delta_{\max}$, since if $N = 0$, the second stopping condition is vacuously satisfied. We now continue to analyzing the resulting performance of each stopping strategy.

3.3 Performance Analysis

In this section, we derive the resulting SNR distributions for each case, under the proposed stopping strategies above. The distributions can be used to compute the expectation with respect to any utility. We will give some examples of this in Section 3.4.

No Channel Information (Case 0)

When the motion is independent of the SNR, the resulting SNR has the unbiased original distribution:

$$f_\gamma^0(\gamma) = f_\gamma^N(\gamma)$$

We will use this later, assuming that whenever the robot drives at constant velocity, it experiences the SNR distribution f_γ^N.

Known SNR Distribution

To derive the resulting SNR distribution, we first compute the distribution of the value of N when the robot arrives at a sample. Using this, we can derive the CDF of the SNR experienced while standing still. Finally, we add the influence of the SNR distribution experienced when driving, as derived above.

To begin, we note that when standing still, the decision instances for the robot occur when $\Delta_{\max} - \Delta(t)$ is an even multiple of $\delta(1 - v_{\text{ref}}/v_{\max})$, *i.e.*, with constant intervals. If the robot decides to drive, $\Delta(t)$ increases by

$$\frac{\delta}{v_{\max}}(v_{\max} - v_{\text{ref}}) = \delta\left(1 - \frac{v_{\text{ref}}}{v_{\max}}\right).$$

Thus, the robot always stands still for a constant time between decision instances. Hence, to find the SNR distribution when standing still, we can find the distribution of the SNR for each stop interval, as they have equal duration. Further, N increases by one for each stop interval, and decreases by one if the robot drives. This will be used in the analysis.

We define N_j as the value of N when the robot arrives at the sample position x_j and stop_j as the event that, at a decision instance, the robot chooses to stand still at x_j. We further define the highest possible value of N as

$$N_{\max} \triangleq \left\lfloor \frac{2\Delta_{\max}}{\delta(1 - v_{\text{ref}}/v_{\max})} \right\rfloor.$$

To model the fact that the robot is forced to drive when $\Delta(t) \leq -\Delta_{\max}$, we can assume that $\bar{\gamma}(N_{\max}) = \infty$. The cases of sparse and dense sampling are now treated separately.

Sparse Sampling (Case 1A)

If and only if the robot moves to the next sample position, N decreases by one. It is therefore a necessary and sufficient condition for $N_j \leq n$ that

$$\gamma_{j-k} \leq \bar{\gamma}(n+k) \;\forall\; k \in \{1, \ldots, k_{\max}\},$$

where $k_{\max} \triangleq N_{\max} - n - 1$. Note that these are all independent events. The condition is necessary since if there is any $k \in \{1, \ldots, k_{\max}\}$ such that $\gamma_{j-k} > \bar{\gamma}(n+k)$, then $N_{j-k+1} > n + k - 1$, which implies that $N_j > n$. It is also sufficient, since it implies that $N_{j-k} \leq n + k$. Hence, the probability that $N_j \leq n$ is

$$P(N_j \leq n) = \prod_{k=1}^{k_{\max}} P(\gamma_{j-k} < \bar{\gamma}(n+k)). \tag{3.3}$$

We now have a distribution for N_j, *i.e.*, the value of N as the robot arrives at x_j. Figure 3.6 shows how N_j and γ_j determine the number of stop intervals that the

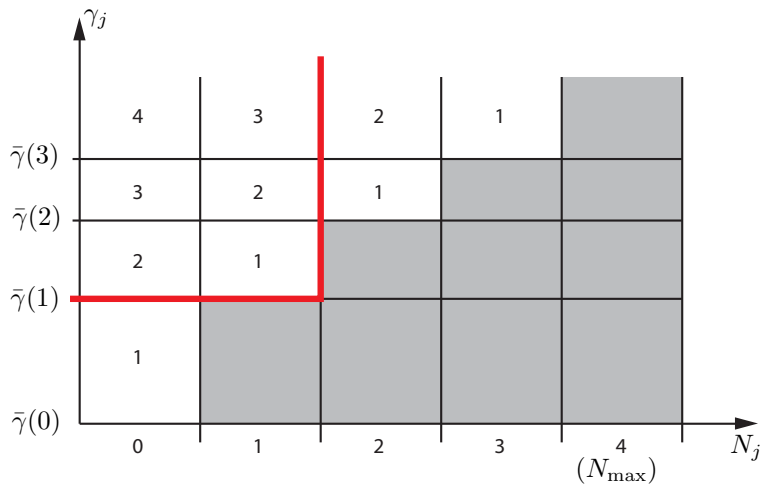


Figure 3.6: The number of time slots that the robot stands still at a sample position x_j is determined by N_j, the value of N when it arrives, and γ_j. This illustrates the case $N_{\max} = 4$ and the thick red line indicates the half-open set where $N_j \leq 1$ and $\gamma_j > \bar{\gamma}(1)$.

robot spends at x_j. The CDF of γ_j, given that the robot decides to stand still, can thus be computed as follows:

$$P(\gamma_j < z|\text{stop}_j)$$
$$= \frac{\sum_{n=0}^{N_{\max}-1} \sum_{k=0}^{k_{\max}} (k+1) P(N_j = n) P(\bar{\gamma}(n+k) < \gamma_j < \bar{\gamma}(n+k+1) \wedge \gamma_j < z)}{\sum_{n=0}^{N_{\max}-1} \sum_{k=0}^{k_{\max}} (k+1) P(N_j = n) P(\bar{\gamma}(n+k) < \gamma_j < \bar{\gamma}(n+k+1))}$$

Here, each term is weighted by $k+1$, which is the number of stop intervals corresponding to each combination of N_j and γ_j. The double sums can be simplified to a single sum, computed over half-open sets as the one indicated by the thick red line in Figure 3.6. This yields

$$P(\gamma_j < z|\text{stop}_j) = \frac{\sum_{n=0}^{N_{\max}-1} P(N_j \leq n) P(\bar{\gamma}(n) < \gamma_j < z)}{\sum_{n=0}^{N_{\max}-1} P(N_j \leq n) P(\gamma_j > \bar{\gamma}(n))},$$

which can be computed using the CDF (2.7), (3.3) and

$$P(\bar{\gamma}(n) < \gamma_j < z) = \max\{0, P(\gamma_j < z) - P(\gamma_j < \bar{\gamma}(n))\}.$$

Hence the PDF for γ when standing still at some sample position x_j can be computed as

$$f_{\text{stop}}^{\text{1A}}(z) = \frac{\mathrm{d}}{\mathrm{d}z} P(\gamma_j < z|\text{stop}_j). \tag{3.4}$$

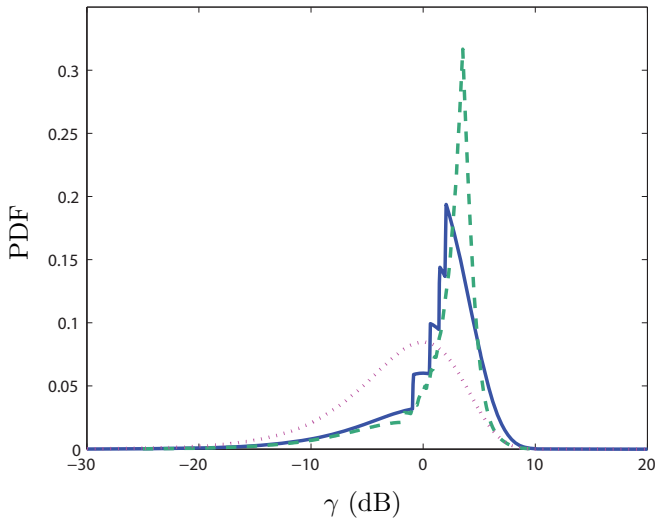


Figure 3.7: Resulting SNR distributions for the cases of *a priori* knowledge of the channel statistics and a maximal tracking error of $\Delta_{\max} = \lambda$. The blue solid line shows $f_\gamma^{1\mathrm{A}}(\gamma)$, with 5 samples taken $\lambda/2$ apart, so they are considered independent. The green dashed line shows $f_\gamma^{1\mathrm{B}}(\gamma)$, with 30 correlated samples taken $\lambda/12$ apart. The Nakagami distribution (dotted magenta line), is included for comparison. It corresponds to driving at constant velocity, as in Case 0.

To find the overall SNR distribution for Case 1A, we recall that the fraction $v_{\mathrm{ref}}/v_{\max}$ of the time is spent driving, when the robot experiences the SNR distribution f_γ^N. The rest of the time is spent standing still at an SNR sample, whose distribution is given by (3.4). The resulting PDF is

$$f_\gamma^{1\mathrm{A}}(\gamma) = \left(1 - \frac{v_{\mathrm{ref}}}{v_{\max}}\right) f_{\mathrm{stop}}^{1\mathrm{A}}(\gamma) + \frac{v_{\mathrm{ref}}}{v_{\max}} f_\gamma^N(\gamma), \tag{3.5}$$

which is illustrated by the blue solid line in Figure 3.7. The figure assumes $\delta = \lambda/2$, $\Delta_{\max} = \lambda$ and $v_{\mathrm{ref}}/v_{\max} = 0.2$.

We now move on to the case of dense sampling, where the samples can no longer be assumed independent.

Dense Sampling (Case 1B)

The analysis is similar to Case 1A, but using conditional probabilities. As above,

$$P(N \leq n|\gamma_j) = P(\gamma_{j-1} < \bar{\gamma}(n+1)|\gamma_j) \times \prod_{k=2}^{k_{\max}} P(\gamma_{j-k} < \bar{\gamma}(n+k)|\gamma_{j-k+1} < \bar{\gamma}(n+k-1)).$$

After simplification, the CDF of γ_j, given that the robot is standing still at x_j is

$$P(\gamma_j < z|\text{stop}_j) = \frac{\sum_{n=0}^{N_{\max}-1} P(N_j \leq n|\bar{\gamma}(n) < \gamma_j < z) P(\bar{\gamma}(n) < \gamma_j < z)}{\sum_{n=0}^{N_{\max}-1} P(N_j \leq n|\gamma_j > \bar{\gamma}(n)) P(\gamma_j > \bar{\gamma}(n))}.$$

The conditional probabilities above can be computed using the bivariate Nakagami PDF $f^N_{\gamma_1,\gamma_2}(\gamma_1, \gamma_2)$ (Simon and Alouini, 1998):

$$P(\gamma_1 < z_1|z_2 < \gamma_2 < z_2') = \frac{\int_0^{z_1} \int_{z_2}^{z_2'} f^N_{\gamma_1,\gamma_2}(s,t)\mathrm{d}t\mathrm{d}s}{\int_0^{\infty} \int_{z_2}^{z_2'} f^N_{\gamma_1,\gamma_2}(s,t)\mathrm{d}t\mathrm{d}s} \tag{3.6}$$

As in Case 1A, the robot will spend a fraction $v_{\text{ref}}/v_{\max}$ of the time driving, and the rest of the time standing still. The resulting overall PDF for the SNR is then

$$f^{\text{1B}}_{\gamma}(\gamma) = \left(1 - \frac{v_{\text{ref}}}{v_{\max}}\right) \frac{\mathrm{d}}{\mathrm{d}z} P(\gamma_j < z|\text{stop}_j) + \frac{v_{\text{ref}}}{v_{\max}} f^N_{\gamma}(\gamma), \tag{3.7}$$

which is plotted as a green dashed line in Figure 3.7. The figure assumes $\delta = \lambda/12$, $\Delta_{\max} = \lambda$ and $v_{\text{ref}}/v_{\max} = 0.2$.

Full Channel Knowledge

Under the Fully Informed Strategy, if the robot chooses to stop at a sample γ_j, this is the best out of $N_{\max}$ samples. Selecting the best out of $N_{\max}$ SNR samples taken with a mobile robot is equivalent to selection combining diversity, where a receiver selects the best out of $N_{\max}$ antennas. For antennas separated enough so they can be independent, the analysis is straightforward, as we will see below. For correlated antennas, as discussed in Chapter 2, we use an approximation to compute the joint CDF.

Sparse Sampling (Case 2A)

To find the PDF of the best out of $N_{\max}$ independent samples, we start by computing the CDF of a single sample. Hence, the PDF of the best sample is

$$f_{\text{stop}}^{\text{2A}}(\gamma) \triangleq \frac{\mathrm{d}}{\mathrm{d}z}\left[P(\gamma < z)^{N_{\max}}\right].$$

As above, the robot will spend a fraction $v_{\text{ref}}/v_{\max}$ of the time driving at constant velocity, so the resulting PDF of the SNR will in this case be

$$f_{\gamma}^{\text{2A}}(\gamma) = \left(1 - \frac{v_{\text{ref}}}{v_{\max}}\right) f_{\text{stop}}^{\text{2A}}(\gamma) + \frac{v_{\text{ref}}}{v_{\max}} f_{\gamma}^{N}(\gamma). \tag{3.8}$$

This is illustrated in Figure 3.8, which assumes that $\delta = \lambda/2$, $\Delta_{\max} = \lambda$ and $v_{\text{ref}}/v_{\max} = 0.2$. We now turn to the case of correlated samples.

Dense Sampling (Case 2B)

To compute the PDF for the best out of $N_{\max}$ correlated SNR samples, we use the approximation of exponential correlation as discussed above. The correlation coefficient of two adjacent SNR samples is $\rho = J_0^2(2\pi\delta/\lambda)$, where δ is the inter-sample distance. We then approximate the correlation between any two samples γ_j and γ_i as $\rho^{|j-i|}$. This allows us to approximate the multivariate Nakagami CDF $P(\gamma_1 < z, \ldots, \gamma_{N_{\max}} < z)$ (Aalo, 1995). Similar to the above case, the resulting PDF is

$$f_{\gamma}^{\text{2B}}(\gamma) = \left(1 - \frac{v_{\text{ref}}}{v_{\max}}\right) f_{\text{stop}}^{\text{2B}}(\gamma) + \frac{v_{\text{ref}}}{v_{\max}} f_{\gamma}^{N}(\gamma), \tag{3.9}$$

where

$$f_{\text{stop}}^{\text{2B}}(\gamma) \triangleq \frac{\mathrm{d}}{\mathrm{d}z} P(\gamma_1 < z, \ldots, \gamma_{N_{\max}} < z).$$

It is illustrated in Figure 3.8, assuming that $\delta = \lambda/12$, $\Delta_{\max} = \lambda$ and $v_{\text{ref}}/v_{\max} = 0.2$.

Comparing Figures 3.7 and 3.8, we see that both the Partially Informed Strategy and the Fully Informed Strategy produce a positive bias in the resulting PDF, compared to the Nakagami distribution. The staircase effect in the PDFs for the Partially Informed Strategy comes from the discrete thresholds, which are denser

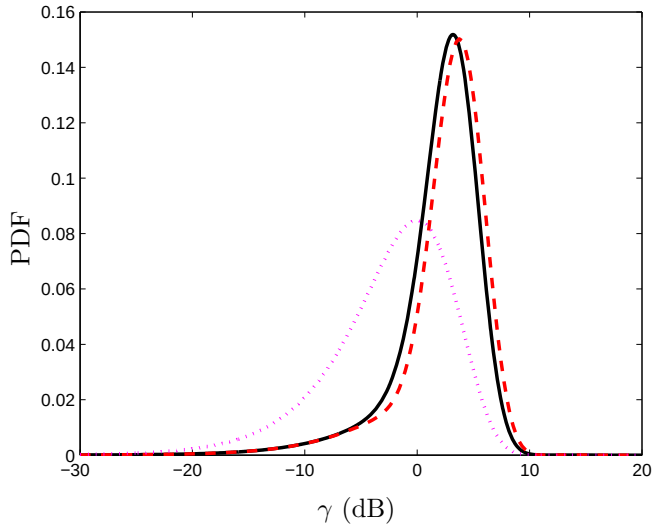


Figure 3.8: An example of probability density functions $f_\gamma^{2\mathrm{A}}(\gamma)$ (Case 2A, black solid line) and $f_\gamma^{2\mathrm{B}}(\gamma)$ (Case 2B, red dashed line). This shows the PDF of the resulting SNR when the robot has full knowledge of the SNR waveform, *e.g.*, if this has been measured on a previous pass. The maximum tracking error is $\Delta_{\max} = \lambda$, so in Case 2A, the robot always drives to the best of the 5 independent samples that it can reach, and in Case 2B it has 30 correlated samples to choose from. The PDF for Nakagami fading (dotted magenta line), is included for comparison. It corresponds to driving at constant velocity, as in Case 0.

in Case 1B than in Case 1A, making the resulting PDF smoother. The quantitative differences between the cases will be clearer in the next section, where we use the PDFs to compute expectations of the channel utilities.

3.4 Simulations

To illustrate the performance of the proposed strategies, we have compared simulations with the analytically computed expected utilities under various tracking constraints. We have also simulated a trajectory of the system under the Partially Informed Strategy, to clarify how it works. But first we describe the simulation setup.

Simulation Setup

In all simulations, we have assumed a distance of $\delta = \lambda/2$ between samples for sparse sampling and $\delta = \lambda/12$ for dense sampling. In the latter case, this yields a correlation coefficient of $\rho = 0.87$ between adjacent samples. We also assume Rayleigh fading ($m = 1$) and, unless specifically stated, an average SNR of $\Omega = 0$ dB. Further, we assume that $v_{\text{ref}}/v_{\max} = 1/5$, so 20% of the time is spent driving. The bandwidth was chosen as $B = 1.15$ Hz so the capacity was normalized to 1 bit/s in Case 0. We assumed QPSK modulation and a packet length of $N_{\text{P}} = 65$ bytes and normalized the throughput by assuming $R_0 = 1$ packet/s. Sparse sampling was simulated by generating independent samples from the distribution f_γ^N. To generate correlated samples, we used the Rayleigh fading simulator proposed by Zheng and Xiao (Zheng and Xiao, 2003), setting the number of scatterers to 100.

Example Trajectory

To illustrate the Partially Informed Strategy, we have simulated a trajectory of the system, with $\Delta_{\max} = \lambda$. We have chosen to maximize the capacity, *i.e.*, $U = U_{\text{C}}$. Figure 3.9 shows the relative position, $\Delta(t)$ and the instantaneous SNR, $\gamma(t)$ during 8 s of the trajectory. The robot stops when it finds a high SNR value, creating long periods of high capacity. This biases the SNR distribution, compared to the Nakagami distribution. When the robot stands still, $\Delta(t)$ decreases. The SNR is constant, but the stop threshold $\bar{\gamma}(N)$ increases as the number, N, of reachable samples in front of the robot increases. Eventually the robot starts driving again, either because the SNR is below the threshold or if $\Delta(t) = -\Delta_{\max}$. Similarly, the robot will always stop before driving past $\Delta(t) = \Delta_{\max}$ since $\bar{\gamma}(N = 0) = 0$.

Utility vs. Tracking Tolerance

To illustrate how the expected utility, $U(\gamma)$, improves when the tracking tolerance, $\Delta_{\max}$, is increased, we have used the PDF:s (2.5), (3.5), (3.7), (3.8), (3.9) to compute the expectation. We have also compared with simulations, for each of the five cases. The results, both for capacity, $U = U_{\text{C}}$, and throughput, $U = U_{\text{T}}$, show that the greatest gain in utility is achieved already when allowing small deviations from the reference. Table 3.2 shows $N_{\max}$, *i.e.*, the maximal number of reachable samples for the robot, for each value of $\Delta_{\max}$.

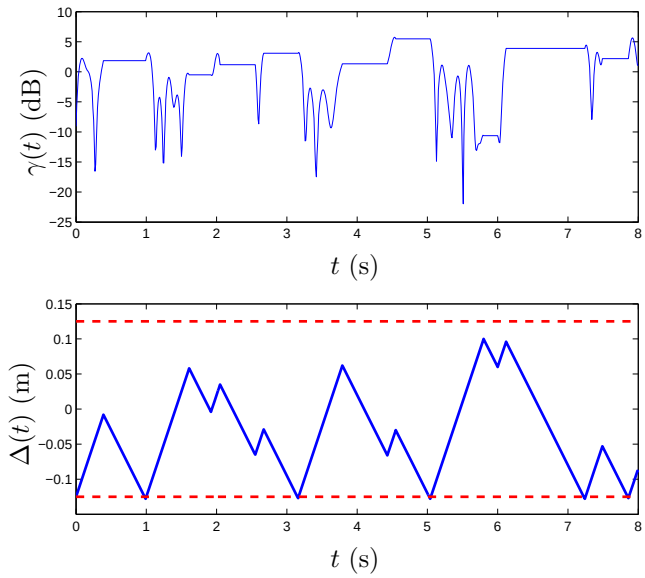


Figure 3.9: An example trajectory when using the Partially Informed Strategy. When the robot drives, the SNR, γ, varies. By stopping at positions where γ is high, it can increase the average link capacity. The relative position, $\Delta(t)$, increases when the robot drives, and decreases when it stands still. The robot samples $\gamma(x_j)$ and stops if it exceeds the stop threshold. The dashed red lines depict $\pm\Delta_{\max}$, which are the tracking error bounds.

Table 3.2: Maximal number of reachable samples, $N_{\max}$

$\Delta_{\max}$	$\lambda/4$	$\lambda/2$	λ	$3\lambda/2$	2λ	3λ	4λ
Independent	1	2	5	7	10	15	20
Correlated	7	15	30	45	60	90	120

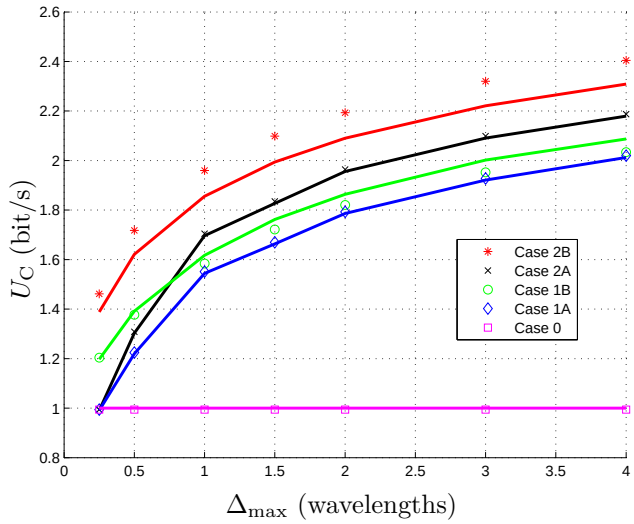


Figure 3.10: Expected link capacity, U_C, as a function of the tracking error tolerance, Δ_{max}, for the five cases considered. The bandwidth B was chosen so the capacity was 1 bit/s in the nominal Case 0. Analytical predictions are drawn as solid lines, and simulations are shown as markers in the same color.

Figure 3.10 shows the resulting expected capacity, U_C, in all five cases. As expected, knowledge of the full waveform yields higher expected capacity than just knowing the SNR distribution. This is true both for sparse sampling (Cases 1A and 2A) and dense sampling (Cases 1B and 2B). Also, dense sampling provides more information than sparse sampling, which results in higher capacity. The differences between simulation and theory for Cases 1B and 2B come from the approximations described above, used to handle the correlation between non-adjacent samples. We finally note that for wireless communication at 2.4 GHz, allowing a tracking error in the order of 0.5 m (4λ) can offer a doubling of the link capacity, even with knowledge only of the fading distribution.

Figure 3.11 shows analytical predictions and simulation results of how the expected throughput, U_T, varies with the tracking error tolerance, Δ_{max}. Here we have assumed a mean SNR of $\Omega = 3$ dB, which yields a throughput of 0.11 packets/s. Similar to the case of capacity, the throughput increases most rapidly for small tracking error tolerances. Both the analytical prediction and the simulations show that Case 1B yields higher throughput than Case 2A. This means that full knowledge of the SNR is less important than sampling densely when optimizing for throughput. A possible explanation is that $U_T(\gamma)$ transitions from 0 to R_0 in a very narrow SNR interval. Thus, exploring and stopping at the first SNR sample with reasonable throughput yields similar performance as finding the highest SNR

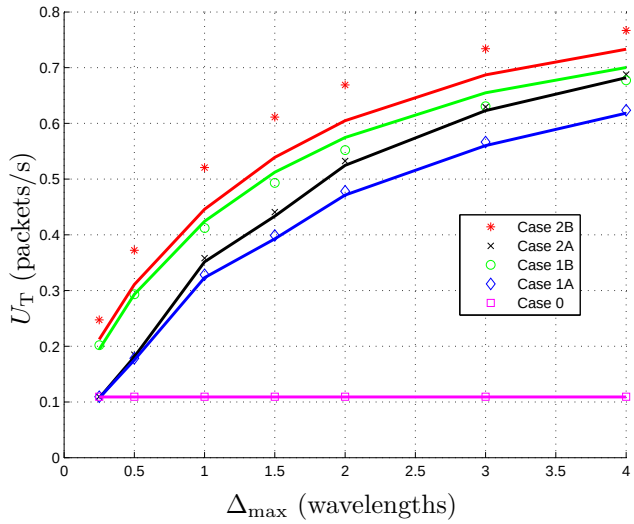


Figure 3.11: Expected link throughput, U_T, as a function of the tracking error tolerance Δ_{max}, for the five cases considered. The packet transmission rate was chosen as $R_0 = 1$ packet/s for normalization. Analytical predictions are drawn as solid lines, and simulations are shown as markers in the same color.

inside the tracking window. As above, we note that for a 2.4 GHz link with low nominal throughput, allowing a tracking error of 0.5 m (4λ) can yield a throughput improvement of five times or more.

3.5 Experimental Evaluation

The first objective of our experiments was to validate the model of static Nakagami fading by measurements in various environments. The second objective was to implement the Partially Informed Strategy with sparse sampling (Case 1A) and compare it to the Uninformed Strategy (Case 0). We have tested this in environments that exhibit static fading, as well as where the fading is more time-varying, to investigate the robustness of the approach. Before going into the experiment results, we describe the setup.

Experiment Setup

We used an indoor differential drive robot, equipped with a laptop and two Tmote Sky transceivers. Specifications for the Tmote Sky are given in Chapter 2. As illustrated in Figure 3.12, one transceiver was used for measurements. It listened to another Tmote Sky which acted as base station, broadcasting $R_0 = 64$ packets/s,

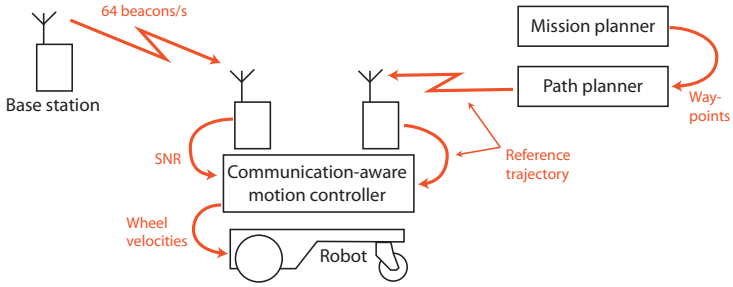


Figure 3.12: Experiment setup. A communication-aware motion controller onboard the robot computes the wheel velocities based on the reference trajectory and the SNR of the beacons from the base station. The reference trajectory in transmitted from a path planner, which finds obstacle-free trajectories between waypoints specified by a mission planner.

each $N_P = 65$ bytes long. The other transceiver, using another channel, received lateral steering commands for the robot from an operator, acting as path planner. Letting the robot autonomously control its forward velocity while manually steering it along the path, allowed quick setup at various test locations, without the need for any navigation other than odometry. The paths were chosen such that different segments were never closer than $\lambda/2$, to avoid undesired correlation between measurements. The robot measured the received signal strength of each received packet. Measurements of the background noise levels in the various locations indicate a constant noise level of -96 dBm in the Tmote Sky, with a standard deviation of about 1 dB.

The experiments were carried out at eleven sites on the KTH campus, chosen to represent different types of surroundings and various degrees of motion in the environment. Below, we list the sites and some estimated parameters for the fading distributions, to validate our communication model. Figures 3.13 and 3.14 contain photos from each site.

Model Validation

The model of Nakagami fading is well-established, but we wanted to validate it and the assumption of static fading at the locations where we later performed the communication-aware motion planning experiments. To do this, we have compared the distribution of the RSS while driving to that when standing still, at eleven different locations on the KTH campus.

In the experiments, we measured the RSS while driving along a path. To isolate the multipath fading from the effects of path loss and shadowing, we subtracted the moving average of the RSS, computed over a window of 1 m. This length was empirically chosen so the path loss and shadowing would be approximately constant inside the window. We then estimated the Nakagami parameter m as in (2.6) for

Figure 3.13: Sites 1 through 7 for the experiments, starting from the top left. These sites had very little motion during the experiments.

Figure 3.14: Sites 8, 9, 10 and 11 for the experiments, starting from the top left. At these sites, people were passing by during the experiments, and Site 11 was significantly more crowded than the others.

the deviations from the moving average and compared the resulting CDF, $F_\gamma^N(\gamma)$, to the empirical distribution function, $F_\gamma(\gamma)$, of our observations. As a metric of the goodness of fit, we have used the Kolmogorov-Smirnov statistic (Massey, 1951)

$$D \triangleq \sup_\gamma |F_\gamma^N(\gamma) - F_\gamma(\gamma)|.$$

Then we measured the RSS while the robot was standing still at five positions along the path, 2 min at each. For each set of measurements, we estimated m for the deviations from the mean. We also estimated m_{tot}, which we define as the Nakagami m-parameter for the total SNR histogram of all five stops. Note that the deviations have a mean of about 0 dB, so m is inversely proportional to the SNR variance. High values of m correspond to a narrow distribution, which means that the assumption on static fading is valid.

The location with the highest value of m_{tot}, *i.e.*, the most static fading, was Site 1, a workshop full of machinery and metal cabinets, but with no people present.

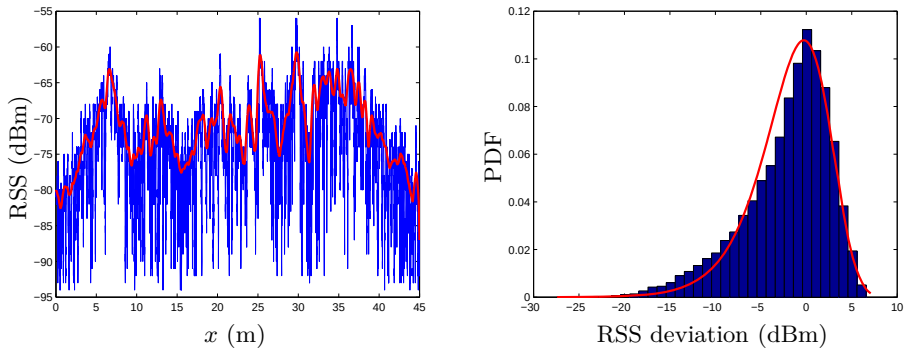


Figure 3.15: Measurements of the received signal strength when driving at constant velocity in Site 1, a cluttered but unattended workshop. The left graph shows the RSS (blue) and the moving average (red). The right graph shows the normalized histogram of the deviations from the moving average (blue) and the corresponding best fit Nakagami PDF (red), with $m = 1.53$.

The results of driving in the workshop are shown in Figure 3.15. On the left are the RSS samples (blue) taken when driving, with the moving average (red) superimposed. On the right is the normalized RSS histogram (blue) with the best fit Nakagami PDF $f_\gamma^N(\gamma)$ (red) superimposed, with $m = 1.53$. The Kolmogorov-Smirnov statistic was $D = 0.048$. Note that $m = 1$ corresponds to Rayleigh fading and that the mean, Ω, is zero, since we consider deviations from the moving average. The RSS recorded when standing still in the workshop is depicted in Figure 3.16. Each subfigure corresponds to a stop of 2 min at some position along the path. The Nakagami parameters, m, are 180, 27, 64, 77 and 240. As the values of m are almost two orders of magnitude higher than those when driving, this indicates that the assumption of static fading is very accurate in this environment.

The least static of the sites, *i.e.*, with the lowest value of m_{tot}, was Site 11, a busy hallway. The experiment was carried out during daytime, with students walking past the robot during the whole experiment. The measured RSS when driving is shown in Figure 3.17 and m was estimated to 1.77. The Kolmogorov-Smirnov statistic was $D = 0.032$. The RSS measured when standing still is shown in Figure 3.18 and m was 17, 19, 19, 8.3 and 13. The values of m are one order of magnitude less than those in the workshop, which makes this the least static of our locations. Nevertheless, even at this site, the RSS variance is about one order of magnitude less when standing still than when driving. In the next subsection, we will see how the various degrees of static fading affect the resulting channel utility.

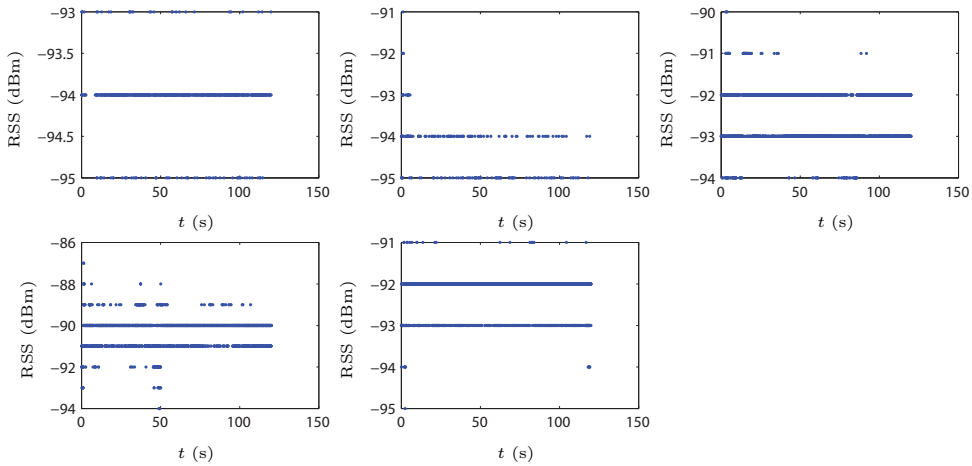


Figure 3.16: Measurements of the received signal strength when standing still at five different locations along the path at Site 1. The Nakagami parameter m for each measurement set is 180, 27, 64, 77 and 240, starting with the top row. High values of m correspond to a narrow RSS distribution, which indicates that this location exhibits static fading.

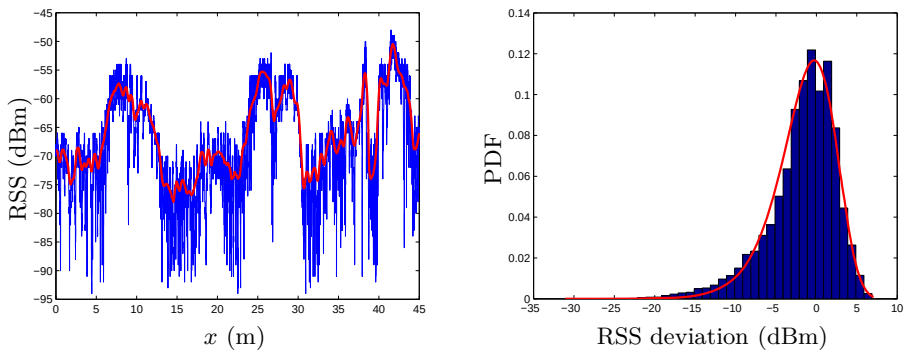


Figure 3.17: Measurements of the received signal strength when driving at constant velocity in Site 11, a crowded hallway. The left graph shows the RSS (blue) and the moving average (red). The right graph shows the normalized histogram of the deviations from the moving average (blue) and the corresponding best fit Nakagami PDF (red), with $m = 1.77$.

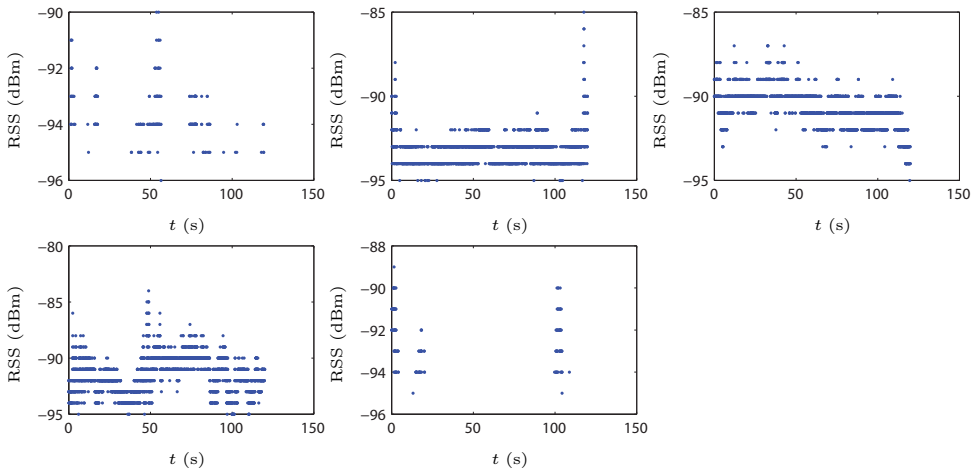


Figure 3.18: Measurements of the received signal strength when standing still at five different locations along the path at Site 11. The Nakagami parameter m is 17, 19, 19, 8.3 and 13, starting with the top row. This shows that this location exhibits less static fading than the workshop. The sparsity of the first and last graph comes from frequent packet losses.

Table 3.3: Measured fading parameters.

Site	Description	Driving		Standing still					
no.	(Activity level)	m	D	m					m_{tot}
1	Workshop (0)	1.53	0.048	180	27	64	77	240	91
2	Corridor (0)	1.92	0.031	100	80	41	18	62	78
3	Storage (0)	1.60	0.021	58	36	360	125	99	75
4	Basement (0–1)	1.36	0.032	117	81	220	46	50	69
5	Hallway (0)	1.96	0.035	12	55	28	92	21	39
6	Basement (0)	1.47	0.033	44	25	24	72	86	36
7	Office (0)	1.26	0.030	40	59	22	32	66	35
8	Hallway (1)	1.54	0.032	47	18	23	26	31	26
9	Lab (1)	1.38	0.024	49	17	27	28	22	23
10	Corridor (1)	1.54	0.030	14	27	68	42	5.2	14
11	Hallway (2)	1.77	0.032	17	19	19	8.3	13	13

Table 3.3 summarizes the results of all validation experiments, with the sites ordered by decreasing m_{tot}. The level of activity at each site is judged on a scale of 0 (empty), 1 (people passing now and then) and 2 (crowded). The table suggests that the estimated Nakagami parameter m_{tot} is useful for reflecting the activity level. The results show that the estimated Nakagami m parameter for the RSS distribution ranges from 1.3 to 2.0, which indicates that the fading is slightly less severe than Rayleigh fading. Each RSS histogram contained about 23 000 samples, so the limit for the Kolmogorov-Smirnov test at the 99% significance level is $D = 0.01$ (Massey, 1951). The measurements thus do not validate the assumption on Nakagami fading, but given that D is only about three times the rejection limit and that the superimposed graphs of the PDF show high similarity between the distributions, we believe that it is a useful approximation. With regards to the assumption on static fading, all sites exhibited values of m that were one or two orders of magnitude larger when standing still than when driving. This indicates that even if the fading is not perfectly static, it makes sense to stop to communicate if the RSS is high. The assumption on static fading is thus meaningful in all locations tested. In the next section, we will see how the degree of static fading affects the resulting channel utility when implementing the Partially Informed Strategy.

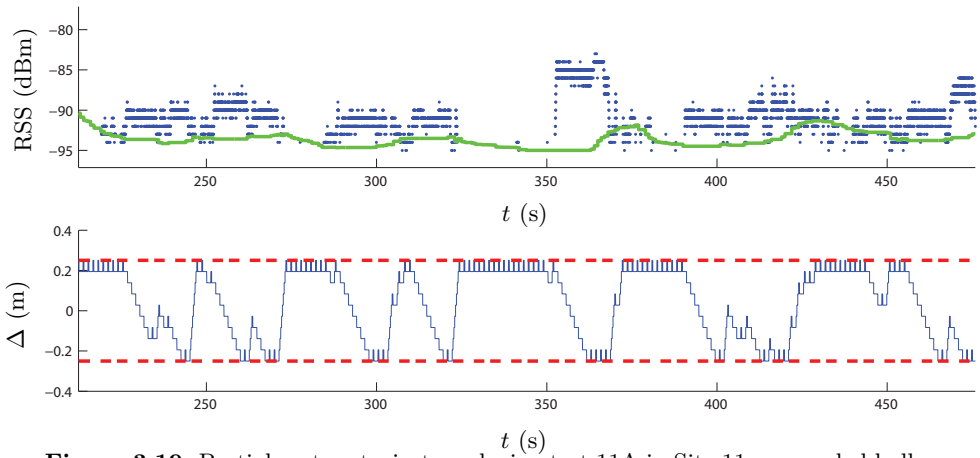


Figure 3.19: Partial system trajectory during test 11A in Site 11, a crowded hallway. The blue dots in the upper graph denote the RSS of received packets and the green line is the lowest stop threshold, corresponding to $\bar{\gamma}(1)$. If the RSS at a sampling point, x_i, is higher than the threshold, the robot stops. Stopping causes periods of high RSS, which increases the average throughput, compared to driving at constant velocity. The lower graph shows the relative position, Δ, which is constrained to $\pm\Delta_{\max}$, shown by red dotted lines.

Results

As a proof of concept and to investigate what happens to the performance of the proposed method when the fading is less static, we have implemented the Partially Informed Strategy for independent samples (Case 1A). As a comparison, we have used the Uninformed Strategy (Case 0). We have allowed a maximum tracking error of $\Delta_{\max} = 2\lambda = 25$ cm and a sample spacing of $\delta = \lambda/4 = 3.1$ cm. The velocities were $v_{\text{ref}} = 2.5$ cm/s and $v_{\max} = 12.5$ cm/s. To adapt the stop thresholds to changes in path loss and shadowing along the path, we have continuously estimated the RSS distribution based on previous measurements. To avoid biasing the estimate, we have only used measurements taken when driving.

Figure 3.19 shows an example trajectory of the system, during test 11A, the first test at Site 11, a crowded hallway. The upper graph shows the RSS (blue) of each received packet. If the robot reached a sampling point x_i and the RSS was higher than the stop threshold, it stopped. As $\Delta(t)$ decreased during the stop, the RSS was compared to increasingly higher thresholds, as defined in the Partially Informed Strategy, until the robot started driving again. The RSS level corresponding to the lowest threshold (green), $\bar{\gamma}(1)$, is included in the graph for illustration. Note how the threshold changed as the estimate of the RSS distribution was updated. Also note the complete outage for 25 s at $t = 325$ s. Due to shadowing, almost all packets were lost in this interval, so the robot followed the forward edge of

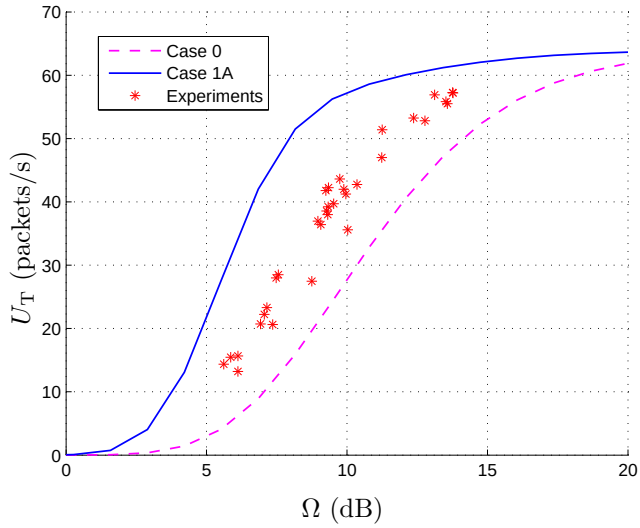


Figure 3.20: Expected throughput, U_T, as function of the mean SNR, Ω, for Cases 0 and 1A when $\Delta_{max} = 2\lambda$. The red stars denote experiment results, equivalent to a power gain of about 3 dB over Case 0.

the tracking window until the outage stopped. This happened very frequently in all tests, reducing the efficiency of the proposed strategy. We believe that these variations due to shadowing are the main reason for the differences between the measured throughput and the analytical prediction, which assumes constant mean SNR, Ω, during the whole experiment.

Table 3.4 summarizes the results of all experiments. It shows the number of the experiment (derived from the site number), a site description, the average throughput of the Uninformed Strategy (Case 0) and that of the Partially Informed Strategy (Case 1A). The maximum throughput is $R_0 = 64$ packets/s. Finally, the table also states the relative throughput improvement of each test. The locations are ordered by decreasing m_{tot} as in Table 3.3, *i.e.*, with the most static fading first.

Figure 3.20 illustrates the experiment results. For comparison, we have plotted how the expected throughput U_T for the Uninformed Strategy (Case 0) and the Partially Informed Strategy with independent samples (Case 1A) vary with Ω for $m = 1.5$ (the median of the measurements). The result of each experiment is shown as a star, at the Ω that corresponds to the throughput of the Uninformed Strategy and the U_T that corresponds to the throughput of the Partially Informed Strategy. The figure shows that the Partially Informed Strategy yields the highest benefits in the transition region, where the throughput is low but not zero. The experiment results are equivalent to an SNR improvement of about 3 dB, *i.e.*, doubling the received signal power.

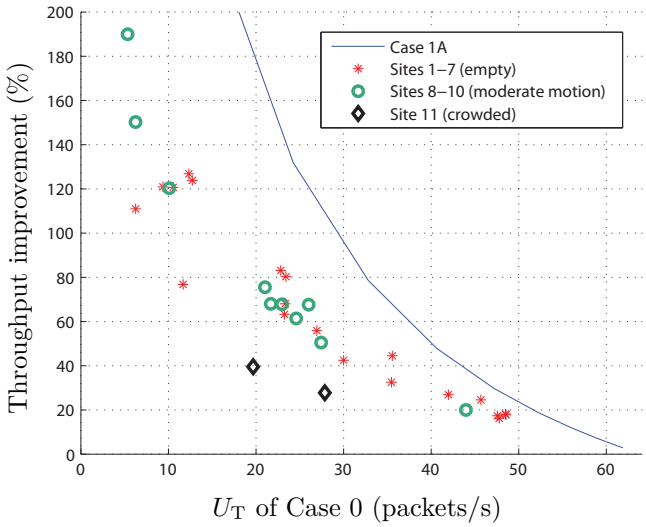


Figure 3.21: Throughput improvement of the experiments, as a function of the nominal throughput of Case 0. The experiment results are categorized according to the judged activity level. The results show that the Partially Informed Strategy can yield significant throughput improvements if the nominal link utility is low. Also, the strategy is robust to moderate levels of motion in the environment.

To further illustrate how the improvements of the Partially Informed Strategy vary with Ω, Figure 3.21 shows the throughput improvement of each experiment, as a function of the nominal throughput, for Case 0. For comparison, we have included the analytic prediction, computed using (3.5), $m = 1.5$ and Ω chosen so that the expected throughput for Case 0 equals the measured throughput. As commented above, this represents an ideal case, when Ω is constant, so there is no change in the path loss or shadowing. The experiment results are categorized according to the judged activity levels, and there is no clear difference between results from sites with activity level 0 or 1. The crowded Site 11 does, however, stand out and yielded less improvements than the trend of the other experiments. This shows that the strategy is robust to moderate violations of the assumption on static fading.

Table 3.4: Experiment results.

Exp. no.	Description (Activity level)	U_T (packets/s) Case 0	Case 1A	Improvement
1A	Workshop	12	28	130%
1B	(0)	11	23	120%
1C		49	57	18%
1D		48	56	16%
2A	Corridor	42	53	27%
2B	(0)	35	47	33%
3A	Storage	36	51	44%
3B	(0)	46	57	25%
3C		23	39	68%
3D		6.3	13	110%
4A	Basement	9.4	21	120%
4B	(0–1)	13	29	120%
5A	Hallway	30	43	42%
5B	(0)	27	42	56%
6A	Basement	23	42	83%
6B	(0)	23	42	80%
6C		48	56	17%
6D		48	57	18%
7A	Office	23	38	63%
7B	(0)	12	21	77%
8A	Hallway	23	39	68%
8B	(1)	25	40	61%
8C		27	41	50%
9A	Lab	5.3	15	190%
9B	(1)	21	37	76%
9C		22	36	68%
9D		44	53	20%
10A	Corridor	26	44	68%
10B	(1)	10	22	120%
10C		6.3	16	150%
10D		4.4	14	230%
11A	Hallway	28	36	28%
11B	(2)	20	27	40%

3.6 Summary

Given a reference trajectory and an associated bound, $\Delta_{\max}$, on the tracking error, we have proposed and analyzed two novel strategies for communication-aware motion planning: If the SNR distribution is known (or can be estimated), the robot can use an optimal stopping approach when sampling, to spend time at positions that maximize the expected channel utility. If the SNR along the trajectory is fully known, the robot should spend as much time as possible at the position with highest SNR inside the tracking window.

The analysis shows that the expected link capacity or throughput can be significantly increased even by allowing small tracking errors, in the order of 2–4 wavelengths. Increasing the tracking tolerance above this yields minor improvements. Having full knowledge of the SNR, as in Cases 2A and 2B, offers some performance improvements, at the price of high requirements on the navigation accuracy and the need to acquire the SNR information beforehand. For both Cases 1A/1B and 2A/2B, increasing the sampling density provides a marginal increase in performance, at the price of significantly higher complexity in the analysis and (for Case 1B) computing the stop thresholds. This is analogous to the analysis of uniform linear antenna arrays, where is can be shown that the spectral efficiency quickly converges as the number of antenna elements in a given space increase (Muharemovic et al., 2008). As shown in Figure 3.20, the proposed strategies are most motivated in the transition region, where the link quality is low but not zero.

Our measurements in various locations validated the model of static multipath fading. The SNR variance was one or two orders of magnitude less when standing still than when driving, which motivates stopping at positions with high SNR to introduce a positive bias in the SNR distribution over time. The assumption on the fading being Nakagami distributed was not formally validated, but the results nevertheless show that it is a good approximation. We have experimentally compared implementations of the proposed Partially Informed Strategy and the nominal Uninformed Strategy. The results show that the Partially Informed Strategy yields throughput improvements of over 100% when the nominal link utility is low, and that it is robust to moderate levels of motion in the environment.

The proposed stop-and-go motion represents a type of antenna diversity over time, which closely parallels other types of diversity, as described in Chapter 2. As mentioned above, the analysis of Cases 2A and 2B directly corresponds to selection combining, where a receiver uses the antenna with the best SNR. And the optimal stopping in Cases 1A and 1B bears similarity to switch-and-examine diversity (Alexandropoulos et al., 2010), except that the robot does not have the option of going back to previous samples.

During the experiments, we observed that shadowing caused periods of complete outage, when the proposed strategies were ineffective. Since the analysis assumes constant path loss and shadowing, this appears to be the main reason for the difference between the measured and predicted throughput improvements. For future work, it would be interesting to increase the tracking tolerance to the spatial scale

of the shadowing, to allow the robot to quickly pass areas with unfavorable shadowing. Another interesting extension would be to use transmitters with higher output power, such as mobile phone base stations, which would allow the robot to operate farther from the base station. This can be expected to lead to more constant path loss and less pronounced shadowing, which would be beneficial for the performance of the proposed strategies.

Chapter 4

Motion Planning under Multipath Fading: Probabilistic Tracking Constraints

In this chapter, we continue the investigation of tracking a reference trajectory in a multipath fading environment. We restrict the problem to a simple strategy that is well suited for implementation on robots with low processing power and slow channel quality sensors: Let the robot drive at a constant velocity v_d for a constant time τ_d and then stop. After stopping, it measures the SNR and then stands still for a time $\tau_s(\gamma)$ before starting over and driving again. By choosing a *stop-length policy* $\tau_s(\gamma)$ that stays longer at positions with high SNR, we can improve the average link quality, compared to driving at constant velocity.

To maintain reference tracking, we will impose a probabilistic tracking error constraint, as opposed to the deterministic constraint in Chapter 3. The stop-length policy is chosen so that the expected velocity equals the reference velocity. Then we propose a cascaded feedback controller to compensate for any drift away from the reference position. This chapter also considers time-triggered stopping, as opposed to stopping when we find high SNR, as in Chapter 3, or stopping on demand when a data buffer is filling up, as in Chapter 5. This has some important advantages. First, time-triggered stopping does not require searching for local maxima of γ. By stopping and then sampling, we can use slow sensors and it is easy for the robot to stand still to maintain the SNR if it is found to be high. Second, as we will see below, the controller architecture is simple and implementable on resource-constrained platforms. Third, by ensuring that the constant driving distance $v_d\tau_d$ between SNR samples is long enough, the samples can be regarded as independent, which facilitates the statistical analysis.

In the following section, we describe the strategy of time-triggered stopping and the proposed controller architecture. We then formulate the problems of designing each subsystem of the communication-aware controller. First we design the rule for stopping times in Section 4.2, then we design a feedback controller for the tracking error in Section 4.3. The last subsystem is a channel estimator, designed in Section 4.4. We report experiments on the system in Section 4.5 and then summarize.

4.1 Preliminaries

Here we define time-triggered stopping and the proposed controller architecture, then we formally state the problem.

We assume that the relative position, $\Delta(t)$, of the robot follows (3.1) and define time-triggered stopping as moving with the velocity

$$v(t) = \begin{cases} 0 \text{ when } t_k \leq t < t_k + \tau_s(\gamma[k]) \\ v_d \text{ else,} \end{cases}$$

for all $k \geq 0$, where the stop time instances are

$$t_k = k\tau_d + \sum_{m=1}^{k} \tau_s(\gamma[m-1])$$

and $\gamma[k]$ denotes the SNR, sampled at time t_k.

Note that $\gamma[k]$ is a discrete-time stochastic process and the distance $v_d\tau_d$ between samples is assumed long enough for the samples to be independent. We assume static Rayleigh fading with distribution (2.4) for simplicity, but a similar approach could be used for a more general SNR distribution. The choices of stop time lengths τ_s do not affect the sampling. To allow comparison with the experiments, we will optimize the throughput, $U_{\mathrm{T}}(\gamma)$, defined in (2.3). We use a packet rate of $R_0 = 64$ packet/s and assume QPSK modulation, with bit error rate given by (2.9).

We are now ready to specify the particular system architecture that we consider in this paper, and formally state the associated control and estimation problems.

System Architecture

The proposed cascaded control structure is illustrated in Figure 4.1. The core of the system is the stop-length controller, which drives the robot and determines the stop durations, depending on the measured SNR, γ. To avoid drift away from the reference position, the stop-length controller should maintain a given mean stop length, with bounded variance. To determine what SNR values are worth stopping for, it needs some information about the distribution of γ. Assuming Rayleigh fading, the only parameter required is the mean SNR, Ω, which is estimated by a channel estimator. If the estimate, $\hat{\Omega}$, is high, the robot will not stop very often. If the estimate is low, the robot will think that many of the samples it finds are unusually high, and thus stop more often. Hence, by adding a bias to the estimate, we can affect the motion of the robot. This is what the tracking controller does, to regulate the tracking error.

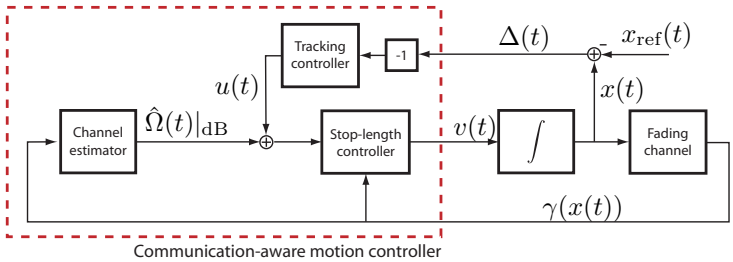


Figure 4.1: Control architecture for time-triggered stopping. The stop-length controller determines how long the robot should stand still at each equidistant sampling position. The decision is based on the instantaneous SNR compared to the average, as computed by the channel estimator. A tracking controller uses feedback to maintain reference tracking.

Problem Formulation

The design of the communication-aware motion controller in Figure 4.1 can be reduced to the following three subproblems:

1. Design a *stop-length controller* by finding a stop-length policy $\tau_s(\gamma)$ that maximizes the expected throughput while maintaining a given mean and variance of τ_s.

2. Design a *tracking controller* that stabilizes the system around $\Delta = 0$.

3. Design a causal *channel estimator* that uses samples of γ compute an estimate $\hat{\Omega}$ of Ω.

In the next section, we describe the resulting stop-length controller. In Section 4.3, a tracking controller is derived and Section 4.4 gives a suggested channel estimator.

4.2 Stop-Length Controller

In this section, we first find the optimal stop-length policy and then approximate it by a simpler threshold policy, better suited for implementation on a resource-constrained robot. We compute the expected performance of both, which shows that the threshold approximation gives negligible loss of performance. This is the inner loop of the system in Figure 4.1. It works on a shorter time-scale than the estimator and feedback controller. In this section, we therefore assume that the fading is weak-sense stationary, so the mean SNR, Ω, is constant. We also assume that it is known, but later we will replace it by an estimate from the channel estimator.

Optimal Stop-Length Policy

The optimal stop-length policy should maximize the throughput and maintain the velocity v_{ref}.

When stopping at t_k, the throughput is $U_{\text{T}}(\gamma[k])$. When driving, the mean throughput is

$$\bar{U}_{\text{T}} \triangleq \text{E}[U_{\text{T}}(\gamma)] = \int_0^\infty [1 - Q(\sqrt{\gamma})]^{8N_P} f_\gamma(\gamma)\ \text{d}\gamma,$$

where N_P is the number of bytes in each packet.

The average throughput of M stop and drive cycles is the total number of packets received, divided by the total time:

$$\frac{\sum_{k=1}^{M} \left(U_{\text{T}}(\gamma[k])\tau_s(\gamma[k]) + \bar{U}_{\text{T}}\tau_d\right)}{\sum_{k=1}^{M} \left(\tau_s(\gamma[k]) + \tau_d\right)}. \tag{4.1}$$

In the limit as $M \to \infty$, according to the strong law of large numbers, this converges with probability one to

$$\frac{\text{E}[U_{\text{T}}(\gamma)\tau_s(\gamma)] + \bar{U}_{\text{T}}\tau_d}{\text{E}[\tau_s(\gamma)] + \tau_d}. \tag{4.2}$$

To maintain the average velocity v_{ref}, we impose the constraint $\text{E}[\tau_s(\gamma)] = \tau_d$. Thus maximizing the average throughput is equivalent to maximizing $\text{E}[U_{\text{T}}(\gamma)\tau_s(\gamma)]$. To reduce the deviation from the reference position, we also limit the stop time variance, denoted as σ^2. The problem of designing the stop-length policy $\tau_s(\gamma)$ can then be stated as:

$$\begin{aligned} \max_{\tau_s(\gamma)} \ & \text{E}[U_{\text{T}}(\gamma)\tau_s(\gamma)] \\ \text{s.t.} \ & \text{E}[\tau_s(\gamma)] = \tau_d \\ & \text{E}[(\tau_s(\gamma) - \tau_d)^2] \le \sigma^2 \\ & \tau_s(\gamma) \ge 0 \ \forall\ \gamma \end{aligned}$$

or equivalently

$$\begin{aligned} \max_{\tau_s(\gamma)} \ & \int_0^\infty U_{\text{T}}(\gamma)\tau_s(\gamma) f_\gamma(\gamma)\text{d}\gamma \\ \text{s.t.} \ & \int_0^\infty \tau_s(\gamma) f_\gamma(\gamma)\text{d}\gamma = \tau_d \\ & \int_0^\infty (\tau_s(\gamma) - \tau_d)^2 f_\gamma(\gamma)\text{d}\gamma \le \sigma^2 \\ & \tau_s(\gamma) \ge 0 \ \forall\ \gamma. \end{aligned}$$

To solve the optimization, we quantize γ in steps of 1 dB, which yields a convex quadratically constrained quadratic program (Boyd and Vandenberghe, 2004). An

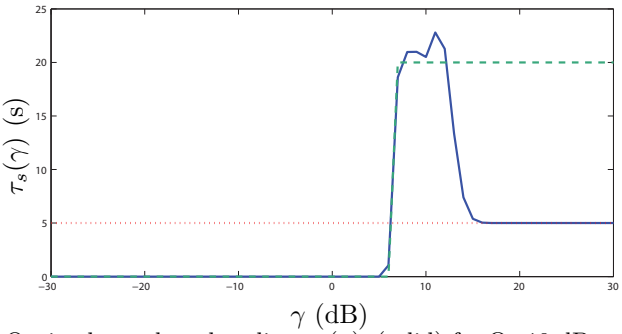


Figure 4.2: Optimal stop-length policy, $\tau_s(\gamma)$, (solid) for Ω=10 dB, τ_d=5 s (dotted), σ^2=75 s^2 and QPSK modulation with packet length $N_P = 21$. The corresponding threshold policy (dashed) has the same expected stop-length and stop-length variance.

example solution, computed with the Matlab `fmincon` solver, for specific values of Ω, τ_d, σ and N_P, is illustrated in Figure 4.2. The solid line is the optimal stop-length policy and the dotted line shows the constant drive time, included as a reference. The dashed line is the threshold approximation, described below.

Threshold Approximation of the Optimal Policy

Solving for the optimal stop-length policy is computationally demanding and the solution depends on Ω, so it cannot be done offline, unless the robot has enough memory to store a complete look-up table. Inspired by the threshold nature of the optimal policy, we instead propose a threshold approximation of the optimal policy:

$$\tau_s(\gamma) = \begin{cases} \alpha\tau_d \text{ if } \gamma > \gamma_{\text{th}} \\ 0 \text{ else.} \end{cases} \tag{4.3}$$

The stop length expectation and variance depend on the parameters α and γ_{th}:

$$\begin{aligned} \mathrm{E}[\tau_s(\gamma)] &= \alpha\tau_d e^{-\gamma_{\text{th}}/\Omega} \\ \mathrm{Var}[\tau_s(\gamma)] &= \alpha^2\tau_d^2 e^{-\gamma_{\text{th}}/\Omega}\left(1 - e^{-\gamma_{\text{th}}/\Omega}\right) \end{aligned} \tag{4.4}$$

By selecting $\alpha = \sigma^2/\tau_d^2 + 1$ and $\gamma_{\text{th}} = \Omega \ln \alpha$, the threshold policy yields the same stop length expectation and variance as the optimal policy. Figure 4.2 shows the threshold approximation of the corresponding optimal policy.

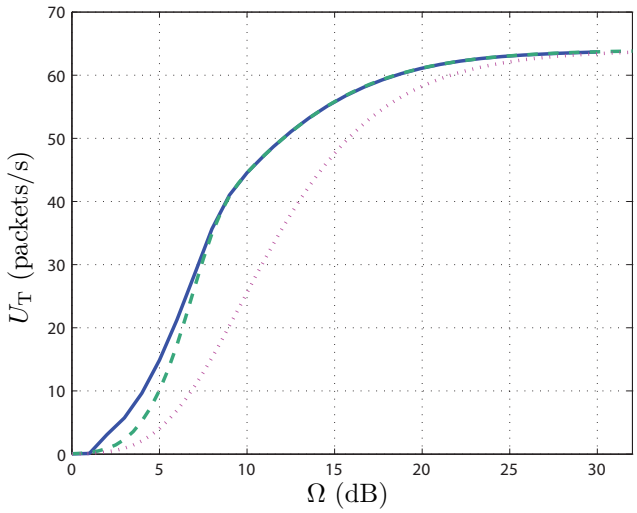


Figure 4.3: Average normalized throughput, U_{T}, as a function of the mean SNR, Ω. The solid blue line represents the optimal stop-length-policy, the dashed green line is the threshold stop-length policy and the dotted magenta line is the case of driving at constant velocity. The figure shows that using the proposed time-triggered motion is particularly beneficial in the transition region, where the signal is weak but still detectable. The parameters are $\tau_d = 5$ s, σ^2=75 s^2 and $N_P = 65$.

Expected Performance

The optimal and threshold policies derived above can be compared with respect to the average throughput. The average throughput (4.2) is plotted in Figure 4.3 for different values of the mean SNR. For comparison, we have also included the throughput $\bar{U}_{\mathrm{T}}$, achived when driving at constant velocity. The figure shows very little difference in throughput between the optimal and threshold policies. We also see that using the proposed strategy can improve the throughput by over 50% compared to just driving, when the SNR is below 10 dB. The price we pay for this improvement is some deviation from the reference position. In the next section, we will discuss how to use feedback to keep this deviation small.

4.3 Tracking Controller

The tracking controller in Figure 4.1 uses the relative position Δ as input and outputs an offset term u that is added to the estimate of the mean SNR. If u is positive, the stop-length policy will increase its threshold for stopping, which makes the robot stop less often and thus move faster. Conversely, a negative u leads to lowering the threshold and thus slows the robot down. This allows us to maintain

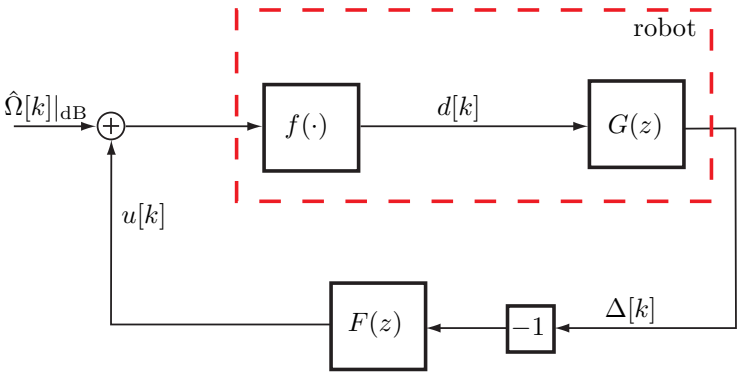


Figure 4.4: Model of the closed-loop system.

reference tracking and also adds robustness to errors in the channel model. If the distribution of γ differs from (2.4), this may introduce a bias in the expected velocity, which the feedback controller can compensate for.

Controller Design

We first derive a model of the system, in Figure 4.4, and then design a controller using pole placement. The controller is formulated in discrete time, sampling irregularly at each instant t_k so that $\Delta[k] \triangleq \Delta(t_k)$. We define $d[k]$ as the change in position during the interval $t_k \leq t < t_{k+1}$, which yields $\Delta[k+1] = \Delta[k] + d[k]$. The transfer function from d to Δ is thus

$$G(z) \triangleq \frac{1}{z-1}. \tag{4.5}$$

The position change $d[k]$ is a stochastic variable whose expectation does not depend on k:

$$\mathrm{E}[d] = \tau_d(v_d - v_{\mathrm{ref}}) - v_{\mathrm{ref}} \int_0^\infty \tau_s(\gamma) f_\gamma(\gamma) \mathrm{d}\gamma.$$

With $\gamma_{\mathrm{th}} = 10^{u/10}\hat{\Omega} \ln \alpha$, (4.4) yields the expectation

$$\mathrm{E}[d] = \tau_d \left(v_d - v_{\mathrm{ref}} - v_{\mathrm{ref}} \alpha^{(1-10^{u/10}\hat{\Omega}/\Omega)} \right).$$

As in Chapter 2, we use the notation $|_{\mathrm{dB}}$ for a value in dB. This allows us to introduce f as the expected value of d with the inputs in decibel:

$$f(\hat{\Omega}|_{\mathrm{dB}} + u) \triangleq \tau_d \left(v_d - v_{\mathrm{ref}} - v_{\mathrm{ref}} \alpha^{(1-10^{(\hat{\Omega}|_{\mathrm{dB}} + u - \Omega|_{\mathrm{dB}})/10})} \right).$$

This static nonlinearity is shown in Figure 4.5. It has maximum slope

$$k_{\max} \triangleq \tau_d v_{\mathrm{ref}} \frac{\ln 10}{10} \alpha^{(1-1/\ln \alpha)}.$$

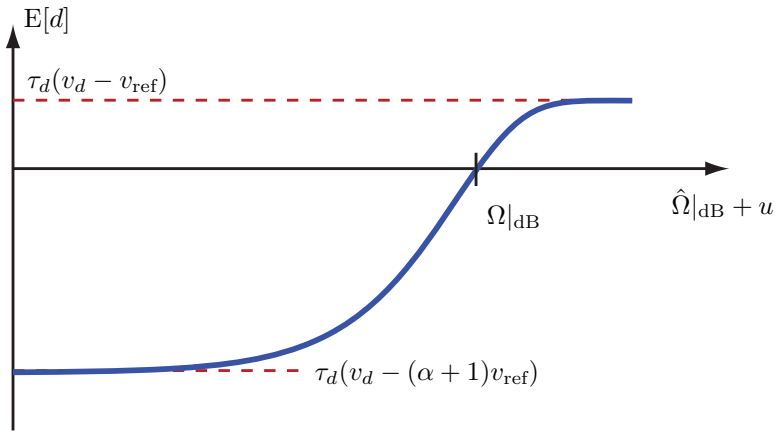


Figure 4.5: The static nonlinearity $f(\hat{\Omega}|_{\text{dB}}+u)$, giving the expected change of relative position per sampling period.

We do linear control design at the working point $\hat{\Omega}|_{\text{dB}} + u = \Omega|_{\text{dB}}$, where the nonlinearity f can be approximated with its derivative

$$k_0 \triangleq \tau_d v_{\text{ref}} \frac{\ln 10 \ln \alpha}{10}.$$

The controller, $F(z)$, is chosen to give the closed-loop transfer function

$$G_{\Omega\Delta}(z) = \frac{k_0(z-1)}{(z-a)^2}$$

from $\tilde{\Omega}|_{\text{dB}}$ to Δ. The zero in $z = 1$ ensures that any bias in $\tilde{\Omega}|_{\text{dB}}$ does not give a static tracking error, and the location $0 < a < 1$ of the double pole determines the time constant of the closed-loop system. We will discuss the choice of this design parameter later. The result is the proportional-integral controller

$$F(z) = \frac{2(1-a)}{k_0} + \frac{(a-1)^2}{k_0(z-1)}. \tag{4.6}$$

We now analyze the stability of the actual nonlinear system to find bounds on a.

Stability Analysis

For the stability analysis, we first model the stochastic variable $d[k]$ with $h[k]+w[k]$, where $h[k]$ is the expected value of $d[k]$, and $w[k]$ represents the unknown variations of $d[k]$. This is illustrated in Figure 4.6. Since

$$d[k] \in [\tau_d(v_d - (\alpha+1)v_{\text{ref}}), \tau_d(v_d - v_{\text{ref}})],$$

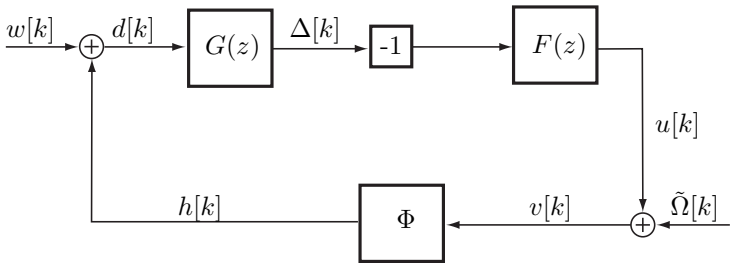


Figure 4.6: A model of the original system, used for the stability analysis.

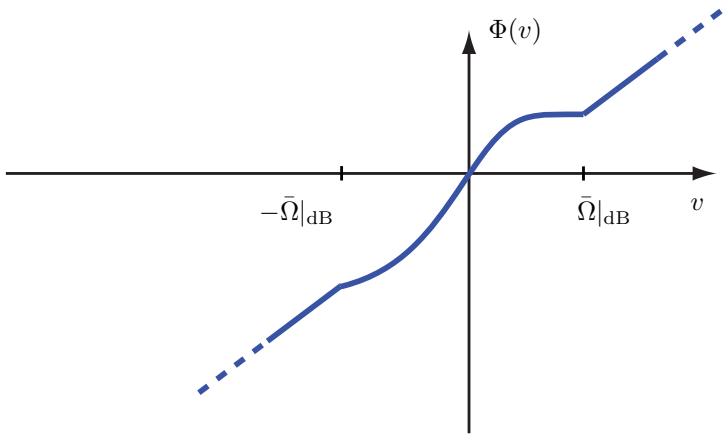


Figure 4.7: The sector nonlinearity Φ.

each sample $w[k]$ is also bounded.

Second, we introduce the estimation error

$$\tilde{\Omega}|_{\mathrm{dB}} \triangleq \hat{\Omega}|_{\mathrm{dB}} - \Omega|_{\mathrm{dB}}$$

as input to the system. We assume it to be constant but unknown. This allows us to replace f by a sector nonlinearity

$$\Phi(v) \triangleq \begin{cases} f(\Omega|_{\mathrm{dB}} + \bar{\Omega}|_{\mathrm{dB}}) + k_0(v - \bar{\Omega}|_{\mathrm{dB}}) & \text{if } v > \bar{\Omega}|_{\mathrm{dB}} \\ f(\Omega|_{\mathrm{dB}} - \bar{\Omega}|_{\mathrm{dB}}) + k_0(v + \bar{\Omega}|_{\mathrm{dB}}) & \text{if } v < -\bar{\Omega}|_{\mathrm{dB}} \\ f(\Omega|_{\mathrm{dB}} + v) & \text{otherwise,} \end{cases} \tag{4.7}$$

where $\bar{\Omega}$ is the maximum permitted estimation error for stability to be guaranteed. The resulting nonlinearity Φ is shown in Figure 4.7. Later, we will comment further on $\bar{\Omega}$.

Proposition 4.3.1 (Tracking Stability)**.** *The system in Figure 4.6, with $G(z)$ as in (4.5), $F(z)$ as in (4.6) and Φ as in (4.7) has finite gain if $0 < a < 1$ and*

$$\operatorname{Re} \frac{F(e^{j\theta})G(e^{j\theta})}{1 + F(e^{j\theta})G(e^{j\theta})k_0/4} > \frac{-1}{k_{max} - k_0/4} \ \forall\ \theta \in [0, 2\pi]. \tag{4.8}$$

Proof. We start by doing a loop transformation, which turns the system into a feedback connection of an asymptotically stable linear system and a sector nonlinearity. Then we apply the circle criterion.

The nonlinearity Φ can be decomposed into a constant gain $\tilde{k} = k_0/4$ and a new nonlinearity $\tilde{\Phi}$:

$$\Phi(v) = \tilde{k}v + \tilde{\Phi}(v).$$

By choosing

$$\bar{\Omega}|_{\mathrm{dB}} = \frac{\tau_d(v_d - v_{\mathrm{ref}})}{\tilde{k}},$$

the nonlinearity $\tilde{\Phi}$ lies in the sector $[0, k_{\max} - \tilde{k}]$. The closed-loop system can then be redrawn as in Figure 4.8, where

$$\tilde{G}(z) = \frac{F(z)G(z)}{1 + \tilde{k}F(z)G(z)},$$

with pole polynomial

$$z^2 - \frac{1}{2}(3 - a)\, z + \frac{1}{4}\left(a^2 + 3\right),$$

so the squared magnitude of the poles is

$$\frac{\sqrt{4a^2 + 12}}{4} < 1.$$

Hence, since $0 < a < 1$, $\tilde{G}(z)$ is asymptotically stable.

The circle criterion (Vidyasagar, 1993, Theorem 6.7.37) states that if $\tilde{G}(z)$ is asymptotically stable and $\tilde{\Phi}$ is a nonlinearity in the sector $[0, k_{\max} - k_0/4]$, the nonlinear feedback system has finite gain if

$$\operatorname{Re} \tilde{G}(e^{j\theta}) > \frac{-1}{k_{\max} - k_0/4} \ \forall\ \theta \in [0, 2\pi].$$

This is equivalent to (4.8) in the proposition. □

Remark 1: The proposition above concerns stability in discrete time. When sampling irregularly, in general the sample interval could become infinitely short, thus making it a poor representation of the actual continuous-time system. This cannot happen in our case, since there is a minimum time $\tau_d > 0$ between each sampling instance.

Remark 2: Because Φ and f differ outside the interval $-\bar{\Omega}|_{\mathrm{dB}} < \tilde{\Omega}|_{\mathrm{dB}} + u < \bar{\Omega}|_{\mathrm{dB}}$, the result above implies local stability of the original system. But, as shown in the

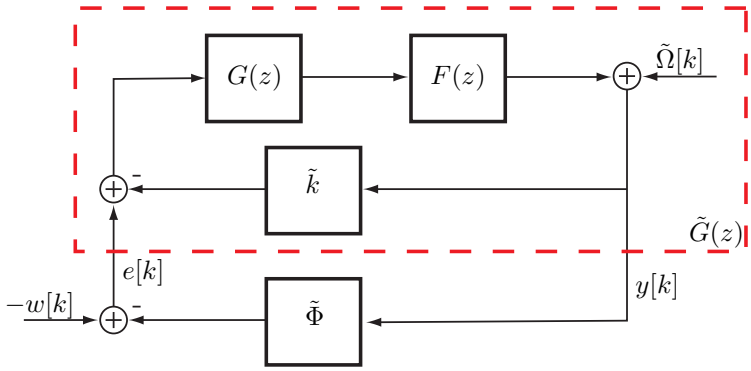


Figure 4.8: Loop transformed system.

examples below, in practice the permitted interval is large enough for the channel estimator error never to fall outside it.

To interpret Proposition 4.3.1, we first note that

$$\Delta(z) = \frac{G(z)}{1 + \tilde{k}F(z)G(z)}E(z) - \frac{\tilde{k}G(z)}{1 + \tilde{k}F(z)G(z)}\tilde{\Omega}|_{\mathrm{dB}}(z).$$

Both transfer functions are asymptotically stable since they share the poles of $\tilde{G}(z)$. They both have a zero in $z = 1$, so the constant estimation error $\tilde{\Omega}|_{\mathrm{dB}}$ does not affect the reference tracking error Δ. Further, since each sample $w[k]$ is bounded, Proposition 4.3.1 implies that $e[k]$ is also bounded. As shown above, the transfer function from e to Δ is asymptotically stable. Thus, the tracking error $\Delta[k]$ due to the variations of $d[k]$ is bounded.

Choosing Closed-Loop Time Constant

The design parameter a in the feedback controller can be chosen by the system designer to set the time constant of the closed loop system. In the preceding section we showed how to check that a certain choice of a maintains stability. Now we go on to discuss how some feasible choices represent different trade-offs between link capacity and reference tracking.

Intuitively, if the closed loop system is too fast, it will cause large changes to the stopping threshold γ_{th}. This means that the decision to stop is mainly dictated by the relative position, not by the instantaneous link capacity. This forces the robot to stay very close to the reference position, without leaving much room to exploit positions where the link capacity is high. Conversely, a slow closed-loop system allows the robot to keep stopping at good positions, giving high link capacity, even if it happens to be far from the reference. How to tune the controller depends on the application.

Table 4.1: Expected performance for different choices of closed-loop pole a.

	Position error	Throughput improvement			
		Ω=0 dB	Ω=3 dB	Ω=6 dB	Ω=10 dB
Eq. (4.2)	N.A.	150%	100%	40%	13%
a=0.99	4.3 m	150%	98%	39%	13%
a=0.95	2.0 m	142%	91%	39%	13%
a=0.9	1.5 m	133%	83%	39%	13%
a=0.8	1.1 m	117%	73%	37%	13%
a=0.5	0.83 m	84%	54%	30%	12%

To illustrate the tradeoff, we have simulated the system running a distance of 5 km, for different mean SNR and with different settings of a. We have then computed the average link capacity (4.1) and the RMS position error

$$\sqrt{\frac{1}{M}\sum_{k=1}^{M}\Delta^2[k]},$$

where M is the number of drive cycles. In the simulations, we have used $\tau_d = 5$ s, $\alpha = 4$, $v_{\text{ref}} = 0.1$ m/s, $v_d = 0.2$ m/s and $N_P = 65$. With these parameters, closed-loop stability can be guaranteed for $a \geq 0.5$ if the estimation error is less than $\bar{\Omega}|_{\text{dB}} = 12$ dB.

Table 4.1 lists the resulting RMS position errors and throughput improvements, compared to just driving at constant speed. For comparison, we have also included the analytical prediction (4.2), corresponding to running in open loop with no feedback control of the tracking error. The simulations and our experience indicate that $a \approx 0.9$ may in general represent a good tradeoff between throughput and tracking performance.

4.4 Channel Estimator

The channel estimator in the control architecture in Figure 4.1 works as a feedforward controller, reacting to changes in the environment. The estimator averages over a fixed number of SNR samples. Let γ_n be the sample taken n samples ago. Then the estimate of the mean SNR is

$$\hat{\Omega} = \frac{1}{M_E}\sum_{n=1}^{M_E}\gamma_n, \tag{4.9}$$

where M_E is the window length of the estimator. Note that the samples γ_n are taken at a higher rate than the samples $\gamma[k]$, taken at each stop instant. Since the

robot will stop at positions with high SNR, we only sample while moving, to avoid biasing the estimate. To minimize the estimator lag, we sample as fast as possible while still maintaining a sample distance of half a wavelength. As motivated above, we then consider the samples as independent.

The window length can be based on two criteria: We can look at the scale of obstacles in the environment and make sure that $M_E\lambda/2$, the spatial window size, is in the same order. Or, we can look at the variance of γ_n (which are i.i.d.) and use the central limit theorem to determine M_E such that the variance of the estimate is in the same order as the accuracy of the one sample which we will compare it to when stopping.

Selecting the fixed window size M_E becomes a tradeoff between adaptability of the estimator versus the accuracy of the estimate. A more elaborate approach would be to have a separate detector for phase changes, such as moving between rooms or turning a corner, where Ω would make abrupt changes. Then the window size could be reduced or the buffer be flushed. If the robot follows the same path several times, it could also use past measurements as a prior, reducing the estimate variance.

4.5 Experiments

The proposed strategy of time-triggered stopping has been experimentally tested, to verify the link throughput improvements and test the properties of the closed loop system. The experiments were made in two different locations, both exhibiting static Rayleigh fading and without persons or objects moving around.

Below, we first describe the experiment setup. We also discuss the issue that we cannot measure the actual SNR, but only the received signal strength. Then we show the results of measuring the throughput improvements. Finally, we illustrate the tracking performance of the feedback controller.

Setup

The experiments were done indoors, at two different locations. A transmitter Tmote Sky was placed about 10 m away, out of sight, from the receiver Tmote Sky, which was mounted on a mobile robot. The robot used a line sensor to follow a closed reference path, 15–20 m long, marked by tape on the floor. See Figure 4.9.

The transmitter sent $R_0 = 64$ packets/s, each with a constant 50 byte payload, making the total packet length 65 bytes. The receiver counted the number of correctly received packets and, when queried by the robot, reported the signal strength of the last packet. If no packet was received within 1/64 s, it reported a packet loss. The average throughput of an experiment run was measured as the total number of received packets divided by the total experiment time.

The first location was a basement lab with reinforced concrete walls and computers and metal lab benches all over the room. The second was a classroom on

Figure 4.9: Experiment setup in the basement lab. The robot follows a tape path on the floor, while the attached Tmote Sky transceiver receives signals from another Tmote Sky in the back of the room.

the ground floor, with wooden desks and plaster walls. To create multipath fading in the classroom, we draped some desks in aluminum foil.

To justify our communication model, we recorded the received signal strength along the path at intervals of 1 cm (lab) and 4 cm (classroom). Figure 4.10 shows the resulting histograms, each based on about 1800 samples. The transmitter was set to high power (0 dBm), which gave only 3 or 4 packet losses in each experiment. For comparison, the corresponding PDF of Rayleigh fading, converted to decibel scale, is overlaid in each graph. The graphs show that Rayleigh fading serves as a useful approximation of the RSS distribution.

Measuring SNR

The proposed strategy assumes that we can measure the SNR of the radio signal and then compare it to the local average to make decisions on stopping or driving. In our system, we actually measured the received signal strength of successfully received packets. This created two problems:

First, the estimator (4.9) got a positive bias since packets with low SNR are more likely to be lost and thus not included in the averaging. This bias was compensated by the feedback controller, but it caused a small transient at the start of each experiment run. To reduce the transient when testing slow controllers, we initialized the controller to $u[0] = -2$ dB.

Second, in scenarios where the mean SNR is low, the stopping threshold γ_{th} can be close to the detection threshold γ_{det} of the receiver. Then, if the controller reduces

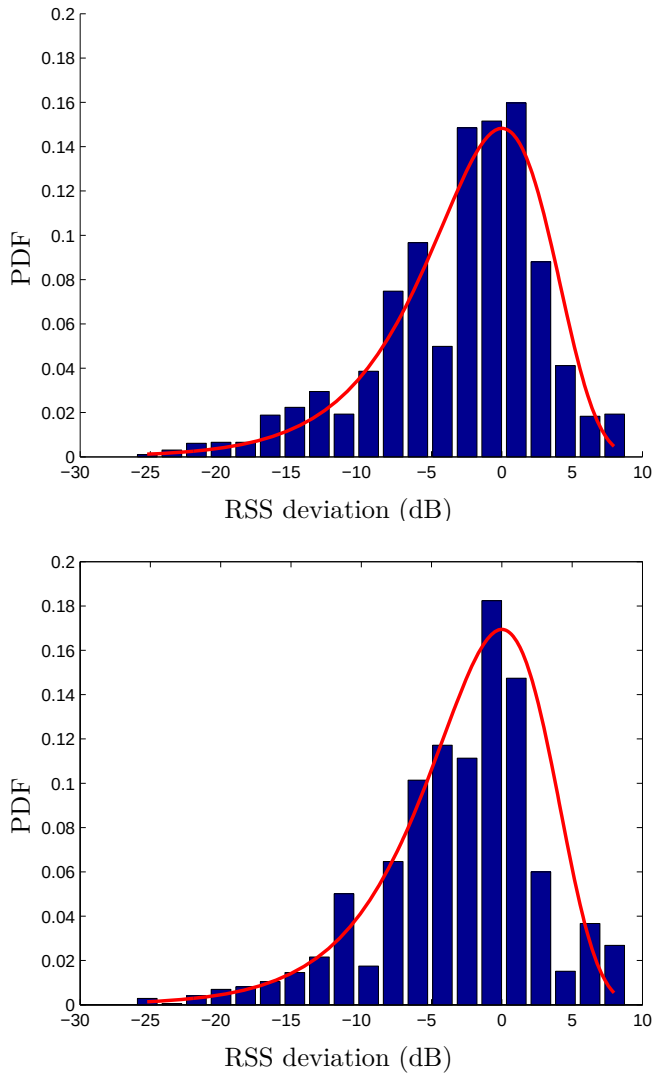


Figure 4.10: Justification the communication model: The histograms show 1800 samples of the deviation from the mean of the received signal strength along the paths in the classroom (top) and lab (bottom). The PDF of Rayleigh fading is included as a reference.

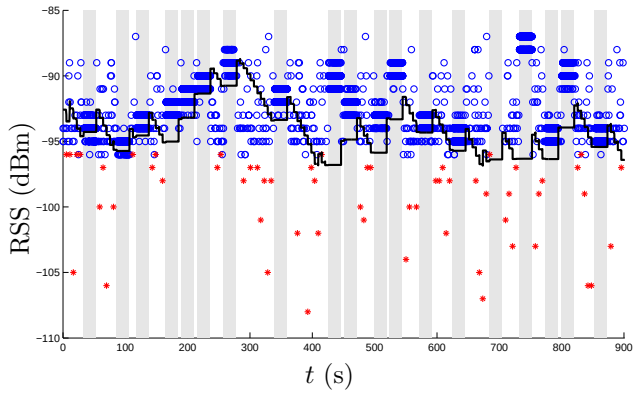


Figure 4.11: An example 15 min run using time-triggered stopping. Each circle represents a received packet and its signal strength. Stars represent random values, replacing lost packets as described earlier. Grey background represents the robot standing still, thus maintaining a near constant signal strength. The solid line is the stopping threshold, γ_{th}. The circles are denser at the high plateaus, since the higher signal strength gives less packet losses. This is what causes the improvement in average throughput.

$u[k]$ to slow the robot down, it might happen that $\gamma_{\text{th}} < \gamma_{\text{det}}$. This effectively cuts off the feedback loop, since changes of γ_{th} do not lead to changes in stop probability. To avoid this, when applying the stop time policy (4.3), we replace each lost packet by a random RSS value

$$\gamma[k]_{dB} = (\gamma_{\text{det}})_{dB} - \mu,$$

where μ is exponentially distributed with mean 5. This approximately recreates the tail of the Rayleigh distribution and maintains the connection between $\Delta[k]$ and $u[k]$. Figure 4.11 shows an example of this.

Throughput Improvement

To test the improvement in throughput, we first let the robot follow the reference path at velocity v_{ref} for 15 min. Then, changing nothing in the room, we made another 15 min run, using time-triggered stopping. The total throughput for each run was then compared, to find the improvement. All experiments used $\alpha = 4$, $\tau_d = 5$ s, $N_P = 65$, $v_{\text{ref}} = 0.1$ m/s, $v_d = 0.2$ m/s and $M_E = 100$ unless specifically stated otherwise.

Several such comparisons were performed in each room. We used different settings of transmitter output power and different placements of the transmitter, to vary the mean SNR over a wide range. We also used tracking controllers with different time constants, to see if the influence of the controller tuning matched the simulations above.

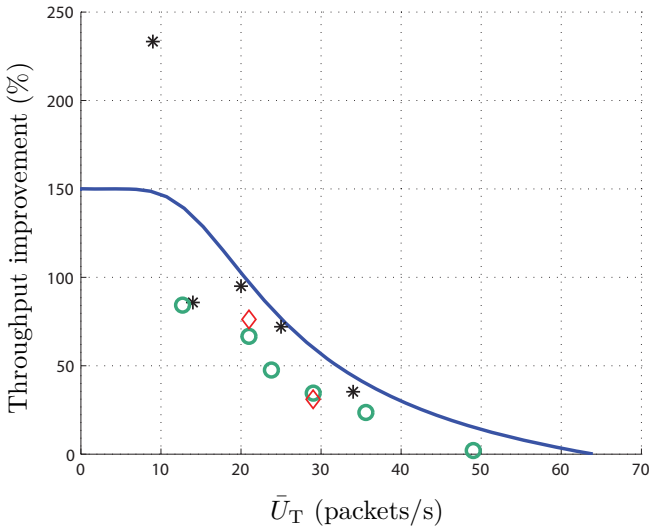


Figure 4.12: The improvement of the throughput when using time-triggered stopping, as a function of $\bar{U}_{\mathrm{T}}$, *i.e.*, the throughput when driving at constant velocity. The solid line is the expected improvement (4.2) for the open-loop case, corresponding to unbounded tracking error. Each marker represents a pair of experimental runs, with and without time-triggered stopping. The experiments were made in a basement lab (stars) and in a ground floor classroom (circles and diamonds).

Figure 4.11 shows the result of one such run, using time-triggered stopping with $a = 0.9$. The average throughput was 35 packets/s, compared to 21 packets/s when driving at constant velocity under the same conditions. Thus the throughput improvement was 67%.

To allow comparison between the achieved results and the expected throughput (4.2) when running in open loop, we have plotted the throughput improvement as a function of the throughput when driving at constant velocity. Figure 4.12 shows the results of experiments in the basement lab and in the ground floor classroom. The expectation of the open-loop case is included for reference. The experiments in the basement used $a = 0.95$ (stars) and those on the ground floor used $a = 0.9$ (circles) or $a = 0.95$ (diamonds). The results follow the trend of the open-loop case, where the improvement from time-triggered stopping increases as the channel quality gets lower. It is expected that the resulting throughput should lie below the open-loop case since the tracking controller sacrifices throughput for reference tracking. At low SNR, there are large variations since there both the driving and stop-and-go throughputs are low, making the ratio very sensitive to random variations. Using $a = 0.9$ or $a = 0.95$ seems to make little difference, as predicted by the simulations in Table 4.1.

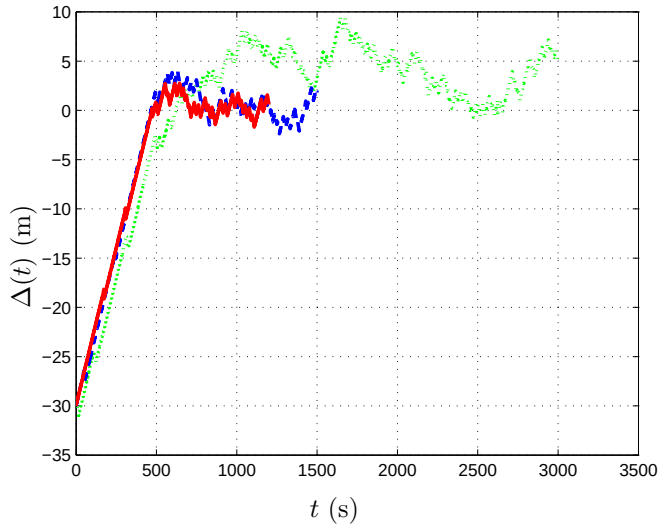


Figure 4.13: Step responses in relative position, $\Delta(t)$, for $a = 0.9$ (red, solid), $a = 0.95$ (blue, dashed) and $a = 0.99$ (green, dotted).

Tracking Stability

To demonstrate reference tracking, we started the robot at a position 30 m behind the reference, using controllers with $a = 0.9$, $a = 0.95$ and $a = 0.99$, respectively. To make these experiments faster, we used $\tau_d = 3$ s. The step responses in relative position are plotted in Figure 4.13. Note that the positive slope is limited to $v_d - v_{\text{ref}}$, corresponding to the robot never stopping.

Figure 4.14 shows the behavior of the controllers with $a = 0.9$ and $a = 0.95$ after the system reached steady state in the step response experiment above. The magnitudes of the position fluctuations match those predicted by simulations in Table 4.1. The recommended value of $a \approx 0.9$ that gave a good tradeoff between tracking and resulting throughput in simulations, appears to yield good performance also in the experiments.

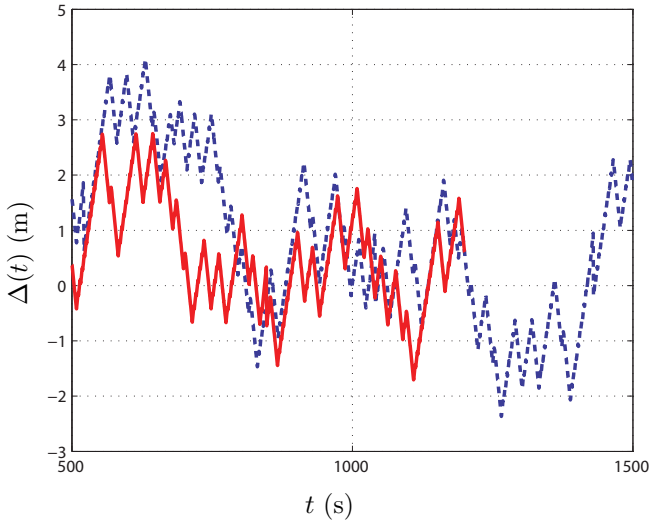


Figure 4.14: Magnification of the steady-state behavior of two of the controllers in Figure 4.13. Using $a = 0.9$ (red, solid line) gives tighter reference tracking than $a = 0.95$ (blue, dashed line), at the expense of a smaller throughput improvement.

4.6 Summary

We have presented a time-triggered stopping strategy for communication-aware trajectory tracking. The robot stops after driving a constant time, measures the instantaneous SNR and makes a decision on how long to stand still before driving again. This decision is affected by the SNR distribution, which may change along the trajectory. The estimator tracks this and outputs an estimate of the average SNR. The decision also depends on the relative position of the robot. Using the tracking controller, this allows the robot to make shorter stops and thus catch up if it falls behind the reference.

Our proposed control architecture makes a tradeoff between link throughput and tracking accuracy, determined by the parameter $0 < a < 1$. High values of a correspond to allowing the robot to deviate far from the reference position, which provides a high improvement in throughput. Small values of a give tight reference tracking, but leave little freedom for communication-awareness, so the throughput improvements are lower.

There are some free parameters in the suggested strategy that must be chosen by the system designer. The reference velocity v_{ref} and the stop time variance σ^2 encode the reference trajectory and its associated tracking accuracy. The drive time τ_d should be chosen much longer than the time to measure the SNR after stopping, so that the measurement time can be neglected. Finally, the driving velocity v_d

should be chosen as high as possible subject to the power constraints of the robot and the requirement that the robot must be able to follow the reference trajectory at this speed. This frees maximal time for the robot to stand still at positions with good channel quality.

To validate the analytical results, we have performed experiments in two locations. The results showed that time-triggered stopping can yield throughput improvements in the range of 50-100% when the channel quality is low. The experiments also showed that the tracking controller effectively compensated deviations from the reference position.

Chapter 5

Motion Planning under Multipath Fading: Hybrid Optimal Control

This is the last chapter on communication-aware motion along a reference trajectory under multipath fading. We now consider a hybrid optimal control formulation, where tracking error and link utility are combined in a cost function. We explicitly model the inflow of sensor data to the robot, assuming that there is a data buffer that stores data until it is sent through the wireless link. This makes it possible to formulate a tradeoff: If the buffer is filling up, the robot can stop at good positions to communicate, at the expense of falling behind the reference. Then it drives to catch up again. Since we do not predict the multipath fading, the only way for the robot to maintain a high signal strength is to stop completely. When it switches to driving, the fading will vary so the robot experiences the mean signal strength. Solving for the optimal hybrid controller is problematic due to the "curse of dimensionality". Therefore we have applied the method of *relaxed dynamic programming*, proposed by Lincoln and Rantzer (2006), which uses an approximate value function with small relative error.

This chapter begins with a derivation of the specific hybrid model we consider. We then formally define the problem. In Section 5.2, we express the system dynamics as a switched linear system and formulate an infinite-horizon hybrid optimal control problem. Using relaxed dynamic programming, we present an algorithm to compute a controller for both the switch sequence and the continuous input. In Section 5.3, we simulate the resulting closed-loop system to illustrate its properties under different non-ideal conditions. Finally we summarize in Section 5.4.

5.1 Preliminaries

Here we define the specific robot motion model of this chapter, as well as the model of the communication buffer. We then formally define the problem.

Robot Model

As before, we consider a robot moving along a given path. We only care about its relative position $\Delta(t)$, with dynamics as in (3.1). To get smooth motion, we model the acceleration as the control input $u(t)$. 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, controlled by the discrete input $\sigma(t) \in \{0, 1\}$. When $\sigma = 0$, the robot is stopped and $\sigma = 1$ corresponds to driving. The dynamics of the relative position are thus

$$\begin{aligned} \dot{\Delta}(t) &= v(t) - v_{\text{ref}} \\ \dot{v}(t) &= \begin{cases} -k_v v(t) \text{ if } \sigma = 0 \\ u \text{ if } \sigma = 1. \end{cases} \end{aligned}$$

The parameter $k_v \gg 1$ models the breaking action, making $v(t)$ converge to zero when stopping. Also note that we do not specifically consider any constraints on $v(t)$, since the robot velocities are moderate even for the unconstrained controller.

Data Buffer Model

We assume that data from the robot sensors are stored in a buffer onboard, and then transmitted over the wireless link to a base station. The buffer has size $z \geq 0$ and inflow rate R_{in}. The outflow from the buffer is equal to the throughput of the wireless link, which we model in a hybrid fashion: When the robot drives without adapting its velocity to the wireless channel, it experiences an average throughput of $\bar{U}_{\text{T}}$, defined in Chapter 4. But when it decides to stop, we assume that it does a very local search to find the highest signal strength before coming to a complete stop. We assume that this corresponds to finding a higher throughput $\hat{U}_{\text{T}}$. The resulting buffer dynamics are

$$\dot{z} = \begin{cases} R_{\text{in}} - \hat{U}_{\text{T}} \text{ if } \sigma = 0 \\ R_{\text{in}} - \bar{U}_{\text{T}} \text{ if } \sigma = 1. \end{cases}$$

Note that the buffer is lossless, which means that no packets are discarded. We can now formally state the problem.

Problem Formulation

Find a controller that determines u, σ such that the tracking error Δ and buffer size z are kept small, as well as the control effort u.

Note that this problem is mostly interesting in the interval $\bar{U}_\mathrm{T} < R_\mathrm{in} < \hat{U}_\mathrm{T}$, since if the inflow is lower than $\bar{U}_\mathrm{T}$ the robot can drive constantly without the buffer filling up. Conversely, if it is greater than $\hat{U}_\mathrm{T}$, some higher-level protocol must discard data to stop the buffer from overflowing. Also note that this does not require accurate navigation, since the robot can always find a high-throughput position by just driving a few centimeters in any direction.

5.2 Hybrid Optimal Control

In this section, we formulate the control problem as a hybrid optimal control problem. We then present relaxed dynamic programming as a way of finding an approximate solution. We state an algorithm that computes a value function from which we can derive a control law. Finally we show how to do this computation in an efficient way.

Switched Linear System

To describe the whole system, we collect the robot and buffer states in the same state vector. We include an integral state Δ_I to allow the controller to attenuate a static error in Δ. We finally add a constant element to the state vector, which allows writing the system on linear form. This yields

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

where the controls are $u \in \mathbb{R}$ and $\sigma \in \{0, 1\}$, and

$$A_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -k_v & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_\mathrm{in} - \hat{U}_\mathrm{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ B_0 = 0$$

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_\mathrm{in} - \bar{U}_\mathrm{T} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ B_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We consider a sampled version of the continuous-time system above, with sampling interval τ. We let $x[n] = x(t)$ and $u[n] = u(t)$ for $n\tau \leq t < (n+1)\tau$. We

also assume that u is kept constant during the sampling intervals and that σ only switches at sampling instances. The discrete dynamics can then be expressed as

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

where

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

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 $\bar{U}_\mathrm{T} < R_\mathrm{in} < \hat{U}_\mathrm{T}$, both Δ and z cannot simultaneously converge to zero, since the robot must occasionally stop to prevent the buffer from overflowing. We therefore introduce a decay factor η^n, with $0 < \eta < 1$, 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

$$\begin{aligned} \min_{\sigma[n], u[n]} \quad & \sum_{n=0}^{\infty} \left(x^T[n] Q x[n] + R u^2[n] \right) \eta^n \\ \text{s.t.} \quad & x[n+1] = f(x[n], u[n], \sigma[n]) \\ & x_4[n] \geq 0 \\ & x[0] = x_0, \end{aligned} \tag{5.1}$$

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 5.3.

Dynamic Programming

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

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

where

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

under the same constraints as in (5.1), but with $x_0 = x$. Once we have $V^*(x)$, we can derive the optimal control law as

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

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:

$$V_{k+1}(x) = \min_{u,\sigma} \left\{ \eta V_k(f(x,u,\sigma)) + \ell(x,u) \right\}. \tag{5.2}$$

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 assumptions on the contractiveness of the system, which we ensure by introducing η, it holds that (Lincoln and Rantzer, 2006)

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

The problem is that, if applied naively to a switched system, 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

$$\begin{aligned} P[n] = \Phi^T P[n+1]\Phi + Q - \Phi^T P[n+1]\Gamma \\ \times \left[\Gamma^T P[n+1]\Gamma + R\right]^{-1} \Gamma^T P[n+1]\Phi \end{aligned} \tag{5.3}$$

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

$$u^*(x[n]) = -\left[R + \Gamma^T P[n+1]\Gamma\right]^{-1} \Gamma^T P[n+1]\Phi\ x[n]. \tag{5.4}$$

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 (5.2):

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

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 $\underline{\alpha} < 1 < \overline{\alpha}$,

$$\begin{aligned} \min_{u,\sigma}\{\eta V_k(f(x,u,\sigma)) + \underline{\alpha}\ell(x,u)\} &\le V_k(x) \\ &\le \min_{u,\sigma}\{\eta V_k(f(x,u,\sigma)) + \overline{\alpha}\ell(x,u)\} \ \forall\ x. \end{aligned} \tag{5.5}$$

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 $\underline{\Pi_k} = \{\underline{P}_1, \ldots, \underline{P}_{N_k}\}$ 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 5.1.

Algorithm 5.1 Relaxed Dynamic Programming

1: $k := 0$, $\Pi_0 = 0_{n\times n}$
2: **while** (5.5) is not fulfilled **do**
3: $k := k + 1$
4: Form $\overline{\Pi}_k$ and $\underline{\Pi}_k$ by propagating the matrices in Π_{k-1} one step backwards in time, both with $\sigma = 0$ and $\sigma = 1$, as defined in (5.3).
5: Sort the sets $\overline{\Pi}_k = \{\overline{P}_1, \ldots, \overline{P}_{N_k}\}$ and $\underline{\Pi}_k = \{\underline{P}_1, \ldots, \underline{P}_{N_k}\}$ so that $\operatorname{tr}\overline{P}_1 \le \ldots \le \operatorname{tr}\overline{P}_{N_k}$ and $\overline{P}_i \ge \underline{P}_i \ \forall\ i$.
6: $\Pi_k := \emptyset$, $i := 0$
7: **while** $i \le N_k$ **do**
8: **if** $\nexists$ a convex combination P of matrices in Π_k such that $P \le \overline{P}_i$ **then**
9: Add $\underline{P}_i$ to Π_k.
10: **end if**
11: $i := i + 1$
12: **end while**
13: **end while**

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.

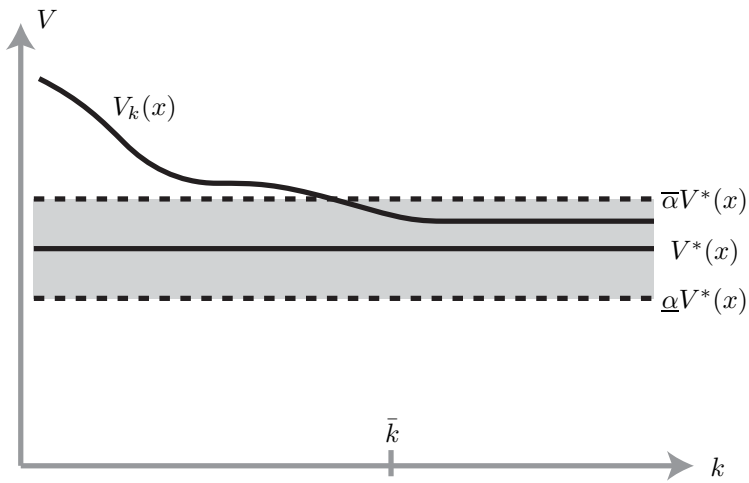


Figure 5.1: The value function $V_k(x)$ converging to the close-to-optimal set. At $k = \overline{k}$, the stopping criterion is fulfilled.

When $V_k(x)$ fulfills the stopping criterion (5.5) at, say, $k = \overline{k}$, it can be applied iteratively to yield that for all $k \geq \overline{k}$

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

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 5.1, 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 5.2. For comparison we also included the number of candidates M_k that would have to be considered using 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 $\eta = 0.9$, 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.

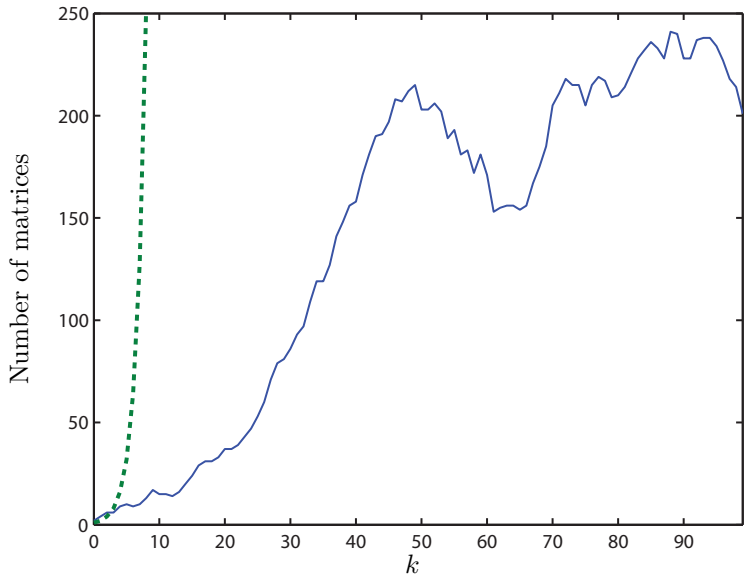


Figure 5.2: The number, N_k, (blue solid line) of matrices in Π_k needed to represent the value function at each iteration step. N_k stops increasing, indicating that (5.5) is fulfilled, after about $k = 50$ iterations. Without discarding any candidates, the complexity would grow as $M_k = 2^k$ (green dashed line), which is illustrated for comparison.

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

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

The optimal continuous control $u^*(x)$ was then computed using (5.4), 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 5.3, for the subset $v = 0$, $\Delta_I = 0$ of the state space.

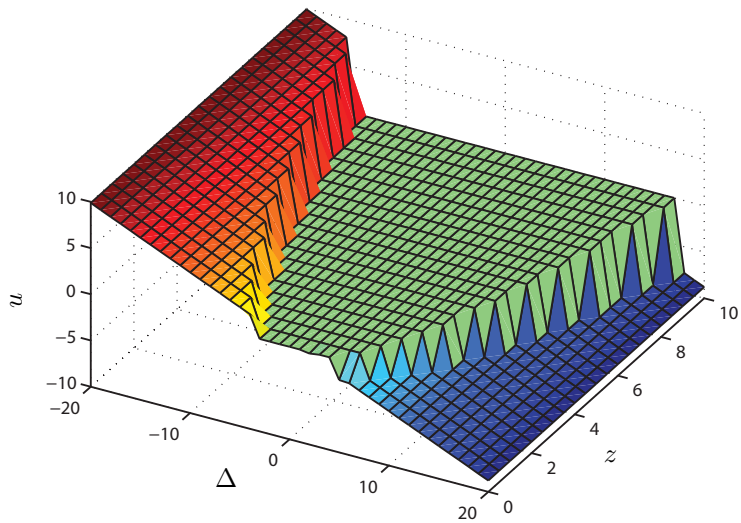


Figure 5.3: The resulting control law for the subset $v = 0$, $\Delta_I = 0$ (corresponding to standing still with an empty integral state in the controller). To also illustrate $\sigma(x)$, we have forced the control to $u = 0$ where $\sigma(x) = 0$. As one would expect, for small Δ, the robot stops. If Δ decreases, the controller accelerates the robot and if Δ becomes too large, it slows the robot down.

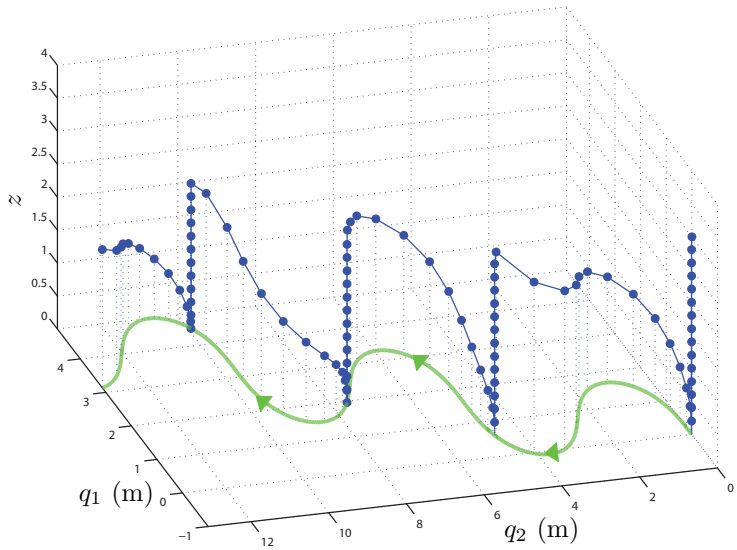


Figure 5.4: A trajectory of the system, at position $q \in \mathbb{R}^2$ and with buffer size z, sampled at regular intervals. The robot follows the reference path (thick solid line) in the plane, while stopping from time to time to reduce the buffer size z, shown as the height of the blue circle. The robot motion is from right to left.

5.3 Simulations

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 throughput 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 $Q = \mathrm{diag}(1, 0, 1, 5, 0)$.

Following a Curved Path

In Figure 5.4, we have simulated the system when following a sinusoidal reference trajectory with reference velocity $v_{ref} = 1$ m/s. The robot performs the "stop-and-go" motion dictated by its communication-aware controller along the reference path. The figure consists of periodical samples of the states of the robot, where the height of the circle 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 $R_{\mathrm{in}} - \hat{U}_{\mathrm{T}} = -1$ and $R_{\mathrm{in}} - \bar{U}_{\mathrm{T}} = 1$ to illustrate the behavior of the system more clearly.

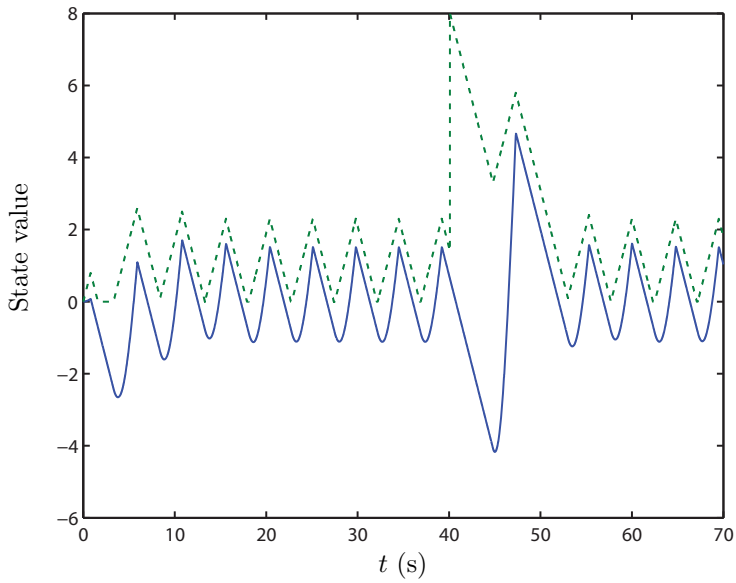


Figure 5.5: An example of a trajectory for the system, starting with an empty buffer and with perfect reference tracking. The position $\Delta(t)$ is the blue solid line and the buffer size $z(t)$ is the green dashed line. The robot approaches a limit cycle with a period time of 5.6 s, where it spends 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 5.5, it approaches a limit cycle with a period time of 5.6 s, where is spends 50% of the time in each mode. The position $\Delta(t)$ oscillates around zero while there is still a small static error in $z(t)$. However, at $t = 40$ s, extra data is added to the buffer, and this impulse disturbance is successfully attenuated. Here we also used $R_{\text{in}} - \hat{U}_{\text{T}} = -1$ and $R_{\text{in}} - \bar{U}_{\text{T}} = 1$.

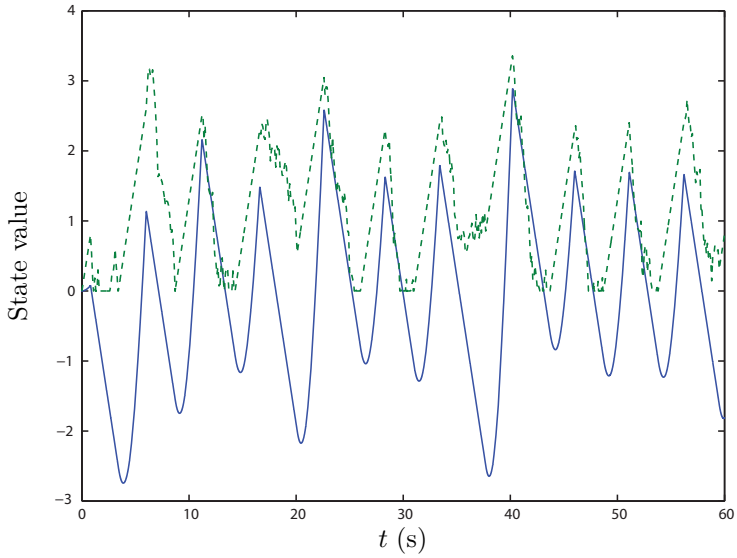


Figure 5.6: A test of how the system performs when zero-mean white gaussian noise with standard variation 2 is added to the link throughput $\hat{U}_{\mathrm{T}}$ in the stop mode. The lag $\Delta(t)$ (blue solid line) oscillates around zero and the buffer size $z(t)$ (green, dashed line) remains bounded.

Robustness to Throughput Variations

As indicated in the derivation of the communication model, the actual link throughput $\hat{U}_{\mathrm{T}}$ 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 $\hat{U}_{\mathrm{T}}$. With $\hat{U}_{\mathrm{T}} = 3$, $\bar{U}_{\mathrm{T}} = 1$ and $R_{\mathrm{in}} = 2$, the simulations indicate that the system still oscillates around $\Delta = 0$ and maintains a bounded buffer size $z(t)$. This is illustrated in Figure 5.6.

5.4 Summary

In this chapter, we have continued the study of how to exploit multipath fading, using a hybrid optimal control formulation where the robot switches between driving and stopping. When driving, it can reduce the tracking error but the outflow from its onboard data buffer is low. When stopping, we assume that it can make a local search to find a point with high throughput, so the buffer size decreases but the tracking error grows. To quantify the tradeoff, we proposed a cost function that includes both the buffer size and the tracking error. The hybrid optimal control problem was solved using relaxed dynamic programming. The resulting controller can be stored in look-up tables and thus used also on resource-constrained

robots. The only prior information needed about the radio link are the average link throughputs $\bar{U}_{\mathrm{T}}$ (when driving) and $\hat{U}_{\mathrm{T}}$ (which can be found by a local search when stopping). No map of the signal strength is needed. The closed-loop system was simulated under various conditions and it maintains a bounded buffer size and zero-mean tracking error.

The robustness to variations in the throughput $\hat{U}_{\mathrm{T}}$ when stopping, can be attributed to feedback. When the robot stops, the decision to start again is governed by the states, including the buffer size. So if the buffer is emptied faster or slower than expected, the stop time will be adjusted accordingly. The value of $\hat{U}_{\mathrm{T}}$ encodes an expectancy, and if the actual throughput differs from this, of course the resulting motion is no longer optimal with respect to the cost (5.1). The tracking error and buffer size, however, are still regulated.

Chapter 6

Motion Planning with Visual Connectivity Constraints

In this chapter, we consider coordination of multiple robots moving along given paths through an obstacle field. Only robots that have a free line of sight between each other can communicate, and the problem is for all robots to traverse their paths while maintaining connectivity of the whole group. In terms of the motion planning architecture considered in this thesis, this corresponds to augmenting the coordination layer so it not only avoids collisions with obstacles or other robots, but also maintains visual connectivity. The result is a reference trajectory for each robot, describing its velocity along the path as a function of time.

As described in Chapter 2, this is a similar formulation as in the path coordination problem, which arises in multi-robot path planning. However, unlike the path coordination problem, we assume that the robot paths are non-intersecting, so there are no collision avoidance constraints on the individual velocities. An example where this is applicable is when identical robots are used as sensors. If the paths intersect, this can be avoided by exchanging the allocation of path segments between robots. Without the velocity constraints due to collision avoidance, it is possible to use this freedom to instead maintain communications.

This chapter describes two different solutions to this problem, both using a configuration-space representation of the system. First we describe a sampling-based solution and show that it can handle problems with many robots. The drawback is that if a problem instance is unsolvable, the solver runs forever. In response to this, we then present an exact solution method, which in finite time reaches a solution or concludes that the problem is unsolvable. The drawback of this is that the worst-case computational complexity is higher and that we need to restrict the problem some, to make the solver feasible. We end by simulations and a discussion of the strengths and weaknesses of the two alternative methods.

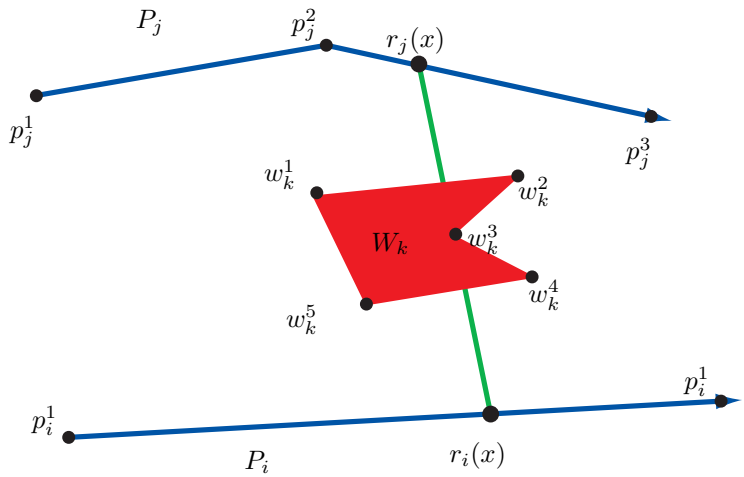


Figure 6.1: Robots i and j (black circles) are at positions $r_i(x)$ and $r_j(x)$, respectively. Each state x_i denotes the distance along the path (blue) that robot i has moved. Due to the obstacle W_k, the robots have an obstructed line of sight (green).

6.1 Preliminaries

In this section, we present models for the motion of the robots, the obstacles and visually constrained communication. Then we formally define the problem of maintaining visual connectivity.

Robot and World Model

We consider a group of N robots, where each robot is given an obstacle-free reference trajectory, as discussed in Chapter 2. The corresponding path for robot i is $P_i \subset \mathbb{R}^2$, consisting of Π_i straight line segments. Each path P_i is defined by its vertices $\{p_i^1, \dots p_i^{\Pi_i+1}\}$, as illustrated in Figure 6.1. The group has configuration $x = (x_1, \dots, x_N)$, where x_i is the position along the path of robot i. The goal configuration is $x_G = (L_1, \dots, L_N)$, where L_i is the length of path P_i. Thus, $0 \leq x_i \leq L_i$. Paths may intersect themselves, but not obstacles or other paths.

The world contains M obstacles. Obstacle W_k, $k \in \{1, \dots, M\}$ is defined as the interior of the (possibly non-convex) polygon with Ω_k vertices $\{w_k^1, \dots w_k^{\Omega_k}\}$. Obstacles may intersect each other. Figure 6.1 illustrates the notation for paths and obstacles.

Communication Model

We study a visibility-based communication model. We define a connectivity graph $G_C(x) = (V_C, E_C(x))$ with vertices $V_C = \{1, \dots, N\}$. The configuration-dependent

set of edges, $E_{\mathrm{C}}(x)$, consists of an undirected link $e = \{i, j\}$ for each pair i, j of robots whose connecting line of sight (LOS) is not obstructed by any obstacles:

$$\{i, j\} \in E_{\mathrm{C}}(x) \Leftrightarrow \mathrm{convhull}(r_i(x), r_j(x)) \cap W_k = \emptyset \ \forall \ k \in \{1, \ldots, M\}.$$

There are no constraints on the communication distance, as long as there is a line of sight. For any configuration, $G_{\mathrm{C}}(x)$ is *connected* if it contains a path between any two vertices. Note that connectivity could also be evaluated over some time interval, like in delay-tolerant networks (Jones et al., 2007), but in this paper we only consider instantaneous connectivity. We further assume that $G_{\mathrm{C}}(0)$ and $G_{\mathrm{C}}(x_{\mathrm{G}})$ are connected, since otherwise the problem is trivially unsolvable.

Problem Formulation

Using the models above, we can now define the problem of path following with continuous connectivity:

Definition 6.1.1 (Path Following with Continuous Connectivity)**.** *Given paths $P_1, \ldots, P_N$ and obstacles $W_1, \ldots, W_M$, find a continuous state trajectory $x : [0, T] \to \mathbb{R}^N$ such that $x(0) = \mathbf{0}$, $x(T) = x_G$ and $G_C(x(t))$ is connected for all $t \in [0, T]$.*

Note that this problem is not guaranteed to have a solution. It is easy to construct problem instances where obstacles make it impossible to maintain connectivity and still reach the goal. We will return to this later and comment on how it affects the choice of solution method. But first we will show how the problem maps to the General Motion Planning Problem, represented in a configuration space.

6.2 Configuration Space Representation

The system has N degrees of freedom, each corresponding to the position of one robot along its path. We thus define the configuration space of the system as $\mathcal{C} = [0, L_1] \times \ldots \times [0, L_N]$. Configurations $x \in \mathcal{C}$ such that $G_{\mathrm{C}}(x)$ is disconnected are defined to be in *outage*. The obstacle region, $\mathcal{C}_{\mathrm{obs}}$, is the set of all configurations in outage. The set of all configurations not in outage is the free space, $\mathcal{C}_{\mathrm{free}}$. The problem under consideration is thus an instance of the General Motion Planning Problem, defined in Chapter 2. Once a solution path τ is found, it can be traversed at a suitable velocity to find the trajectory $x(t)$. The choice of method to solve this problem depends on, among other things, the geometry of $\mathcal{C}_{\mathrm{obs}}$ and the requirements on completeness of the solution.

Under the general problem formulation of Definition 6.1.1, the obstacles can be described as possibly non-convex semi-algebraic sets. This can be shown as in Figure 6.2, with two path segments and an obstacle. There, $\hat{e}_i$ and $\hat{e}_j$ are unit vectors in the direction of each path, so the position in the plane of robot i is $q_i(x) = p_i^1 + x_i\hat{e}_i$. The line of sight between robots i and j intersects the obstacle vertex w_k^n when, for some real λ,

$$p_i^1 + x_i\hat{e}_i + \lambda(w_k^n - p_i^1 - x_i\hat{e}_i) = p_j^1 + x_j\hat{e}_j.$$

If we let $v = w_k^n - p_i^1$ and $d = p_j^1 - p_i^1$, we get

$$\begin{bmatrix} \lambda \\ x_j \end{bmatrix} = \begin{bmatrix} v - x_i\hat{e}_i & -\hat{e}_j \end{bmatrix}^{-1} (d - x_i\hat{e}_i).$$

The inverse above does not exist in the special case when P_j is parallel to the line of sight from robot i to w_k^n, but this is easy to handle separately. Otherwise, we get the following condition for when the line of sight is on the obstacle side of w_k^n:

$$x_i(w_k^n - p_j^1)^\perp \cdot \hat{e}_i - x_j(w_k^n - p_i^1)^\perp \cdot \hat{e}_j - x_i x_j(\hat{e}_i \cdot \hat{e}_j^\perp) > v^\perp \cdot d, \tag{6.1}$$

where we define the orthogonality operator as a 90° counterclockwise rotation:

$$v^\perp \triangleq \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} v.$$

Note that if $\hat{e}_i \parallel \hat{e}_j$, the boundary of the half-space is a plane. We will make extensive use of this later in the chapter.

For a given pair $\{i, j\}$ of robots and an obstacle W_k, the intersection of the half-spaces (6.1) corresponding to all vertices of W_k, defines a region in $\mathcal{C}$ where that obstacle causes the link $\{i, j\}$ not to exist. Finding $\mathcal{C}_{\text{obs}}$ is then a combinatorial problem: For each set of links such that G_{C} cannot be connected without them, the intersection of their corresponding regions is a $\mathcal{C}$-obstacle. A very simple example is shown to the right in Figure 6.2, where three half-spaces intersect to form a non-convex $\mathcal{C}$-obstacle in $\mathbb{R}^2$. Note that with only two robots, we cannot pass the $\mathcal{C}$-obstacle, but in higher dimensions there may be a free path around it.

To summarize, this discussion shows that $\mathcal{C}_{\text{obs}}$ will in general be non-convex. As the number of robots grows, there is a combinatorial explosion in the number of, possibly intersecting, $\mathcal{C}$-obstacles. We now discuss how this affects the choice of motion planning method.

Of the exact motion planning methods described in Chapter 2, some, such as visibility graphs, are specifically tailored for 2D problems. Since the class of problems we consider typically has at least three robots, these methods are not suitable here. Further, potential-based methods explicitly represent the $\mathcal{C}$-obstacles, which may be inefficient since there will be so many. Instead, it appears more compact to represent $\mathcal{C}_{\text{free}}$, such as in exact cell decomposition methods. For general dimensions

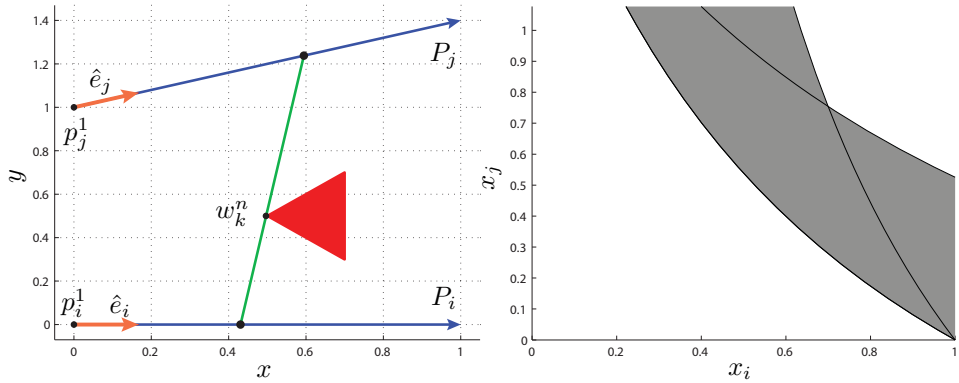


Figure 6.2: Notation for finding the half-space of $\mathcal{C}$ where the line of sight between robots i and j is on the obstacle side of the vertex w_k^n (left). The intersection of the half-spaces of each vertex is the $\mathcal{C}$-obstacle (grey) in the corresponding configuration space (right).

and obstacle geometries, the best such methods have worst-case time complexity that is exponential in the dimensionality. For our class of problems, this means that they do not scale very well with the number of robots.

To make the time complexity scale better, an alternative is to use sampling-based methods. We note that each problem is only solved once for each world geometry, so multi-query roadmap methods are not motivated here. Instead, one could use an RRT method. It is insensitive to the obstacle geometry and its time-complexity scales well with the dimensionality of the problem. The main drawback is that it is not complete, so in the very possible event that there is no solution, the solver runs forever.

To explore the strengths and weaknesses of exact and sampling-based methods, we have chosen to develop both an RRT solver and an exact solver based on cell decomposition. To make the exact solver feasible, we will restrict the problem some, as will be described in Section 6.4. But first we present the RRT solver.

6.3 Sampling-Based Solution

By connecting randomly chosen points in $\mathcal{C}_{\text{free}}$, we can construct a tree that eventually contains a path from the start to the goal configurations. The only problem-specific component is the collision detector, which we refer to as the outage detector. It determines if a new sample can be connected by a straight line to the nearest part of the tree without colliding with a $\mathcal{C}$-obstacle. We will present the RRT algorithm and then describe how the outage detector can be implemented. We end this section by discussing the computational complexity of the outage detector.

Rapidly Exploring Random Tree

The RRT algorithm builds a rapidly exploring random tree, which is a tree graph $G_{\mathrm{R}} = (V_{\mathrm{R}}, E_{\mathrm{R}})$, with Γ vertices $V_{\mathrm{R}} = \{r_1, \ldots, r_\Gamma\} \in \mathcal{C}_{\mathrm{free}}$ and edges $E_{\mathrm{R}} \in V_{\mathrm{R}} \times V_{\mathrm{R}}$. If there is an edge $\{r_i, r_j\} \in E_{\mathrm{R}}$, the straight line between r_i and r_j is contained in $\mathcal{C}_{\mathrm{free}}$. The number of vertices, Γ, grows as the tree is constructed, as described below in Algorithm 6.1, from LaValle (2006).

The algorithm iteratively constructs a tree that fills $\mathcal{C}_{\mathrm{free}}$. In each iteration, a random point $y \in \mathcal{C}$ is chosen. The function NEAREST(G_{R}, y) returns the configuration $x \in V_{\mathrm{R}}$ that is closest to y. If y is closer to a point between two configurations r_i, r_j such that $\{r_i, r_j\} \in E_{\mathrm{R}}$, than to a configuration x, the edge $\{r_i, r_j\}$ is split, a new vertex is inserted there and that vertex is returned. Then the outage detector FIRST_OUTAGE(x, y) returns a configuration $\tilde{y}$ on the straight line from x to y, as close to y as possible such that the line from x to $\tilde{y}$ is contained in $\mathcal{C}_{\mathrm{free}}$. If there is progress, so $\tilde{y} \neq x$, we add this new configuration and the corresponding edge to G_{R}.

Algorithm 6.1 Rapidly Exploring Random Tree (LaValle, 2006)

$\Gamma := 1$
$V_{\mathrm{R}} := \{\mathbf{0}\}$
$E_{\mathrm{R}} := \emptyset$
loop
  Randomly select $y \in \mathcal{C}$
  $x :=$ NEAREST(G_{R}, y)
  $\tilde{y} :=$ FIRST_OUTAGE(x, y)
  if $\tilde{y} \neq x$ **then**
    $\Gamma := \Gamma + 1$
    $V_{\mathrm{R}} := V_{\mathrm{R}} \cup \tilde{y}$
    $E_{\mathrm{R}} := E_{\mathrm{R}} \cup \{x, \tilde{y}\}$
  end if
end loop

Note that Algorithm 6.1 will run forever, giving an RRT that is arbitrarily close to any point in $\mathcal{C}_{\mathrm{free}}$. To make it terminate in finite time, every 100th iteration, we replace the random y with x_{G} and abort if $\tilde{y} = x_{\mathrm{G}}$. Then the RRT contains a path to the goal. Next, we describe how the outage detector can be implemented.

Outage Detector

The outage detector answers the question of how far the system can go from a configuration x towards another, y, before intersecting a $\mathcal{C}$-obstacle, *i.e.*, going into outage.

Since under Algorithm 6.1, $x \in V_{\mathrm{R}}$, the communication graph $G_{\mathrm{C}}(x)$ is always connected. Hence, it is sufficient to check how far the system can go towards y

before any link $\{i,j\} \in E_{\mathrm{C}}$ is blocked by an obstacle. We update $\tilde{y}$ to this location and search $G_{\mathrm{C}}(\tilde{y})$ for an indirect path between vertices i and j. If it exists, the link $\{i,j\} \in E_{\mathrm{C}}$ was redundant, so we can continue. The iteration ends if $G_{\mathrm{C}}(\tilde{y})$ is disconnected or we reach y, as summarized in Algorithm 6.2.

Algorithm 6.2 $\tilde{y} = \mathrm{FIRST\_OUTAGE}(x, y)$

1: $\tilde{y} := x$
2: **while** $\tilde{y} \neq y$ and $G_{\mathrm{C}}(\tilde{y})$ is connected **do**
3: $\quad \gamma^* := \min_{e \in E_{\mathrm{C}}(\tilde{y})} \min_{k \in K(e)} \min_{n \in \{1,\ldots,\Omega_k\}} \gamma(e, w_k^n, \tilde{y}, y)$
4: $\quad \tilde{y} := \tilde{y} + (y - \tilde{y}) \min(1, \gamma^*)$
5: **end while**

To check $G_{\mathrm{C}}(x)$ for connectivity in the iteration condition, we do a breadth-first search with robot i as the root. Algorithm 6.2 uses the set

$$K(e) \triangleq \{k : \exists\, x \in \mathcal{C} : \mathrm{convhull}(r_i(x), r_j(x)) \cap W_k \neq \emptyset\},$$

which are the indices of all obstacles that may block the link $e = \{i,j\}$. It also uses the function $\gamma(e, w_k^n, x, y)$, which, for a link $e = \{i,j\}$, is defined as

$$\begin{aligned}\gamma(e, w_k^n, x, y) \triangleq \min\{g : (1-\lambda)\,[(1-g)r_i(x) + g r_i(y)] \\ + \lambda\,[(1-g)r_j(x) + g r_j(y)] = w_k^n, g \geq 0, 0 < \lambda < 1\}.\end{aligned}$$

Finding candidate solutions g requires finding the intersections of two rectangular hyperbola or, in degenerate cases, of two straight lines. As illustrated in Figure 6.3, $\gamma(e, w_k^n, x, y)$ is the fraction of the straight line from x to y that the system can move before the LOS between robots i and j intersects the obstacle vertex w_k^n. For simplicity, we assume that both robots i and j move on one single segment of their paths. In the case of multi-segment paths, the computation is done separately over each interval of g corresponding to different combinations of path segments. If the problem is infeasible, we let $\gamma = \infty$. Note that we do allow solutions where $\gamma > 1$. This allows the result to be reused to speed up computations, as described in Section 6.3.

When implementing Algorithm 6.2, there is a numerical issue that requires special attention. Many new vertices $\tilde{y} \neq y$ are generated on the boundary of $\mathcal{C}$-obstacles. Answering an outage detector query where x is on the boundary of $\mathcal{C}_{\mathrm{free}}$ is numerically sensitive, due to limited precision. For all vertices generated by collisions with $\mathcal{C}$-obstacles, it may therefore be useful to associate information on which link $e = \{i,j\}$ came in conflict with which obstacle vertex w_k^n. Then subsequent queries starting from x can first check if the direction of movement is legal with respect to e and w_k^n. This is equivalent to

$$\min_{m \in \{1,\ldots,\Omega_k\} \setminus n} \gamma(e, w_k^m, x, y) = \infty.$$

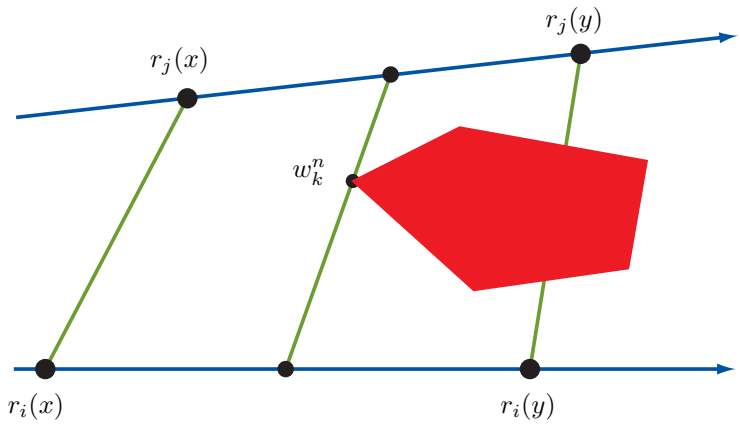


Figure 6.3: The core of the outage detector: For a link $e = \{i, j\}$, at what fraction $\gamma(e, w_k^n, x, y)$ of the motion from configuration x to y does the line of sight of robots i and j intersect the obstacle vertex w_k^n?

If the criterion is not fulfilled, it means that movement from x in the direction of y will eventually cause the line of sight of the link e to intersect another vertex of W_k.

Computational Complexity

In worst case, finding γ^* in each iteration of Algorithm 6.2 requires checking M obstacles per link and N^2 links. If there is more than one iteration, $\gamma(e, w_k^n, x, y)$ is never recomputed for a link. Instead, one only needs to subtract γ^* to get an updated value. Evaluating the iteration condition on connectivity requires testing at most N^2 links against M obstacles in each iteration. In worst case, there could be $O(MN^2)$ iterations. Since computing $\gamma(e, w_k^n, x, y)$ and testing a link against an obstacle are constant-time operations, the worst-case time complexity of each query to the collision detector is $O(M^2N^4)$.

As mentioned earlier, the RRT algorithm is only probabilistically complete, so its worst-case execution time is unbounded. As described by LaValle (2006), the running time of course increases with the dimensionality N of $\mathcal{C}$, but also heavily depends on the presence of narrow bottlenecks in $\mathcal{C}_{\text{free}}$. In our problem, this corresponds to high obstacle density, so only a small set of robot configurations allows the group to pass without going into outage. Below, we will illustrate how the obstacle density affects the solution times.

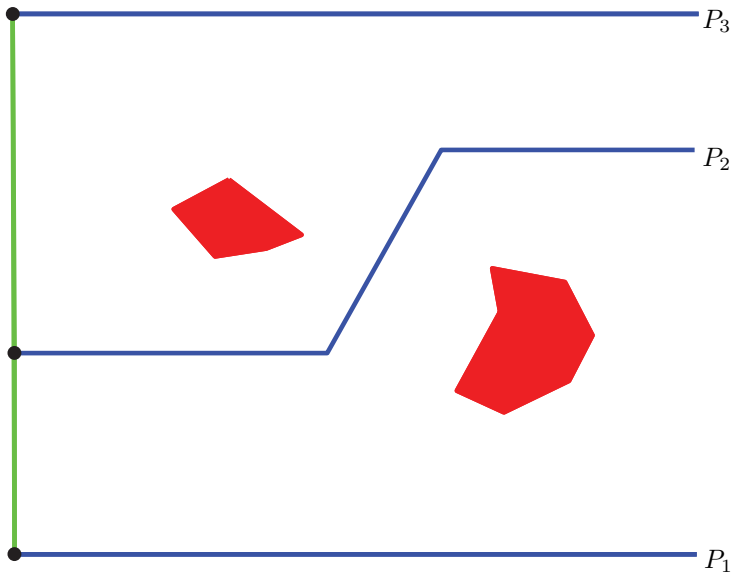


Figure 6.4: A simple example scenario with three robots and two obstacles. The robots are depicted in the initial configuration.

Simulations

In this section, we show some simulations to illustrate the solution method. We also demonstrate how the solution times increase when the obstacle density is higher, creating narrower bottlenecks in $\mathcal{C}$. All simulations were made in Matlab, and we have not optimized the implementation for speed. It mainly serves as a proof of concept and to elucidate the relative differences in typical solution times.

As a small illustrative example, Figure 6.4 shows a scenario with two obstacles and three paths. The robots are depicted in the start configuration. The corresponding configuration space is shown in Figure 6.5. It shows the $\mathcal{C}$-obstacles, drawn as point clouds. When the system is at a configuration x inside a $\mathcal{C}$-obstacle, it means that the group is in outage, *i.e.*, that not enough robots have a clear line of sight for the communication graph $G_{\mathrm{C}}(x)$ to be connected. The RRT G_{R} was grown between the obstacles, starting at $(0, 0, 0)$ and expanding through free space towards the goal x_{G}. Its vertices are drawn as circles and the edges are drawn as straight lines connecting the circles. Figure 6.6 shows eight snapshots of the resulting solution, with dashed red lines depicting lines of sight that are blocked. A movie of the resulting robot motion can be found at `http://www.ee.kth.se/~lindhe`.

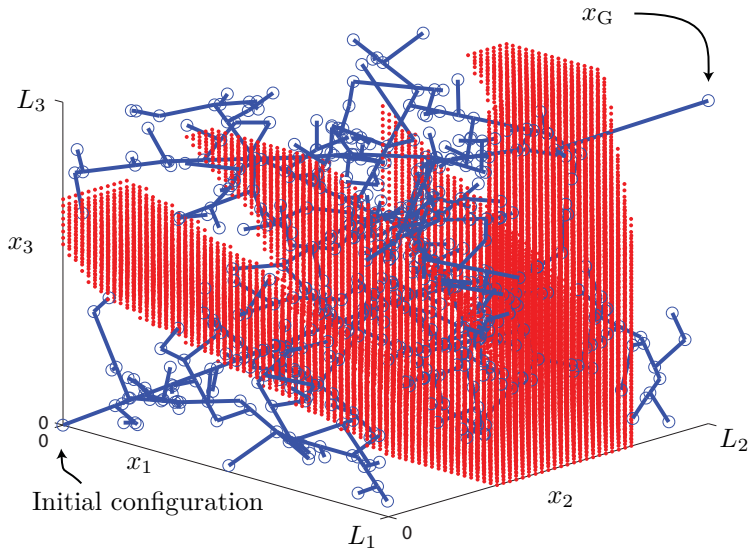


Figure 6.5: Configuration space corresponding to the example in Figure 6.4. The RRT vertices are shown as blue circles, joined by edges in the form of blue lines. The $\mathcal{C}$-obstacles are shown as red point clouds.

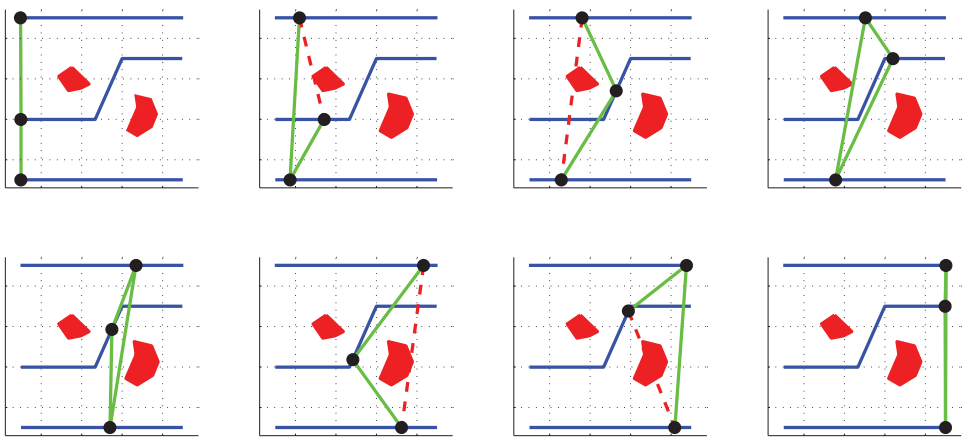


Figure 6.6: Snapshots of a solution trajectory for the example in Figure 6.4. The robots stay connected during the whole trajectory, even though some lines of sight (red dashed lines) are obstructed by obstacles.

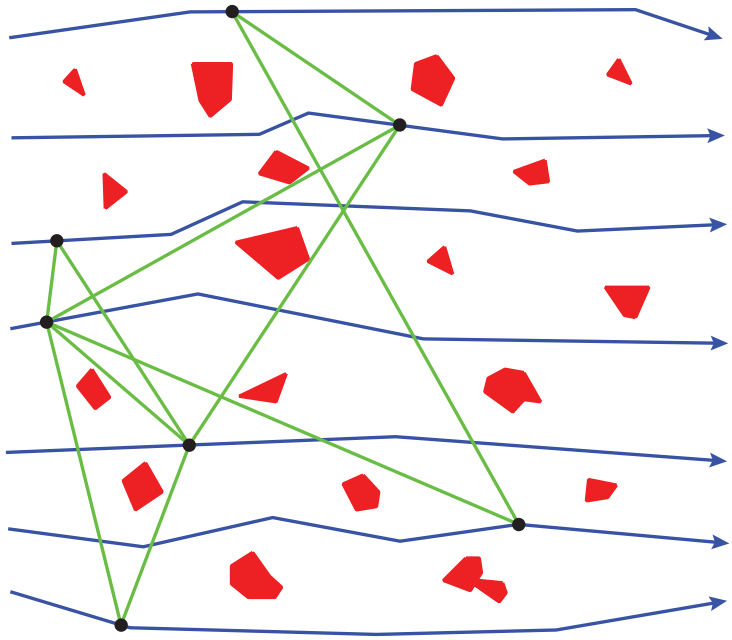


Figure 6.7: This problem instance with seven robots and 19 obstacles took 2 min 24 s to solve. The figure shows a snapshot of the solution.

To illustrate the size of problems that is feasible, we also constructed a larger example scenario, in Figure 6.7. It contains 7 robots and 19 obstacles and it took 2 min 24 s to compute a solution on a laptop with an Intel Core 2 Duo processor at 2.2 GHz and 2 GB of RAM. A movie of the solution can be found at the same location as above.

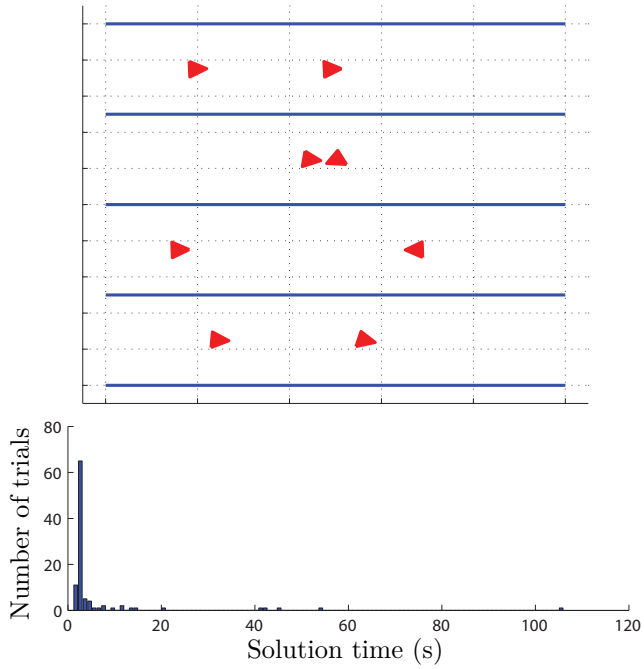


Figure 6.8: Solution times for 100 randomized scenarios with small obstacles. One example realization is shown above.

Then, to demonstrate how the solution times depend on the obstacle density, we defined a scenario with five paths and two triangular obstacles between each path. The triangles were vertically centered between the paths, but the horizontal position and the orientation were randomized for each trial. We first made 100 trials with small obstacles, where the base of the triangles was 20% of the distance between paths. Figure 6.8 shows a histogram of the solution times, along with one realization of a scenario. All scenarios were solved and the mean solution time was 6.3 s.

As a comparison, we made 100 similar trials with larger obstacles, where the base of each triangle was 50% of the distance between paths. Figure 6.9 shows the resulting solution time histogram and an example realization. (Note the different time scale from Figure 6.8.) The maximum size of the RRT was bounded to 50 000 nodes, and with this termination rule, 89% of the scenarios were solved. The number of nodes was roughly proportional to the solution time, and the search was terminated after about 700 s. The mean solution time for the solved scenarios was 78 s.

We finally note that in similar experiments, with all 100 trials using the same scenario, the randomness of the RRT caused a similar spread in solution times, both for scenarios with small and large obstacles. This is shown i Figure 6.10 and

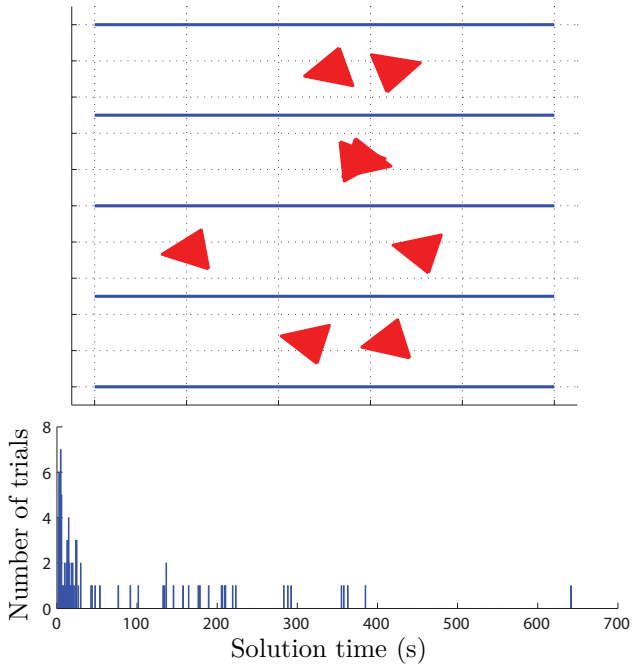


Figure 6.9: Solution times for 100 randomized scenarios with larger obstacles. One example realization is shown above. Larger obstacles cause narrow bottlenecks in the configuration space, slowing down the RRT solver.

Figure 6.11 for the base of the triangular obstacles being 20% and 50% of the path distance, respectively. This shows that the spread in Figure 6.8 and Figure 6.9 is not mainly due to variations in the difficulty of the randomized scenarios, but the inherent randomness of the RRT.

These tests illustrate two things: First, as expected, the solution times increase when the obstacles become denser, since there are narrower bottlenecks that the RRT needs to pass through. Second, the solution time distribution has a long tail of solvable scenarios where the solution takes very long to find. As mentioned in Section 6.2, this is a known problem with this class of sampling-based solvers. In our application, this means it will be difficult to set a termination time when the problem is assumed unsolvable. Not only does the solution time inherently vary in the RRT method, but the distribution also depends on the obstacle density. Because of this, we now go on to present an exact solution method.

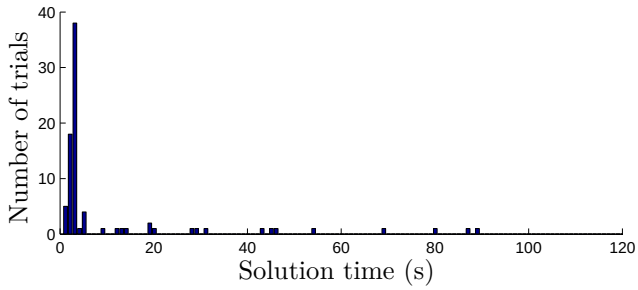


Figure 6.10: Solution times for 100 trials with the RRT solver on the same scenario, with five paths and two small triangular obstacles between each path. For comparison with Figure 6.8, 26 trials that took longer than 120 s are not included. Three trials were aborted before finding a solution. The spread is due to the randomness of the RRT.

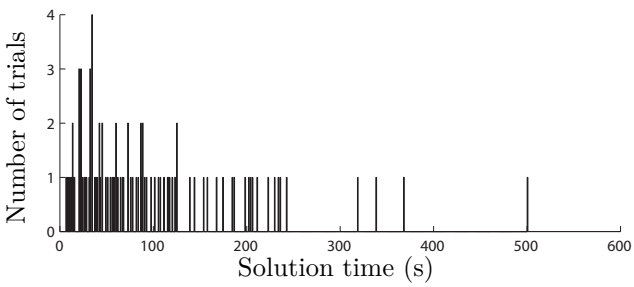


Figure 6.11: Solution times for 100 trials of the RRT solver on the same scenario, with five paths and two *large* triangular obstacles between each path. Two trials were aborted before finding a solution. The spread is due to the randomness of the RRT.

6.4 Exact Solution

Our proposed exact solution is similar to cell decomposition methods, as described in Chapter 2. But in contrast to the standard methods, the cells are allowed to intersect, so they do not form a partition of $\mathcal{C}_{\text{free}}$. Before presenting the cell decomposition and how to search the resulting graph of adjacent cells, we introduce two restrictions to the general problem in Definition 6.1.1. At the end of this section, we will comment further on what is gained by this, and how the solution relates to the general problem.

Restricted Problem Formulation

We first introduce the restriction that the paths $P_1, \ldots, P_N$ are parallel. We will see later that this restriction allows us to represent the cells as convex sets, making it simple to test for adjacency of cells. We can then, without loss of generality, orient the coordinate system so that the position of robot i is $q_i(t) = (x_i^0 + x_i(t), y_i)$, where we assume that x_i^0 is constant and y_i is constant and unique. This fixed vertical ordering allows numbering the robots such that $y_1 < y_2 \ldots < y_N$. The *neighbors* of robot i are $i-1$ and $i+1$ if these exist. The convex hull of the paths of robots i and $i+1$ is called *corridor* i.

Second, we restrict the types of connectivity that we consider. We define the graph $G_{\text{C}}(x)$ to be *locally connected* if and only if there exists a $j \in \{1, \ldots, N-1\}$ such that

$$\{i, i+1\} \in E_{\text{C}}(x) \, \forall \, i \in \{1, \ldots, N-1\} \setminus j$$

and

$$\{\{j, j+1\}, \{j-1, j+1\}, \{j, j+2\}\} \cap E_{\text{C}}(x) \neq \emptyset.$$

This means that there is a link between all neighbors except possibly a single pair $\{j, j+1\}$, which must then be indirectly connected by any of the two-hop links $\{j-1, j+1\}$ or $\{j, j+2\}$. This makes routing simple and still allows flexibility to pass obstacles by local rerouting. Introducing this restricted notion of connectivity will make the search of the cell adjacency graph faster. To simplify the presentation, analogously to the problem in Definition 6.1.1, we assume that the initial and final configurations are locally connected, since otherwise the problem is trivially infeasible.

Based on the models defined earlier and the restrictions above, we can now formulate the problem of following parallel paths while preserving local connectivity:

Definition 6.4.1 (Parallel Path Following with Local Connectivity)**.** *Given parallel paths $P_1, \ldots, P_N$ and obstacles $W_1, \ldots, W_M$, find a time-continuous state trajectory $x : [0, T] \to \mathbb{R}^N$, such that $x(0) = \mathbf{0}$, $x(T) = x_G$ and $G_C(x(t))$ is locally connected for all $t \in [0, T]$.*

As before, instances of this problem may be unsolvable. The proposed solution method either finds a solution or in finite time concludes that there is none. In this

case, a planner on a higher level could adjust some paths, relax the connectivity constraint or add more robots. In the following, we will describe how the configuration space can be decomposed into cells and then present the proposed solution strategy.

Cell Decomposition of Free Space

Following from the above definition of local connectivity, free space is the union of two types of connected subsets, or cells: First there are cells with nearest-neighbor connectivity, *i.e.*, where all links $\{i, i+1\}$ exist, called *n-cells*. Then there are cells where a two-hop link of type $\{i, i+2\}$ exists, allowing one nearest-neighbor link to be broken so the robots can pass an obstacle. We call these *t-cells*, since they act as tunnels between the n-cells. Regions where n-cells and t-cells intersect correspond to configurations where it is possible to switch network topology without losing connectivity. We now formally define cells and show that they are convex polyhedra, so free space can be described as the union of overlapping convex polyhedra.

To fully specify a cell we must state the network topology and how every link in the topology interacts with each obstacle, *i.e.*, on which side of the obstacle it passes. Let T be a minimal locally connected topology, *i.e.*, a set of links $\{i, j\}$ such that (V_C, T) is locally connected but connectivity is lost if any link in T is removed. This makes T a tree topology. Also let $L(e,k) : V_C \times V_C \times \{1,\dots,M\} \to \{-1,0,1\}$ be a LOS constraint, i.e., a rule that defines how the line of sight corresponding to link e should relate to obstacle k:

$$\begin{cases} L = 1 \Leftrightarrow \text{ the LOS should be on the left of } W_k \\ L = -1 \Leftrightarrow \text{ the LOS should be on the right of } W_k \\ L = 0 \Leftrightarrow \text{ no constraint} \end{cases}$$

Note that right and left are well-defined for instances of our problem, since all paths lead from left to right. The empty constraint $L(e,k) = 0$ is used when $e \notin T$, or if there is no x such that e intersects W_k. With a slight abuse of notation, we say that the rule $L(e,k)$ is *satisfied* if and only if link e and obstacle k are related as specified by $L(e,k)$. The empty constraint $L(e,k) = 0$ is always considered satisfied. Now we can define a cell:

Definition 6.4.2. *A cell* (T, L) *is a set*

$$\{x \in \mathcal{C}_{free} : T \subseteq E_C(x) \wedge L(e,k) \text{ satisfied } \forall\, e \in T, k \in \{1,\dots,M\}\}.$$

Each cell can be expressed by a set of linear inequality constraints, derived as in Figure 6.2. In this special case, (6.1) shows that the LOS between robots i and $j > i$ is to the *left* of the obstacle vertex w_k^n if and only if

$$-(x_i^0 + x_i)(w_{k,y}^n - y_j) + (x_j^0 + x_j)(w_{k,y}^n - y_i) < w_{k,x}^n (y_j - y_i).$$

So the cell (T, L) can be expressed as

$$x \in (T,L) \Leftrightarrow \forall\, e=\{i,j\} \in T, k \in \{1,\dots,M\}, n \in \{1,\dots,\Omega_k\}:$$
$$-L(e,k)(x_i^0+x_i)(w_{k,y}^n-y_j)+L(e,k)(x_j^0+x_j)(w_{k,y}^n-y_i) < L(e,k)w_{k,x}^n(y_j-y_i). \tag{6.2}$$

We can now define the different types of cells: An n-cell is a cell where $T = T_0$, where

$$T_0 = \bigcup_{i\in\{1,\dots,N-1\}} \{i, i+1\}.$$

A t-cell has one of the two following types of topologies, depending on which replacement link is used:

$$T_{i+} = T_0 \setminus \{i,i+1\} \cup \{i,i+2\}$$
$$T_{i-} = T_0 \setminus \{i,i+1\} \cup \{i-1,i+1\}$$

As an example, Figure 6.14b shows a configuration that is in the intersection of a T_{3-} cell and a T_0 cell. If the obstacles are numbered from below, the nonempty LOS constraints for the T_{3-} cell are $L(\{1,2\},1) = 1$, $L(\{2,3\},2) = 1$, $L(\{2,4),2\} = 1$ and $L(\{2,4\},3) = 1$. In the following, we will alternatively denote n-cells as T_0 cells and t-cells as $T_{i\pm}$ cells.

Figure 6.12 shows a simple example scenario, which corresponds to the configuration space representation in Figure 6.13. All three nonempty n-cells are drawn in red and the green t-cell, acting as a tunnel between two n-cells, has topology T_{2-}.

Now that we have defined the cell decomposition of $\mathcal{C}_{\text{free}}$, we turn to the problem of searching for a sequence of n- and t-cells that form a sequence from the initial to the final configuration.

Searching the Cell Adjacency Graph

When two cells intersect, so the system can pass between them without losing local connectivity, we consider them adjacent. Starting from a cell that contains the initial configuration, we build a tree graph of adjacent cells, until we find a cell containing the final configuration or the graph is fully explored. The latter case means that there is no solution. We now give an algorithm for exploring the graph and derive two properties of the graph that simplify the exploration.

We define the cell adjacency graph as $(V_{\text{A}}, E_{\text{A}})$, where the vertex set V_{A} is a set of explored cells (T, L). It is composed of two disjoint subsets, V_{A}^{E} and V_{A}^{U}. They contain cells whose neighbors are all explored or not, respectively. $E_{\text{A}} \in V_{\text{A}} \times V_{\text{A}}$ is the set of edges, such that the cells (T, L) and (T', L') intersect if there is an edge $\{(T, L), (T', L')\} \in E_{\text{A}}$. To get efficient paths and reduce the search time, we search for a path that passes the minimum number of cells. This is achieved with A* searching (LaValle, 2006), using two cost functions: Let $g(T, L)$ be defined for

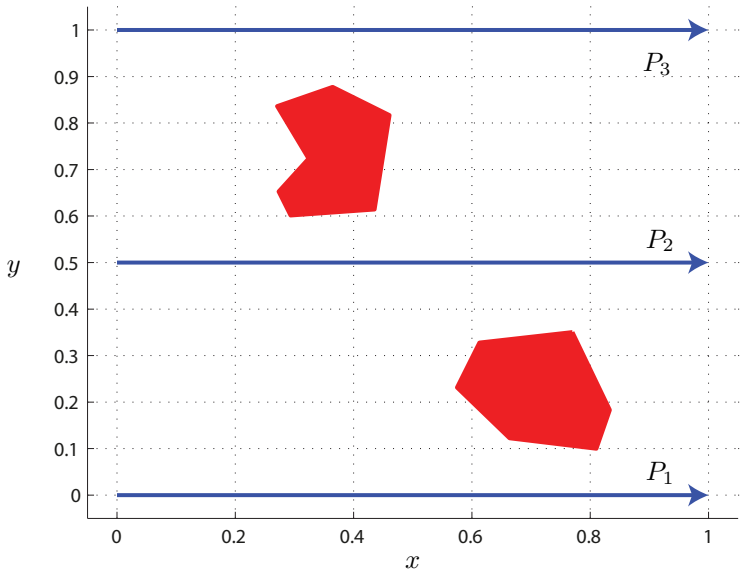


Figure 6.12: An example scenario with three paths and two obstacles. The corresponding configuration space representation is illustrated in Figure 6.13.

all $(T, L) \in V_{\mathrm{A}}$ as the minimum number of edges from the start to (T, L). Also let $h(T, L)$ be the minimum number of edges between (T, L) and the goal cell. To find h, we note that it takes a sequence of at least a T_0 and a $T_{i\pm}$ cell to pass obstacles in a corridor. Then the path needs to go through a T_0 cell to pass obstacles in another corridor. So for a T_0 cell, $h(T_0, L)$ is equal to two times the number of corridors where there are still obstacles to the right of the LOS. For a $T_{i\pm}$ cell, it is two times the number of corridors, except corridor i, where there are still obstacles on the right of the LOS, plus one. If a feasible path is found, we can solve for a point in each cell intersection in polynomial time, using (6.2), and then form a path through $\mathcal{C}_{\mathrm{free}}$ by linear interpolation between the points. This path can be traversed at arbitrary velocity to get the trajectory $x_i(t)$ for each robot.

Algorithm 6.3 uses NEW_NEIGHBORS(T, L), which returns the set of all cells that intersect (T, L) but are not in V_{A}. We now derive two properties of the graph, which reduce the number of possible neighbor cells that must be considered:

Property 6.4.3. *Two cells (T, L) and (T', L') are disjoint if there exist $e \in T \cap T'$ and $k \in \{1, \ldots, M\}$ such that $L(e, k) \neq L'(e, k)$.*

This follows since no x fulfills (6.2) for both L and L' in this case. Property 6.4.3 means that from a T_0 cell, we only need to consider adjacency to $T_{i\pm}$ cells where the LOS constraints are the same, except for the link that replaces $(i, i+1)$. No T_0 cells are adjacent.

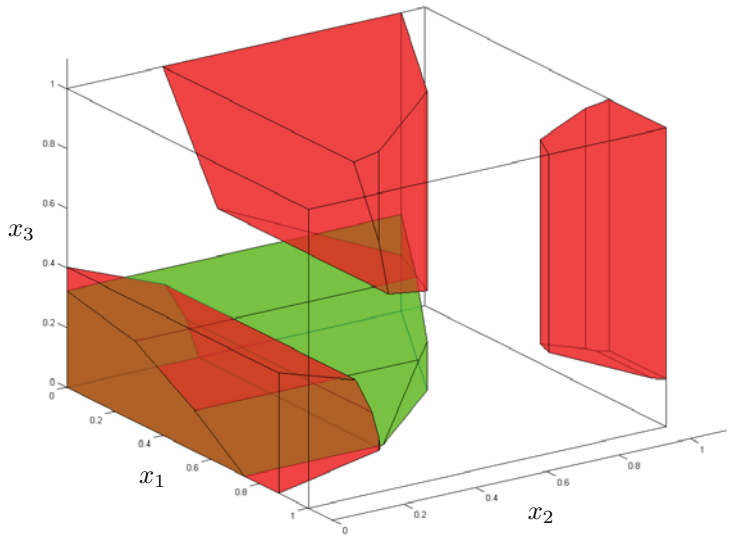


Figure 6.13: The configuration space corresponding to the scenario in Figure 6.12. There are three disjoint n-cells (red), *i.e.*, regions of $\mathcal{C}$ where the robots have nearest-neighbor connectivity. One green t-cell is also shown, where the links $\{1, 2\}$ and $\{1, 3\}$ are available. This topology can be used to pass between n-cells with maintained connectivity.

Algorithm 6.3 A* Search in the Cell Adjacency Graph

Find a cell (T, L) that contains $x = \mathbf{0}$
$V_{\mathrm{A}}^{\mathrm{E}} := \emptyset$, $V_{\mathrm{A}}^{\mathrm{U}} := (T, L)$, $E_{\mathrm{A}} := \emptyset$
loop
  if $V_{\mathrm{A}}^{\mathrm{U}} = \emptyset$ **then**
    Terminate, there is no solution
  end if
  $(T, L) := \mathrm{argmin}_{(T', L') \in V_{\mathrm{A}}^{\mathrm{U}}} g(T', L') + h(T', L')$
  $V_{\mathrm{A}}^{\mathrm{E}} := V_{\mathrm{A}}^{\mathrm{E}} \cup (T, L)$
  $V_{\mathrm{A}}^{\mathrm{U}} := V_{\mathrm{A}}^{\mathrm{U}} \setminus (T, L) \cup \mathrm{NEW\_NEIGHBORS}(T, L)$
  $E_{\mathrm{A}} := E_{\mathrm{A}} \cup \{(T, L), \mathrm{NEW\_NEIGHBORS}(T, L)\}$
  if $x_{\mathrm{G}} \in \mathrm{NEW\_NEIGHBORS}(T, L)$ **then**
    Terminate, a path is found
  end if
end loop

Property 6.4.4. *The intersection of t-cells with topologies $T_{i\pm}$ and $T_{j\pm}$, where $i \neq j$, is a subset of an n-cell.*

In the intersection, all links $T_{i\pm} \cup T_{j\pm}$ must be available. Since $T_0 \subset T_{i\pm} \cup T_{j\pm}$, the intersection is a subset of an n-cell. Together with Property 6.4.3, this means that from a T_{i+} cell, without loss of generality, we can consider only adjacency with T_{i-} or T_0 cells, and conversely from a T_{i-} cell. We finally note that many cells can be empty, but this also means that the intersection with any existing cell is empty.

Solution Completeness and Complexity

The cell adjacency graph will contain finitely many cells, so Algorithm 6.3 will in finite time either find a solution or conclude that the problem is unsolvable.

To determine the worst-case time complexity of searching the graph, we let m_i be the number of obstacles in corridor i. Obviously, $\sum_{i=1}^{N-1} m_i = M$. A T_0 cell has one link through each corridor and it is uniquely defined by which obstacles in the corridor are on the left or right of the link. Thus there are

$$\prod_{i=1}^{N-1} 2^{m_i} = 2^M$$

possible T_0 cells. A T_{i+} cell has nearest-neighbor links in each corridor, except corridor i. Instead, the link $\{i, i+2\}$ passes corridors i and $i+1$, which gives

$$2^{m_i} 2^{m_{i+1}} \prod_{j \in \{1,\dots,N-1\} \setminus i} 2^{m_j} = 2^{m_i+1} 2^M \leq 2^{2M}$$

possible combinations. Similarly, there are at most 2^{2M} T_{i-} cells.

Due to Property 6.4.3, all possible neighbors of a T_0 cell can be constructed by removing one link $\{i, i+1\}$ and replacing it with either $\{i, i+2\}$ or $\{i-1, i+1\}$. The former can have $2^{m_i} 2^{m_{i+1}}$ combinations of LOS constraints, and the latter can have $2^{m_{i-1}} 2^{m_i}$ combinations. In total, that gives

$$\sum_{i=1}^{N-2} 2^{m_i} 2^{m_{i+1}} + \sum_{i=2}^{N-1} 2^{m_{i-1}} 2^{m_i} = 2 \sum_{i=1}^{N-2} 2^{m_i} 2^{m_{i+1}} \leq 2N 2^M$$

possible neighbors of each T_0 cell. Every T_{i+} cell can have at most 2^{m_i} T_0 cells and $2^{m_{i-1}} 2^{m_i}$ T_{i-} cells as neighbors. Analogously, a T_{i-} cell can have the same number of T_0 cell neighbors and at most $2^{m_i} 2^{m_i+1}$ T_{i+} cell neighbors. So a $T_{i\pm}$ cell has at most $2 \cdot 2^M$ possible neighbors.

Thus there are at most $2^M \cdot 2N \cdot 2^M$ edges from an n-cell to a t-cell. And there are at most $2^{2M} \cdot 2 \cdot 2^M$ edges from a t-cell to an n-cell. Testing for adjacency is a linear programming problem with $\mathcal{O}(NM)$ constraints as in (6.2). Its time complexity is polynomial in the number of constraints (Boyd and Vandenberghe, 2004). We summarize this discussion in the following proposition:

Proposition 6.4.5 (Completeness and Complexity)**.** *Algorithm 6.3 is complete. Its worst-case time complexity is exponential in the number of obstacles, M, and polynomial in the number of robots, N.*

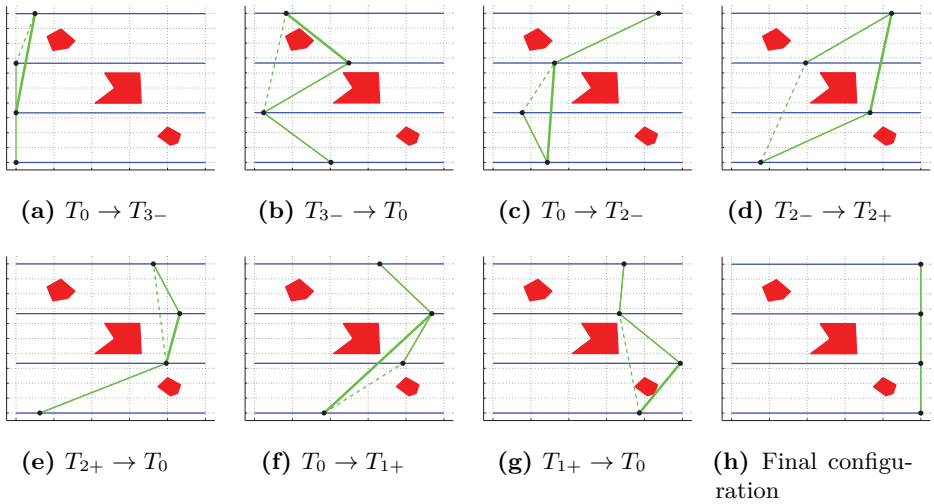
As mentioned in Chapter 2, the complexity of exact cell decomposition methods is usually dominated by the dimensionality of the state space, since this affects the number of possible neighbors of each cell. In our case, the number of neighbors instead depends on the number of obstacles, so this is the main limitation. Note, however, that the problem can be trivially decomposed if there is a corridor without any obstacles, so in practice, M is lower bounded by $N - 1$. In the conclusions, we will comment on how larger problems could be decomposed to reduce the number of obstacles. But first, we describe some simulations.

Simulations

To illustrate the method and some of its properties, we include two simulations. The first is a simple scenario to show the solution trajectory, and the second is a larger example to show what problem sizes are practically solvable. All simulations are made in Matlab on a laptop with an Intel Core i7 processor running at 2.7 GHz and with 8 GB of RAM. Movies of the resulting trajectories are available at `http://www.ee.kth.se/~lindhe`.

Figure 6.14 shows a small example with four robots and three obstacles, which took 2.9 s to solve. The figure shows snapshots of the motion, at every instance when the robots move between cells and switch topologies. The blue lines are the robot paths, black circles are robots and green lines show the LOS links. Dashed lines are the links that are given up at each switch, and wider lines are the new links.

A larger example is given in Figure 6.15, with six paths and ten obstacles. It took 9 min 26 s to solve, and the resulting cell adjacency graph had 381 T_0 cells. (For efficiency, only the T_0 cells are stored and once a solution path is found, the sequence of $T_{i\pm}$ cells is reconstructed along the path.) The solution path passes 21 cells.



(a) $T_0 \to T_{3-}$ **(b)** $T_{3-} \to T_0$ **(c)** $T_0 \to T_{2-}$ **(d)** $T_{2-} \to T_{2+}$

(e) $T_{2+} \to T_0$ **(f)** $T_0 \to T_{1+}$ **(g)** $T_{1+} \to T_0$ **(h)** Final configuration

Figure 6.14: Snapshots of four robots passing three obstacles. At each snapshot but the last, the robots are switching topologies, as labeled underneath. The link that is given up after the switch is dashed, and the new link that is established is drawn wider.

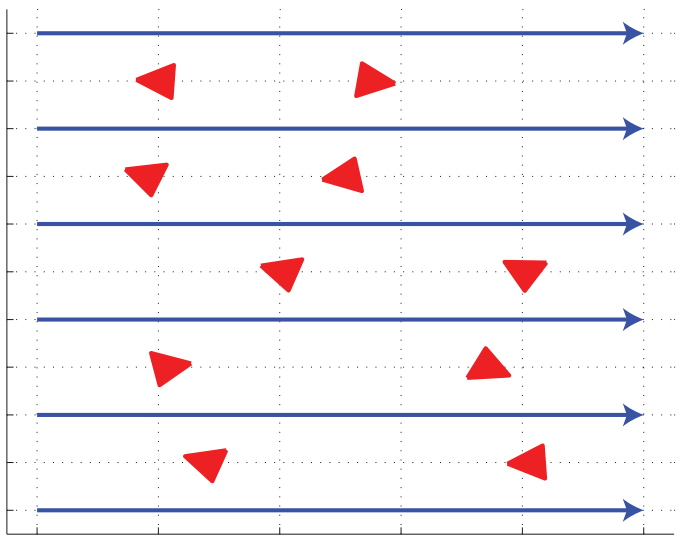


Figure 6.15: A larger example scenario with six paths and ten obstacles, solved in 9 min 26 s.

6.5 Summary

We have presented two methods of computing visibility-constrained reference trajectories, suitable for integration in the coordination layer of a motion planner. Both methods construct a tree inside the free space, starting from the initial configuration and trying to reach the final configuration. In the RRT case, the tree is grown by adding straight line edges from the tree towards randomly chosen points. This strategy works well in cases with sparse $\mathcal{C}$-obstacles and its time complexity scales favorably with the number of obstacles and robots. The main drawback is that it cannot in finite time conclude that a problem instance is unsolvable. Also, the time to compute a solution exhibits a large spread due to the randomness of the method, making it difficult to set an empirical termination time for the solver. The exact solution, on the other hand, represents free space as a tree of partially intersecting convex cells. The cells can in finite time fill all reachable parts of $\mathcal{C}_{\text{free}}$, making it possible to determine that there is no solution. The drawbacks of the method are the high worst-case running time and that it applies only to a restricted class of problems. In Chapter 7, we discuss how the problem could be decomposed by using a receding horizon approach. We will now comment further on the restrictions of the problem.

The proposed cell decomposition solution utilizes two key assumptions: that of parallel paths and the restriction to only consider local connectivity. By assuming parallel paths, we get convex cells, so path planning inside a cell is trivial and solving for the intersection of two cells is a convex problem. In the case of non-parallel paths, it would be an interesting direction of future research to investigate other parameterizations of $\mathcal{C}$, which may yield convex cells. We expect the results on adjacency and complexity of the cell graph to carry over also to this case. Extending that to multi-segment paths would mean expanding the configuration space into multiple hyperrectangles, each of which corresponds to a combination of segments. The other assumption, restricting to local connectivity, gives simple network routing and also lowers the number of possible cells, making the cell adjacency tree faster to search. It would be straightforward to allow a larger class of connected topologies. It would just make the search tree larger.

We note that that it would be hard to compare the sampling-based and exact methods on the same scenario, for two reasons. First, the comparison would be unfair since the scenario must have parallel paths, which the exact method is tailored for, unlike the RRT method, which can handle a more general class of problems. Second, the exact method is insensitive to narrow bottlenecks and for given N and M, its solution time only depends on the topology of the cell adjacency graph. We have seen that the solution time of the RRT method, on the other hand, depends strongly on the obstacle density. So the comparison would be very sensitive to the problem geometry.

In the introduction to this chapter, we commented that this problem bears similarities to the path coordination problem. A solution to this was proposed by Siméon et al. (2002), exploiting a cylindrical property of the $\mathcal{C}$-obstacles, namely that colli-

sions between robots i and j are uniquely determined by the projection of $\mathcal{C}$ onto the two-dimensional subspace (x_1, x_j). They also use a decomposition of the robots into smaller interacting subsets. In our general problem formulation of Definition 6.1.1, the $\mathcal{C}$-obstacles are not cylindrical. The cylindrical property does, however, apply when we restrict the problem to local connectivity in Definition 6.4.1. Connectivity for robots i and $i+1$ is then uniquely determined by a projection of $\mathcal{C}$ onto the four-dimensional subspace $(x_{i-1}, x_i, x_{i+1}, x_{i+2})$. The proposed decomposition is however not applicable to neither problem formulation, since the connectivity constraint guarantees that there are no non-interacting subsets of robots.

Chapter 7

Conclusions

This chapter contains a summary of the work presented in the thesis, some conclusions and suggestions for possible extensions. We first discuss how the results fit into the layered architecture of communication-aware motion planning. Then we summarize the work on exploiting multipath fading. Finally, we discuss the proposed methods for maintaining visual connectivity.

7.1 Communication-Aware Motion Planning

This thesis assumes a layered architecture for robot motion planning, where a mission planner computes waypoints that must be reached to complete some high-level task. A coordination layer translates the waypoints into reference trajectories for each robot, such that the robots avoid collisions with obstacles or each other. Finally, a motion controller in each robot tracks its reference trajectory, using feedback to compensate for disturbances or model errors. We suggest doing communication-aware motion planning by modifying the layers to also take communication constraints into account, while maintaining the interfaces of the architecture.

We have proposed and analyzed two communication-aware components that are suitable for integration in a layered motion planner. First we considered how to modify the motion controller, to exploit multipath fading by stopping at positions where the SNR is high. The tradeoff between communication and reference tracking can be quantified to ensure that the tracking accuracy required by the coordination layer is maintained. Second, we considered how the coordination layer can determine the velocity for each robot along its path, so that visual connectivity is maintained, despite obstacles. The result is a reference trajectory for each robot, which can be sent to the motion planner. The proposed components are tested in simulations and experiments.

Both multipath fading and shadowing can be compensated for, by increased transmission power, multi-hop relaying or hardware measures, such as diversity. This thesis aims at providing another alternative, by using the motion of the robot. This can be used as a complement or replacement to traditional methods of im-

proving communications, depending on the application requirements. One segment where this could be useful is for low-cost, power-constrained robots that want to use cheap radios transmitting at low power, but still maintain reliable high-bandwidth communications for streaming sound or images. Whenever the nominal link quality is too low, the robots could switch to communication-aware motion planning if needed.

7.1.1 Extensions

The two main themes in this thesis, exploiting multipath fading and maintaining visual connectivity, are complementary. In some situations it is possible to maintain visibility, in which case multipath fading will not be very pronounced, but in some cases there are not enough robots to do this without sacrificing mission completion. Then the robots could switch to stop-and-go motion to make the best of the multipath fading instead. We have chosen not to focus on the effects of path loss on the wireless link. The reason for this, as indicated in Chapter 2, is that the interconnection between path loss and motion planning has already been thoroughly studied in the robotics literature. We believe that, to arrive at truly applicable and versatile tools for communication-aware motion planning, the challenge of integrating all of path loss, shadowing and multipath fading largely remains, both in planners as well as in realistic simulators.

A first step towards more versatile communication-aware motion planning could be to combine the methods presented in this thesis. An interesting line of future research would be to reformulate the problem of visibility-constrained connectivity to minimizing the duration of the longest outage. During periods of outage, the robots could mitigate the multipath fading by stop-and-go motion and multi-user diversity.

7.2 Exploiting Multipath Fading

When a robot is following a reference trajectory through an environment that exhibits static multipath fading, it can improve the average quality of the wireless link to a base station by stopping at positions where the signal strength is high. To maintain reference tracking, it should only stop for a limited time, and then catch up again with the reference position. We have formulated this in three different ways, leading to three different methods for finding a communication-aware velocity controller for the robot. We have previously categorized them according to how the constraints on reference tracking are formulated, but below we also discuss how they differ in what triggers the stopping.

Channel-, Time- or State-Triggered Stopping

The proposed methods for stop-and-go motion represent a feedback approach, where the robot uses its position and measurements of the SNR to control its motion. The advantage of this is that the fading does not need to be predicted, which is difficult in general settings. Using feedback also adds robustness to errors in the models of motion and communication. The robot can compensate if it moves slower than expected, or if the statistics of the fading change over time. We now clarify how the SNR measurements control the stopping of the robot in each proposed method.

In Chapter 3, we considered hard bounds on the tracking error and assumed that the robot could sample the SNR at equidistant points. If the robot knew only the distribution of the SNR, we proposed an optimal stopping strategy, where the threshold for stopping depended on the tracking error. The more the robot was lagging behind, the higher the threshold for standing still. If the robot instead had complete knowledge of the SNR waveform, it should simply stand still at the best position that did not violate the tracking bounds. Both approaches can be described as *channel-triggered stopping*. We computed the resulting link capacity and throughput, as a function of the tracking bounds. Finally, we performed experiments in a number of locations, to test the performance of the optimal stopping strategy. The results showed that it could improve the throughput by over 100%, compared to the nominal case of driving at constant velocity. As expected, the resulting throughput improvements depend strongly on how well the link performs in the nominal case. But they also show that the approach is robust to moderate levels of motion in the environment, which violates the assumption on the fading being static.

In Chapter 4, the tracking requirements were formulated in a probabilistic manner: The robot stopped for a constant time if the SNR exceeded a given threshold, and both the stop time and threshold level were chosen to yield a specified expected velocity. To maintain reference tracking, this was embedded in an architecture with a feedback controller that controlled the position of the robot by adding a bias to the threshold. To allow for a slower channel quality sensor, the robot stopped after a given time and only sampled the SNR while standing still. If the SNR was below the stopping threshold, the robot immediately resumed driving. This was called *time-triggered stopping*. We implemented this architecture on a robot, and in cases with low channel quality, the results showed throughput improvements of 50–100% compared to the nominal case.

The last chapter on multipath fading was Chapter 5, where we formulated an optimal control problem with both throughput and reference tracking in the cost function. The robot was assumed to have a data buffer with constant inflow, where the outflow was equal to the wireless throughput. The robot motion and the buffer were modeled as a switched linear system, where the SNR varied according to a known distribution when the robot was moving and assumed a constant value of the local maximum when the robot was standing still. A controller was found using hybrid optimal control, and the resulting performance was illustrated in simulations. The robot stopped if the buffer was large compared to the tracking error, and

otherwise focused on driving to catch up with the reference. This could be described as *state-triggered stopping*.

Comparison

One of the main differences between the proposed methods was how the tradeoff between tracking and communication was controlled: In Chapter 3, we simply set the maximum tracking error and we also analyzed how it affected the resulting channel capacity or throughput. In Chapter 4, the tradeoff was controlled by the closed-loop pole a, as well as the stop time variance σ^2. Simulations illustrated the expected result that a faster pole (smaller a) yielded better reference tracking, but this meant that the tracking controller interfered more with the stop-time policy. A smaller variance meant that the robot made shorter stops, which also left less freedom to exploit good positions. Finally, in Chapter 5, the tradeoff between communication and tracking was controlled by the choice of weights, Q, in the cost function (5.1).

With regards to implementation, channel-triggered stopping required that the robot could measure the SNR when driving. As the multipath fading varies quickly with the position of the robot, this precluded averaging over several received packets if the robot was moving fast. This made the stopping decision more noise-sensitive than the time- or state-triggered stopping. In the case of full knowledge of the SNR waveform, the main difficulty of channel-triggered stopping would instead be navigation: To exploit previous measurements of the multipath fading, the robot would need to be able to reproduce its position with accuracy in the order of a fraction of a wavelength. Time-triggered stopping, on the other hand, placed less demands on the channel sensor as well as the navigation. But it would be more difficult to integrate in a layered motion-planning architecture because there were more parameters to set and no hard bounds on the resulting tracking error. This also held true for state-triggered stopping, which caused as much tracking error as needed to maintain a bounded buffer size. To implement state-triggered stopping, the system would have to be monitored against buffer overflow. If the channel conditions were too bad so the buffer could not be balanced by stop-and-go motion, the inflow would need to be reduced, by turning sensors off or reducing the sampling rate or resolution.

The general approach suggested above, to spend more time at positions where the fading is beneficial, is a form of diversity. It could be compared to other diversity schemes traditionally used to mitigate multipath fading, as described in Chapter 2. The general idea of diversity is to create multiple independent channels that can be combined to improve the link. The more channels, the better the resulting link performance. This is why our approach, which can be viewed as a type of switched antenna diversity over time, is useful as a complement to other techniques, to increase the number of independent channels. As an example, a robot could be equipped with 3 antennas for diversity, or have 3 frequencies to choose between. But if it could also choose any of 3 sampling positions, that would offer 9 independent

channels. The advantage of our approach is that by leveraging the robot mobility, it does not require any extra hardware, unlike antenna diversity. As robots get smaller, it could also be difficult to fit multiple antennas on them, with sufficient separation to get good diversity. Then using motion for diversity may be a more efficient alternative. Finally, as modern radio circuits are moving towards higher frequencies, the allowed deviation required for a given throughput will decrease, since it is proportional to the carrier wavelength.

Extensions

Chapters 3–5 focused on the channel between a single robot and its base station. But similar principles could be applied in multi-robot networks, where robots also communicate between each other. A simple extension is when robots are deployed to static locations and the communication topology is a tree. Starting from the root, the robots could then sequentially make small movements to improve the channel to their parent with respect to multipath fading. For other topologies, one could use a distributed optimization framework, as proposed by Vieira et al. (2011). If two moving robots are communicating, they could jointly decide to stop when the channel is good. In a static environment, this is equivalent to the case of a base station and a single robot. And if both robots have the same reference velocity, the tradeoff between communication and reference tracking is identical, so they can make the same stopping decisions.

The approaches above were derived under the assumption of static multipath fading, but the principles carry over to more general propagation effects as well. Making small deviations from a given reference position is relevant in any situation where the signal strength varies over short distances. As an example, the tracking error bounds could be extended to overcome shadowing. A possible application would be a robot searching office rooms, knowing that the signal strength is better in the corridor or near windows. The principle of channel-triggered optimal stopping could also be used for underwater robots, that surface regularly to try to communicate.

7.3 Maintaining Visual Connectivity

We have proposed two methods for computing the velocities for N robots, moving along given paths, such that they maintain visual connectivity and still reach their goals. In both cases, we translated the problem to motion planning in an N-dimensional configuration space. The $\mathcal{C}$-obstacles corresponded to configurations where the robots were not connected, which must be avoided. The first proposed method solved the motion planning problem using the RRT algorithm, which built a tree that filled the free space by expanding in the direction of randomly chosen sample points. When the goal configuration was reachable from the tree, the solution path could be extracted. This method could handle high-dimensional configuration spaces, but in the likely case that there was no solution, the solver ran forever. We therefore also proposed an exact method, which could detect an unsolvable problem in finite time. This method divided the free space into overlapping convex cells, each corresponding to a certain communication topology and a certain relation between all obstacles and the lines of sight between robots. It was then possible to construct a graph of adjacent cells. If there was a sequence of cells linking the cells that contained the initial and goal configurations, a path could easily be extracted. This method had higher computational complexity, so it could only handle cases with up to about six robots. It also required that the problem was constrained to only allowing parallel paths.

As described in the thesis problem formulation, the methods for maintaining visual connectivity are intended for integration with the coordination layer of a robot motion planner. This represents an open-loop control structure, using given paths and a map of the obstacles to compute reference trajectories for each robot. An important drawback of open-loop control is that it is sensitive to disturbances. If a robot is delayed, the others will continue following their trajectories, possibly losing connectivity. And the approach is not robust to errors in the map, which can also lead to loss of connectivity. To improve robustness, the proposed methods could be implemented in a closed-loop way, using a receding horizon framework. The coordination layer would then solve for partial trajectories and replan regularly, based on new information in the map and the updated robot positions. Below, we comment further on this extension.

Extensions

In contrast to most cell decomposition methods, constructing the cells is trivial in the exact method, but testing for adjacency is computationally expensive. This is why the number of obstacles, M, which determines the branching factor of the search tree, has higher impact on the complexity of the problem than the dimension N of the configuration space. This suggests decomposing large problems by only considering the next $\tilde{M} < M$ obstacles, rather than decomposing them along the y-dimension by dividing the robots into subgroups. Formulating this as a receding horizon problem and studying its completeness properties remains an open problem

for future research.

The RRT framework is very flexible, in that it is straightforward to introduce additional constraints. One such constraint would arise if intersecting paths were allowed. The outage detector could then be augmented so it returns the configuration x that is as close to y as possible without going into outage *or* having an inter-robot collision. Algorithm 6.1 would not change. Similarly, it would be possible to enforce constraints on the network topology. As an example, the outage detector could be reconfigured to only consider two robots as connected if the path between them in the communication graph has a given maximum number of hops.

Another example of the RRT flexibility is that one can exchange the local solver in the outage detector. In our proposed outage detector, we have chosen to use a very simple straight-line local solver to find paths between two candidate points in $\mathcal{C}$. It is important to note that this is not the only possible choice. A potential-based method, where we allow local minima, could possibly yield better results, at the expense of slower queries to the outage detector. The tradeoff between efficiency of the exploration and the computational complexity of the outage detector (more generally called collision detector) is discussed by Geraerts (2006).

We finally note that the formulated problem constitutes half of the solution to the full visibility-constrained path planning problem. Before choosing velocities, we would need to find collision-free paths. The advantage of the proposed exact solver would be more apparent in this setting, since it would need to signal if the paths do not allow maintained connectivity, so they can be replanned.

Bibliography

V. A. Aalo. Performance of maximal-ratio diversity systems in a correlated Nakagami-fading environment. *IEEE Transactions on Communications*, 43(8): 2360–2369 (1995).

G. C. Alexandropoulos, P. T. Mathiopoulos, and N. C. Sagias. Switch-and-examine diversity over arbitrarily correlated Nakagami-m fading channels. *IEEE Transactions on Vehicular Technology*, 59(4): 2080–2087 (2010).

G. C. Alexandropoulos, N. C. Sagias, F. I. Lazarakis, and K. Berberidis. New results for the multivariate Nakagami-m fading model with arbitrary correlation matrix and applications. *IEEE Transactions on Wireless Communications*, 8(1): 245–255 (2009).

D. Anisi, P. Ögren, and X. Hu. Cooperative minimum time surveillance with multiple ground vehicles. *IEEE Transactions on Automatic Control*, 55(12): 2679–2691 (2010).

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

J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6): 628–649 (1991).

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

F. Chaumette and S. Hutchinson. Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4): 82–90 (2006).

Y. S. Chow. *Great expectations: The theory of optimal stopping*. Houghton Mifflin (1971).

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

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

N. Correll, J. Bachrach, D. Vickery, and D. Rus. Ad-hoc wireless network coverage with networked robots that cannot localize. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Kobe, Japan (2009).

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).

B. d'Andréa Novel, G. Campion, and G. Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *International Journal of Robotics Research*, 14(6): 543–559 (1995).

C. Dixon and E. W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. *Mobile Networks and Applications*, 14(3): 281–291 (2009).

M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2(1–4): 477–521 (1987).

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

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

P. Fabiani, P. Gonzalez-Banos, J. Latombe, and D. Lin. Tracking a partially predictable target with uncertainties and visibility constraints. *Journal of Robotics and Autonomous Systems*, 38(1): 31–48 (2002).

T. Fugen, J. Maurer, T. Kayser, and W. Wiesbeck. Capability of 3-D ray tracing for defining parameter sets for the specification of future mobile communications systems. *IEEE Transactions on Antennas and Propagation*, 54(11): 3125–3137 (2006).

A. Ganguli, J. Cortés, and F. Bullo. Visibility-based multi-agent deployment in orthogonal environments. In *Proceedings of the American Control Conference*. New York City, USA (2007).

A. Ganguli, J. Cortés, and F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, 25(2): 340–352 (2009).

R. Geraerts. *Sampling-based Motion Planning: Analysis and Path Quality*. Ph.D. thesis, Utrecht University (2006).

A. Ghaffarkhah and Y. Mostofi. Communication-aware motion planning in mobile networks. *IEEE Transactions on Automatic Control*, 56(10): 2478–2485 (2011).

A. Goldsmith. *Wireless Communications*. Cambridge University Press (2005).

G. Hollinger and S. Singh. Multi-robot coordination with periodic connectivity. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Anchorage, USA (2010).

J. Hopcroft, J. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". *The International Journal of Robotics Research*, 3(4): 76–88 (1984).

M. A. Hsieh, A. Cowley, R. V. Kumar, and C. J. Taylor. Maintaining network connectivity and performance in robot teams. *Journal of Field Robotics*, 25(1-2): 111–131 (2008).

W. C. Jakes, editor. *Microwave Mobile Communications*. IEEE Press (1974).

M. Ji and M. Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4): 693–703 (2007).

E. P. C. Jones, L. Li, J. Schmidtke, and P. A. S. Ward. Practical routing in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 6(8): 943–959 (2007).

K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research*, 5(3): 72–89 (1986).

G. K. Karagiannidis, D. A. Zogas, and S. A. Kotsopoulos. An efficient approach to multivariate Nakagami-m distribution using Green's matrix approximation. *IEEE Transactions on Wireless Communications*, 2(5): 883–889 (2003).

J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers (1991).

S. M. LaValle. *Planning Algorithms*. Cambridge University Press (2006). URL `http://planning.cs.uiuc.edu/`.

S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6): 912–925 (1998).

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

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

M. Lindhé, T. Keviczky, and K. H. Johansson. Multi-robot path following with visual connectivity. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers*. Pacific Grove, USA (2011).

S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42: 661–668 (2006).

F. J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253): 68–78 (1951).

M. McClure, D. R. Corbett, and D. W. Gage. The DARPA LANdroids program. In *Proceedings of the SPIE Unmanned Systems Technology Conference XI*. Orlando, USA (2009).

A. F. Molisch. *Wireless Communications*. John Wiley and Sons Ltd. (2005).

L. Moser. On a problem of Cayley. *Scripta Mathematica*, 22: 289–292 (1956).

Y. Mostofi. Decentralized communication-aware motion planning in mobile networks: An information-gain approach. *Journal of Intelligent and Robotic Systems*, 56(2): 233–256 (2009).

Moteiv Corporation. Tmote Sky datasheet 1.02 (2006). URL `http://www.moteiv.com`. Last visited 2007-05-19.

T. Muharemovic, A. Sabharwal, and B. Aazhang. Antenna packing in low-power systems: Communication limits and array design. *IEEE Transactions on Information Theory*, 54(1): 429–440 (2008).

R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *International Journal of Robotics Research*, 26(3): 233–253 (2007).

U. Nilsson, P. Ögren, and J. Thunberg. Optimal positioning of surveillance UGVs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice, France (2008).

P. O'Donnell and T. Lozano-Perez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Scottsdale, USA (1989).

P. Ögren, D. Anisi, D. Berglund, D. Dimarogonas, H. Gustavsson, L. Hedlin, L. Hedström, X. Hu, K. H. Johansson, F. Katsilieris, V. Kaznov, P. Lif, M. Lindhé, U. Nilsson, M. Persson, M. Seeman, P. Svenmarck, and J. Thunberg. Results from the project AURES: Autonomous UGV-system for reconnaissance and surveillance. Technical Report FOI-R–2783–SE, Swedish Defence Research Agency (2009). ISSN-1650-1942.

R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3): 401–420 (2006).

L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi. Decentralized cooperative policy for conflict resolution in multi-vehicle systems. *IEEE Transactions on Robotics*, 23(6): 1170–1183 (2007).

S.-M. Park, J.-H. Lee, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. In *Lecture Notes in Computer Science*, volume 2076 (2001).

J. G. Proakis and M. Salehi. *Communication Systems Engineering*. Prentice-Hall, 2nd edition (2002).

D. Puccinelli, M. Brennan, and M. Haenggi. Reactive sink mobility in wireless sensor networks. In *First ACM Workshop on Mobile Opportunistic Networking*. Puerto Rico (2007).

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).

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

G. Roussos, D. V. Dimarogonas, and K. J. Kyriakopoulos. 3D navigation and collision avoidance for nonholonomic aircraft-like vehicles. *International Journal of Adaptive Control and Signal Processing*, 24(10): 900–920 (2010).

J. T. Schwartz and M. Sharir. On the piano mover's problem: III. coordinating the motion of several independent bodies. *The International Journal of Robotics Research*, 2(3): 46–75 (1983).

T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9): 1384–1399 (1992).

M. K. Simon and M.-S. Alouini. A simple integral representation of the bivariate Rayleigh distribution. *IEEE Communications letters*, 2(5): 128–130 (1998).

T. Siméon, S. Leroy, and J.-P. Laumond. Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics*, 18(1): 42–49 (2002).

J. Smith, M. Olivieri, A. Lackpour, and N. Hinnerschitz. RF-mobility gain: concept, measurement campaign, and exploitation. *IEEE Wireless Communications*, 16(1): 38–44 (2009).

P. J. Smith, P. A. Dmochowski, M. Chiani, and A. Giorgetti. On the number of independent channels in a diversity system. In *Proceedings of the IEEE Wireless Communications and Networking Conference*. Sydney, Australia (2010).

D. P. Spanos and R. M. Murray. Robust connectivity of networked vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*. New Orleans, USA (2004).

M. Stachura and E. W. Frew. Cooperative target localization with a communication-aware unmanned aircraft system. *AIAA Journal of Guidance, Control, and Dynamics*, 34(5): 1352–1362 (2011).

G. L. Stüber. *Principles of mobile communication*. Kluwer academic publishers (1996).

E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Pasadena, USA (2008).

E. A. Stump and B. M. Sadler. Persistent surveillance using mutually-visible robotic formations. *Proceedings of the SPIE*, 7694 (2010).

P. Svestka and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Nagoya, Japan (1995).

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

O. Tekdas, P. A. Plonski, N. Karnad, and V. Isler. Maintaining connectivity in environments with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Anchorage, USA (2010*a*).

O. Tekdas, W. Yang, and V. Isler. Robotic routers: Algorithms and implementation. *International Journal of Robotics Research*, 29(1): 110–126 (2010*b*).

C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4): 509–521 (1998).

B. Tovar, L. Guilamo, and S. M. LaValle. Gap navigation trees: Minimal representation for visibility-based tasks. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, volume 17 (2004).

B. Tovar and S. M. LaValle. Visibility-based pursuit-evasion with bounded speed. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*. New York City, USA (2006).

P. Varaiya. A question about hierarchical systems. In T. Djaferis and I. Schick, editors, *System Theory: Modeling, Analysis and Control*. Kluwer (2000).

M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice-Hall (1993).

M. A. M. Vieira, M. E. Taylor, P. Tandon, M. Jain, R. Govindan, G. S. Sukhatme, and M. Tambe. Mitigating multi-path fading in a mobile mesh network. *Ad Hoc Networks* (2011).

P. Viswanath, D. N. C. Tse, and R. Laroia. Opportunistic beamforming using dumb antennas. *IEEE Transactions on Information Theory*, 48(6): 1277–1294 (2002).

C. W. Warren. Multiple robot path coordination using artificial potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Cincinnati, USA (1990).

M. M. Zavlanos, H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Hybrid control for connectivity preserving flocking. *IEEE Transactions on Automatic Control*, 54(12): 2869–2875 (2009).

Q. T. Zhang and H. G. Lu. A general analytical approach to multi-branch selection combining over various spatially correlated fading channels. *IEEE Transactions on Communications*, 50(7): 1066–1073 (2002).

Y. Zhang and R. Vaughan. Ganging up: Team-based aggression expands the population/performance envelope in a multi-robot system. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Orlando, USA (2006).

Y. R. Zheng and C. Xiao. Simulation models with correct statistical properties for Rayleigh fading channels. *IEEE Transactions on Communications*, 51(6): 920–928 (2003).

"*The problem with quotes from the Internet is that the sources are difficult to verify.*"

Abraham Lincoln