Diss. ETH No. 26090

SCALABLE LOCALIZATION AND COORDINATION OF ROBOT SWARMS

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Michael David Hamer
MSc ETH in Robotics, Systems and Control

born on 26 October 1987
citizen of Australia

accepted on the recommendation of

Prof. Dr. Raffaello D'Andrea, examiner
Prof. Dr. Roland Siegwart, co-examiner

2019

Scalable Localization and Coordination of Robot Swarms

Michael David Hamer

Institute for Dynamic Systems and Control
ETH Zurich
2019

Abstract

As the capabilities of robots and their control systems improve, applications involving the use of large robot swarms in semi-structured environments become increasingly viable. Such applications include the progressive automation of warehouses, factories, mine sites and hospitals. Despite differences in context and application, these problems all require accurate localization and timely coordination of large fleets of robots.

In outdoor applications, satellite-based localization (e.g. GPS) is a core technology driving the development of autonomous vehicles and facilitating the progressive robotization of industries such as agriculture, mining, inspection and freight. Satellite-based localization enables such applications by providing robots with the ability to quickly and independently measure their absolute position. In indoor environments, satellite-based localization is unavailable, making absolute positioning in such environments challenging.

The first contribution of this thesis is the development of a scalable, "indoor GPS"-like system using ultra-wideband radio technology. The topology of this system is similar to that of GPS: fixed-position radio modules installed in the environment regularly transmit radio signals, fulfilling a similar role to that of GPS satellites; while mobile robots move within the coverage area and localize themselves based on the received signals. Much like GPS, the system therefore scales to support an unlimited number of robots. Theoretical developments presented in this thesis are supported by experimental results, including a demonstration of the system's functionality, in which multiple nano-quadcopters are flown simultaneously within a space.

Generating collision-free trajectories for large swarms of robots operating in close proximity is a similarly challenging problem, since robot trajectories are coupled through collision avoidance constraints, making the problem computationally expensive and time consuming to solve using classical optimization techniques. The second contribution of this thesis is a method to quickly generate such trajectories by leveraging the parallel-computation architecture of modern graphics processing units. The effectiveness and scalability of this method is demonstrated in two simulation-based case studies: a benchmark problem requiring a swarm of 200 quadcopters to traverse a maze; and an example in which a fleet of 100 robots with bicycle dynamics must change their formation. In both cases, the method easily handles nonlinear dynamics and constraints, and generates feasible, collision-free trajectories for the entire swarm in a matter of seconds.

The developments and contributions presented in this thesis provide a pathway towards the application of these technologies to the localization and coordination of large robot swarms in indoor environments.

Zusammenfassung

Die Weiterentwicklung von Robotern und ihren Steuerungssystemen ermöglicht die Anwendung großer Roboterschwärme in semi-strukturierten Umgebungen. Zu diesen Anwendungen gehört die fortschreitende Automatisierung von Lagerhallen, Fabriken, Bergwerken und Krankenhäusern. Trotz der unterschiedlichen Einsatzgebieten erfordern all diese Anwendungen genaue Lokalisierung und Koordination großer Roboterflotten.

In Außenanwendungen ist die satellitengestützte Lokalisierung (z.B. GPS) eine Kerntechnologie, welche die Entwicklung autonomer Fahrzeuge vorantreibt und die fortschreitende Robotisierung von Industrien wie Landwirtschaft, Bergbau, Inspektion und Fracht ermöglicht. Solche Anwendungen werden durch die Fähigkeit der Roboter ermöglicht, ihre absolute Positionen schnell und unabhängig voneinander zu messen. In Innenräumen ist eine satellitengestützte Ortung nicht möglich, was eine absolute Positionierung in solchen Situationen erschwert.

Der erste Beitrag dieser Arbeit ist die Entwicklung eines, "indoor GPS"-ähnlichen Systems mit Hilfe der Ultrabreitband-Funktechnologie. Die Funktionsweise dieses Systems ähnelt jener eines GPS: In der Umgebung installierte Funkmodule mit fester Position senden regelmäßig Funksignale und erfüllen eine ähnliche Rolle wie GPS-Satelliten; Roboter können sich innerhalb des Versorgungsgebiets bewegen und sich anhand der empfangenen Signale lokalisieren. Ähnlich wie bei einem GPS kann das System eine unbegrenzte Anzahl von Robotern unterstützen. Theoretische Resultate, die in dieser Arbeit vorgestellt werden, werden durch experimentelle Ergebnisse unterstützt, einschließlich einer Demonstration der Funktionalität des Systems, bei der mehrere Nanoquadrokopter gleichzeitig in einem Raum geflogen werden.

Die Planung kollisionsfreier Trajektorien für große Roboterschwärme ist ebenfalls ein herausforderndes Problem. Um Kollisionen zu vermeiden, müssen die Trajektorien gemeinsam optimiert werden. Dies ist unter der Verwendung klassischer Optimierungstechniken rechnerisch aufwendig und zeitintensiv. Der zweite Beitrag dieser Arbeit ist eine Methode zur schnellen Planung solcher Trajektorien durch die Nutzung der Parallelberechnungsarchitektur moderner Grafikprozessoren. Die Effizienz und Skalierbarkeit dieser Methode wird in zwei simulationsbasierten Fallstudien demonstriert: ein Benchmark-Problem, bei dem ein Schwarm von 200 Quadrokopter ein Labyrinth durchqueren muss; und ein Beispiel, bei dem eine Flotte von 100 Fahrradrobotern ihre Formation ändern muss. In beiden Fällen geht die Methode problemlos mit nichtlinearer Dynamik und Bedingungen um und ist fähig in wenigen Sekunden realisierbare, kollisionsfreie Trajektorien für den gesamten Schwarm zu planen.

Die Arbeit zeigt Lösungen auf, wie Roboterschwärme in Innenräumen lokalisiert und koordiniert werden können.

Acknowledgements

This thesis would not have been possible without the support and contributions of many people.

Firstly, I would like to thank Professor Raffaello D'Andrea (Raff) for putting his trust in me by accepting me firstly as an intern, then as a master's student, and finally as a doctoral candidate. During my time at the Institute for Dynamic Systems and Control, Raff has always given me the opportunity and freedom to explore my own interests and ideas, and in doing so I have learned so much more than I would have, had I followed a prescribed path. Secondly, I would like to thank Professor Roland Siegwart for agreeing to act as co-examiner and for his willingness to spend time reviewing this thesis.

During my time at the Institute for Dynamic Systems and Control, I have had the pleasure to work with many friendly, helpful and extremely intelligent colleagues, to whom I am grateful for creating and maintaining such a unique research environment: Angela, Sebastian, Ray, Sergei, Markus H., Nico, Gajan, Philipp, Markus W., Mark, Federico, Max, Robin, Michi, Dario, Tony, Weixuan, Rajan, Matthias, and Carlo. I would like to give particular thanks to Tony for the many engaging discussions regarding ultra-wideband radio, for helping me troubleshoot issues with the radio modules, and for always being there when I needed input or a second eye on something.

The Institute for Dynamic Systems and Control supports its doctoral candidates with excellent support staff. I would like to express my gratitude to Michi, Mac, Gregor, Timon, Clara and Elias for their technical support in realizing my various ideas, and to Helen and Katharina for their administrative support. A mention goes also to all the students and interns who have supported me with their ideas and projects over the years.

Finally, I wish to give my profound thanks to my parents Megan and Peter, whose encouraging upbringing sent me to Europe on the quest for knowledge and adventure; to my younger brother Jason, who set me a target and beat me to it; to my grandparents Audrey and David and my uncle Andrew, whose stories of adventure never failed to inspire; to Nora, who has supported me through every step of the journey; and to all my friends, both near and far.

Contents

Foreword

This thesis documents the research carried out by the author during his doctoral studies under the supervision of Professor Raffaello D'Andrea at the Institute for Dynamic Systems and Control at ETH Zurich between October 2013 and February 2019.

This thesis is divided into two stand-alone parts, focused on the localization (Part A) and the coordination of robot swarms (Part B). This thesis presents a mix of theoretical and practical results that progress the state-of-the-art research in these two fields. The work presented in this thesis is based on material published by the author in two peer-reviewed journal papers, three peer-reviewed conference papers, co-authored in two additional peer-reviewed publications, and investigated through 13 student projects supervised during his doctoral studies.

These two thesis parts are put into context by two introductory chapters, Chapter I, which introduces and motivates this work; and Chapter II, which outlines the key contributions made by the author during his doctoral studies. Being stand-alone, each thesis part provides a more thorough introduction to the topic and a review of related research, and is concluded by a technical discussion of open questions and future research directions. The thesis is concluded by Chapter III, which provides an overview of the major thesis contributions, and a general outlook on future research.

I

Introduction

Improvements in the capabilities of robots and their control systems have allowed robots to operate in increasingly cluttered environments and in increasingly close proximity to each other. Recent examples of such applications include the automation of warehouse logistics using fleets of mobile robots [1]; the use of service robots for goods transport in hospitals [2], [3]; the robotization of cleaning tasks (e.g. in households, supermarkets); and the progressive automation of mining sites and agriculture. Despite differences in context and scale, many of these examples share a requirement for accurate localization in semi-structured environments, and rely on the coordination of fleets of robots to achieve a higher-level goal.

This thesis addresses these requirements in two related, yet stand-alone parts. Part A of the thesis presents the development of an ultra-wideband (UWB) radio localization system, which allows a theoretically unlimited number of robots to localize within an environment that has been outfitted with a small number of fixed-position modules. Part B of the thesis addresses computational challenges in generating trajectories for robot swarms and proposes a method of solving such problems using parallel computation, which can be scaled to swarms of hundreds to thousands of robots. Each thesis part begins with a technical introduction to the topic and a review of related research, and is concluded by an outlook on future research directions. A general overview of each thesis part follows.

Localization of Robot Swarms

Satellite-based localization systems (e.g. GPS) have revolutionized modern industry and have become ubiquitous in our personal lives. This technology is a core enabler of autonomous vehicles (e.g. cars, boats, planes), and is instrumental to the increasing robotization and automation of the agriculture, mining, freight and transportation industries, to name just a few. The success of GPS technology is largely due to its scalability: an unlimited number of devices can receive GPS signals and each device can use these signals to compute its position relative to the satellite coordinate system. Since it is a satellite-based localization system, the application of GPS technology is limited to outdoor applications in which signals sent by satellites travel in a direct path to each receiving device.

Part A of the thesis is based on the author's publications [1], [5], [6] (see Chapter II) and presents the development of a localization system for indoor application, which shares many similarities with GPS. The proposed system is based on UWB radio technology, which has been used in communication and radar applications since the 1960's [4] due to its ability to accurately timestamp the transmission and reception of radio pulses. The use of UWB as a means of localization is a much newer field, driven by its legalization [5] and standardization [6], [7] in the early 2000s, and by the advent of small, low-power UWB radio transceivers such as the Decawave DW1000 [8] used in this thesis.

In the proposed system, an indoor environment is outfitted with a number of stationary radio transceivers, which play a similar role to the satellites in GPS. Devices (in the case of this thesis, robots) within the coverage area can localize themselves based only on received signals. Much like GPS, this allows the system to support an unlimited number of robots. Furthermore, the stationary radio transceivers require no physical connection for synchronization and initially no knowledge of their position, since both are measured and refined during operation. This makes the proposed system easy to deploy, maintain and extend.

Furthermore, since each robot is able to directly measure its position within the environment, sensors mounted on the robot such as inertial measurement units, wheel encoders, or cameras, can be used to improve the robot's state estimate and increase its awareness of the local environment. This is akin to the approach taken by autonomous vehicles (e.g. self-driving cars), in which GPS position measurements are augmented with cameras, LIDARs, wheel encoders and other sensory capabilities to allow the vehicle to navigate in its local environment. Much like GPS, the localization technology proposed in Part A is therefore viewed as complementary to other technologies, and allows an unlimited number of robots to localize themselves within a semi-structured environment at low cost and with low hardware and computational complexity.

Part A begins with a more thorough overview of UWB technology (Chapters 1 and 2) and a review of relevant literature (Chapter 3), before presenting a systems-based approach to the development of the UWB localization system (Chapters 4–6) and associated robot state estimation framework (Chapter 7). An alternative method of coordinating transmissions is presented in Chapter 8, which allows the system to scale to a large number of stationary transceivers, while maintaining its scalability to an unlimited number of robots. Results presented in Part A are interspersed with the presentation of pertinent experimental results, with the functionality of the system demonstrated by multiple nano-quadcopters localizing and flying simultaneously within a space. Part A is concluded by a discussion of future research directions in Chapter 9.

Coordination of Robot Swarms

Having developed a system that enables robot swarms to localize within a semi-structured environment, the focus in Part B of the thesis is turned to the coordination of such swarms through the time-efficient generation of feasible, collision-free trajectories that transition a swarm of robots from an initial state to a defined goal state. Such problems have historically proven difficult to solve in a time-efficient manner, since the requirement for collision avoidance introduces a non-convex coupling between robot trajectories and requires trajectories to be optimized jointly. Classic optimization approaches such as mixed integer linear optimization (e.g. [9]) or sequential convex optimization (e.g. [10]) scale poorly to such high-dimensional problems.

Part B of the thesis is based on [2] and presents a novel method to quickly generate collision-free trajectories for swarms of hundreds of robots through cluttered environments. As introduced in Part B, Chapter 3, the proposed method leverages existing trajectory generation methods to initialize individual robot trajectories without considering collisions, and then uses back-propagation and gradient descent to optimize these trajectories until feasibility, when collision constraints are considered. This formulation of the problem can be efficiently parallelized and can thus be solved quickly on a tensor or graphics processing unit (GPU). Furthermore, the method is straightforward to express using tensor computation frameworks and can leverage advances in computation and computational methods, driven by the machine-learning community.

The effectiveness of this method is demonstrated in two simulation-based case studies, presented in Chapter 4: a benchmark problem involving a swarm of 200 quadcopters traversing a maze, and a fleet of 100 bicycle robots (a common benchmark for non-holonomic systems) changing their formation. In both cases, the method requires seconds to generate feasible, collision-free trajectories for the entire swarm. These case studies are used to demonstrate the method's application to nonlinear systems; to discuss the implementation of various state and input constraints; and to address various performance caveats and methods of increasing the method's convergence speed. The source code for these case studies has been published in [11]. Part B concludes with discussion of future research directions in Chapter 5.

Bibliography: Introduction

[1] D'Andrea, R., "Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead", *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012.

[2] Ozkil, A. G., Fan, Z., Dawids, S., Aanes, H., Kristensen, J. K., Christensen, K. H., "Service robots for hospitals: A case study of transportation tasks in a

hospital", in *2009 IEEE International Conference on Automation and Logistics*, IEEE, 2009, pp. 289–294.

[3] Bačík, J., Ďurovský, F., Biroš, M., Kyslan, K., Perduková, D., Padmanaban, S., "Pathfinder–development of automated guided vehicle for hospital logistics", *IEEE Access*, vol. 5, pp. 26 892–26 900, 2017.

[4] Barrett, T. W., "History of ultrawideband (UWB) radar & communications: Pioneers and innovators", in *Proc. Progress in Electromagnetics Symposium*, 2000, pp. 1–42.

[5] Federal Communications Commission, *First Report and Order 02-48*, Apr. 2002.

[6] IEEE Standard for information Technology, *IEEE Std 802.15.4a*, 2007.

[7] Sahinoglu, Z., Gezici, S., "Ranging in the IEEE 802.15. 4a standard", in *Wireless and Microwave Technology Conference*, 2006, pp. 1–5.

[8] Decawave, (2019). DW1000 UWB radio IC: User manual, [Online]. Available: `https://www.decawave.com/wp-content/uploads/2019/04/DW1000-User-Manual.pdf` (visited on Apr. 1, 2019).

[9] Earl, M. G., D'Andrea, R., "Iterative MILP methods for vehicle-control problems", *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[10] Augugliaro, F., Schoellig, A. P., D'Andrea, R., "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1917–1922.

[11] **Hamer, M.** (2019). Fast generation of collision-free trajectories for robot swarms using GPU acceleration: Software release. `http://mikehamer.info/swarm-trajectories` and `https://github.com/mikehamer/swarm-trajectories`.

II

Contributions

The scientific contributions of this thesis can be broadly classified into

- the localization of a swarm of robots using ultra-wideband radio, discussed in Part A of this thesis; and
- the coordination of a swarm of robots through the fast generation of collision-free trajectories, discussed in Part B of this thesis.

The major results of this thesis were published in two journal papers [1], [2] and two conference papers [5], [6]. In addition, the theoretical contributions and infrastructure resulting from this doctoral work directly contributed to two related journal papers [3], [R.2], two related conference papers [7], [R.4], and directly supported the two master's theses, five semester projects, two bachelor's theses, and four student internships that I supervised during my doctoral studies.

Publications on which this thesis is based

Journal publications

[P.1] **Hamer, M.**, D'Andrea, R., "Self-calibrating ultra-wideband network supporting multi-robot localization", *IEEE Access*, vol. 6, pp. 22 292–22 304, 2018.

[P.2] **Hamer, M.**, Widmer, L., D'Andrea, R., "Fast generation of collision-free trajectories for robot swarms using GPU acceleration", *IEEE Access*, vol. 7, pp. 6679–6690, 2019.

Conference publications (peer reviewed)

[P.3] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "A robot self-localization system using one-way ultra-wideband communication", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 3131–3137.

[P.4] Mueller, M. W., **Hamer, M.**, D'Andrea, R., "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1730–1736.

- [P.3] resulted from the master's thesis [MT.1], which was directly supervised by the author

- Equal contributions were made to [P.4], which combined two distinct research fields

Publications related to and supported by the results of this thesis

Journal publications

[R.1] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Angle of arrival estimation based on channel impulse response measurements", *(in preparation)*, 2019.

[R.2] Ledergerber, A., D'Andrea, R., "Calibrating away inaccuracies in ultra wideband range measurements: A maximum likelihood approach", *IEEE Access*, vol. 6, pp. 78 719–78 730, 2018.

Conference publications (peer reviewed)

[R.3] Hoeller, D., Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Augmenting ultra-wideband localization with computer vision for accurate flight", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 734–12 740, 2017.

[R.4] Ledergerber, A., D'Andrea, R., "Ultra-wideband range measurement model with gaussian processes", in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2017, pp. 1929–1934.

Additional publications

Journal publications

[A.1] Augugliaro, F., Lupashin, S., **Hamer, M.**, Male, C., Hehn, M., Mueller, M. W., Willmann, J. S., Gramazio, F., Kohler, M., D'Andrea, R., "The flight assembled architecture installation: Cooperative construction with flying machines", *IEEE Control Systems*, vol. 34, no. 4, pp. 46–64, 2014.

Conference publications (peer reviewed)

[A.2] **Hamer, M.**, Waibel, M., D'Andrea, R., "Knowledge transfer for high-performance quadrocopter maneuvers", in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 1714–1719.

Other publications (not peer reviewed)

[A.3] Förster, J., **Hamer, M.**, D'Andrea, R., "System identification of the Crazyflie 2.0 nano quadrocopter", Bachelor's thesis, ETH Zurich, 2015.

Student supervision

Masters thesis

The master's thesis is a six-month, full-time project.

[MT.1] Ledergerber, A., "A robot self-localization system using one-way ultra-wideband communication", Master's thesis, ETH Zurich, 2014.

[MT.2] Jia, Z., "Distributed control and coordination of quadcopter swarms", Master's thesis, ETH Zurich, 2017.

Semester project

A semester-long, part-time project conducted during the master's studies

[SP.1] Zahner, M., "Application of machine learning to quadrocopter slalom flying", Semester project, ETH Zurich, 2013.

[SP.2] Grob, M., Stadler, M., "Crazyflie onboard estimation & control with single point localization", Semester project, ETH Zurich, 2016.

[SP.3] Stadler, M., Grob, M., "Crazyflie slalom flight using single point localization", Semester project, ETH Zurich, 2016.

[SP.4] Ma, Y., "Learning local policies from global solutions", Semester project, ETH Zurich, 2018.

[SP.5] Widmer, L., "Fast computation of swarm transition trajectories", Semester project, ETH Zurich, 2018.

Bachelors thesis

The bachelor's thesis is a three-month, full-time project.

[BT.1] Estandia, A., "Mechanical design of a quadrocopter D.I.Y. kit", Bachelor's thesis, ETH Zurich, 2015.

[BT.2] Förster, J., "System identification of the crazyflie nano-quadcopter", Bachelor's thesis, ETH Zurich, 2015.

Internship

Internships are practical projects lasting four to six months

[IN.1] Sun, J., "Ultra-wideband API development", Internship, ETH Zurich, 2014.

[IN.2] Alawieh, A., "Modelling systematic ultra-wideband ranging errors", Internship, ETH Zurich, 2014.

[IN.3] Lee, S. K., "Position control of a crazyflie nano-quadcopter", Internship, ETH Zurich, 2015.

[IN.4] Buono, A., "Improving nano-quadcopter motor control using IR reflectance sensors", Internship, ETH Zurich, 2017.

Open-source involvement

Publications

Full source code for the two case studies presented in Part B of the thesis has been made available at `https://github.com/mikehamer/swarm-trajectories/`.

Contributions

During the course of the author's doctoral research, numerous major contributions were made to the Crazyflie open-source nano-quadcopter firmware, which is based on the FreeRTOS real-time operating system. These contributions facilitated the development of the Crazyflie from a hobbyist nano-quadcopter, into the widely-used research platform that it is today. These contributions were committed via pull-request to the main Crazyflie repository at `https://github.com/bitcraze/crazyflie-firmware`, and are listed below in chronological order with a reference to the relevant commit hash and a brief description of the contribution.

Commit	Contribution
`[663e7f8]`	**Critical Bug Fix:** I2C contention & timeouts cause the system to hang when under load.
`[f364475]`	**Critical Bug Fix:** Syslink unhandled interrupt causes the system to hang when under load.
`[1bc21ff]`	**Critical Bug Fix:** Radio link drops packets at high rates.
`[cb07ba6]`	**Critical Bug Fix:** Radio link crash when missing communications link.
`[93203d8]`	**Critical Bug Fix:** I2C incorrect delays & I2C underflow cause the system to hang when under load.
`[c2c5137]`	**Enhancement:** Improved EXTI handling to support more complicated expansion boards (such as the UWB board).
`[882bc96]`	**Critical Bug Fix:** Incorrect semaphore logic in Syslink results in a race condition with interrupts on a warm restart of the system, which can result in dereferencing a null pointer.
`[c8a372f]`	**Enhancement:** Convert expansion board SPI driver to use DMA. Required for UWB expansion board.
`[97a54d2]`	**Enhancement:** Rewrite IMU driver to use interrupts rather than polling.
`[fc88c6a]`	**Major Contribution:** Kalman filter, as described in Part A, Chapter 7. The linked contribution is the first in a series that add onboard state estimation to the Crazyflie platform. The contributed Kalman filter is now a core part of the firmware and is fundamental to all recent hardware and software developments on the Crazyflie.

Part A
Localization of Robot Swarms

This thesis part is based on material published in

[P.1] **Hamer, M.**, D'Andrea, R., "Self-calibrating ultra-wideband network supporting multi-robot localization", *IEEE Access*, vol. 6, pp. 22 292–22 304, 2018

[P.3] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "A robot self-localization system using one-way ultra-wideband communication", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 3131–3137

[P.4] Mueller, M. W., **Hamer, M.**, D'Andrea, R., "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1730–1736

and supported the publications

[R.1] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Angle of arrival estimation based on channel impulse response measurements", *(in preparation)*, 2019

[R.2] Ledergerber, A., D'Andrea, R., "Calibrating away inaccuracies in ultra wideband range measurements: A maximum likelihood approach", *IEEE Access*, vol. 6, pp. 78 719–78 730, 2018

[R.3] Hoeller, D., Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Augmenting ultra-wideband localization with computer vision for accurate flight", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 734–12 740, 2017

[R.4] Ledergerber, A., D'Andrea, R., "Ultra-wideband range measurement model with gaussian processes", in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2017, pp. 1929–1934

1

Overview

The first part of this thesis presents the development of a localization system that permits an unlimited number of robots to localize themselves simultaneously within a given area that has been outfitted with a network of stationary ultra-wideband radio (UWB) modules, herein known as anchors. This part begins with an introduction to UWB technology in Chapter 2, within which the focus is primarily on the aspects of the technology that make it suitable for radio-based localization. This chapter is largely based on the author's research, development, understanding and thought processes, and covers what is now considered to be background knowledge to the understanding of research in this field.

After establishing an understanding of UWB technology and its application to localization systems, a review of the current state of research is presented in Chapter 3, within which a particular focus is placed on research and developments related and relevant to this thesis.

Chapters 4–8 contain the novel and unique contributions of this thesis. In Chapter 4, a systems-based approach is used to model the inter-module clock behavior and to then derive a clock synchronization scheme for pairs of UWB modules.

Chapter 5 shows how pairwise clock synchronization of UWB modules can be combined to allow a network of UWB anchors to wirelessly synchronize to a network logical clock and to coordinate their transmissions according to a time-division (TDMA) schedule.

Chapter 6 shows how UWB anchors can measure distances between each other and use these (biased and noisy) distance measurements to initialize their positions within a pre-defined coordinate system. It is then demonstrated how each anchor can improve its position estimate using distributed gradient descent on global positioning error. The ability of the UWB anchor network to self-localize enables it to be set up quickly and dynamically, since anchor positions are computed as a by-product of operation and do not need to be manually measured or otherwise known beforehand.

Having shown how a network of UWB anchors can synchronize and construct a coordinate system, Chapter 7 demonstrates how robots can use measurements of packet reception times to localize themselves within the network's coordinate system. Since robots

are passive receivers in this system and are able to compute their position based only on received and local information, a theoretically unlimited number of robots can operate simultaneously and without the need for central coordination or centralized localization infrastructure.

The system presented in Chapters 4–7 relies on anchors coordinating their transmissions according to a predefined TDMA schedule. Chapter 8 presents initial results and discussion as to how similar results can be achieved by anchors randomly transmitting messages, a method that significantly reduces software complexity and allows trivial scaling to large, partially-connected networks.

This part of the thesis is concluded in Chapter 9, within which the current limitations of the system are discussed and directions for future research are proposed.

All chapters are interspersed with the presentation of pertinent experimental results based on a network of eight UWB anchors (with hardware detailed in Chapter 2), supporting the operation of a fleet of Crazyflie 2.0 nano-quadcopters [1] (with hardware detailed in Chapter 7). This setup is depicted in Fig. 1.1.

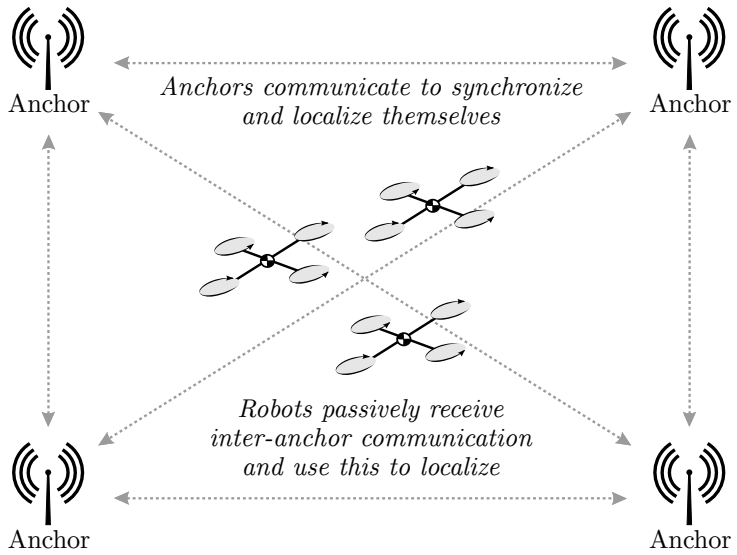


Figure 1.1: This figure shows the problem addressed in this part of the thesis. A network of stationary UWB anchors communicate, allowing them to synchronize their clocks, measure their pairwise distances, and construct a coordinate system. Through passive reception of these communications, a theoretically unlimited number of robots are able to localize themselves within the space. The topology of this system is akin to that of an "indoor-GPS" system.

2

Localization using Ultra-wideband Radio

2.1 Ultra-wideband radio

The basic theory and technology underpinning modern UWB radio systems (originally known as *impulse* radio) has been in development since the late 1960's [2]. This technology is based on the transmission of impulses of radio energy that, being so short in duration, have a wide bandwidth [3]. This wide bandwidth enables low energy operation and robustness to interference, while the short duration of a pulse allows its reception to be accurately timestamped and multi-path components to be filtered [4], [5]. Owing to these benefits, ultra-wideband radio has seen historical application as both a communication system [6] and as a form of radar [7]. More recently, UWB radio has found application in the context of indoor localization systems, driven by the legalization of its unlicensed usage by the U.S. Federal Communications Commission in 2002 [8], the development of a communication and localization standard by the IEEE [9], [10] and the advent of small, low-power UWB radio transceivers, such as the Decawave DW1000 [11] used in this thesis.

This section provides a brief overview of UWB technology, with particular focus on aspects of its operation that make it suitable for radio-based localization. Readers are referred to [3] and [12] for an in-depth discussion of UWB technology, and to [13] for an in-depth review of UWB localization strategies. Experiments in this thesis were conducted using the Decawave DWM1000 UWB module and readers are referred to [11, Appendix 1], [14, Chapter 4] and [15, Section 2] for details on how the DW1000 implements the UWB radio technology discussed herein. An overview of the DW1000 configuration used in this thesis is given in Section 2.1.4.

2.1.1 Communication

UWB radio communication is based on the transmission of short duration pulses based on, for example, a Gaussian mono-cycle waveform (e.g. [6]), a damped sinusoid [12, Chapter 2], or on a raised cosine pulse as reported to be used in the DW1000 [14], [16]. These pulses are generated in the UWB module's baseband frequency and modulated by the carrier frequency, as depicted in Fig. 2.1. By transmitting a pulse and modulating its polarity a single bit of information can be transmitted.

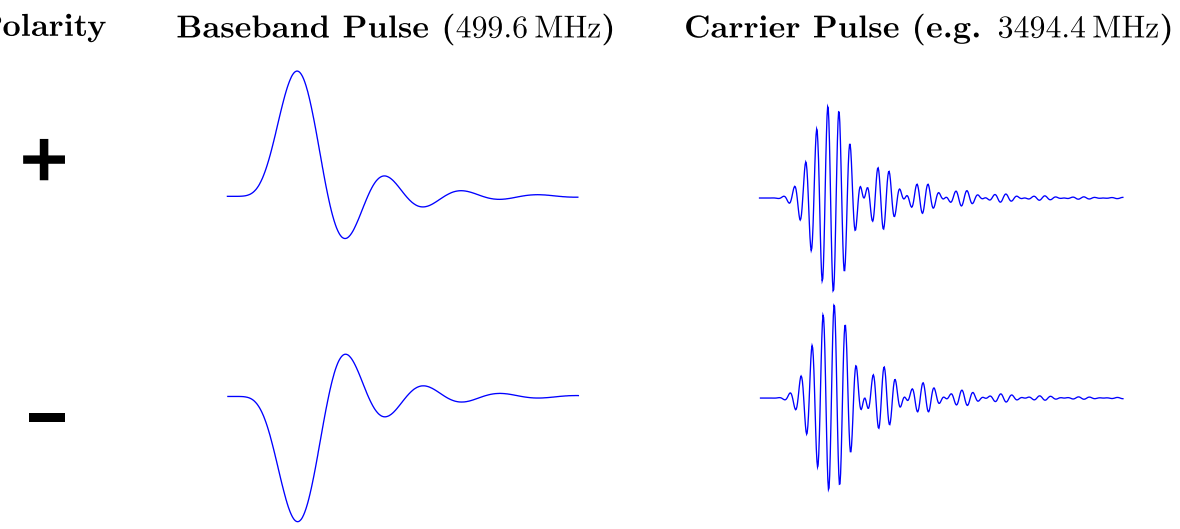


Figure 2.1: This figure shows an example of the UWB pulse shape used by the DW1000. This pulse is generated in the DW1000's baseband frequency (499.2 MHz), and is modulated by the desired channel center frequency (in this example, 3494.4 MHz). By modulating the pulse's polarity, a single bit of information can be encoded in each pulse. This figure has been adapted from [14].

In order to communicate data, multiple pulses are transmitted in sequence with spacing determined by the predefined *pulse repetition frequency*. Transmission of equally-spaced pulses introduces undesirable harmonics in the frequency domain [12, Chapter 1]. The magnitude of these harmonics can be reduced by offsetting the transmission time of each pulse slightly, such that pulses are not equally spaced. In addition to whitening the frequency spectrum, this offsetting allows additional information to be encoded by the offsets of pulses in the pulse train, a technique known as pulse position modulation [12, Chapter 1]. This allows a single pulse to encode two bits of information (based on its polarity, and its offset), as shown in Fig. 2.2. In the DW1000, these additional bits are used for forward error correction to improve robustness to noise [14, Chapter 4], [15, Section 2].

2.1.2 CIR estimation

By transmitting a known pulse sequence and observing how the sequence is affected by the transmission channel (including both transmission and reception antennas), a receiver

Data	T_{symbol}	
	T_0	T_1
00	+	0
01	0	+
10	−	0
11	0	−

Figure 2.2: A simplified representation of two-bit data transmission by modulating the polarity and position of a transmitted pulse. The pulse train is divided into segments of width T_{symbol}, each of which is subdivided into two possible pulse locations. A + indicates that a positive pulse is sent; a − indicates that a negative pulse is sent; and a 0 indicates that no pulse is sent. To further smooth the frequency spectrum, the positioning of pulses within the respective pulse location is additionally shifted by a pseudo-random offset [11].

can estimate the channel's impulse response (CIR). The DW1000 refers to this known sequence as the "preamble" sequence, and it is selected to have minimal cross-correlation in order to facilitate accurate synchronization between receiver and transmitter, and allow estimation of the CIR. Fig. 2.3 shows examples of the estimated CIR magnitude envelope, generated upon reception of a UWB packet from a stationary transmitter. Three individual estimates are shown in the figure in red, yellow and cyan. The CIR would ideally be a single peak, indicative of a single, direct transmission path to the receiver. Multiple peaks in the CIR indicate that the signal has taken multiple paths to reach the receiver due to reflections off objects in the environment [17], [18]. In addition, the CIR estimate is affected by background noise, interference with other radio systems and by the frequency responses of the transmitting and receiving electronics [19]. These effects are further discussed in Section 2.4.

2.1.3 Packet timestamping

As shown in Fig. 2.4, a DW1000 UWB packet is separated into three portions: the synchronization header, beginning with the preamble and terminated by a start frame delimiter (SFD) sequence; the physical header (PHY), including instructions about the length and datarate of the following data; and the data itself, including a two-byte Reed-Solomon error-correction code enabling an additional layer of error detection and correction [14].

The application of UWB radio to localization is facilitated by the ability to accurately schedule packet transmission, and accurately timestamp packet reception. The DW1000 defines a transmission timestamp as the time at which the PHY sequence begins transmission, and a reception timestamp as the time at which reception of this sequence begins. On the DW1000, transmissions can be scheduled to occur at an exact timestamp, and the reception timestamp is estimated using a proprietary thresholding approach to compute the time at which the CIR magnitude exceeds a certain threshold [11].

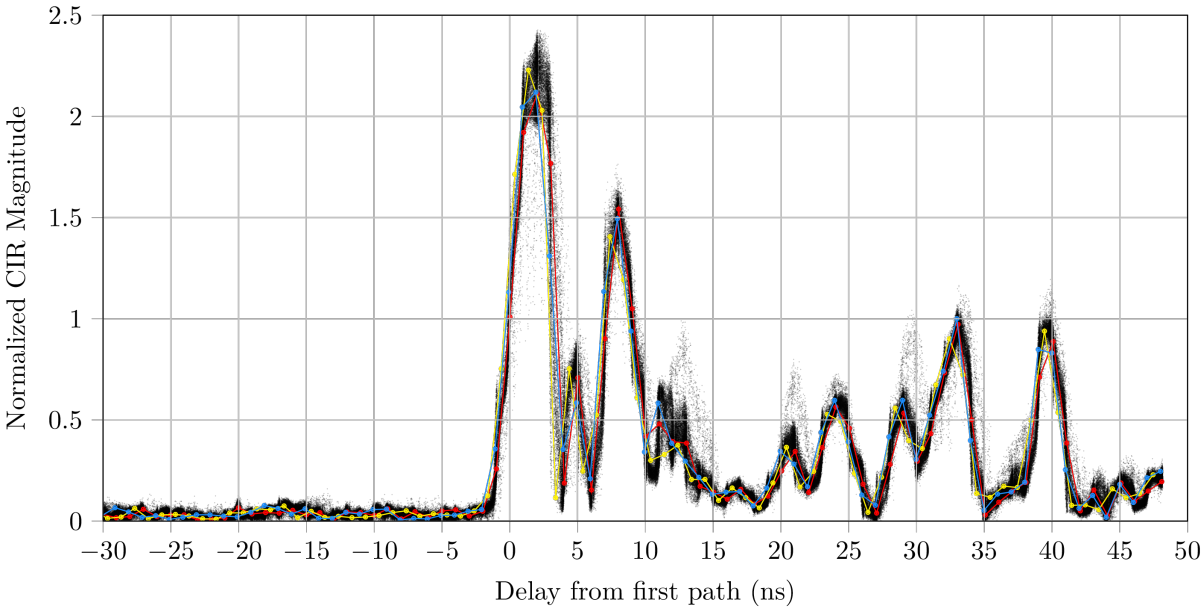


Figure 2.3: This figure shows measurements of the channel impulse response (CIR) magnitude, generated upon reception of a UWB packet. The plot is aligned to the first path occurring at $t = 0$, as measured by Decawave's proprietary algorithm. This plot is composed of hundreds of individual samples of the CIR, three of which are shown in red, yellow and cyan. Ideally, the CIR would be a single peak, however reflections of the transmitted signal off objects in the environment can be observed in the CIR as secondary peaks. Additionally, measurements of the CIR are affected by background noise, attenuation due to propagation distance, and by the frequency response of the transmitting and receiving antennas.

Figure 2.4: A DW1000 UWB packet is separated into three portions: 1. The synchronization header, consisting of preamble and start frame delimiter (SFD) sequences, enables receiving modules to synchronize to the transmission; 2. The physical header (PHY) is used to transmit the packet's length and datarate, and is additionally used to mark the transmission and reception timestamps; and 3. the packet data, which additionally includes a two-byte Reed-Solomon error-correction code. Further information is available in [11].

2.1.4 Hardware overview of the ultra-wideband modules

Experiments in this thesis were conducted using the Decawave DWM1000 UWB module, which consists of a Decawave DW1000 UWB radio connected to a Patron UWB surface-mount antenna. This UWB module was connected via SPI to an STM32F4 microprocessor (32-bit, 168MHz, single-precision floating-point unit, based on the ARM Cortex M4F), as shown in Fig. 2.5.

The DWM1000 UWB module has a timestamping precision of 15.65 ps, in which time a radio pulse propagates 4.7 mm in air. These timestamps are generated based on ticks from a 63.898 GHz clock signal, which is derived from a 38.4 MHz quartz crystal oscillator [11]. As will be later discussed, the accuracy of localization is directly related to the frequency stability of the clocks.

Each UWB module was configured as shown in Table 2.1, resulting in a 0-byte packet air time of approximately 98 µs with an additional 1.18 µs per byte of payload data [11].

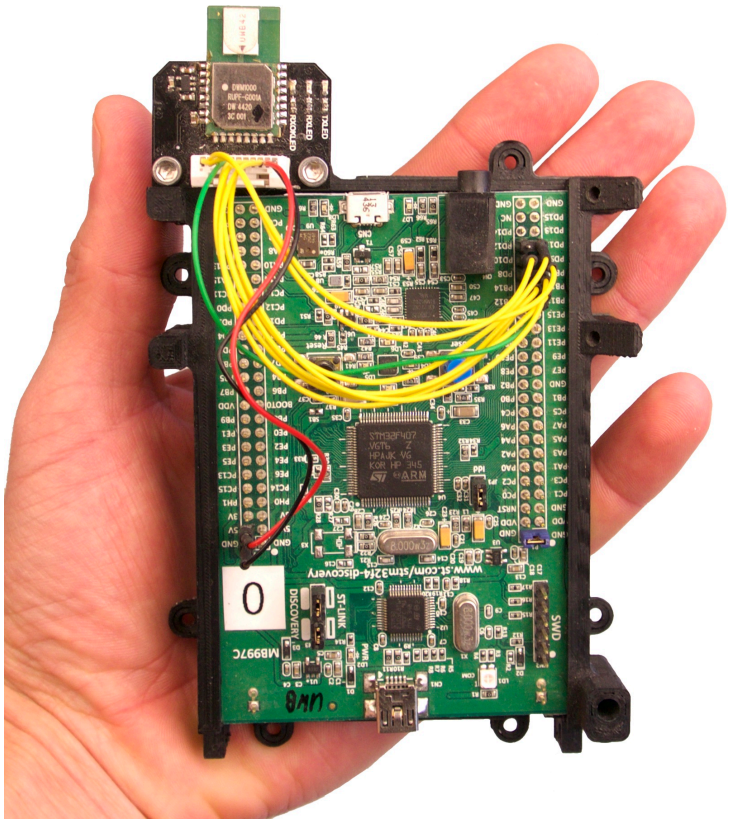


Figure 2.5: Each UWB module consists of an STM32F4 microprocessor (32-bit, 168MHz, single-precision floating-point unit, based on the ARM Cortex M4F) connected to a Decawave DWM1000 UWB radio module.

Table 2.1: DW1000 settings used for UWB experiments.

Parameter	**Value**
DW1000 Channel	#7
• Center Frequency	6489.6 MHz
• Nominal Bandwidth	1081.6 MHz
• Effective Rx Bandwidth	$\sim$ 900 MHz
DW1000 Preamble Code	#17
Pulse Repetition Frequency	64 MHz
Preamble Length	64 symbols
Start Frame Delimiter	8 symbols (non-standard SFD)
Data Rate	6.8 Mbit s^{-1}

2.2 Distance measurement using UWB radio

Having introduced the technology underpinning UWB radio, the technology's applicability to localization now becomes the focus. For the remainder of this thesis, the underlying technology is abstracted and UWB radio is simply treated as a mechanism for transmitting messages at an exact and predefined time instant and for timestamping the arrival of these messages at a receiving antenna. Using this abstraction, the remainder of this chapter progressively introduces the concepts required to understand both the application of UWB radio to localization, as well as the inherent difficulties, uncertainties, and inaccuracies involved.

2.2.1 Distance measurement with synchronized clocks

The discussion of UWB localization begins with the ideal case, in which the clocks of all UWB modules are synchronized and run at real-time speed, and in which reception timestamps are noiseless. This scenario is depicted in Fig. 2.6, in which modules A and B are separated by a distance of $\mathrm{d_{AB}} = \mathrm{d_{BA}}$ meters, corresponding to a propagation delay of $\delta_{\mathrm{AB}} = \delta_{\mathrm{BA}}$ seconds.

In this setup, a UWB packet is transmitted from Module A at the predefined time $t^{\mathtt{T}}$ (with superscript `T` used to denote a transmission timestamp), and is received at Module B at time $t^{\mathtt{R}}$ (with a superscript `R` used to denote a reception timestamp). Under the assumption of real-time clocks and a noiseless reception timestamp, and assuming that packet transmission times are communicated as part of the packet data, the distance between the two UWB modules can be calculated as

$$\mathrm{d_{BA}} = c \cdot \delta_{\mathrm{BA}} = c \cdot (t^{\mathtt{R}} - t^{\mathtt{T}}), \tag{2.1}$$

where c denotes the speed of light in air. Calculation of distance in this way is only possible if the timestamps generated by each module's clock are synchronized to the same time reference and can thus be directly compared.

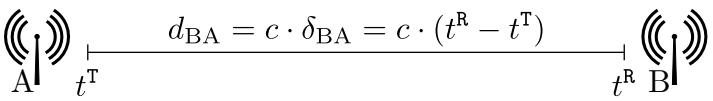


Figure 2.6: A UWB packet is transmitted from Module A at time $t^{\mathtt{T}}$ and received at Module B at time $t^{\mathtt{R}}$. Under the assumption of synchronized, real-time clocks and a noiseless reception timestamp, the distance $\mathrm{d_{BA}}$ between the two UWB modules is simple to calculate.

2.2.2 Non-ideal clock behavior

In practice, each module's clock will begin counting at a different time instant, resulting in an offset between the modules' clocks; and, each module's clock, being based on a physical process (e.g. the frequency of a crystal's oscillations), will run at a slightly different and non real-time rate. The major influences on oscillator frequency are oscillator age, temperature, and voltage. Since timestamps are scaled by the speed of light when used to calculate distance, clock non-idealities have a significant effect on the accuracy and precision of distance measurements. The frequency stability of the chosen physical oscillator, whether a crystal oscillator, temperature-compensated crystal oscillator, oven-controlled crystal oscillator, or a chip-scale atomic oscillator, is therefore decisive for the achievable accuracy of the localization system.

With reference to the UWB modules used in this research, it was assumed that each UWB module was equipped with a hardware clock of sufficient stability; that each UWB module could read the value of its clock exactly; could schedule the transmission of a packet to occur at an exact time; and could timestamp the reception of a packet.

The measurement of Module A's hardware clock at real-time t is denoted $h_{\mathrm{A}}(t)$. A packet sent from a module is denoted by the transmitting module's lowercased ID, with a subscript used to denote the packet's index; for example, a_k denotes the kth packet transmitted from Module A. Please note that if it is clear from context, this subscript will be omitted when referring to the most recent packet.

The transmission or reception of a packet is referred to as an event and, as previously introduced, transmission and reception events are denoted by a superscript $\mathtt{T}$ or $\mathtt{R}$, respectively. Using this notation, $a_k^{\mathtt{T}}$ refers to the transmission of packet a_k from Module A, and $a^{\mathtt{T}}$ to the most recent transmission event from Module A.

For future notational simplicity, $f[\epsilon] \coloneqq f(t_\epsilon)$ is defined to be a sample of the continuous-time process $f(\cdot)$ at the real-time instant corresponding to the occurrence of the discrete event ϵ. Using this notation, $h_{\mathrm{B}}[a^{\mathtt{R}}]$ is the value of Module B's clock at the real-time instant at which the most recent packet transmitted by Module A was received at Module B.

2.2.3 Distance measurement with unknown clock offsets

Returning to the case presented in Fig. 2.6, it is assumed that the hardware clocks of modules A and B run at a real-time rate, but with an offset to real-time, denoted by θ

$$\begin{aligned} h_{\mathrm{A}}(t) &\coloneqq t + \theta_{\mathrm{A}} \\ h_{\mathrm{B}}(t) &\coloneqq t + \theta_{\mathrm{B}}. \end{aligned} \tag{2.2}$$

Recalling (2.1), it is noted that this offset causes a bias in distance measurement, proportional to the relative offset of the two clocks $\theta_{\mathrm{BA}} = \theta_{\mathrm{B}} - \theta_A$

$$\begin{aligned} c \cdot (h_{\mathrm{B}}[a^{\mathrm{R}}] - h_{\mathrm{A}}[a^{\mathrm{T}}]) &= c \cdot (t_{a^{\mathrm{R}}} + \theta_{\mathrm{B}} - t_{a^{\mathrm{T}}} - \theta_{\mathrm{A}}) \\ &= c \cdot \delta_{\mathrm{BA}} + c \cdot (\theta_B - \theta_A). \end{aligned} \tag{2.3}$$

If communication is unidirectional and clocks are unsynchronized, a clock offset or a measurement bias is indistinguishable from the inter-module distance, and only their addition is observable. Section 2.3 addresses this statement in the context of localization.

Consider now the two-way communication shown in Fig. 2.7 and note that Module A can calculate the inter-module distance as

$$\mathrm{d}_{\mathrm{AB}} = c \cdot \delta_{\mathrm{AB}} = \tfrac{c}{2}(h_{\mathrm{A}}[b^{\mathrm{R}}] - h_{\mathrm{A}}[a^{\mathrm{T}}] - \Delta_{\mathrm{A}}). \tag{2.4}$$

Unfortunately, Module A cannot directly measure Δ_{A}; however, under the assumption that the rates of Module A's clock and Module B's clock are equal, $\Delta_{\mathrm{A}} = \Delta_{\mathrm{B}}$, which allows d_{AB} to be calculated as

$$\begin{aligned} \mathrm{d}_{\mathrm{AB}} &= \tfrac{c}{2}(h_{\mathrm{A}}[b^{\mathrm{R}}] - h_{\mathrm{A}}[a^{\mathrm{T}}] - \Delta_{\mathrm{A}}) \\ &= \tfrac{c}{2}(h_{\mathrm{A}}[b^{\mathrm{R}}] - h_{\mathrm{A}}[a^{\mathrm{T}}] - \Delta_{\mathrm{B}}) \\ &= \tfrac{c}{2}\left((h_{\mathrm{A}}[b^{\mathrm{R}}] - h_{\mathrm{A}}[a^{\mathrm{T}}]) - (h_{\mathrm{B}}[b^{\mathrm{T}}] - h_{\mathrm{B}}[a^{\mathrm{R}}])\right). \end{aligned} \tag{2.5}$$

However, the assumption of negligible rate difference is unrealistic when considering that non-atomic oscillators are typically used in such applications.

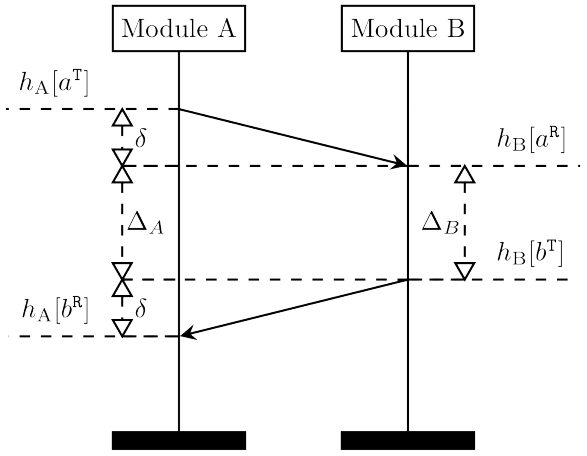


Figure 2.7: In the case when hardware clocks have a relative offset of θ_{BA}, a distance measurement in a single direction will be biased by $\pm c \cdot \theta_{\mathrm{BA}}$ meters. Two-directional communication renders this bias observable and enables accurate distance measurement.

2.2.4 Distance measurement with unknown clock rates

As shown in Fig. 2.8, differing clock rates can be compensated for algorithmically. By scheduling two replies, each spaced by Δ_B seconds, Module B enables Module A to measure the equivalent delay Δ_A in its own clock, and thus calculate the inter-module distance as

$$\begin{aligned} \mathrm{d_{AB}} &= \tfrac{c}{2}(h_\mathrm{A}[b_0^\mathtt{R}] - h_\mathrm{A}[a_0^\mathtt{T}] - \Delta_\mathrm{A}) \\ &= \tfrac{c}{2}((h_\mathrm{A}[b_0^\mathtt{R}] - h_\mathrm{A}[a_0^\mathtt{T}]) - (h_\mathrm{A}[b_1^\mathtt{R}] - h_\mathrm{A}[b_0^\mathtt{R}])). \end{aligned} \tag{2.6}$$

Furthermore, Module A can estimate its relative clock rate ϕ_AB as

$$\phi_{AB} \approx \frac{\Delta_\mathrm{A}}{\Delta_\mathrm{B}} = \frac{h_\mathrm{A}[b_1^\mathtt{R}] - h_\mathrm{A}[b_0^\mathtt{R}]}{h_\mathrm{B}[b_1^\mathtt{T}] - h_\mathrm{B}[b_0^\mathtt{T}]}. \tag{2.7}$$

Hereinafter, this algorithm is referred to as the "two-way ranging" algorithm.

In reality, since the rates of Module A's clock and Module B's clock are time varying, these calculations are only accurate to first order, and only to within the accuracy permitted by measurement noise on the reception timestamps. These uncertainties are further investigated in Section 2.4. Despite these uncertainties, this algorithm is sufficiently accurate to enable the localization and control of a quadcopter using UWB radio, as we showed in [P.4].

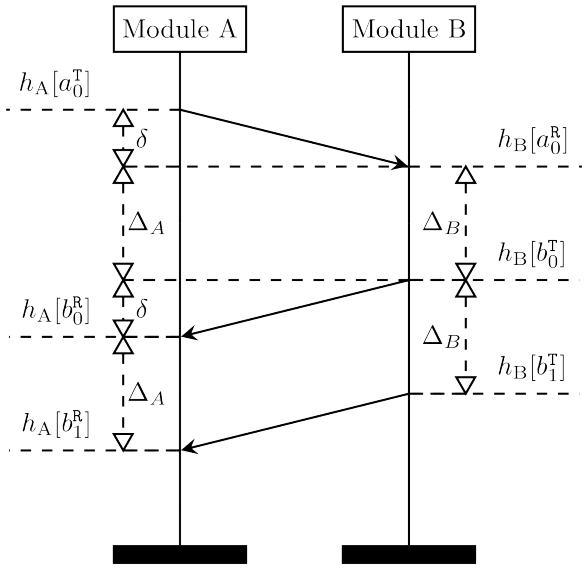


Figure 2.8: The two-way ranging algorithm. By scheduling replies b_0 and b_1 with transmission times $h_\mathrm{B}[b_0^\mathtt{T}]$ and $h_\mathrm{B}[b_1^\mathtt{T}]$ spaced by Δ_B seconds, Module B enables Module A to measure the equivalent delay in its own clock and calculate the inter-module distance.

2.2.5 Distance measurement using pseudo two-way ranging

In Fig. 2.8 it is demonstrated how receiving a repeated reply from Module B allowed Module A to measure its relative clock rate ϕ_{AB}, and thus estimate the inter-module distance. This concept is extended in Fig. 2.9, in which both Module A and Module B transmit packets at arbitrary times. By treating any two replies (e.g. b_0 and b_1) as the "repeated reply" in the two-way ranging algorithm, Module A can estimate its relative clock rate as in (2.7), and estimate the inter-module distance as

$$\begin{aligned}
\mathrm{d_{AB}} &= \tfrac{c}{2}(h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}] - \Delta_{\mathrm{A}_0}) \\
&= \tfrac{c}{2}(h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}] - \phi_{\mathrm{AB}}\Delta_{\mathrm{B}_0}) \\
&= \tfrac{c}{2}\left(h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}] - \frac{\Delta_{\mathrm{A}_1}}{\Delta_{\mathrm{B}_1}}\Delta_{\mathrm{B}_0}\right) \\
&= \tfrac{c}{2}\left(h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}] - \frac{h_{\mathrm{A}}[b_1^{\mathtt{R}}] - h_{\mathrm{A}}[b_0^{\mathtt{R}}]}{h_{\mathrm{B}}[b_1^{\mathtt{T}}] - h_{\mathrm{B}}[b_0^{\mathtt{T}}]}(h_{\mathrm{B}}[b_0^{\mathtt{T}}] - h_{\mathrm{B}}[a_0^{\mathtt{R}}])\right) \\
&= \tfrac{c}{2}\left(h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}] - \phi_{\mathrm{AB}}(h_{\mathrm{B}}[b_0^{\mathtt{T}}] - h_{\mathrm{B}}[a_0^{\mathtt{R}}])\right).
\end{aligned} \tag{2.8}$$

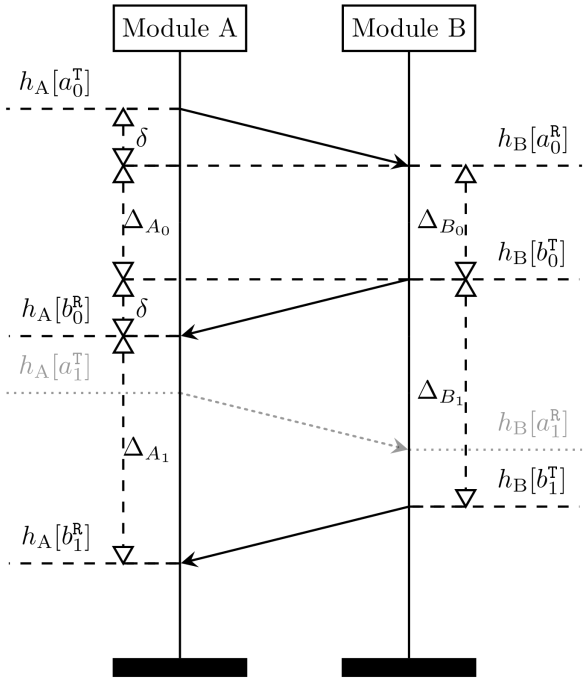


Figure 2.9: Module A records the transmission and reception timestamps of any two packets sent by Module B. Once its relative rate is known, Module A can use two-way ranging to compute its distance from Module B. This algorithm is referred to as the pseudo two-way ranging algorithm.

2.3 Localization using UWB radio

Consider the two dimensional environment depicted in Fig. 2.10, in which a fleet of mobile robots moves within a space surrounded by three fixed-position modules hereinafter referred to as anchors. Given known anchor positions (x_A, y_A), (x_B, y_B), and (x_C, y_C), each robot desires to compute its position within the space.

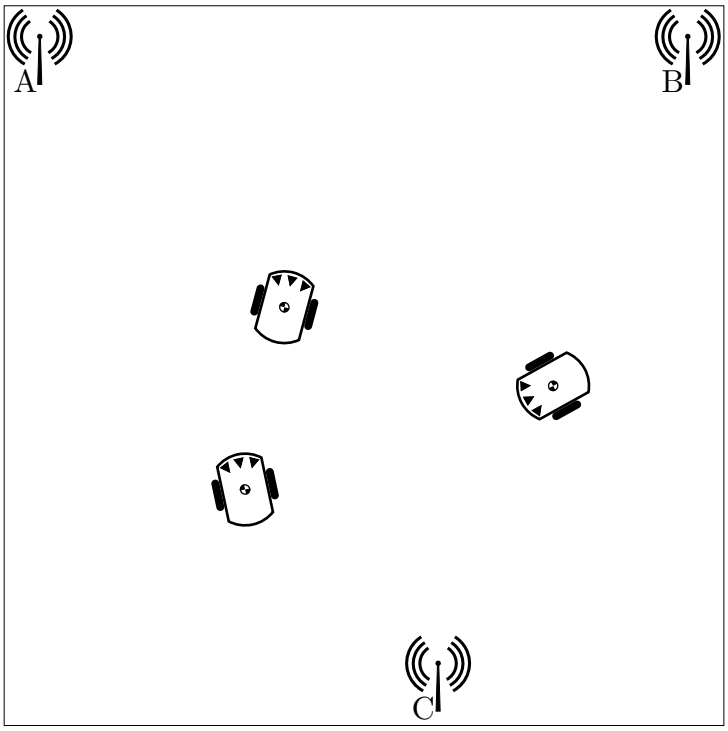


Figure 2.10: A fleet of mobile robots moves within a space, surrounded by three fixed-position UWB anchors. Given known anchor positions, each robot desires to localize itself within the space.

2.3.1 Trilateration using two-way ranging

The first and arguably most well known method of localization is trilateration, whereby each robot calculates its distance to each anchor using, for example, the pseudo two-way ranging algorithm presented in Section 2.2.5.

As shown in Fig. 2.11, the distance between the robot and each anchor defines a set of circles centered at each anchor's known location and with radius equal to the respective robot-anchor distance

$$(x_R - x_A)^2 + (y_R - y_A)^2 = d_{RA}^2 \tag{2.9}$$
$$(x_R - x_B)^2 + (y_R - y_B)^2 = d_{RB}^2 \tag{2.10}$$
$$(x_R - x_C)^2 + (y_R - y_C)^2 = d_{RC}^2, \tag{2.11}$$

where d_{RA}, d_{RB} and d_{RC} are computed using pseudo two-way ranging (Section 2.2.5). By computing the intersection of these circles, the robot's position can be calculated.

Although this method is accurate and robust to noise [P.3], it requires each robot to actively communicate with each of the anchors, and therefore does not scale to large fleets of robots.

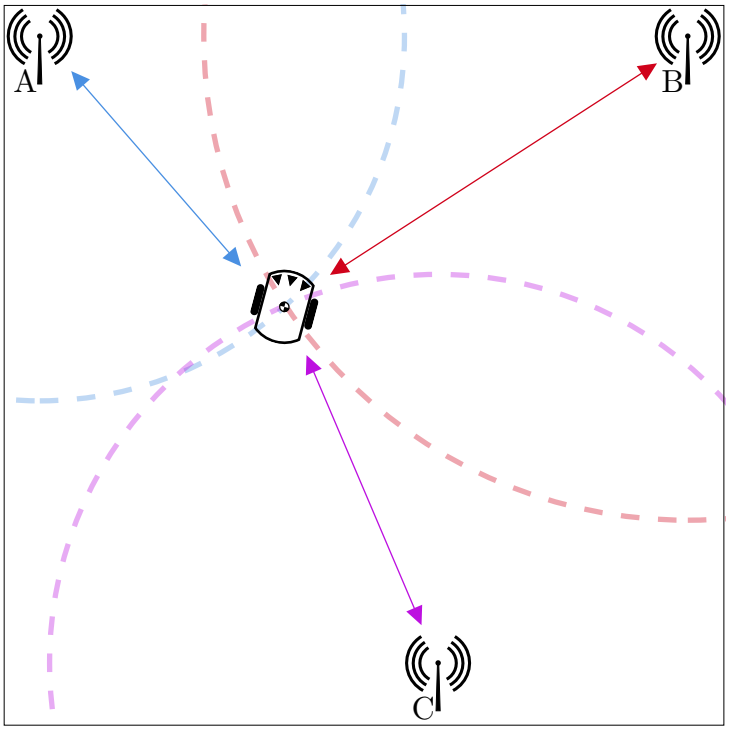


Figure 2.11: A mobile robot measures its distance to each anchor using two-way ranging. The robot's position is calculated as the intersection of circles centered at each anchor's known position, and with radius equal to the robot-anchor distance. Although accurate and robust to noise, this method does not scale to a large number of robots, since each robot is required to actively communicate.

2.3.2 Localization using one-way communication

The problem of scaling can be solved by robots localizing themselves based only on received information, since in such a situation, robots are not required to be active in the communication process. As mentioned in Section 2.2.3, if communication is unidirectional and clocks unsynchronized, a clock offset or a measurement bias is indistinguishable from the inter-module distance, and some form of clock synchronization is therefore required to enable localization. To this end, a system logical clock **S** is introduced, to which each clock in the system should synchronize. Such a situation is depicted in Fig. 2.12, in which a set of synchronized anchors broadcast messages, and robots "eavesdrop" on this communication.

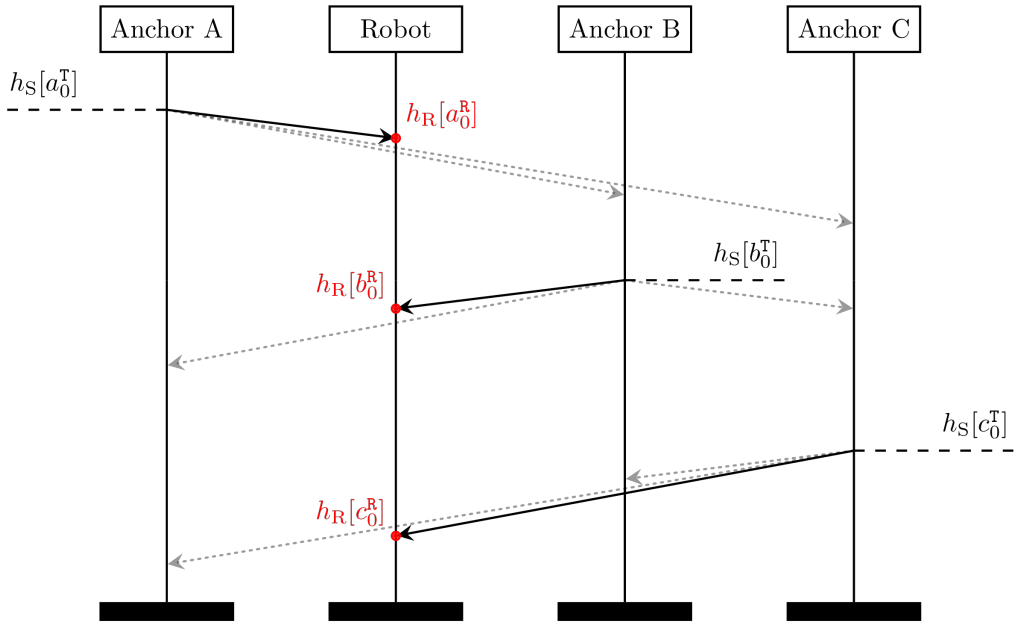


Figure 2.12: The difference between two reception timestamps is related to the difference in the robot's distance to the two transmitting anchors, and the robot can therefore use this information to localize itself. Since the robot is not active in the communication, multiple robots can localize themselves simultaneously, and the system can therefore support a theoretically unlimited number of robots.

Let the clock of an arbitrary UWB module I be related to the system logical clock at real-time t as

$$h_{\mathbf{S}}(t) = \phi_{\mathbf{SI}}(t) \cdot h_{\mathrm{I}}(t) + \theta_{\mathbf{SI}}(t), \tag{2.12}$$

where $\phi_{\mathbf{SI}}(t)$ and $\theta_{\mathbf{SI}}(t)$ are respectively the time-varying rate and offset of Module I's clock with respect to the logical clock. Using this relation, the robot's distance to anchors A, B, and C is given by

$$\begin{aligned}
\mathrm{d_{RA}} &= c \cdot \left((\phi_{\mathbf{SR}}[a_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[a_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[a_0^{\mathtt{R}}]) - h_{\mathbf{S}}[a_0^{\mathtt{T}}] \right) \\
&= c \cdot \left((\phi_{\mathbf{SR}}[a_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[a_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[a_0^{\mathtt{R}}]) - (\phi_{\mathbf{SA}}[a_0^{\mathtt{T}}] \cdot h_{\mathrm{A}}[a_0^{\mathtt{T}}] + \theta_{\mathbf{SA}}[a_0^{\mathtt{T}}]) \right) && (2.13) \\
\mathrm{d_{RB}} &= c \cdot \left((\phi_{\mathbf{SR}}[b_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[b_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[b_0^{\mathtt{R}}]) - h_{\mathbf{S}}[b_0^{\mathtt{T}}] \right) \\
&= c \cdot \left((\phi_{\mathbf{SR}}[b_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[b_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[b_0^{\mathtt{R}}]) - (\phi_{\mathbf{SB}}[b_0^{\mathtt{T}}] \cdot h_{\mathrm{B}}[b_0^{\mathtt{T}}] + \theta_{\mathbf{SB}}[b_0^{\mathtt{T}}]) \right) && (2.14) \\
\mathrm{d_{RC}} &= c \cdot \left((\phi_{\mathbf{SR}}[c_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[c_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[c_0^{\mathtt{R}}]) - h_{\mathbf{S}}[c_0^{\mathtt{T}}] \right) \\
&= c \cdot \left((\phi_{\mathbf{SR}}[c_0^{\mathtt{R}}] \cdot h_{\mathrm{R}}[c_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}[c_0^{\mathtt{R}}]) - (\phi_{\mathbf{SC}}[c_0^{\mathtt{T}}] \cdot h_{\mathrm{C}}[c_0^{\mathtt{T}}] + \theta_{\mathbf{SC}}[c_0^{\mathtt{T}}]) \right). && (2.15)
\end{aligned}$$

In the (unrealistic) case of perfect clock synchronization, that is if $\phi_{\mathbf{SI}}(t)$ and $\theta_{\mathbf{SI}}(t)$ are known exactly for all t and all modules I, the above distances can be calculated exactly and the robot's position can be solved via trilateration as in Section 2.3.1.

If packets are transmitted at a high-enough frequency, robot motion and clock drift can be considered to be negligible between two subsequent packets. It is therefore assumed that $\phi_{\mathbf{SR}}[a_0^{\mathtt{R}}] \approx \phi_{\mathbf{SR}}[b_0^{\mathtt{R}}] \approx \phi_{\mathbf{SR}}[c_0^{\mathtt{R}}] \approx \phi_{\mathbf{SR}}$ and $\theta_{\mathbf{SR}}[a_0^{\mathtt{R}}] \approx \theta_{\mathbf{SR}}[b_0^{\mathtt{R}}] \approx \theta_{\mathbf{SR}}[c_0^{\mathtt{R}}] \approx \theta_{\mathbf{SR}}$. Subtracting (2.13) from (2.14) and (2.14) from (2.15) yields the equalities

$$\begin{aligned} \mathrm{d}_{\mathrm{RB}} - \mathrm{d}_{\mathrm{RA}} &= c \cdot \big((\phi_{\mathbf{SR}} \cdot h_{\mathrm{R}}[b_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}) - (\phi_{\mathbf{SR}} \cdot h_{\mathrm{R}}[a_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}) - (h_{\mathbf{S}}[b_0^{\mathtt{T}}] - h_{\mathbf{S}}[a_0^{\mathtt{T}}])\big) \\ &= c \cdot \big(\phi_{\mathbf{SR}}(h_{\mathrm{R}}[b_0^{\mathtt{R}}] - h_{\mathrm{R}}[a_0^{\mathtt{R}}]) - (h_{\mathbf{S}}[b_0^{\mathtt{T}}] - h_{\mathbf{S}}[a_0^{\mathtt{T}}])\big)\,, \text{ and} \qquad (2.16) \\ \mathrm{d}_{\mathrm{RC}} - \mathrm{d}_{\mathrm{RB}} &= c \cdot \big((\phi_{\mathbf{SR}} \cdot h_{\mathrm{R}}[c_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}) - (\phi_{\mathbf{SR}} \cdot h_{\mathrm{R}}[b_0^{\mathtt{R}}] + \theta_{\mathbf{SR}}) - (h_{\mathbf{S}}[c_0^{\mathtt{T}}] - h_{\mathbf{S}}[b_0^{\mathtt{T}}])\big) \\ &= c \cdot \big(\phi_{\mathbf{SR}}(h_{\mathrm{R}}[c_0^{\mathtt{R}}] - h_{\mathrm{R}}[b_0^{\mathtt{R}}]) - (h_{\mathbf{S}}[c_0^{\mathtt{T}}] - h_{\mathbf{S}}[b_0^{\mathtt{T}}])\big)\,. \qquad (2.17) \end{aligned}$$

Knowing the anchors' positions, the difference in packet transmission times, and having an estimate for the relative rate of the robot's clock $\phi_{\mathbf{SR}}$, each of the above equalities is satisfied by the robot's position lying on a hyperboloid, defined by a constant difference in

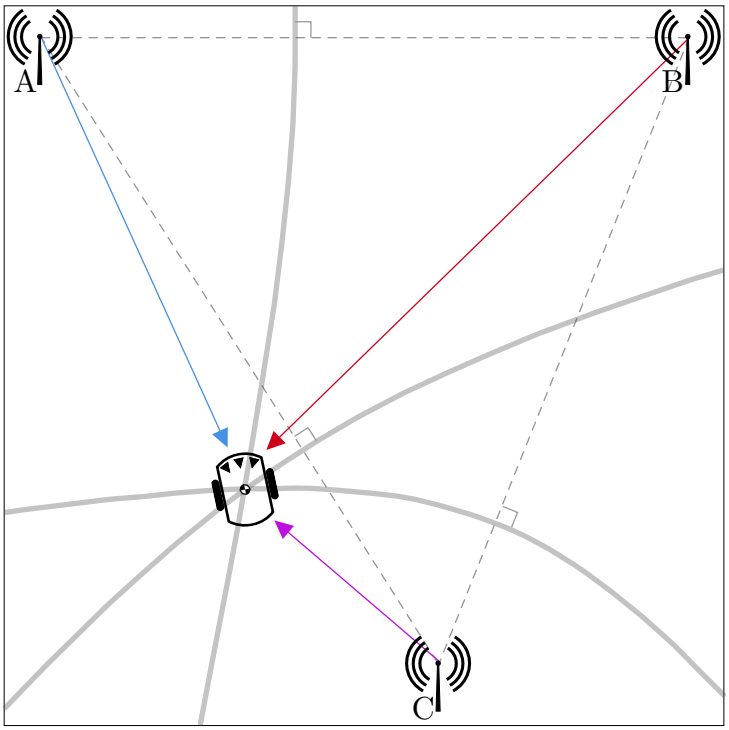


Figure 2.13: A mobile robot localizes itself using time-difference of arrival localization. The robot's position is calculated as the intersection of hyperboloids. Compared with trilateration using two-way ranging, this method is less accurate and more sensitive to noise, however scales to a large number of robots, since robots are not required to actively communicate.

distance between the two anchors. By computing the intersection of these hyperboloids, as shown in Fig. 2.13, the robot can be localized. Note that for completeness, the hyperboloid given by the subtraction of packets c_0 and a_0 is also shown; however, as a linear combination of the other two measurements, this does not contribute additional information. This approach to localization is known as multilateration, or time-difference of arrival (TDOA) localization.

In order to localize in this manner, transmission timestamps need to be synchronized, and the robot's relative clock rate estimated. A first attempt to address these synchronization requirements was published by the author in [P.3] and is shown in Fig. 2.14. This scheme relies on a double transmission from Anchor A to enable other anchors to synchronize their transmissions as

$$\begin{aligned} h_{\mathbf{S}}[a_0^{\mathtt{T}}] &= h_{\mathrm{A}}[a_0^{\mathtt{T}}] \\ &= h_{\mathrm{A}}[a_{-1}^{\mathtt{T}}] + \Delta_{\mathbf{S}} && (2.18) \\ h_{\mathbf{S}}[b_0^{\mathtt{T}}] &= h_{\mathbf{S}}[a_0^{\mathtt{T}}] + \Delta_{\mathbf{S}} \\ h_{\mathrm{B}}[b_0^{\mathtt{T}}] &= h_{\mathrm{B}}[a_0^{\mathtt{T}}] + \Delta_{\mathrm{B}} \\ &= (h_{\mathrm{B}}[a_0^{\mathtt{R}}] - \delta_{\mathrm{BA}}) + (h_{\mathrm{B}}[a_0^{\mathtt{R}}] - h_{\mathrm{B}}[a_{-1}^{\mathtt{R}}]) && (2.19) \\ h_{\mathbf{S}}[c_0^{\mathtt{T}}] &= h_{\mathbf{S}}[a_0^{\mathtt{T}}] + 2\Delta_{\mathbf{S}} \\ h_{\mathrm{C}}[b_0^{\mathtt{T}}] &= h_{\mathrm{C}}[a_0^{\mathtt{T}}] + 2\Delta_{\mathrm{C}} \\ &= (h_{\mathrm{C}}[a_0^{\mathtt{R}}] - \delta_{\mathrm{CA}}) + 2(h_{\mathrm{C}}[a_0^{\mathtt{R}}] - h_{\mathrm{C}}[a_{-1}^{\mathtt{R}}]), && (2.20) \end{aligned}$$

where anchor locations (and thus inter-anchor propagation times δ) are known.

Given synchronized transmissions and a measurement of the transmission spacing $\Delta_{\mathbf{S}}$ in its own clock Δ_{R}, the robot can compute (2.16) and (2.17) as

$$\begin{aligned} \mathrm{d_{RB}} - \mathrm{d_{RA}} &= c \cdot \left(\frac{\Delta_{\mathbf{S}}}{\Delta_{\mathrm{R}}} (h_{\mathrm{R}}[b_0^{\mathtt{R}}] - h_{\mathrm{R}}[a_0^{\mathtt{R}}]) - \Delta_{\mathbf{S}} \right) && (2.21) \\ &\approx c \cdot \left((h_{\mathrm{R}}[b_0^{\mathtt{R}}] - h_{\mathrm{R}}[a_0^{\mathtt{R}}]) - \Delta_{\mathrm{R}} \right), \text{ and} && (2.22) \\ \mathrm{d_{RC}} - \mathrm{d_{RB}} &= c \cdot \left(\frac{\Delta_{\mathbf{S}}}{\Delta_{\mathrm{R}}} (h_{\mathrm{R}}[c_0^{\mathtt{R}}] - h_{\mathrm{R}}[b_0^{\mathtt{R}}]) - \Delta_{\mathbf{S}} \right) && (2.23) \\ &\approx c \cdot \left((h_{\mathrm{R}}[c_0^{\mathtt{R}}] - h_{\mathrm{R}}[b_0^{\mathtt{R}}]) - \Delta_{\mathrm{R}} \right). && (2.24) \end{aligned}$$

Note that the relative clock rate $\frac{\Delta_{\mathrm{R}}}{\Delta_{\mathbf{S}}}$ is typically within the range of $1 \pm 1 \times 10^{-4}$ for a simple crystal oscillator, and its effect on distance difference can therefore be neglected.

As we show in [P.3], the above messaging sequence is sufficient for a quadcopter to estimate and maintain a position in space. However, this sequence is highly affected by noise in the reception timestamps, and if no additional filtering is performed, requires that all modules receive both packets a_{-1} and a_0.

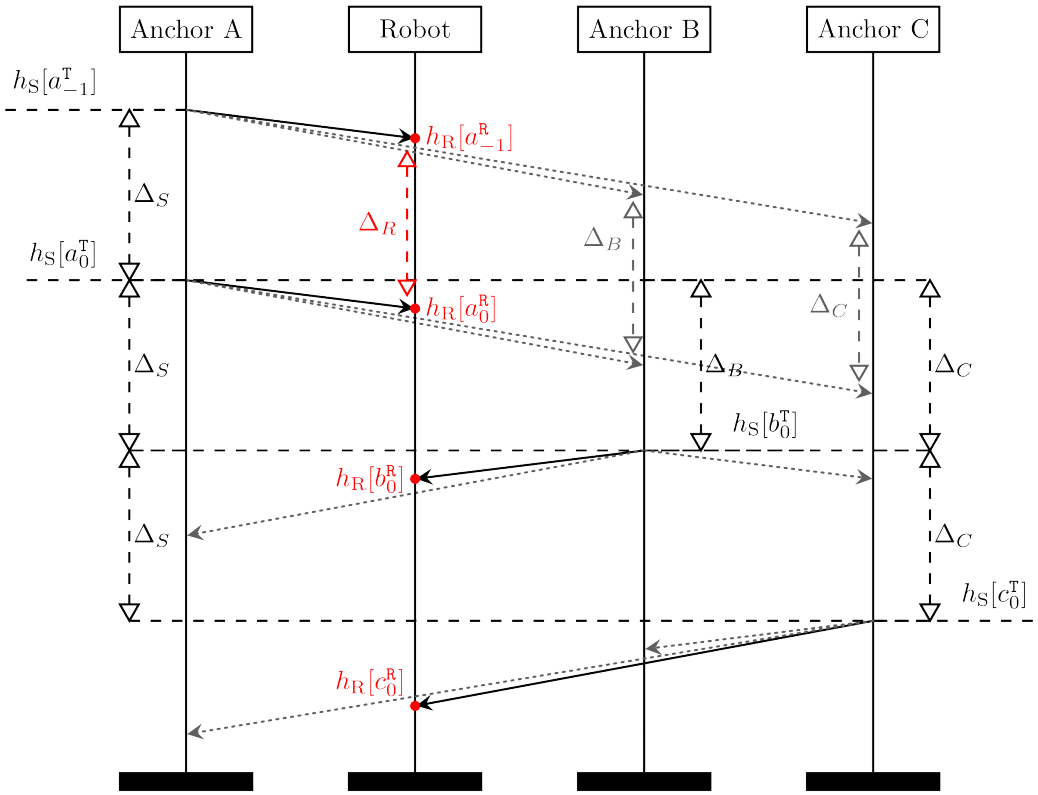


Figure 2.14: By transmitting two packets at the start of each messaging sequence, Anchor A defines the timing of the sequence. Assuming anchor positions are known, anchors can synchronize their transmission times to be evenly spaced, and each robot can localize itself based on the difference in packet arrival time.

2.3.3 Centralized vs Decentralized localization

In the previous discussion, algorithms are presented in the context of "decentralized localization", a situation in which a robot must determine its own position. This is advantageous if the robot requires low-latency, high-frequency knowledge of its position, for example for state estimation and/or control. In such cases, if a centralized system would need to know the robots position (for example to implement higher level task or trajectory planning, or facilitate collision avoidance), the robot can transmit its position estimate at a lower rate over a high-bandwidth, higher latency wireless communications channel, such as 802.11 WiFi.

Use cases such as asset tracking are an example of "centralized localization". In such cases, the position of modules must be known centrally and individual modules do not need to know their position. Although it is not discussed in detail and will not be further covered in this thesis, all algorithms shown thus far can be applied to this use case.

2.3.4 Closed-form vs Recursive localization

In the previous discussion, as an introduction to the topic and for purposes of clarity, the focus has been on computing the location of a UWB module directly and in closed form. In the context of dynamical systems, it may however be advantageous to recursively estimate the position of a module by using individual measurement equations to update the state of a Bayesian filter (e.g. a Kalman filter). This approach additionally allows module dynamics and other sensor measurements to contribute to the accurate estimation of the module's position. This was the approach taken by the author in [P.1], [P.3], [P.4], and is discussed in more detail in Chapter 7.

2.3.5 Model-based clock synchronization & localization

In addition to recursively estimating the position of a UWB module using Bayesian filtering, the relative behavior of two UWB clocks can be estimated in a similar fashion, treating transmission and reception timestamps as noisy measurements of an underlying clock process. This approach was published by the author in [P.1] and is discussed in more detail in Chapter 4.

2.4 Localization uncertainties

In this section, experiments are performed to demonstrate the effects of uncertainties and non-idealities on localization performance. Experiments were performed for two different anchor pairings, with anchors placed at a distance of 5 meters and instructed to range using the pseudo two-way ranging algorithm presented in Section 2.2.5.

2.4.1 Measurement noise

Packet transmission can be scheduled exactly, and transmission timestamps are as such noiseless; however, the estimation of packet reception time is affected by the quality of the CIR estimate. In Section 2.1, the CIR is investigated with discussion as to how the signal's "first path" reception time can be identified using methods such as thresholding. In the (realistic) case in which background noise affects the signal during propagation, an appropriate noise signal could positively interfere, and cause this threshold to be reached earlier; or can negatively interfere, and delay this threshold being reached. Thus, this noise directly affects the measured reception timestamps, as shown in Fig. 2.15. This measurement noise is investigated in greater detail in Section 4.2.

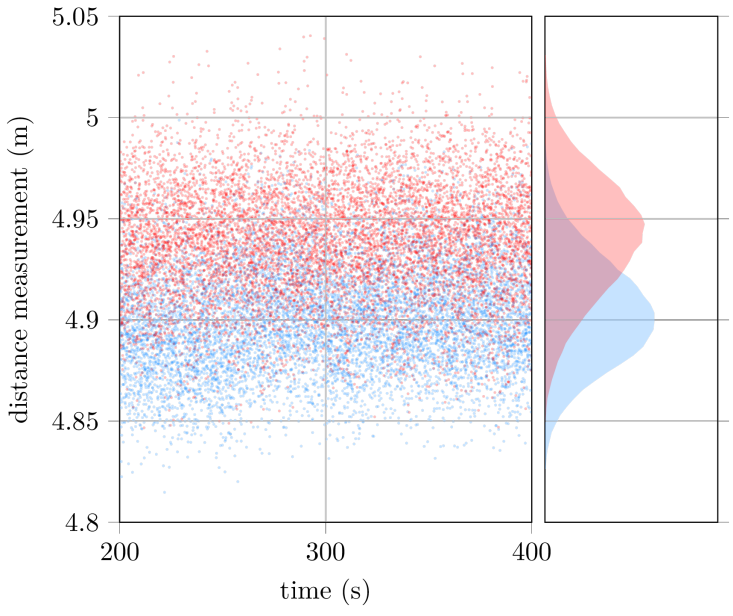


Figure 2.15: Two modules were placed 5 meters apart and the distance between each was calculated using pseudo two-way ranging. This experiment was repeated for two different anchor pairs, with results shown in blue and red. Anchors were given 200 seconds to reach a steady-state temperature before recording started. These results show the measurement noise present when reading reception timestamps, as well as the module-dependent bias.

2.4.2 Clock variation

As described in previous sections, the relative rate of a pair of UWB modules results in the same time period being measured differently by each module. Scaled by the speed of light, a small difference in these measurements can result in a large error in distance. Taking the experiment of Fig. 2.15 as an example, if the inter-module distance were to be computed without compensating for the relative clock rate ϕ_{AB} as

$$\mathrm{d_{AB}} = \tfrac{c}{2}\left((h_{\mathrm{A}}[b_0^{\mathtt{R}}] - h_{\mathrm{A}}[a_0^{\mathtt{T}}]) - (h_{\mathrm{B}}[b_0^{\mathtt{T}}] - h_{\mathrm{B}}[a_0^{\mathtt{R}}])\right), \tag{2.25}$$

Module A would calculate its average distance to Module B to be 2.26 m, and Module B would calculate its average distance to Module A to be 7.56 m. Both the bias compared to the average rate-adjusted measurement (4.94 m in both directions), as well as the discrepancy between these two measurements depend on the relative steady-state clock rates of the two modules. It is therefore very important to consider and compensate for relative clock rates, either algorithmically through a repeated message or by tracking clock rates using a clock model, as will be discussed in Chapter 4.

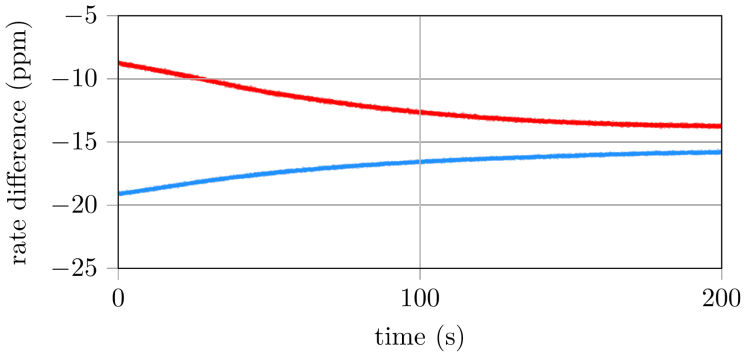
Compensation for differing clock rates using a repeated message assumes that clock rates remain constant between the reception of the first and second replies. This assumption is reasonable during steady-state operation; however, during the first minutes of operation when modules are still warming, their clock rates can change significantly between messages. This effect is shown in Fig. 2.16, and motivates the use of a state estimator and identified clock dynamics (in Chapter 4) to track not only clock rate, but also clock acceleration.

2.4.3 Measurement bias & environmental effects

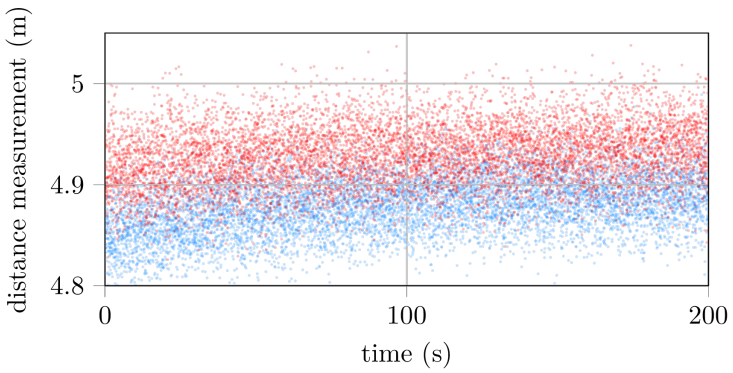
An example of the module-dependent measurement bias is shown in Fig. 2.15. The difference between modules is caused by slight manufacturing differences interacting with each module's timestamping process, resulting in a systematic bias in the reception timestamps generated by each module.

In particular, each module's antenna transfer function has a noticeable effect on the measured distance [19]. The antennas used in this thesis are omnidirectional, and ideally should have a constant transfer function for all orientations. Fig. 2.17 and Fig. 2.18 show the case in which the antenna of one UWB module is rotated around its vertical, and its forward-facing axis. These figures demonstrate the effect of non-idealities in the antenna transfer function on the pairwise distance measurement, and suggest that measurement biases are a function of the relative orientation of the pair of UWB modules, and vary significantly throughout the range of possible orientations.

The interaction of environmental reflections with the shape of the CIR has an additional influence on measured distance. As shown in Fig. 2.3, signal reflections from objects in the environment can be observed in the measured CIR. If these reflections occur too



(a) Change in relative clock rate during warmup phase

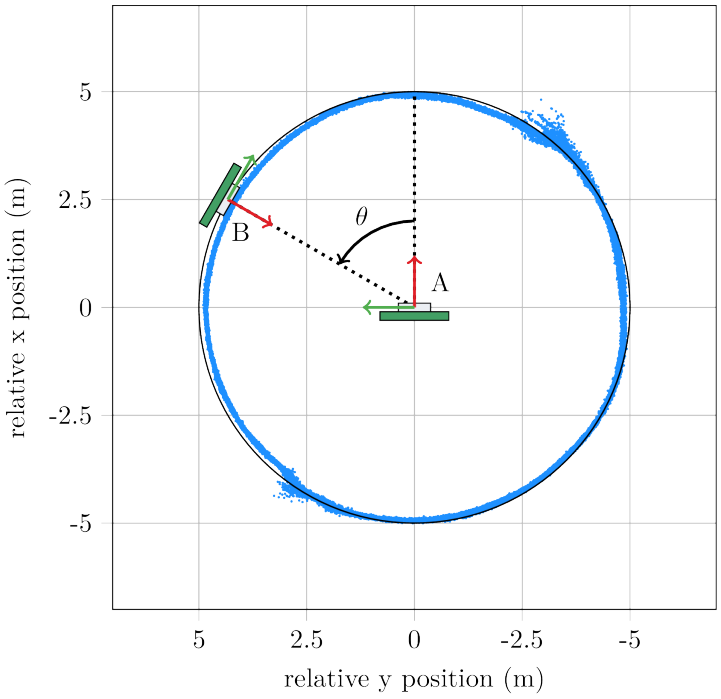


(b) Change in distance measurement during warmup phase

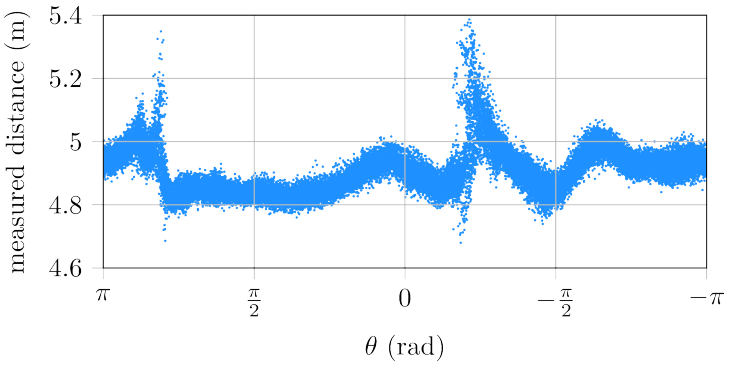
Figure 2.16: This figure shows the first minutes of ranging operation over a distance of 5 m. During these first minutes, changes in the modules' temperatures affect their relative clock rates, as seen in subfigure (a). Although the two-way ranging algorithm compensates for differing clock rates to first order, changes in module clock rates between reception of the two replies introduce second order effects, which are noticeable during this period of warming, as shown in subfigure (b). After approximately the first three minutes of operation, modules reach a steady-state and behave as previously shown in Fig. 2.15.

closely to the main lobe they will affect the accuracy of the timestamping process and result in a systematic measurement bias. This implies that the measurement bias is not only dependent on the relative orientation a pair of UWB modules, but also on their placement within the environment.

This section has presented a series of simple experiments that exemplify the uncertainties and non-idealities present in an UWB localization system. Many of these non-idealities are unobservable during ranging operation and are therefore difficult to compensate for. Chapter 7 presents results showing the effect that these non-idealities have on a quadcopter flying within a space. In particular, the results in Chapter 7 show how these systematic measurement biases vary throughout the space and result in systematic, position-dependent estimation errors.

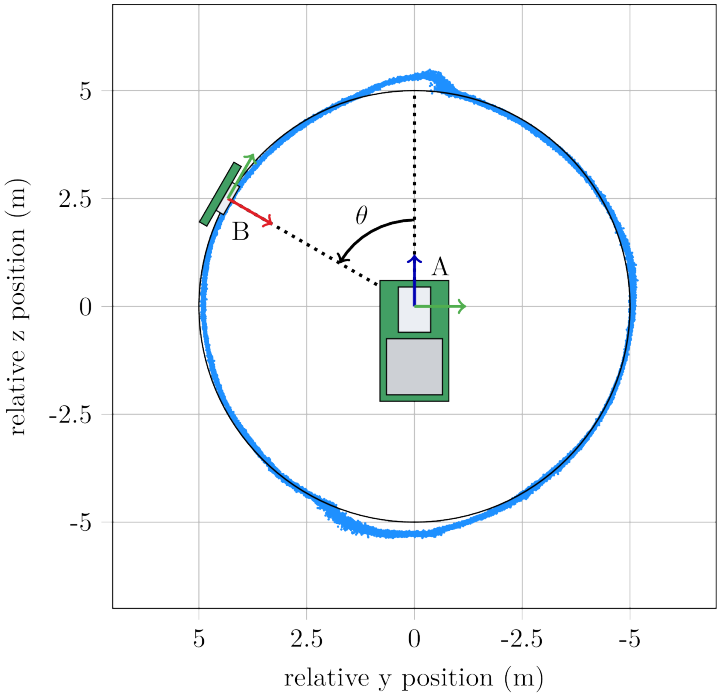


(a) The position of Module B measured by and relative to Module A

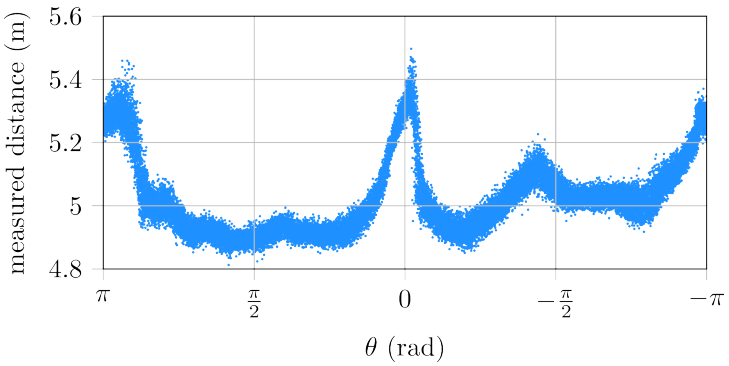


(b) Measured distance as a function of angle

Figure 2.17: In this experiment, Module B was placed 5 meters from Module A, and Module A was rotated around its vertical axis as shown. The position of Module B relative to Module A is expressed in Module A's body coordinate system, giving the effect of rotating Module B around Module A. In this figure, systematic variations in measured distance are observed to be a function of the relative orientation of the two modules, an effect caused by interactions between background noise and the timestamping process, coupled with a sub-optimal, angle-dependent antenna transfer function.



(a) The position of Module B measured by and relative to Module A



(b) Measured distance as a function of angle

Figure 2.18: The experiment of Fig. 2.17 was repeated with rotation around Module A's forward-facing axis, as shown. The position of Module B relative to Module A is expressed in Module A's body coordinate system, giving the effect of rotating Module B around Module A. This figure shows a different axis of rotation, and further exemplifies the angle-dependent transfer function. Distortions around $\theta = 0$ and $\theta = \pm\pi$ correspond to regions of low antenna gain.

3

Literature Review

Having introduced the background knowledge required to understand research in the field of UWB localization, this chapter provides an overview of the field's state at the time of writing. Particular focus is placed on research relevant to the contributions of this thesis, which is found at the intersection of two broad research fields: clock modeling and synchronization in wireless networks; and UWB-radio-based localization. Although many of the concepts presented in this thesis are related to ideas found in existing literature, the core and novel contribution of this thesis to existing knowledge lies in the adaptation and combination of these areas for usage in UWB localization.

3.1 Clock synchronization

Clock synchronization of networked devices is a thoroughly-studied problem and many robust algorithms have been proposed. The Network Time Protocol [20], for example, is deployed world-wide and has proven robust in the context of synchronizing devices across the internet. Synchronization of wireless sensors can pose additional difficulties such as limited bandwidth, energy, memory, and computation; potential motion of the sensors; and signal interference or multi-path problems. Tailored approaches to the synchronization of wireless sensor networks have therefore been a target of more recent research, with early results summarized, for example, in the review papers [21] and [22].

Of particular relevance to UWB localization is research focused on the *gradient clock synchronization* property. This property was proposed in [23] and requires that the logical clock skew between two nodes in a network be bounded by a non-decreasing function of their distance; that is, closer nodes are more closely synchronized. This is in contrast to standard clock synchronization algorithms, which aim to minimize global skew and thus often require maintaining or optimizing global state, rather than relying only on local information. Algorithms addressing the gradient clock synchronization property have been suggested in, for example, [24]–[27]. In [25] it is shown that a simple approach based on distributed averaging performs well in the context of gradient clock synchronization, while requiring minimal computation and storage, and remaining robust to changes in network topology. These properties make the approach of [25] ideal for application to clock synchronization in UWB localization networks. This is demonstrated in Chapter 5,

within which a similar method is used to synchronize a network of UWB anchors.

This approach to network synchronization relies on accurate pairwise clock synchronization. Within this thesis, this pairwise clock behavior is tracked using a Kalman filter, an approach similar to, for example [28]–[30], as well as similar to [31], who additionally note the effectiveness of a Kalman filter at compensating for packet loss. Similar to [31], this thesis uses a clock model based on a system identification of the pairwise clock behavior; however, the implementation in this thesis differs in that system identification is used to determine parameters for a continuous-time clock model and exact discretization is used to arrive at the discrete time dynamics and process noise covariance. This allows the model to correctly handle varying reception periods, as is encountered during clock synchronization or due to packet loss.

3.2 UWB-based robot localization

Early results using a UWB radio network as a means of robot localization include [32], who use centralized TDOA to localize a transmitting agent; with [33] suggesting the use of a particle filter to combine a dynamics model, IMU measurements, and UWB range measurements in order to facilitate agent tracking; and [34], who extend this particle-filter based approach to the case of mobile-robot tracking, while also presenting an analysis of tracking performance in line-of-sight (LOS) and non-line-of-sight (NLOS) environments.

A similar analysis of UWB performance in NLOS environments is presented in [35], and is further expanded upon in the subsequent work [36]–[38]. In these works, small robots use UWB to localize themselves within an area. Accurate localization is facilitated by a UWB measurement model incorporating the probability of both LOS and NLOS measurements, as well as by robots measuring and sharing their pairwise distances and bearings.

Many of these early results employ custom electronics to transmit and receive UWB signals. Further examples of early UWB localization systems can be found in, for example [39], who demonstrate that robots receiving measurements from externally-synchronized, actively-transmitting anchors are able to self-localize using a TDOA approach; and [40], who demonstrate the feasibility of the inverse approach (active robots, passive anchors), while also analyzing the importance of anchor placement for TDOA-localization.

In more recent years, the development of small, low-power UWB ICs (for example, the Decawave DW1000 [41], as used in this thesis) has simplified the integration of UWB localization with more-standard robotic systems. Combined with increases in the performance of embedded processors, UWB localization has become a promising solution for real-time, on-board localization and control. We demonstrated this in [P.4] by outfitting a standard quadcopter with a UWB module and demonstrating that by using two-way ranging (Fig. 2.8) for range measurement and an extended Kalman filter for

state estimation, the quadcopter was able to fly a predefined trajectory using only onboard measurements and control. A similar approach was used by [42] who demonstrated controlled trajectory tracking in a multitude of indoor and outdoor environments; and by [43] who demonstrated the flight of multiple quadcopters by coordinating their ranging requests using TDMA. We demonstrated similar results using TDOA localization in [P.3], using the scheme of Fig. 2.14 to enable a quadcopter to localize itself based on passive observation of anchor transmissions. In [44] the inverse problem is addressed, and an actively-transmitting ground robot is centrally localized using a passive network of anchors.

One of the largest issues with UWB localization, as identified by many of the aforementioned papers and briefly discussed in Section 2.4, are time-stamping inaccuracies leading to biased, noisy, or incorrect range measurements. Recent work in [45], [46] suggests that multiplexing antennas (each mounted in a different orientation) and transmission channels can be used to generate a diversity of ranging measurements and improve measurement accuracy and precision. Related work in [47] considers a frequency-domain band-stitching approach to improve the accuracy of timestamping. In [R.2] and [R.4] an angle-dependent model of antenna biases is used to compensate for timestamping inaccuracies.

The fusion of UWB localization with other sensors has also been investigated as a way to mitigate UWB biases. In particular, vision sensors have received significant attention. In [R.3], we show that by augmenting UWB localization with visual odometry, local maneuver accuracy can be improved. The inverse problem is tackled in [48], where a visual SLAM system is augmented with UWB localization to reduce the computational overhead of vision-based localization, assist with visual loop closure, and to mitigate problems with visual localization in feature-sparse environments or environments with many specular reflections.

In the following chapters of this thesis the existing literature in the space of UWB-based localization is extended by:

1. developing a synchronization algorithm for wireless, stationary UWB anchors, allowing them to synchronize with enough accuracy to facilitate accurate distance measurement;
2. leveraging this synchronization to allow anchors to localize themselves and construct a coordinate system; and
3. enabling robots to localize themselves within the space based only on passive reception of the anchors' communications, thus allowing a theoretically unlimited number of robots to localize simultaneously.

As has been presented in this chapter, existing research has tackled each one of these points to some extent; the novel contribution of this thesis lies in the adaptation, extension, and combination of these into a scalable indoor localization system for large robot swarms.

4

Pairwise Clock Modeling & Tracking

As discussed in Chapter 2, pairwise clock synchronization is required for distances (or differences in distances) to be accurately measured. In this chapter pairwise clock behavior is modeled using a systems-based approach and, using this model, a time-varying Kalman filter is developed to track this relative clock behavior. This pairwise synchronization is later leveraged for network self-localization and by robots to localize themselves within the anchor network.

4.1 Notation

A connected network of anchors is denoted by $\mathcal{A}$. For purposes of explanation and without loss of generality, anchors IDs are denoted by capital letters, with I and J used to refer to arbitrary anchors within $\mathcal{A}$. Full-connectivity of the network is not required, and $\mathcal{A}_{\mathrm{I}}$ is used to denote the set of anchors capable of bi-directional communication with Anchor I. Recall from Chapter 2 that $\mathrm{d}_{\mathrm{IJ}} = \mathrm{d}_{\mathrm{JI}}$ denotes the distance between two anchors I and J, and that $\delta_{\mathrm{IJ}} = \delta_{\mathrm{JI}}$ denotes the equivalent propagation delay.

Recall that each UWB anchor $\mathrm{J} \in \mathcal{A}$ possesses a hardware clock, whose value at real-time t is denoted using $h_{\mathrm{J}}(t)$ and whose value at the discrete transmission or reception event ϵ is denoted $h_{\mathrm{J}}[\epsilon]$. This notation is extended to clock rates and accelerations and thus

$$\dot{h}_{\mathrm{J}}[\epsilon] \coloneqq \frac{dh_{\mathrm{J}}}{dt}[\epsilon] \tag{4.1}$$

defines the real-time clock rate of Anchor J's clock at the occurrence of event ϵ. Since there is no real-time reference in the system, this rate cannot be measured; however, for purposes of synchronization only relative rates must be tracked, for example the rate of Anchor J's clock relative to the rate of Anchor I's clock

$$\frac{dh_{\mathrm{J}}}{dt}[\epsilon] / \frac{dh_{\mathrm{I}}}{dt}[\epsilon] = \frac{dh_{\mathrm{J}}}{dh_{\mathrm{I}}}[\epsilon]. \tag{4.2}$$

For future notational simplicity, this relative rate is denoted by

$$\dot{h}_{\mathrm{J}}^{(\mathrm{I})}[\epsilon] \coloneqq \frac{dh_{\mathrm{J}}}{dh_{\mathrm{I}}}[\epsilon]. \tag{4.3}$$

It follows that

$$\ddot{h}_{\mathrm{J}}^{(\mathrm{I})}[\epsilon] \coloneqq \frac{d^2h_{\mathrm{J}}}{dh_{\mathrm{I}}^2}[\epsilon], \text{ and} \tag{4.4}$$

$$\dddot{h}_{\mathrm{J}}^{(\mathrm{I})}[\epsilon] \coloneqq \frac{d^3h_{\mathrm{J}}}{dh_{\mathrm{I}}^3}[\epsilon], \tag{4.5}$$

respectively denote the relative acceleration and jerk of Anchor J's clock with respect to Anchor I's clock.

4.2 Modeling and tracking relative clock behavior

Fig. 4.1 shows the case in which Anchor I transmits packet i_k at its hardware clock time $h_{\mathrm{I}}[i_k^{\mathtt{T}}]$, and Anchor J receives this packet at time $h_{\mathrm{J}}[i_k^{\mathtt{R}}]$, measured in its hardware clock. Recall that, in the case of uni-directional communication, propagation delay δ_{IJ} cannot be distinguished from the relative clock offset θ_{IJ}. As such, the inter-anchor synchronization implicitly includes this delay, which is then compensated for once two-directional communication has begun.

The behavior of Anchor J's clock is modeled with respect to Anchor I's clock as a third-order linear system driven by noise

$$\dddot{h}_{\mathrm{J}}^{(\mathrm{I})}(t) = \nu(t), \tag{4.6}$$

where $\nu(t) \sim \mathcal{N}(0, \sigma^2)$ is the continuous-time process noise driving the inter-clock relationship. This third-order model assumption allows the value, relative rate and relative acceleration of Anchor J's clock with respect to Anchor I's clock to be tracked. Tracking relative acceleration is particularly important in the first few minutes of network operation as the clocks of both radios warm from room temperature to a more steady state, causing significant changes in the clocks' rates, as shown in Fig. 2.16.

Since exact transmission scheduling allows the transmission timestamp $h_{\mathrm{I}}[i_k^{\mathtt{T}}]$ to be known exactly, and since noise only enters the system through the reception timestamp $h_{\mathrm{J}}[i_k^{\mathtt{R}}]$, the above system is a standard third-order linear system with Anchor I's clock forming the time-basis of the system and measurements $h_{\mathrm{J}}[i_k^{\mathtt{R}}]$ being corrupted by noise.

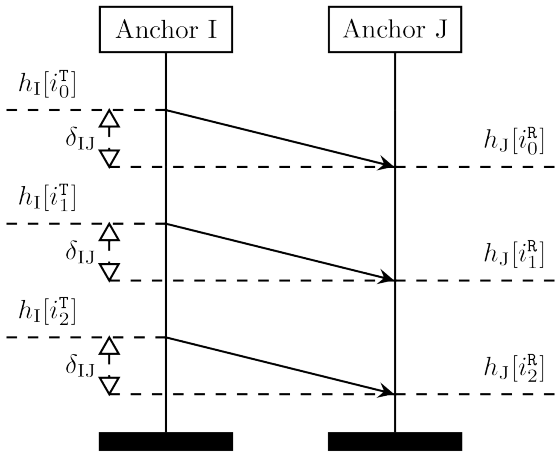


Figure 4.1: In order for Anchor J to synchronize to Anchor I, it timestamps the arrival of packets from Anchor I. By assuming the inter-clock behavior has third-order dynamics, a Kalman filter was developed and tuned to allow Anchor J to track the value, rate and acceleration of its clock relative to the clock of Anchor I. Synchronizing each pair of clocks is the first step towards achieving network synchronization.

Since measurements $h_{\mathrm{J}}[i_k^{\mathrm{R}}]$ only occur at the reception of a packet, an exact discretization of (4.6) is computed at the reception of packet i_k, for the sampling time $\Delta \coloneqq h_{\mathrm{I}}[i_k^{\mathrm{T}}] - h_{\mathrm{I}}[i_{k-1}^{\mathrm{T}}]$.

Letting

$$q_{\mathrm{J}}^{(\mathrm{I})}[i_k^{\mathrm{R}}] \coloneqq \begin{bmatrix} h_{\mathrm{J}}[i_k^{\mathrm{R}}] & \dot{h}_{\mathrm{J}}^{(\mathrm{I})}[i_k^{\mathrm{R}}] & \ddot{h}_{\mathrm{J}}^{(\mathrm{I})}[i_k^{\mathrm{R}}] \end{bmatrix}^{\mathsf{T}} \tag{4.7}$$

this exact discretization results in the discrete-time state-space model

$$\begin{aligned} q_{\mathrm{J}}^{(\mathrm{I})}[i_k^{\mathrm{R}}] &= F\, q_{\mathrm{J}}^{(\mathrm{I})}[i_{k-1}^{\mathrm{R}}] + \omega[i_k^{\mathrm{R}}] \\ z_{\mathrm{J}}[i_k^{\mathrm{R}}] &= H\, q_{\mathrm{J}}^{(\mathrm{I})}[i_k^{\mathrm{R}}] + \xi[i_k^{\mathrm{R}}], \end{aligned} \tag{4.8}$$

with

$$F = \begin{bmatrix} 1 & \Delta & \frac{1}{2}\Delta^2 \\ 0 & 1 & \Delta \\ 0 & 0 & 1 \end{bmatrix}, \; H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{4.9}$$

where $z_{\mathrm{J}}[i_k^{\mathrm{R}}]$ is a measurement of Anchor J's clock at the reception of packet i_k; where $\xi[\cdot] \sim \mathcal{N}(0, \varsigma^2)$ is additive measurement noise, which corrupts reception timestamps; and

where $\omega[\cdot] \sim \mathcal{N}(0, \Sigma)$ is the discretized process noise driving the system and has covariance

$$\Sigma = \sigma^2 \begin{bmatrix} \frac{1}{20}\Delta^5 & \frac{1}{8}\Delta^4 & \frac{1}{6}\Delta^3 \\ \frac{1}{8}\Delta^4 & \frac{1}{3}\Delta^3 & \frac{1}{2}\Delta^2 \\ \frac{1}{6}\Delta^3 & \frac{1}{2}\Delta^2 & \Delta \end{bmatrix}. \tag{4.10}$$

Modeling the inter-clock behavior in continuous-time and discretizing upon packet reception allows the synchronization algorithm to implicitly and correctly account for packet loss; for variation in hardware clock rates (e.g. due to temperature changes); and for varying transmission periods.

Since the inter-clock behavior has been modeled using a linear system with the assumption of white Gaussian noise, Anchor J implements a discrete-time Kalman filter to track its hardware clock relative to the hardware clock of Anchor I using the standard equations for an autonomous linear system with Gaussian noises:

Prediction Step:

$$\begin{aligned} q^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k] &= F(\Delta)\; q^{(\mathrm{I})}_{\mathrm{J,m}}[i^{\mathtt{R}}_{k-1}] \\ P^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k] &= F(\Delta)\; P^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_{k-1}]\; F^{\intercal}(\Delta) + \Sigma(\Delta) \end{aligned} \tag{4.11}$$

Measurement Update:

$$\begin{aligned} K^{(\mathrm{I})}_{\mathrm{J}}[i^{\mathtt{R}}_k] &= P^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k]\; H^{\intercal} \left(H\; P^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k]\; H^{\intercal} + \varsigma^2\right)^{-1} \\ P^{(\mathrm{I})}_{\mathrm{J,m}}[i^{\mathtt{R}}_k] &= (I^{3\times 3} - K^{(\mathrm{I})}_{\mathrm{J}}[i^{\mathtt{R}}_k]\; H) P^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k] \\ q^{(\mathrm{I})}_{\mathrm{J,m}}[i^{\mathtt{R}}_k] &= q^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k] + K^{(\mathrm{I})}_{\mathrm{J}}[i^{\mathtt{R}}_k] \left(z_{\mathrm{J}}[i^{\mathtt{R}}_k] - H\; q^{(\mathrm{I})}_{\mathrm{J,p}}[i^{\mathtt{R}}_k]\right). \end{aligned} \tag{4.12}$$

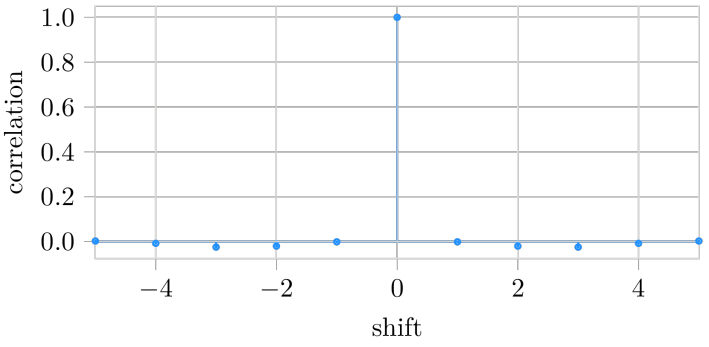
Note that the dependence of the discrete-time model on the sampling period Δ allows the Kalman filter to correctly handle packets arriving at a non-constant rate. The reader is referred to, for example, [49] for further background on the Kalman filter.

4.3 Tuning for the experimental system

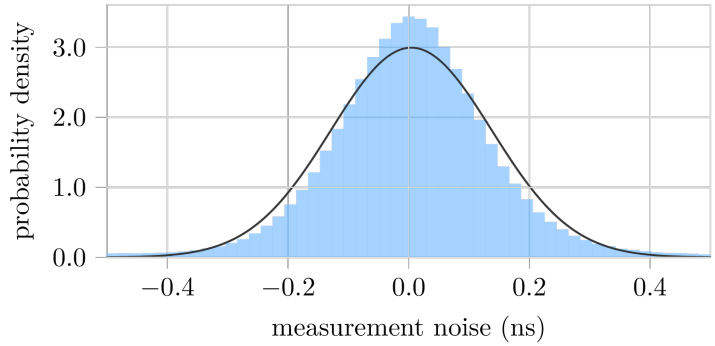
Since the behavior of a Kalman filter is determined by the ratio of the process and measurement noise, tuning the filter for a real system requires determining the standard deviation of the process noise σ, and the standard deviation of the measurement noise ς.

This was achieved experimentally by mounting a pair of anchors in positions representative of the localization system's setup (5 m distance between anchors, with anchors mounted approximately 15 cm from the floor or walls), and instructing one anchor of the pair to transmit at random intervals, while the other anchor recorded receptions. Measurements were then used to update the Kalman filter as in (4.12). Since it is the ratio of σ and ς that determines the Kalman filter's behavior, σ was fixed at 1, and ς was adjusted such that the resulting measurement noise was maximally white. Finally, from the standard deviation of the resulting white measurement noise, the real value of ς could be determined and the real value of σ derived.

On the experimental platform, this tuning resulted in a measurement noise with standard deviation $\varsigma = 0.13\text{ns}$ (equivalent to a distance measurement error of 40mm), as shown in Fig. 4.2, and a process noise with standard deviation of $\sigma = 51\text{ns}\,\text{s}^{-3}$.



(a) Autocorrelation of the resulting measurement noise



(b) Distribution of the resulting measurement noise

Figure 4.2: Anchor I tracks its clock relative to the clock of Anchor J using a Kalman filter, based on the assumption of third-order random-walk clock dynamics. The Kalman filter's noise characteristics were tuned to maximize the whiteness of the measurement noise after filtering, resulting in measurement noise with a standard deviation of 0.13ns (equivalent to a distance measurement error of 40mm).

5

TDMA Transmission Scheduling

Having discussed how pairs of anchors can synchronize their clocks, this chapter will discuss how anchors can synchronize to a network logical clock and coordinate their transmissions according to a predefined time-division multiple-access (TDMA) schedule.

In order to coordinate transmissions according to a TDMA schedule and avoid packet collisions, the clocks of every UWB module in the network must be synchronized to a common timescale. Synchronization in the proposed network architecture follows four distinct phases:

1. Upon startup, an anchor listens for incoming packets. If no packets are received within a specified time-interval, the anchor will begin transmitting. If packets are received, the anchor first synchronizes individually to each transmitting anchor, as described in Chapter 4.
2. Once the receiving anchor is synchronized with each transmitting anchor, it then synchronizes to the network's consensus-based logical clock, as described in Section 5.1.
3. Once synchronized to the network's logical clock, the receiving anchor begins transmitting in accordance with the network's TDMA schedule. Other anchors in the network then begin synchronizing to the newly transmitting anchor and incorporating its clock rate into the network's consensus-based logical clock.
4. Finally, the network synchronization can be improved by each anchor sharing the times at which it has received packets from other anchors, allowing the packet propagation time between pairs of anchors to be measured and accounted for in the synchronization, as discussed in Section 5.2.

After the network is synchronized and all anchors are transmitting in accordance with the TDMA schedule, anchors are able to self-localize within the network (Chapter 6), and robots are able to localize themselves based on TDOA measurements (Chapter 7).

5.1 Network clock synchronization

For purposes of example and without loss of generality, consider a network of four anchors labeled A, B, C and D, and consider the case in which anchors A, B and C are

synchronized and transmitting, and Anchor D wishes to begin transmitting, needing first to synchronize itself to the network. The first step in the synchronization process is Anchor D synchronizing individually to each anchor in $\mathcal{A}_{\mathrm{D}}$ using the method discussed in Chapter 4.

After synchronizing to each anchor individually, Anchor D is now able to synchronize to the network's logical clock. This is accomplished using a consensus-based approach and in a manner similar to the gradient clock synchronization algorithm presented in [25].

Each anchor $\mathrm{I} \in \mathcal{A}$ models its relationship to the network's logical clock (denoted $\mathbf{S}(t)$) as an affine function with parameters ϕ_{I} and θ_{I}, such that at a given time instant t, the network's logical clock can be expressed as a function of the anchor's hardware clock as

$$\mathbf{S}(t) = \phi_{\mathrm{I}}(h_{\mathrm{I}}(t) + \theta_{\mathrm{I}}). \tag{5.1}$$

Synchronization with the network's logical clock at time t is therefore achieved by calculating an appropriate ϕ_{I} and θ_{I}, herein referred to as synchronization parameters.

Considering the case of Anchor D synchronizing to the network's logical clock, let

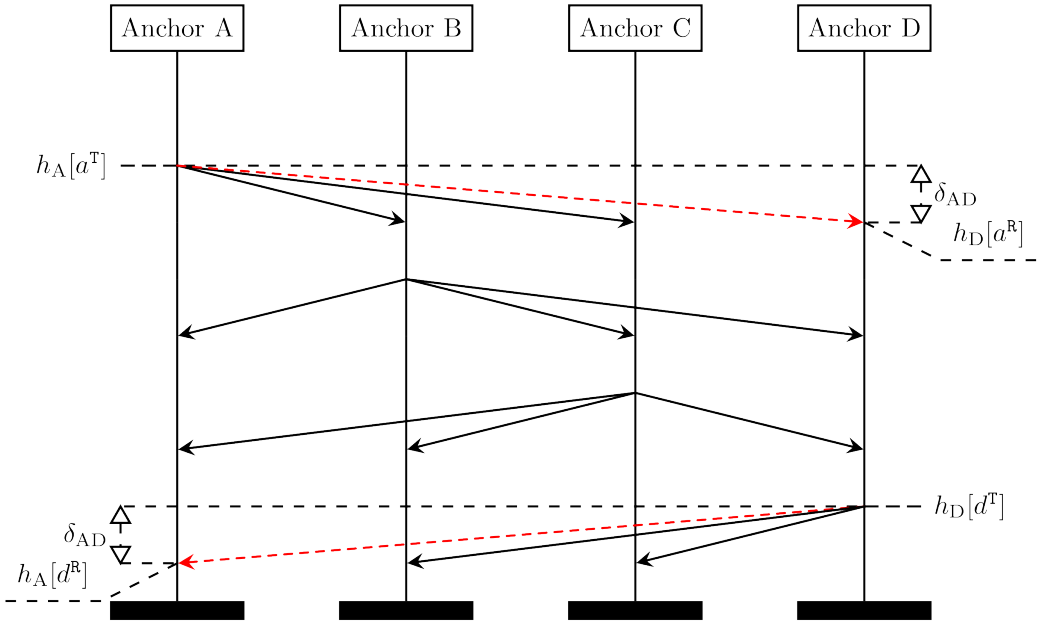


Figure 5.1: After synchronizing to the network's logical clock, anchors are able to transmit in accordance with a known TDMA schedule. Propagation delays between anchors are made observable by each anchor communicating packet reception times. The dashed red transmissions exemplify this: by Anchor D communicating the reception timestamp $h_{\mathrm{D}}[a^{\mathrm{R}}]$ and transmission timestamp $h_{\mathrm{D}}[d^{\mathrm{T}}]$, Anchor A is able to compute δ_{AD}. By knowing the propagation delays between anchors, network synchronization can be improved and the distance between anchors can be estimated, allowing for network self-localization.

$h_{\mathrm{D}}[p^{\mathrm{R}}]$ be the timestamp of the most recently received packet p from any other anchor. Upon reception of p, Anchor D predicts its synchronization to each anchor $\mathrm{I} \in \mathcal{A}_{\mathrm{D}}$ forward to the time of the most recent reception event $h_{\mathrm{D}}[p^{\mathrm{R}}]$. Referring to (4.8), this is accomplished by solving

$$h_{\mathrm{D}}[p^{\mathrm{R}}] = h_{\mathrm{D}}[i^{\mathrm{R}}] + \dot{h}_{\mathrm{D}}^{(\mathrm{I})}[i^{\mathrm{R}}]\Delta_{\mathrm{I}} + \tfrac{1}{2}\ddot{h}_{\mathrm{D}}^{(\mathrm{I})}[i^{\mathrm{R}}]\Delta_{\mathrm{I}}^2 \tag{5.2}$$

for Δ_{I}, the time progression of Anchor I's clock corresponding to a progression of Anchor D's clock by $h_{\mathrm{D}}[p^{\mathrm{R}}] - h_{\mathrm{D}}[i^{\mathrm{R}}]$. Finally Anchor D computes

$$\dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathrm{R}}] = \dot{h}_{\mathrm{D}}^{(\mathrm{I})}[i^{\mathrm{R}}] + \Delta_{\mathrm{I}}\ddot{h}_{\mathrm{D}}^{(\mathrm{I})}[i^{\mathrm{R}}], \tag{5.3}$$

to arrive at the relative rate of the two clocks at the current time.

Having predicted the synchronization of its neighbors forward using the clock model of Chapter 4, Anchor D can update its synchronization parameters using a consensus-based approach. For purposes of consensus, the synchronization parameters of an arbitrary Anchor I must be converted into Anchor D's hardware clock, firstly by equating the rates

$$\begin{aligned}
\phi_{\mathrm{D}}(h_{\mathrm{D}}[p^{\mathrm{R}}] + \theta_{\mathrm{D}} - \delta_{\mathrm{ID}}) &= \phi_{\mathrm{I}}(h_{\mathrm{I}}[p^{\mathrm{R}}] + \theta_{\mathrm{I}}) \\
\tfrac{d}{dh_{\mathrm{I}}}\phi_{\mathrm{D}}(h_{\mathrm{D}}[p^{\mathrm{R}}] + \theta_{\mathrm{D}} - \delta_{\mathrm{ID}}) &= \tfrac{d}{dh_{\mathrm{I}}}\phi_{\mathrm{I}}(h_{\mathrm{I}}[p^{\mathrm{R}}] + \theta_{\mathrm{I}}) \\
\phi_{\mathrm{D}}\tfrac{\mathrm{d}h_{\mathrm{D}}}{\mathrm{d}h_{\mathrm{I}}}[p^{\mathrm{R}}] &= \phi_{\mathrm{I}}\tfrac{\mathrm{d}h_{\mathrm{I}}}{\mathrm{d}h_{\mathrm{I}}}[p^{\mathrm{R}}] \\
\phi_{\mathrm{D}}\dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathrm{R}}] &= \phi_{\mathrm{I}} \\
\phi_{\mathrm{D}} &= \frac{\phi_{\mathrm{I}}}{\dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathrm{R}}]};
\end{aligned} \tag{5.4}$$

and then by equating the offsets

$$\begin{aligned}
\phi_{\mathrm{D}}(h_{\mathrm{D}}[p^{\mathrm{R}}] + \theta_{\mathrm{D}} - \delta_{\mathrm{ID}}) &= \phi_{\mathrm{I}}(h_{\mathrm{I}}[p^{\mathrm{R}}] + \theta_{\mathrm{I}}) \\
h_{\mathrm{D}}[p^{\mathrm{R}}] + \theta_{\mathrm{D}} - \delta_{\mathrm{ID}} &= \frac{\phi_{\mathrm{I}}}{\phi_{\mathrm{D}}}(h_{\mathrm{I}}[p^{\mathrm{R}}] + \theta_{\mathrm{I}}) \\
\theta_{\mathrm{D}} &= \dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathrm{R}}](h_{\mathrm{I}}[p^{\mathrm{R}}] + \theta_{\mathrm{I}}) - h_{\mathrm{D}}[p^{\mathrm{R}}] + \delta_{\mathrm{ID}}.
\end{aligned} \tag{5.5}$$

Note that although this synchronization compensates for the propagation time δ_{ID} between anchors, this value is not observable until Anchor D has begun transmitting. Due to the relatively small magnitude of this value in comparison to the network's transmission period (nanoseconds compared with milliseconds), Anchor D computes an initial synchronization under the assumption that $\delta_{\mathrm{ID}} = 0$ for all $\mathrm{I} \in \mathcal{A}_{\mathrm{D}}$, such that it can begin transmitting, thus rendering the true value of δ_{ID} observable.

Consensus is achieved by averaging the synchronization parameters of all anchors

$$\phi_{\mathrm{D}}^{*} = \tfrac{1}{|\mathcal{A}_{\mathrm{D}}|+1}\left(\phi_{\mathrm{D}} + \sum_{\mathrm{I}\in\mathcal{A}_{\mathrm{D}}} \frac{\phi_{\mathrm{I}}}{\dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathtt{R}}]}\right), \text{ and} \tag{5.6}$$

$$\theta_{\mathrm{D}}^{*} = \tfrac{1}{|\mathcal{A}_{\mathrm{D}}|+1}\left(\theta_{\mathrm{D}} + \sum_{\mathrm{I}\in\mathcal{A}_{\mathrm{D}}} \left(\dot{h}_{\mathrm{D}}^{(\mathrm{I})}[p^{\mathtt{R}}](h_{\mathrm{I}}[p^{\mathtt{R}}] + \theta_{\mathrm{I}}) - h_{\mathrm{D}}[p^{\mathtt{R}}] + \delta_{\mathrm{ID}}\right)\right), \tag{5.7}$$

where ϕ_{I} and θ_{I} are communicated to Anchor D by Anchor I in the contents of each packet. This update rule resembles the gradient clock synchronization update from [25], which was shown to provide accurate clock synchronization between neighboring anchors; to scale to connected networks without requiring complete connectivity; and to converge. Convergence was shown by noting the row-stochasticity of the update matrix.

5.2 Synchronization refinement

After Anchor D has computed an initial synchronization, it is able to begin transmitting in accordance with the predefined TDMA schedule. In this thesis, a round-robin TDMA scheme is implemented, in which anchors are allocated a predefined time-slice for transmission of width 1.25 ms.

After Anchor D begins transmitting, propagation times between each anchor become observable if transmission and reception timestamps are shared in the contents of each packet. Consider the red, dashed transmissions shown in Fig. 5.1: by Anchor D communicating the reception time $h_{\mathrm{D}}[a^{\mathtt{R}}]$ and transmission time $h_{\mathrm{D}}[d^{\mathtt{T}}]$ to Anchor A, Anchor A is able to estimate the propagation time δ_{AD} from a single pair of measurements as

$$2\delta_{\mathrm{AD}} = \left(h_{\mathrm{A}}[d^{\mathtt{R}}] - h_{\mathrm{A}}[a^{\mathtt{T}}]\right) - \tfrac{1}{2}\left(\dot{h}_{\mathrm{A}}^{(\mathrm{D})}[d^{\mathtt{R}}] + \dot{h}_{\mathrm{A}}^{(\mathrm{D})}[a^{\mathtt{T}}]\right)\left(h_{\mathrm{D}}[d^{\mathtt{T}}] - h_{\mathrm{D}}[a^{\mathtt{R}}]\right), \tag{5.8}$$

where the relative clock rate $\dot{h}_{\mathrm{A}}^{(\mathrm{D})}(t)$ is tracked by Anchor A as discussed in Chapter 4. After computing the propagation delay to each anchor, Anchor A incorporates these delays into the network synchronization (Section 5.1). Since anchors are known to be stationary, measurements of propagation delay can be low-pass filtered to reduce noise.

On the experimental system, the performance of the network synchronization is judged by Anchor A estimating the reception time of the next packet based on its current synchronization parameters and the known TDMA schedule. Upon arrival of the packet, Anchor A computes the error between the actual and the expected reception time, which was found to be normally distributed with a mean of 0.77 ps (0.23 mm) and standard deviation of 50 ps (15 mm). This is shown in Fig. 5.2.

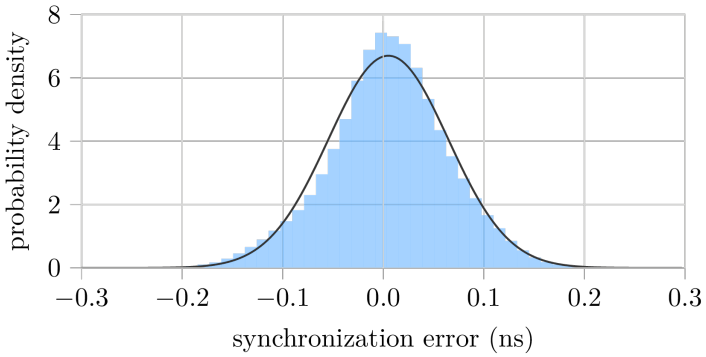


Figure 5.2: Anchor A estimates the reception time of the next packet, based on its synchronization to the network's logical clock. Upon reception of the packet, Anchor A computes the error between actual and estimated arrival time, which is normally distributed with a mean of 0.77 ps (0.23 mm) and standard deviation of 50 ps (15 mm). This provides a metric to assess the quality of network synchronization.

6

Network Self-Localization

As shown in Chapter 5, by synchronizing clocks and sharing reception and transmission timestamps, each anchor can measure the propagation delay to every other anchor with which it can directly communicate. This chapter describes how, having measured these propagation delays, anchors can localize themselves within the anchor network, and how this localization can be refined in a distributed manner to minimize the global localization error. The ability of the anchor network to self-localize enables it to be setup quickly and easily, since anchor positions are computed as a by-product of operation and do not need to be manually measured or otherwise known beforehand.

6.1 Relating distance to propagation delay

As shown in Section 2.4, the measurement of propagation delay between two UWB modules is affected by both noise and by a systematic bias. This bias is a function of the environment and of the relative pose of the antenna pair, as was previously shown in Fig. 2.17 and Fig. 2.18. If the bias was known, the distance between UWB modules I and J could be estimated as

$$\mathrm{d_{IJ}} = c\,\delta_{\mathrm{IJ}} - \beta_{\mathrm{IJ}}, \tag{6.1}$$

where β_{IJ} is the known measurement bias. Unfortunately, this bias is not observable, given the measurements available from the localization network [50], since a change in bias is indistinguishable from a change in distance between the anchors. Therefore, a simplifying assumption is that

$$\mathrm{d_{IJ}} = c\,\delta_{\mathrm{IJ}} - \bar{\beta}, \text{ for all anchors I, J} \in \mathcal{A} \tag{6.2}$$

where c is the speed of light and $\bar{\beta}$ is the pre-calibrated average bias encountered when processing and timestamping the reception of a packet. Despite accounting for the average measurement bias, measurement biases remain within the range of ±300mm, as can be seen in Table 6.1 at the end of this chapter.

6.2 Estimating anchor positions from inter-anchor distances

In order for anchors to localize themselves within the localization network, a coordinate system must be defined. This can be achieved by manually setting the positions of four anchors, or by constraining their placement such that their locations may be inferred from the available localization data.

In this thesis, the anchor coordinate system is defined to have Anchor A positioned at the origin, Anchor B along the positive x-axis, Anchor C in the positive y-direction, and Anchor D in the positive z-direction. All other anchors can be placed arbitrarily within the space. These assumptions define the coordinate system of the network and allow the positions of anchors in the network to be determined by minimizing the global positioning error

$$\mathcal{J} \coloneqq \sum_{\mathrm{I}\in\mathcal{A}} \sum_{\mathrm{J}\in\mathcal{A}_\mathrm{I}} \left(\mathrm{d}_{\mathrm{IJ}} - \|\mathbf{p}_\mathrm{I} - \mathbf{p}_\mathrm{J}\|_2\right)^2, \tag{6.3}$$

where $\mathbf{p}_\mathrm{I} \coloneqq (x_\mathrm{I}, y_\mathrm{I}, z_\mathrm{I})$ is the position of Anchor I. Global positioning error is minimized using a distributed and iterative approach, rather than a centralized global optimization. This approach is computationally lightweight and scales well to large anchor networks.

Self-localization begins by initializing the position of anchors A, B and C as

$$(x_\mathrm{A}, y_\mathrm{A}, z_\mathrm{A}) = (0, 0, 0) \tag{6.4}$$

$$(x_\mathrm{B}, y_\mathrm{B}, z_\mathrm{B}) = (\mathrm{d}_{\mathrm{BA}}, 0, 0) \tag{6.5}$$

$$(x_\mathrm{C}, y_\mathrm{C}, z_\mathrm{C}) = \left(\frac{x_\mathrm{B}^2 + \mathrm{d}_{\mathrm{CA}}^2 - \mathrm{d}_{\mathrm{CB}}^2}{2x_\mathrm{B}}, \sqrt{\mathrm{d}_{\mathrm{CA}}^2 - x_\mathrm{C}^2}, 0\right). \tag{6.6}$$

The position of each other anchor can then be initialized with respect to these anchors as

$$x_\mathrm{I} = \frac{\mathrm{d}_{\mathrm{IA}}^2 - \mathrm{d}_{\mathrm{IB}}^2 + x_\mathrm{B}^2}{2x_\mathrm{B}} \tag{6.7}$$

$$y_\mathrm{I} = \frac{\mathrm{d}_{\mathrm{IA}}^2 - \mathrm{d}_{\mathrm{IC}}^2 + x_\mathrm{C}^2 + y_\mathrm{C}^2}{2y_\mathrm{C}} - \frac{x_\mathrm{C} x_\mathrm{I}}{y_\mathrm{C}} \tag{6.8}$$

$$z_\mathrm{I} = \pm\sqrt{\mathrm{d}_{\mathrm{IA}}^2 - x_\mathrm{I}^2 - y_\mathrm{I}^2}, \tag{6.9}$$

noting that Anchor D is constrained to have $z_D > 0$, and where the z-ambiguity of other anchors is resolved by observation of their distance to Anchor D.

After initialization, this position estimate is further refined using distributed gradient descent, whereby each anchor updates its position in an iterative manner by calculating and descending the cost gradients $\frac{\partial \mathcal{J}}{\partial x_\mathrm{I}}$, $\frac{\partial \mathcal{J}}{\partial y_\mathrm{I}}$, and $\frac{\partial \mathcal{J}}{\partial z_\mathrm{I}}$. Each anchor communicates its position periodically to all other anchors in the network, facilitating the minimization of (6.3) in a distributed fashion.

It should be noted that the above method of initialization is dependent on each anchor being able to communicate with (and thus measure distance to) anchors A, B, C and D. Furthermore, since the initial positions are computed in closed-form from the locations of anchors A, B and C, the initialization is both heavily affected by errors in these positions and is sensitive to their placement. This sensitivity is investigated using the Cramér-Rao lower bound in, for example [P.3], [51], [52]. To mitigate both these issues, initialization can be computed based on three neighboring anchors, selected such that the available Fisher information is maximized.

Further improvements on the above algorithm are possible by exploiting known structure in anchor placement. In many situations, anchor placement may be semi-structured; for example, it may be known that a subset of anchors share the same x, y or z coordinate, perhaps being placed along a wall or ceiling; as a further example, it might be known that a subset of anchors lie within a specific half-space. The presented formulation allows anchor positions to be constrained to known coordinates, or for additional penalties to be included in the cost function to model inequality or equality constraints on anchor coordinates or inter-anchor distances.

6.3 Experimental validation

In the experimental setup, eight anchors were placed in an approximately rectangular setup with side-dimensions of approximately $6\text{m} \times 7\text{m} \times 3.5\text{m}$. The ground-truth position of each anchor was measured by hand. The anchor network was reinitialized ten times, and positions resulting from self-localization were recorded. Across all ten trials, the presented synchronization and anchor localization algorithms enabled anchors within the network to self-localize with a position root mean squared error of 97mm, resulting from an x-position error of $-30 \pm 54\text{mm}$, y-position error of $19 \pm 46\text{mm}$, and z-position error of $-34 \pm 45\text{mm}$; and an inter-anchor distance estimation error of $-12 \pm 76\text{mm}$. Note that these errors are within the tolerances of manually measuring the anchor's actual positions, and are far less than the inter-anchor distance measurement bias of $\pm 300\,\text{mm}$. Given the structured placement of anchors it was possible to further constrain the anchor coordinates that are known to be equal. Given the already accurate localization, no additional constraints were imposed on anchor positions, with the exception of Anchor E, which was placed on the floor and thus constrained to have $z = 0$.

Table 6.1 presents a summary of range measurements upon which the self-localization is based, while Table 6.2 presents the results of the self-localization procedure discussed in this section. Comparing measured and estimated distances gives an idea of the influence and variation of measurement bias within the space, and of the potential bias influencing measurements received by a robot.

Table 6.1: Distance measurement and estimation errors relative to Anchor C. Anchor positions were measured by hand and a ground truth Euclidean distance between each pair of anchors was calculated. The network was reinitialized ten times and the results of distance measurement and estimation were recorded. Measurement errors are calculated as the difference between the ground truth distance to Anchor C and the pairwise distance measurement, as derived from the time-of-flight measurement (6.2). Estimation errors are calculated based on the difference between the ground truth distance to Anchor C and the Euclidean distance between the anchors' estimated positions. Errors are shown as mean±standard deviation, with mean and standard deviation computed across the ten trials. Note that the DWM1000 is able to measure distances with a precision of 4.7 mm [11].

ID	Measurement Error (mm)	Estimation Error (mm)
A	-89.08 ± 4.3	-2.48 ± 3.6
B	205.46 ± 4.7	33.66 ± 4.2
C	*	*
D	-34.30 ± 3.7	35.90 ± 4.6
E	-217.87 ± 4.8	-61.07 ± 3.4
F	29.52 ± 7.3	77.92 ± 5.5
G	55.52 ± 5.6	53.42 ± 3.2
H	153.87 ± 6.0	44.27 ± 4.9

Table 6.2: Position errors after network self localization. A network of eight anchors were placed in an approximately rectangular setup with positions measured by hand. The network was reinitialized ten times and the results of self-localization were recorded. Errors were calculated as the difference between the hand-measured position and the estimated position. Errors are shown as mean±standard deviation, with mean and standard deviation computed across the ten trials. Constrained positions are denoted by a *. Note that anchors D, F, G and H were placed on the ceiling, making hand measurement of their absolute positions difficult; this is reflected in their relatively high errors. Since Anchor E was placed on the floor, it was constrained to have $z = 0$.

ID	x error (mm)	y error (mm)	z error (mm)
A	*	*	*
B	54.94 ± 9.79	*	*
C	2.23 ± 0.00	-2.48 ± 3.63	*
D	66.36 ± 8.87	21.51 ± 4.47	99.11 ± 4.13
E	-58.83 ± 3.44	22.06 ± 4.97	*
F	133.68 ± 5.23	-80.94 ± 4.68	25.51 ± 4.65
G	11.75 ± 8.76	-7.05 ± 4.38	119.52 ± 2.46
H	31.14 ± 5.52	-110.87 ± 4.62	35.18 ± 3.33

7

(Multi-)Robot Localization

Having now synchronized and self-localized, the anchor network can support the operation of a theoretically unlimited number of robots within the space. By passively listening to the network traffic (Fig. 5.1) and recording the reception time of measurements, a robot is able to compute the difference in packet time of flight between pairs of anchors. This time difference is proportional to the difference in the robot's distance to the transmitting anchors. By collecting multiple measurements from multiple pairs of anchors, a robot is able to localize itself.

7.1 Robot localization using time-difference-of-arrival measurements

With reference to Fig. 5.1, let $h_{\mathrm{R}}[a^{\mathtt{R}}]$ and $h_{\mathrm{R}}[d^{\mathtt{R}}]$ be the times at which the robot receives the latest packets from anchors A and D respectively. It is assumed that each packet contains both the position of its transmitting anchor and its transmission time expressed in the network's logical clock. This additionally provides the robot with times $\mathbf{S}[a^{\mathtt{T}}]$ and $\mathbf{S}[d^{\mathtt{T}}]$, as well as the positions of both anchors involved in the communication. Letting ϕ_{R} denote the robot's UWB clock rate relative to the anchor network's logical clock rate, and $\boldsymbol{x} \coloneqq (\, x, y, z \,)$ its position within the anchor coordinate system

$$\begin{aligned} \mathbf{S}[a^{\mathtt{T}}] + \delta_{\mathrm{AR}} &= \phi_{\mathrm{R}}(h_{\mathrm{R}}[a^{\mathtt{R}}] + \theta_{\mathrm{R}}), \text{ and} \\ \mathbf{S}[d^{\mathtt{T}}] + \delta_{\mathrm{DR}} &= \phi_{\mathrm{R}}(h_{\mathrm{R}}[d^{\mathtt{R}}] + \theta_{\mathrm{R}}). \end{aligned}$$

Computing the time difference of arrival of the above packets yields

$$\begin{aligned} (\mathbf{S}[d^{\mathtt{T}}] - \mathbf{S}[a^{\mathtt{T}}]) + (\delta_{\mathrm{DR}} - \delta_{\mathrm{AR}}) &= \phi_{\mathrm{R}}(h_{\mathrm{R}}[d^{\mathtt{R}}] - h_{\mathrm{R}}[a^{\mathtt{R}}]) \\ c(\mathbf{S}[d^{\mathtt{T}}] - \mathbf{S}[a^{\mathtt{T}}]) &= c \cdot \phi_{\mathrm{R}}(h_{\mathrm{R}}[d^{\mathtt{R}}] - h_{\mathrm{R}}[a^{\mathtt{R}}]) - (d_{\mathrm{DR}} - d_{\mathrm{AR}}) \\ c(\mathbf{S}[d^{\mathtt{T}}] - \mathbf{S}[a^{\mathtt{T}}]) &= c \cdot \phi_{\mathrm{R}}(h_{\mathrm{R}}[d^{\mathtt{R}}] - h_{\mathrm{R}}[a^{\mathtt{R}}]) + \|\mathbf{p}_{\mathrm{A}} - \boldsymbol{x}\|_2 - \|\mathbf{p}_{\mathrm{D}} - \boldsymbol{x}\|_2, \end{aligned} \tag{7.1}$$

which relate the TDOA measurements to the position of the robot $\boldsymbol{x}$. By gathering multiple such measurements, a robot can localize itself by either solving for its position

directly (using, for example, nonlinear least squares); or can use individual measurements to update a state estimator. In both these cases ϕ_R, the relative rate of the robot's clock to the system clock, must be estimated or tracked (for example by including it as an additional state in the state estimator and tracking its value over time).

7.2 Quadcopter experimental system

For these experiments, a Bitcraze Crazyflie 2.0 nano-quadcopter [1] (shown in Fig. 7.1) was flown within the eight-anchor setup described in Section 6.3 and with anchor positions given in Table 6.2. The presented method of localization is, however, applicable to any number of robots operating within a 3D space spanned by at least four anchors. This minimum of four being required to define the three-dimensional coordinate system. Due to the sensitivity of the time difference of arrival localization method to noise [P.3], it is recommended that robots operate within the convex hull of the anchors.

This section details the quadcopter's control and estimation software, which run on its 168 MHz, ARM Cortex-M4F microprocessor (STM32F405, single-precision floating point unit, 196 kB RAM, 1 MB flash). Localization was enabled by connecting a Decawave DWM1000 UWB radio module to the quadcopter's microprocessor via SPI, allowing it to receive UWB packets sent by the anchors. State estimates were sent at 30 Hz from the robot to a laptop computer for logging purposes only. The laptop did not communicate with the quadcopters, nor was it required for their flight, since trajectory planning, control, localization, and state estimation were all performed onboard.

Figure 7.1: One of the Crazyflie nano-quadcopters used for these experiments. A Decawave DWM1000 UWB radio is mounted on the quadcopter, allowing it to eavesdrop on UWB communications between anchors.

7.2.1 Quadcopter model

The quadcopter is modeled as a rigid body with mass m, whose motion is governed by the Newton-Euler equations (see e.g. [53]). The orientation of the quadcopter with respect to the inertial frame is denoted by the rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$, and recall that the quadcopter's position in the inertial frame is denoted by the vector $\boldsymbol{x} \in \mathbb{R}^3$.

The cumulative thrust produced by the four propellers is denoted f, and is always positive and aligned to the quadcopter's body z-axis. With $\boldsymbol{e}_3 \coloneqq (\,0, 0, 1\,)^\intercal$ the thrust vector in the quadcopter's body frame is therefore $\boldsymbol{f} \coloneqq f\boldsymbol{e}_3$. The magnitude of gravitational acceleration is denoted by g and the vector of gravitational acceleration in the inertial frame by $\boldsymbol{g} \coloneqq -g\boldsymbol{e}_3$. A quadcopter's dynamics are therefore described by

$$m\ddot{\boldsymbol{x}} = \mathbf{R}\,\boldsymbol{f} + m\boldsymbol{g} \tag{7.2}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,[\![\boldsymbol{\omega}\times]\!], \tag{7.3}$$

where $\boldsymbol{\omega} = (\,\omega_1, \omega_2, \omega_3\,)$ is the angular velocity of the quadcopter expressed in its body frame and measured by its rate gyroscope, and where $[\![\boldsymbol{\omega}\times]\!]$ denotes the matrix form of the cross product given by

$$[\![\boldsymbol{\omega}\times]\!] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{7.4}$$

Note that for simplicity and since flight is relatively slow, aerodynamic effects such as body drag and blade flapping are ignored in the above model [54].

The angular acceleration of the quadcopter evolves as a function of its current angular velocity, its inertia, and of the torques generated by its propellors. As these equations are not required for the estimator design, they will not be repeated here and instead the reader is directed to [54] for further information on the modeling of multirotor aerial vehicles.

7.2.2 Quadcopter state estimation

7.2.2.1 Overview To control a quadcopter accurately within the space, an estimate of its position, velocity, orientation and angular velocity is required. The quadcopter's rate gyroscope is assumed to be calibrated and to have zero bias, and gyroscope measurements $\boldsymbol{z}_{\text{gyro}}$ are assumed to be affected by additive, white measurement noise. Akin to [P.4], the rate gyroscope measurements are used directly as the estimate of the quadcopter's angular

velocity, giving

$$\hat{\boldsymbol{\omega}} = \boldsymbol{z}_{\text{gyro}}, \tag{7.5}$$

in which a caret is used to indicate a state estimate, and in which the covariance of $\hat{\boldsymbol{\omega}}$ is equal to the covariance of the gyroscope measurements.

An extended Kalman filter using the attitude-error formulation of [P.4] is used to estimate the quadcopter's position, velocity, and orientation. The estimator's nine-dimensional stochastic state $\boldsymbol{\xi}$ is therefore

$$\boldsymbol{\xi} \coloneqq (\, \boldsymbol{x}, \boldsymbol{v}, \boldsymbol{\delta} \,). \tag{7.6}$$

where $\boldsymbol{x}$ and $\boldsymbol{v}$ are respectively the quadcopter's position and velocity expressed in the inertial coordinate system and $\boldsymbol{\delta}$ is a three-dimensional attitude error representation, which encodes uncertainty about the quadcopter's orientation. The quadcopter's orientation $\mathbf{R}_{\text{ref}}$ is tracked separately. After each prediction step and measurement update of the state estimator, the newly-estimated attitude error $\hat{\boldsymbol{\delta}}$ is used to update the quadcopter's orientation reference as detailed in Section 7.2.2.4, before being reset to zero.

This attitude error formulation allows the quadcopter's attitude to be estimated within a standard Kalman filter framework, without issues of singularities in the attitude representation (e.g. as would be the case for a representation using Euler angles [55]), and without requiring constraints to be placed on the attitude representation (e.g. as would be required for representation of the attitude as a quaternion or rotation matrix). Readers are referred to, for example [56] and [57] for further information on this formulation.

7.2.2.2 Prediction step During the prediction step of the Kalman filter, the estimated states are updated according to

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{v}} \tag{7.7}$$

$$\dot{\hat{\boldsymbol{v}}} = \frac{1}{m}\mathbf{R}_{\text{ref}} \left(\mathbf{I} + [\![\hat{\boldsymbol{\delta}}\times]\!] \right) \boldsymbol{f} + \boldsymbol{g} \tag{7.8}$$

$$\dot{\hat{\boldsymbol{\delta}}} = \left(\mathbf{I} + \tfrac{1}{2}[\![\hat{\boldsymbol{\delta}}\times]\!] \right) \hat{\boldsymbol{\omega}} \tag{7.9}$$

As previously noted, measurements from the quadcopter's angular rate gyroscope are treated as an input in (7.9), with the zero-mean noise on the rate gyroscope encoded as process noise using the standard extended Kalman filter formulation [49]. Note that a first-order approximation is used for the evolution of $\hat{\boldsymbol{\delta}}$ due to the expected low body-rates of the quadcopter and regularity with which attitude resets are performed [58].

By linearizing the above equations around the current estimate $\hat{\boldsymbol{\xi}}$, the Jacobian nec-

essary for the Kalman filter's covariance prediction step can be computed.

After the prediction step has been completed, the attitude reset step is performed (Section 7.2.2.4) in order to update the estimate of the quadcopter's orientation.

7.2.2.3 UWB measurement update In order to update the quadcopter's state estimate using TDOA measurements (7.1), the behavior of the quadcopter's UWB clock with respect to the UWB network's logical clock must be tracked. Assuming that the UWB clock behavior is uncorrelated with the quadcopter's other states and that the quadcopter's displacement between packet receptions is negligible, this can be tracked with a separate Kalman filter as discussed in Chapter 4.

Both the estimated UWB clock behavior and the estimate of the quadcopter's physical states are updated using TDOA measurements by linearizing (7.1) about the quadcopter's current state and updating the extended Kalman filter as per the standard framework [49]. Due to the unobservable measurement bias in each measurement, the measurement noise standard deviation was set to 200 mm rather than 40 mm (as suggested by Fig. 4.2).

Measurement outliers were detected and rejected by tracking the mean and standard deviation of the measurement history from each anchor individually, and rejecting measurements more than three standard deviations from the mean.

After measurement updates have been completed, the attitude reset step is performed in order to update the estimate of the quadcopter's orientation.

7.2.2.4 Attitude error reset After both prediction and measurement update steps, the quadcopter's estimated attitude $\mathbf{R}_{\text{ref}}$ is updated according to [57] as

$$\mathbf{R}_{\text{ref}} \leftarrow \mathbf{R}_{\text{ref}} \left(\mathbf{I} + [\![\hat{\boldsymbol{\delta}} \times]\!] \right), \tag{7.10}$$

and the covariance of the attitude error as

$$\text{Var}[\hat{\boldsymbol{\delta}}] \leftarrow \left(\mathbf{I} - \tfrac{1}{2} [\![\hat{\boldsymbol{\delta}} \times]\!] \right) \text{Var}[\hat{\boldsymbol{\delta}}] \left(\mathbf{I} - \tfrac{1}{2} [\![\hat{\boldsymbol{\delta}} \times]\!] \right)^{\mathsf{T}}. \tag{7.11}$$

The covariance of the attitude error thus evolves over time to encode the uncertainty about the quadcopter's attitude. Finally, the estimated attitude error $\hat{\boldsymbol{\delta}}$ is reset to zero.

7.2.3 Quadcopter control

Since quadcopter control is not a focus of this thesis and since control performance is not critical to the experimental results presented in this chapter, this section gives a brief overview of the control implementation, and the reader is directed to the cited papers for further information.

The quadcopter is controlled using the cascaded control approach from [59]. At the highest level, the quadcopter is supplied with a reference position trajectory. In the case

of these experiments, this reference trajectory was generated onboard. Tracking this reference trajectory is achieved using the high-level control strategy from [59], which generates a desired thrust vector for the attitude controller.

Since a quadcopter can only produce thrust along its body's z axis, the direction of the acceleration vector computed by the high-level controller defines the quadcopter's desired pitch and roll, with the quadcopter's yaw—the rotation around its body z axis—being a free variable. The attitude controller generates a desired body rate through a comparison of the desired attitude with the estimated attitude. The work of [60] was adapted for this purpose.

At the lowest level, the desired body rates are mapped to the desired body torque through the quadcopter's known inertia [A.3] and finally, this desired body torque along with the desired cumulative thrust can be mapped to the desired motor forces [59].

7.3 Experimental results

7.3.1 Single Quadcopter

To verify the performance of the system, a quadcopter was commanded to fly a circle of radius $r = 1\,\mathrm{m}$ with a period of $T = 4\,\mathrm{s}$ (corresponding to the quadcopter flying with a linear velocity of $\frac{2\pi r}{T} \approx 1.57\,\mathrm{m\,s^{-1}}$). The quadcopter flew the circle 20 times in succession before landing. The error between the quadcopter's estimated and actual position (as measured by a Vicon motion capture system with sub-millimeter positioning accuracy) was calculated for each location on the circle (parameterized using the angle θ) and error statistics were calculated over the 20 runs. In order to investigate the variation of these statistics over time, this experiment was repeated on a different day, with a reinitialized localization network. Fig. 7.2 shows the estimation error mean and standard deviation (shaded region) for both trials. This analysis shows that estimation error appears to contain position-dependent effects on the order of $\pm 100\,\mathrm{mm}$, and that these effects are relatively constant in the short-term (as indicated by the standard deviation) as well as over the longer-term (as indicated by a comparison of the two trials).

Both the magnitude of these effects and their position dependence corroborate the observations of [P.3] and [P.4], and can be explained by UWB measurement biases being affected by the quadcopter's position and orientation relative to each anchor's antenna. Variation in these biases observed between trials is likely due to slight variation in the placement and orientation of the anchors, and variation in their calculated positions (on the order of $\pm 40\,\mathrm{mm}$ per axis for each anchor, see Section 6.3). The offset in the z-axis can be explained by accelerometer biases and discrepancies between expected and actual thrust, whose effects on the state estimate are amplified by the artificially-high TDOA measurement noise.

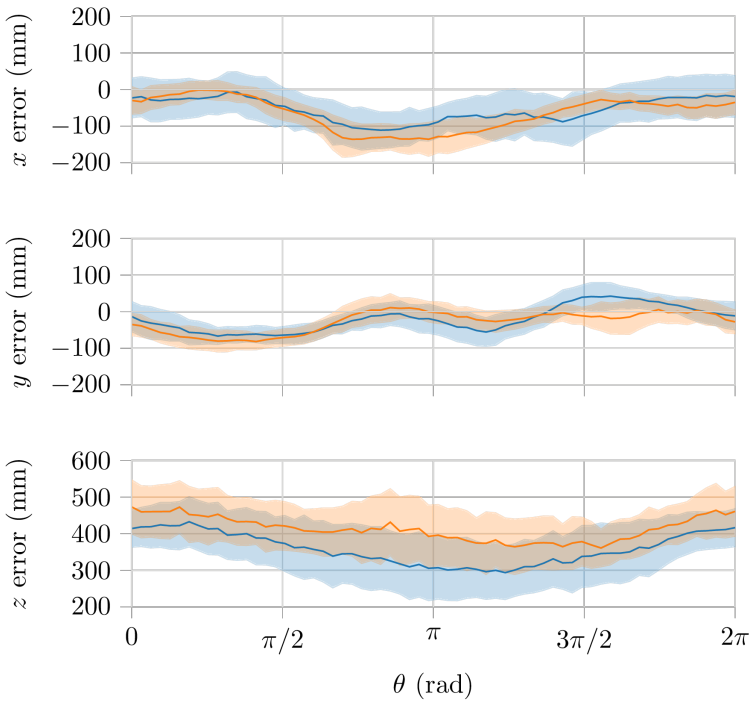
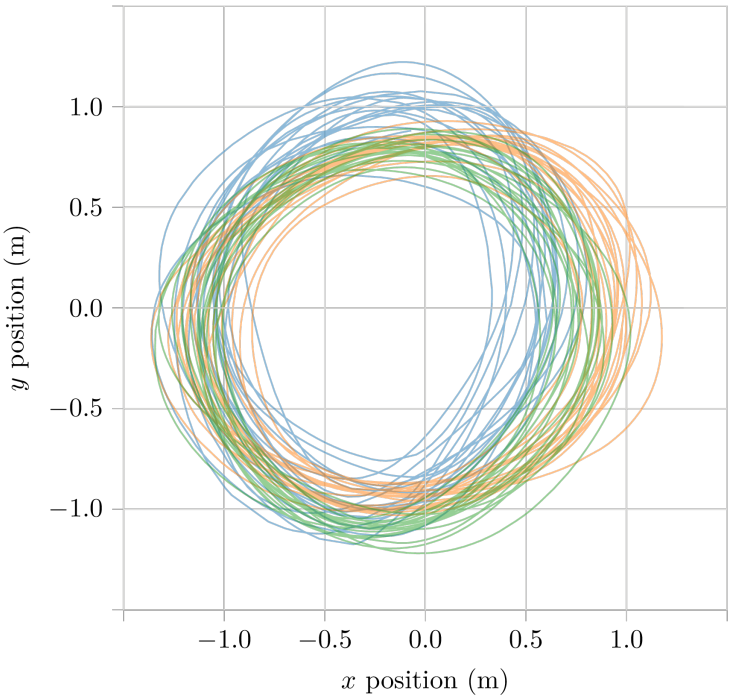


Figure 7.2: This figure shows two trials of an experiment in which a quadcopter flew a circle 20 times before landing. Trials were performed a day apart, and the anchor network reset before each trial. The error between the actual and estimated x, y and z positions were calculated as function of the quadcopter's location on the circle. These plots show the mean and standard deviation (shaded region) of these errors. Note that errors vary with the quadcopter's position, and are repeatable. This variation as a function of position is likely caused by UWB measurement biases varying throughout the space.

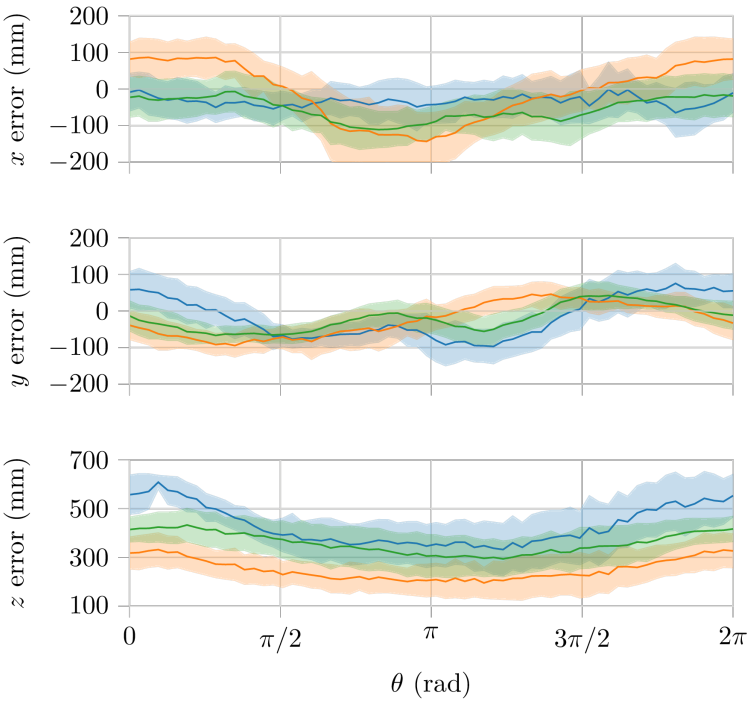
7.3.2 Multiple Quadcopters

Since quadcopters localize themselves based only on received UWB signals the network supports the simultaneous operation of many quadcopters. To demonstrate this, the previous experiment was repeated using three quadcopters flying the same circle simultaneously. Since the anchor network maintains a network time, and since each quadcopter knows its ID, quadcopters were able to compute their reference position as a function of network time and thus avoid collisions with other quadcopters. A video of this experiment can be viewed at `http://mikehamer.info/uwb-system`.

As in the previous experiment, errors between each quadcopter's actual and estimated position were computed as a function of its position on the circle. The mean and standard deviation of these errors are shown in Fig. 7.3. As previously discussed, these estimation errors exhibit a dependence on the quadcopter's position; furthermore, similarities can be seen between the quadcopters.



(a) Three quadcopters simultaneously fly a circle with a radius of 1 m, centered at (0, 0)



(b) Estimation errors are repeatable and position dependent

Figure 7.3: To demonstrate the system's ability to support the operation of multiple robots, three quadcopters were commanded to simultaneously fly a circle. Estimation errors were calculated as a function of each quadcopter's position on the circle. These errors show a dependence on position, and display similarities between quadcopters.

8

ALOHA Localization

The previous chapters discussed how anchors can coordinate their transmissions to a TDMA schedule by first synchronizing to each other, and then synchronizing to a network logical clock, and how transmissions made according to this TDMA schedule can be used for TDOA localization. Having anchors synchronize to a TDMA schedule is advantageous since interference between transmissions is avoided, thus allowing network throughput to be maximized. However, increases in transmission rate require increased processing on receivers while yielding diminishing returns in terms of localization accuracy. In the case of a TDMA schedule, increased throughput and the avoidance of packet collisions comes at the cost of increased implementation complexity, and decreased system flexibility.

A method of anchor self-localization using pseudo two-way ranging was proposed in Chapter 6, and a method of robot localization using TDOA measurements in Chapter 7. Note that both these methods require only synchronization of the timestamps involved in each calculation and as such only pairwise synchronization of the anchors is required. The additional algorithmic complexity of anchors maintaining synchronization to a network logical clock and coordinating to a TDMA schedule is therefore not inherently required for network self-localization or for robot localization.

A potential direction for future research is *ALOHA*-style localization, in which anchors randomly transmit packets rather than adhering to a TDMA schedule, similar to the *ALOHA* networking protocol [61]. Such an approach would greatly simplify the anchor network's logic, and increase robustness to packet interference. Furthermore, a protocol relying on random transmissions allows the system to easily handle the dynamic addition or removal of anchors, and thus to easily scale to large networks and to partially connected networks with overlapping areas of coverage. In contrast, if a TDMA-based schedule were employed, the negotiation and coordination of TDMA slots would add significant algorithmic complexity.

8.1 Random transmission scheme

Consider the case shown in Fig. 8.1, in which an anchor transmits packets of length A at random intervals. The random delay between packet transmissions is a random variable D distributed uniformly on $[L, H]$:

$$p_D(d) = \text{UNIFORM}_{[L,H]}(d) = \begin{cases} \frac{1}{H-L} & L \leq d \leq H \\ 0 & \text{otherwise} \end{cases} \tag{8.1}$$

where L is the minimum delay between packet transmissions, and where $H > L$ is the maximum delay between packet transmissions. Note that the delay is measured from the beginning of the transmission, and as such includes the time required to transmit the packet. It follows that $L > A$. By tuning H and L, the average transmission rate of each anchor can be adjusted, which in turn affects the probability of packets from different anchors colliding. The average throughput of the network can therefore be controlled by adjusting these parameters, as discussed in the next sections.

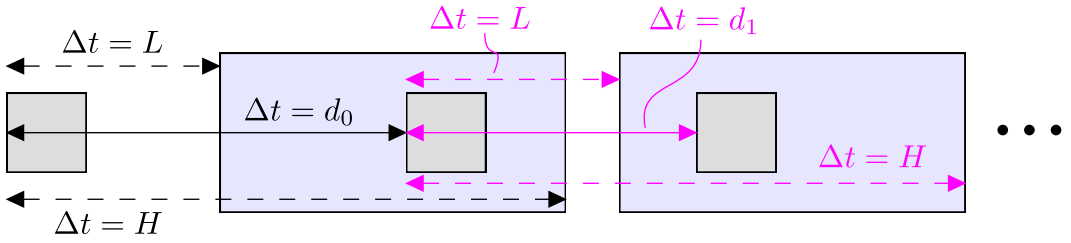


Figure 8.1: This figure depicts packet transmissions (gray squares) from a single anchor. Packets are transmitted at least L seconds and at most H seconds after the transmission of the previous packet begins. This range is shaded in blue. The transmission delay between packets is selected uniformly from this range.

8.1.1 Distribution of anchor time

Each anchor transmits a series of packets as shown in Fig. 8.2. Packet transmissions are separated by delays d, which are sampled uniformly from $[L, H]$. If a second anchor attempts to transmit a packet at time t, the probability of a packet collision is much higher if this transmission occurs during a shorter delay (e.g. d_2), than if it occurs during a longer delay (e.g. d_3). It is therefore important to model how an anchor's time is distributed, and how this distribution informs the probability of a packet collision occurring.

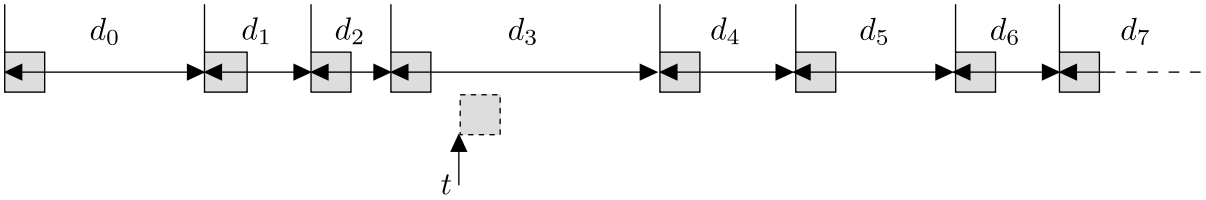


Figure 8.2: A series of packets is transmitted, with transmissions separated by delays d. A second anchor transmits at time t. The probability of a packet collision is higher if the transmission occurs during a shorter delay (e.g. d_2) than during a longer delay (e.g. d_3).

Let $[d_1, \ldots, d_M]$ be a sequence of M independently sampled delays (e.g. as shown in Fig. 8.2), and let

$$T = \sum_{m=1}^{M} d_m \tag{8.2}$$

be the duration of the sequence. By the law of large numbers

$$\frac{\mathrm{E}[T]}{M} \to \mathrm{E}[D] = \frac{H+L}{2} \tag{8.3}$$

as $M \to \infty$, where $\mathrm{E}[\cdot]$ denotes expectation.

Let a packet from a second anchor be transmitted at time t, and let t be uniformly distributed on $[0, T]$. Let d_t denote the first anchor's delay at time t. Then for $L \leq d \leq H$

$$\Pr(d \leq d_t \leq d + \mathrm{d}d) \to \frac{\frac{\mathrm{d}d}{H-L} \cdot M \cdot d}{\frac{H+L}{2} \cdot M} \tag{8.4}$$

$$= \frac{2d}{(H-L)(H+L)}\mathrm{d}d \tag{8.5}$$

as $M \to \infty$, where $\Pr(\cdot)$ is used to denote probability. The numerator of (8.4) is the expected amount of time the anchor spends in delays within the range $d \leq d_t \leq d + \mathrm{d}d$, and the denominator is the expected duration of the delay sequence. It follows that for an infinite length sequence

$$p_{d_t}(d) = \begin{cases} \frac{2d}{(H-L)(H+L)} & L \leq d \leq H \\ 0 & \text{otherwise} \end{cases} \tag{8.6}$$

is the probability density function of an anchor finding itself in a delay of duration d.

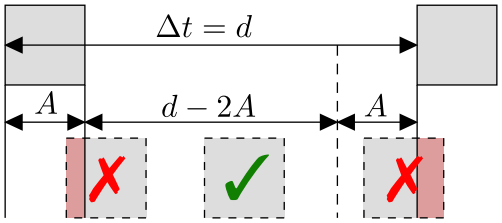


Figure 8.3: This figure shows transmissions from two anchors, and depicts the range of possible transmission times that will not result in a collision between the two anchors' packets. This figure shows that within a cycle of length d, there is a period of length $2A$ within which a second anchor cannot transmit without causing a collision.

8.1.2 Probability of packet collision

Let a first anchor be in a delay of duration d, which begins at time t_0. Let a second anchor begin transmission at time t. As depicted in Fig. 8.3, since the first A seconds of a delay cycle are spent transmitting and since a subsequent cycle begins with a transmission, a collision will result if $t \leq t_0 + A$ or $t \geq t_0 + d - A$.

It follows that the conditional probability of a collision, given that the first anchor is in a cycle of delay d is

$$\Pr(\text{collision} \,|\, d) = \frac{2A}{d}. \tag{8.7}$$

The probability of a packet collision Pr(collision) can be computed by marginalizing out the delay length d via p_{d_t}:

$$\begin{aligned} \Pr(\text{collision}) &= \int_{-\infty}^{\infty} \Pr(\text{collision} \,|\, d) p_{d_t}(d) \, \mathrm{d}d \\ &= \int_{L}^{H} \frac{2A}{d} \frac{2d}{(H-L)(H+L)} \, \mathrm{d}d \\ &= \frac{4A}{H+L}. \end{aligned} \tag{8.8}$$

8.1.3 Maximizing network throughput

Given a network of N anchors, the probability of a successful transmission is the probability that no collision occurs with any other anchor:

$$\begin{aligned} \Pr(\text{success}) &= (1 - \Pr(\text{collision}))^{N-1} \\ &= \left(1 - \frac{4A}{H+L}\right)^{N-1} \end{aligned} \tag{8.9}$$

Given a selection of H and L, the average rate at which any given anchor attempts to transmit is $F = \frac{2}{H+L}$ packets per second, and the average success rate of a single anchor's

transmissions S is therefore

$$\begin{aligned} S &= F \cdot \Pr(\text{success}) \\ &= F\left(1 - \frac{4A}{H+L}\right)^{N-1} \\ &= F\left(1 - 2AF\right)^{N-1} \end{aligned} \tag{8.10}$$

which has extrema when

$$\frac{\mathrm{d}S}{\mathrm{d}F} = (1-2AF)^{N-2}(1-2ANF) = 0. \tag{8.11}$$

This is satisfied for $F = (2A)^{-1}$, which yields 0 throughput; and for $F = (2AN)^{-1}$, which yields maximum throughput. Since $F = \frac{2}{H+L}$, it follows that to maximize throughput, anchors should select a combination of H and L such that $H + L = 4AN$, where N is the number of transmitting anchors.

8.1.4 Minimizing repeated collisions

Assuming that H and L are selected to maximize throughput, consider now the probability of a repeated packet collision occurring. Not all combinations of H and L are equal in this regard, as can be seen by considering the extreme case, in which $H = L$ (i.e. there is no randomness in an anchor's delay between cycles). In this situation, if the transmissions of two anchors collide, their next transmissions are also guaranteed to collide, since both anchors delay by an equal amount.

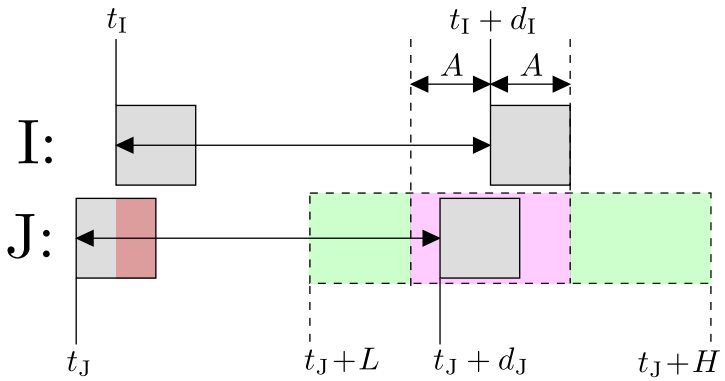


Figure 8.4: This figure shows the case in which a transmission from Anchor I beginning at time t_I collides with a transmission from Anchor J beginning at time t_J. Assuming Anchor I transmits a second packet after a delay of d_I, this figure shows the region of possible transmission delays d_J which will (shaded red) and will not (shaded green) result in a repeated collision. By selecting $H - L$ to be as large as possible, the probability of a repeated collision is minimized.

Consider the case depicted in Fig. 8.4, in which Anchor I's transmission, beginning at time t_I, results in a collision with Anchor J's packet, whose transmission must therefore begin at some time within the time period $t_\mathrm{J} \in [t_\mathrm{I} - A,\, t_I + A]$. Given that Anchor I finds itself in a cycle of delay d_I, the probability of its next packet succeeding is the probability that the next transmission from Anchor J does not cause a collision; that is, that

$$t_\mathrm{J} + d_\mathrm{J} \notin [t_\mathrm{I} + d_\mathrm{I} - A,\, t_\mathrm{I} + d_\mathrm{I} + A]. \tag{8.12}$$

As depicted graphically in Fig. 8.4, the range of allowable transmission times is shown in green, and the total range of possible transmission times has width $H - L$. The ratio of these, and thus the probability of a successful transmission given a previous collision, is given by

$$\begin{aligned}
\Pr(\text{success} \,|\, \text{collision}) &= \Pr(t_\mathrm{J} + d_\mathrm{J} < t_\mathrm{I} + d_\mathrm{I} - A) + \Pr(t_\mathrm{J} + d_\mathrm{J} \geq t_\mathrm{I} + d_\mathrm{I} + A) \\
&= \frac{1}{H - L}(\max(0,\, (t_\mathrm{J} + H) - (t_\mathrm{I} + d_\mathrm{I} + A)) + \max(0,\, (t_\mathrm{I} + d_\mathrm{I} - A) - (t_\mathrm{J} + L))) \\
&\geq \frac{1}{H - L}((t_\mathrm{J} + H) - (t_\mathrm{I} + d_\mathrm{I} + A) + (t_\mathrm{I} + d_\mathrm{I} - A) - (t_\mathrm{J} + L)) \\
&= 1 - \frac{2A}{H - L}.
\end{aligned} \tag{8.13}$$

This lower bound, and therefore the probability of a success following a collision, is maximized by maximizing $H - L$.

8.1.5 Optimal parameter selection

The above results provide a simple approach to handle the dynamic addition or removal of anchors from the network while maintaining high throughput and a low occurrence of repeated collisions: when a change in the number of transmitting anchors N is detected, anchors should select H and L such that

1. $H + L = 4AN$ to maximize throughput;
2. $H - L$ is as large as possible in order to maximize the probability of a success following a collision; and
3. constraints on H and L are fulfilled.

Since H is unconstrained from above, these criteria are satisfied by selecting L to be as small as possible given the constraints, and selecting $H = 4AN - L$. This simple adaptation scheme could be extended to the case of partially-connected networks with overlapping regions of coverage by having anchors adjust their transmission rate based on their "1-hop" network size.

8.2 Investigation of random transmission throughput

The random transmission scheme is evaluated using a network of $N = 2$, $N = 6$ and $N = 12$ anchors with anchor hardware and settings as described in Section 2.1.4. Each packet included a 1 byte anchor ID and an 8 byte timestamp, giving a total packet air time of $A = 108.6\,\mu s$. A minimum delay of $L = 250\,\mu s$ was required between transmission times to account for transmission of the previous packet, time spent processing on the microprocessor, and scheduling of the next packet. L was held constant for all experiments. The average single-anchor transmission rate $F = \frac{2}{H+L}$ was varied by selecting appropriate values of H.

Fig. 8.5 shows the measured single anchor throughput S for $N = 2$, $N = 6$ and $N = 12$ anchors, as the average transmission rate F is varied. This figure provides a comparison of measured data with theoretical values given by (8.10). The difference between the

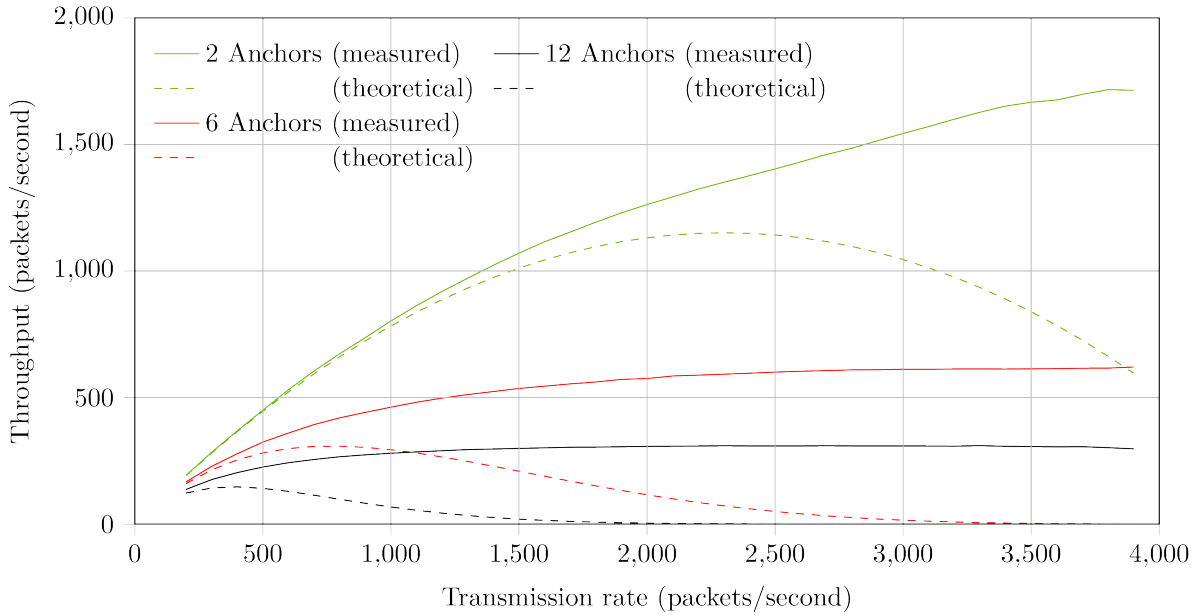


Figure 8.5: This figure shows the single-anchor throughput S for $N = 2$ (green), $N = 6$ (red) and $N = 12$ (black) transmitting anchors, as the average transmission rate F is varied. Total network throughput can be calculated by multiplying each curve by the respective number of anchors. Dashed lines show the theoretical throughput according to (8.10), while a solid line shows the measured throughput. If packet collisions did not occur, the throughput would be equal to the transmission rate; however, a comparison indicates that many packets are lost due to packet collisions, particularly as the average transmission rate increases. The discrepancy between the measured (solid line) and theoretical results (dashed line) indicates, however, that many packets are received correctly despite collisions occurring. This is likely due to the forward error correction employed by the DW1000 (see Section 2.1).

theoretical throughput and measured throughput for high transmission rates implies that some packets are received despite packet collisions occurring. Since UWB technology employs multiple layers of forward error correction, not every symbol must be received correctly in order to synchronize and communicate. It is therefore hypothesized that a packet collision might affect the quality of synchronization but not necessarily prevent packet reception. If this hypothesis is correct, a decrease in synchronization quality should be observed as packet collisions increase in likelihood (with increasing F).

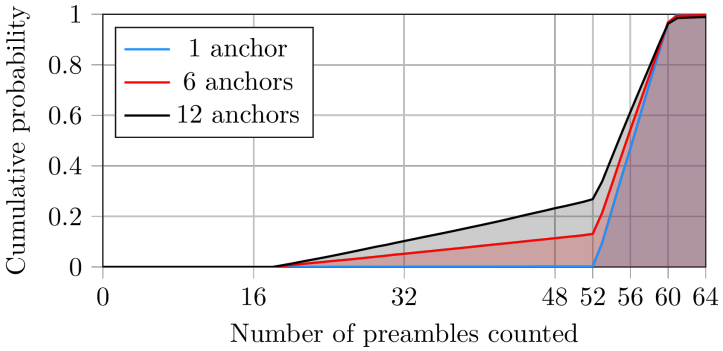
One measure of synchronization quality is the number of preamble symbols accumulated during the synchronization phase, a measure investigated in Fig. 8.6. This investigation confirms the hypothesis that an increase in F (and thus an increase in the probability of packet collisions) results in a decrease in synchronization quality.

This reduction in synchronization quality can further be observed in Fig. 8.7, in which the estimated CIR magnitude is compared for various transmission rates. In this figure, a decrease in CIR quality can be observed to occur with increases in transmission rate (indicated by an increase in the 5th and 95th percentile envelope). This decrease in CIR quality will result in less accurate reception timestamps, despite the packet data being correctly received.

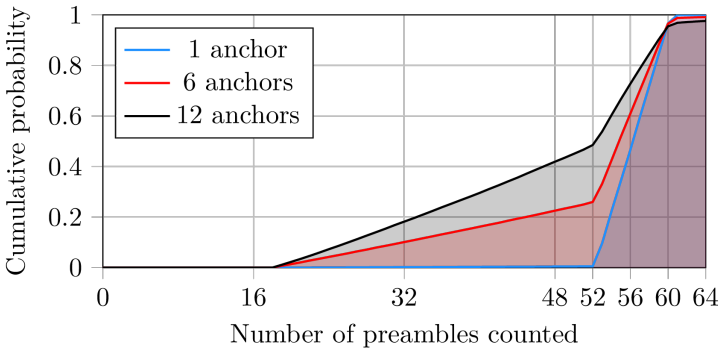
Considering the single-anchor results shown in Fig. 8.6, in the ideal case with no packet collisions (i.e. using one anchor) at least 52 preamble symbols are used for synchronization. As the probability of packet collisions increases (with increasing frequency), the probability of packets having fewer than 52 symbols used for synchronization also increases. Based on this, results of the previous throughput experiments (Fig. 8.5) are filtered, discarding measurements if fewer than 52 symbols were used in the synchronization. These results are shown in Fig. 8.8.

8.3 Method benefits and drawbacks

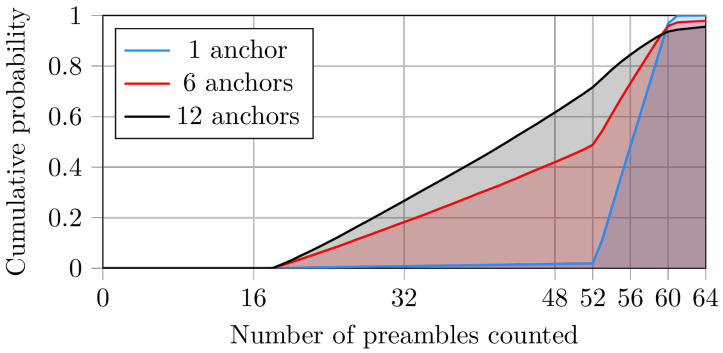
Based on these initial findings, the application of random transmission to UWB localization networks appears to be a promising research direction that could significantly reduce the complexity and increase the robustness of localization networks. By varying H and L, a desired network throughput can be achieved, allowing anchors to easily adjust to the dynamic addition and removal of anchors from the network. Despite these advantages, an appropriate mechanism of filtering correctly received but poorly timestamped packets is crucial for the method's success in the context of localization.



(a) Average transmission rate of $F = 700$ packets/second

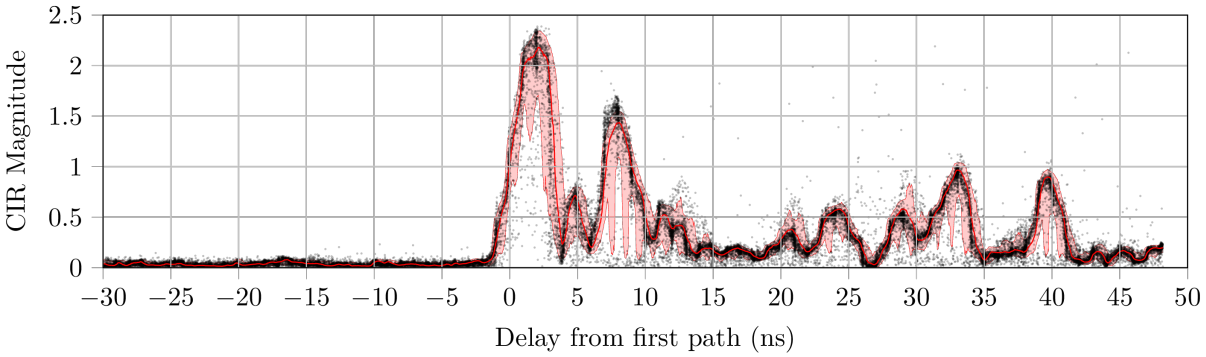


(b) Average transmission rate of $F = 1400$ packets/second

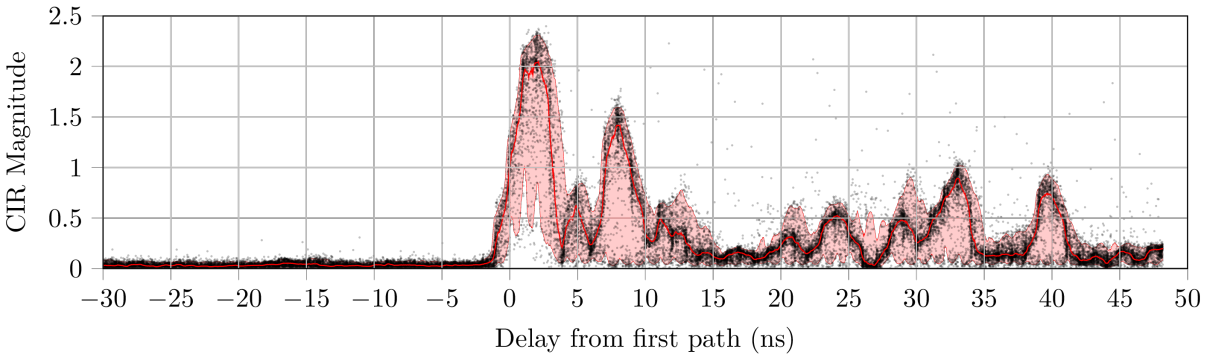


(c) Average transmission rate of $F = 2800$ packets/second

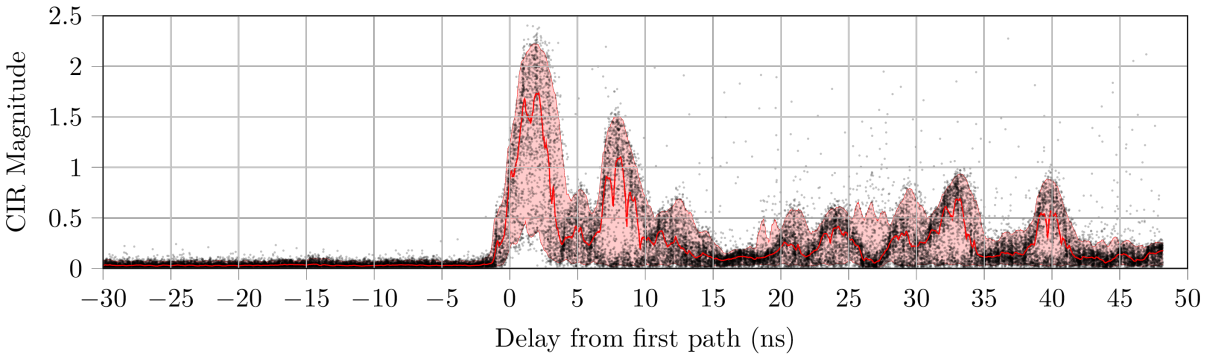
Figure 8.6: These figures show the degradation in synchronization quality occurring as the probability of packet collisions increases with increasing transmission rate. Since a single anchor transmitting (shown in blue) is not affected by packet collisions, a high level of synchronization quality is achieved for all average transmission rates F. Comparing this to the case when $N = 6$ (red) and $N = 12$ (black) anchors transmit randomly, a decrease in synchronization quality can be observed as the average transmission rate increases. This is indicated by an increased probability of packets being received with fewer preamble symbols.



(a) Average transmission rate of $F = 700$ packets/second



(b) Average transmission rate of $F = 1400$ packets/second



(c) Average transmission rate of $F = 2800$ packets/second

Figure 8.7: Six anchors transmitted randomly, with average transmission rates of $F = 700$, $F = 1400$, and $F = 2800$ packets/second. Measurements of the channel impulse response (CIR) magnitude were collected upon reception of a packet from the first of the six anchors. These scatter plots are composed of many individual samples of the CIR to the same anchor (black dots), with the region shaded in translucent red indicating the 5th and 95th percentile bounds of these measurements. The median CIR measurement is denoted with a thick red line. A comparison of the three plots shows that measurements of the CIR become increasingly noisy and inaccurate as the average transmission rate increases. This is in line with the expectation of synchronization quality decreasing as transmission rate is increased. This plot should be compared with Fig. 2.3, in which only a single anchor transmits and thus packet collisions do not occur.

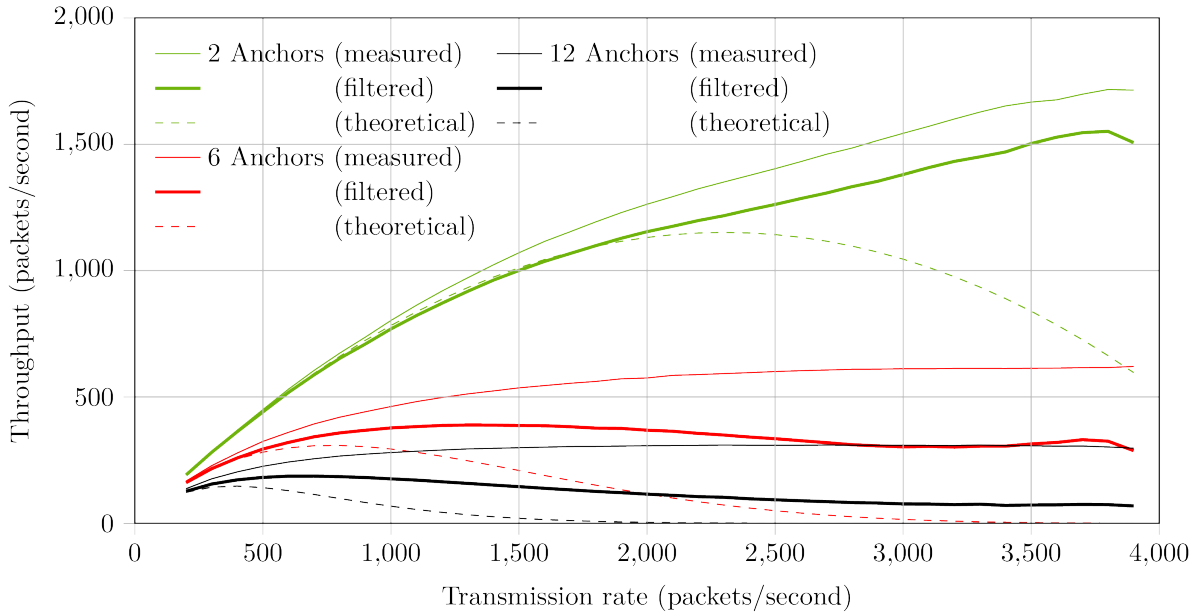


Figure 8.8: This figure presents results from the experiment of Fig. 8.5, with an additional series (thick line) showing the results of filtering packets based on synchronization quality. Based on the single-anchor results of Fig. 8.6, received packets were discarded if fewer than 52 preambles were used in the synchronization.

9

Future Work

9.1 Compensation for systematic biases

The experimental results presented in Chapter 7 demonstrate that multiple robots are capable of operating simultaneously within a space. These results also show that each robot is similarly affected by a systematic bias, resulting in a position-dependent offset between the robot's estimated and actual position. Reduction of these systematic biases is an important next step in improving the accuracy of UWB localization. As discussed in Chapter 4, UWB range measurements are affected by a white measurement noise with standard deviation 40 mm. However, since multiple UWB range measurements are required to localize and since each of these measurements has a different measurement bias, smooth trajectory tracking was only achieved if the robot's Kalman filter was detuned to treat each UWB range measurement as having a measurement noise with standard deviation 200 mm. If the magnitude of these measurement biases could be decreased, the robot's localization system would not need to be as drastically detuned.

Since these measurement biases are systematic, it should be possible to compensate at least partially for their influence. Possible approaches include:

1. Quantification of these biases through comparison with ground truth data, and modeling measurement bias as a function of a receiving module's pose relative to a transmitting module. In addition to pose data, this model may include signal characteristics such as received signal strength or measurements of the CIR. Initial results [R.2] suggest that this is a promising direction of research, and that learned models are general enough to be applied to localization networks in which ground-truth data is not available.
2. Using other sensors (e.g. an IMU, an optical flow sensor, or visual odometry) to provide additional information to the state estimate and applying techniques similar to iterative learning control to facilitate on-line learning of position-dependent biases. Initial results in [R.3] suggest this approach to be powerful, particularly when combined with visual odometry as a source of ground-truth data.

9.2 *ALOHA* Localization

Based on the findings presented in Chapter 8, the application of random transmission to UWB localization networks appears to be a promising research direction that could significantly reduce the complexity and increase the robustness and scalability of localization networks.

Ironically, the robustness of UWB radio as method of communication, owing to its multiple layers of forward error correction, also presents one of the largest challenges in implementing this localization scheme. As shown in Fig. 8.6 and Fig. 8.7, due to this robustness to interference, packet collisions can result in poor synchronization quality rather than a complete loss of the packet and thus, correct reception of a packet does not imply a collision-free transmission.

Although data communication in this setting is more robust than predicted, reduced synchronization quality leads to a reduction in the accuracy of reception timestamps, which can affect localization accuracy. One measure that could be used to determine reception quality is the number of preamble symbols used during the synchronization process. Results shown in Fig. 8.8 indicate that this is a promising first step in rejecting poor measurements, however, that it does not fully capture the problem. One potential focus of future research is on further quantifying the effects of packet collisions on timestamp quality, and therefore the effects on clock tracking and localization accuracy.

Bibliography: Part A

[1] Bitcraze AB, (2018). Crazyflie 2.0, [Online]. Available: https://www.bitcraze.io/crazyflie-2/.

[2] Barrett, T. W., "History of ultrawideband (UWB) radar & communications: Pioneers and innovators", in *Proc. Progress in Electromagnetics Symposium*, 2000, pp. 1–42.

[3] Sachs, J., *Handbook of ultra-wideband short-range sensing: theory, sensors, applications*. John Wiley & Sons, 2013.

[4] Bennett, C. L., Ross, G. F., "Time-domain electromagnetics and its applications", *Proceedings of the IEEE*, vol. 66, no. 3, pp. 299–318, 1978.

[5] Win, M. Z., Scholtz, R. A., "Impulse radio: How it works", *IEEE Communications letters*, vol. 2, no. 2, pp. 36–38, 1998.

[6] Withington, P., Fullerton, L. W., "An impulse radio communications system", in *Ultra-Wideband, Short-Pulse Electromagnetics*, H. L. Bertoni, L. Carin, and L. B. Felsen, Eds. Boston, MA: Springer US, 1993, pp. 113–120.

[7] Taylor, J. D., *Introduction to ultra-wideband radar systems*. CRC press, 1994.

[8] Federal Communications Commission, *First Report and Order 02-48*, Apr. 2002.

[9] IEEE Standard for information Technology, *IEEE Std 802.15.4a*, 2007.

[10] Sahinoglu, Z., Gezici, S., "Ranging in the IEEE 802.15. 4a standard", in *Wireless and Microwave Technology Conference*, 2006, pp. 1–5.

[11] Decawave, (2019). DW1000 UWB radio IC: User manual, [Online]. Available: https://www.decawave.com/wp-content/uploads/2019/04/DW1000-User-Manual.pdf (visited on Apr. 1, 2019).

[12] Ghavami, M., Michael, L., Kohno, R., *Ultra wideband signals and systems in communication engineering*. John Wiley & Sons, 2007.

[13] Etzlinger, B., Wymeersch, H., *Synchronization and Localization in Wireless Networks*, 1. Now Publishers, Inc., 2018, vol. 12, pp. 1–106.

[14] Decawave, (2018). APS010 application note: Wireless sensor networks and the DW1000, [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/10/APS010_DW1000-and-Wireless-Sensor-Networks_v1.1.pdf (visited on Oct. 1, 2018).

[15] McElroy, C., Neirynck, D., McLaughlin, M., "Comparison of wireless clock synchronization algorithms for indoor location systems", in *2014 IEEE International Conference on Communications Workshops (ICC)*, IEEE, 2014, pp. 157–162.

[16] Kulmer, J., Hinteregger, S., Großwindhager, B., Rath, M., Bakr, M. S., Leitinger, E., Witrisal, K., "Using Decawave UWB transceivers for high-accuracy multipath-assisted indoor positioning", in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2017, pp. 1239–1245.

[17] Cramer, R. J.-M., Win, M. Z., Scholtz, R. A., "Impulse radio multipath characteristics and diversity reception", in *IEEE International Conference on Communications*, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEE), vol. 3, 1998, pp. 1650–1654.

[18] Cramer, R. J.-M., Win, M. Z., Scholtz, R. A., "Evaluation of the multipath characteristics of the impulse radio channel", in *Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (Cat. No. 98TH8361)*, IEEE, vol. 2, 1998, pp. 864–868.

[19] Sipal, V., John, M., Neirynck, D., McLaughlin, M., Ammann, M., "Advent of practical UWB localization: (R)Evolution in UWB antenna research", in *The 8th European Conference on Antennas and Propagation (EuCAP 2014)*, IEEE, 2014, pp. 1561–1565.

[20] Mills, D. L., "Internet time synchronization: The network time protocol", *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.

[21] Elson, J., Römer, K., "Wireless Sensor Networks: A New Regime for Time Synchronization", *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 149–154, Jan. 2003.

[22] Sivrikaya, F., Yener, B., "Time synchronization in sensor networks: A survey", *IEEE Network*, vol. 18, no. 4, pp. 45–50, Jul. 2004.

[23] Fan, R., Lynch, N., "Gradient clock synchronization", in *In Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC). ACM*, 2004.

[24] Locher, T., Wattenhofer, R., "Oblivious Gradient Clock Synchronization", in *In Proc. 20th International Symposium on Distributed Computing (DISC*, 2006, pp. 520–533.

[25] Sommer, P., Wattenhofer, R., "Gradient clock synchronization in wireless sensor networks", in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IEEE Computer Society, 2009, pp. 37–48.

[26] Lenzen, C., Locher, T., Wattenhofer, R., "Tight Bounds for Clock Synchronization", *J. ACM*, vol. 57, no. 2, 8:1–8:42, Feb. 2010.

[27] Kuhn, F., Locher, T., Oshman, R., "Gradient Clock Synchronization in Dynamic Networks", *Theory of Computing Systems*, vol. 49, no. 4, pp. 781–816, Nov. 2011.

[28] Giorgi, G., Narduzzi, C., "Performance analysis of Kalman filter-based clock synchronization in IEEE 1588 networks", in *Control and Communication 2009 International Symposium on Precision Clock Synchronization for Measurement*, Oct. 2009, pp. 1–6.

[29] Kirsch, F., Vossiek, M., "Distributed Kalman filter for precise and robust clock synchronization in wireless networks", in *IEEE Radio and Wireless Symposium, 2009. RWS '09*, Jan. 2009, pp. 482–485.

[30] Hamilton, B. R., Ma, X., Zhao, Q., Xu, J., "ACES: Adaptive Clock Estimation and Synchronization Using Kalman Filtering", in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '08, New York, NY, USA: ACM, 2008, pp. 152–162.

[31] Abubakari, H., Sastry, S., "IEEE 1588 style synchronization over wireless link", in *Control and Communication 2008 IEEE International Symposium on Precision Clock Synchronization for Measurement*, Sep. 2008, pp. 127–130.

[32] Schroeder, J., Galler, S., Kyamakya, K., "A low-cost experimental ultra-wideband positioning system", in *2005 IEEE International Conference on Ultra-Wideband*, Sep. 2005, pp. 632–637.

[33] Jourdan, D. B., Deyst, J. J., Win, M. Z., Roy, N., "Monte Carlo localization in dense multipath environments using UWB ranging", in *2005 IEEE International Conference on Ultra-Wideband*, Sep. 2005, pp. 314–319.

[34] González, J., Blanco, J. L., Galindo, C., Ortiz-de-Galisteo, A., Fernández-Madrigal, J. A., Moreno, F. A., Martínez, J. L., "Mobile robot localization based on Ultra-Wide-Band ranging: A particle filter approach", *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 496–507, May 2009.

[35] Prorok, A., Arfire, A., Bahr, A., Farserotu, J., Martinoli, A., "Indoor navigation research with the Khepera III mobile robot: An experimental baseline with a case-study on ultra-wideband positioning", in *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sep. 2010, pp. 1–9.

[36] Prorok, A., Tomé, P., Martinoli, A., "Accommodation of NLOS for ultra-wideband TDOA localization in single-and multi-robot systems", in *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference On*, IEEE, 2011, pp. 1–9.

[37] Prorok, A., Gonon, L., Martinoli, A., "Online model estimation of ultra-wideband TDOA measurements for mobile robot localization", in *Robotics and Automation (ICRA), 2012 IEEE International Conference On*, IEEE, 2012, pp. 807–814.

[38] Prorok, A., Martinoli, A., "Accurate indoor localization with ultra-wideband using spatial models and collaboration", *The International Journal of Robotics Research*, pp. 547–568, Nov. 2013.

[39] Segura, M., Hashemi, H., Sisterna, C., Mut, V., "Experimental demonstration of self-localized Ultra Wideband indoor mobile robot navigation system", in *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sep. 2010, pp. 1–9.

[40] Zwirello, L., Schipper, T., Harter, M., Zwick, T., "UWB Localization System for Indoor Applications: Concept, Realization and Analysis", *Journal of Electrical and Computer Engineering*, vol. 2012, e849638, May 2012.

[41] Ye, T., Walsh, M., Haigh, P., Barton, J., O'Flynn, B., "Experimental impulse radio IEEE 802.15. 4a UWB based wireless sensor localization technology: Characterization, reliability and ranging", in *ISSC 2011, 22nd IET Irish Signals and Systems Conference, Dublin, Ireland. 23-24 Jun 2011*, Insitution of Engineering and Technology, 2011.

[42] Guo, K., Qiu, Z., Miao, C., Zaini, A. H., Chen, C.-L., Meng, W., Xie, L., "Ultra-Wideband-Based Localization for Quadcopter Navigation", *Unmanned Systems*, vol. 04, no. 01, pp. 23–34, Jan. 2016.

[43] Nguyen, T. M., Zaini, A. H., Guo, K., Xie, L., "An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied environments", in *International Micro Air Vehicle Conference and Competition 2016*, 2016.

[44] Tiemann, J., Eckermann, F., Wietfeld, C., "ATLAS - an open-source TDOA-based Ultra-wideband localization system", in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct. 2016, pp. 1–6.

[45] Kempke, B., Pannuto, P., Dutta, P., "PolyPoint: Guiding Indoor Quadrotors with Ultra-Wideband Localization", in *Proceedings of the 2nd International Workshop on Hot Topics in Wireless*, New York, NY, USA: ACM, 2015, pp. 16–20.

[46] Kempke, B., Pannuto, P., Campbell, B., Dutta, P., "SurePoint: Exploiting Ultra Wideband Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization", in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, ser. SenSys '16, Stanford, CA, USA: ACM, 2016, pp. 137–149.

[47] Kempke, B., Pannuto, P., Dutta, P., "Harmonium: Asymmetric, bandstitched UWB for fast, accurate, and robust indoor localization", in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, IEEE Press, 2016, p. 15.

[48] Wang, C., Zhang, H., Nguyen, T.-M., Xie, L., "Ultra-Wideband Aided Fast Localization and Mapping System", *arXiv:1710.00156 [cs]*, Sep. 2017. arXiv: 1710.00156 [cs].

[49] Bar-Shalom, Y., Li, X.-R., Kirubarajan, T., *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[50] Freris, N. M., Graham, S. R., Kumar, P. R., "Fundamental Limits on Synchronizing Clocks Over Networks", *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.

[51] Gezici, S., Tian, Z., Giannakis, G., Kobayashi, H., Molisch, A., Poor, H., Sahinoglu, Z., "Localization via ultra-wideband radios: A look at positioning aspects for future sensor networks", *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, Jul. 2005.

[52] Patwari, N., Ash, J. N., Kyperountas, S., Hero, A. O., Moses, R. L., Correal, N. S., "Locating the nodes: Cooperative localization in wireless sensor networks", *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 54–69, 2005.

[53] Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics Second Edition*. AIAA, 2007.

[54] Mahony, R., Kumar, V., Corke, P., "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor", *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[55] Stuelpnagel, J., "On the parametrization of the three-dimensional rotation group", *SIAM review*, vol. 6, no. 4, pp. 422–430, 1964.

[56] Markley, F. L., "Attitude error representations for Kalman filtering", *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[57] Mueller, M. W., Hehn, M., D'Andrea, R., "Covariance correction step for kalman filtering with an attitude", *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2301–2306, 2016.

[58] Shuster, M. D., "A survey of attitude representations", *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[59] Lupashin, S., Hehn, M., Mueller, M. W., Schoellig, A. P., Sherback, M., D'Andrea, R., "A platform for aerial robotics research and demonstration: The Flying Machine Arena", *Mechatronics*, vol. 24, no. 1, pp. 41–54, Feb. 2014.

[60] Brescianini, D., Hehn, M., D'Andrea, R., "Nonlinear quadrocopter attitude control", ETH Zurich, Tech. Rep., 2013.

[61] Abramson, N., "The ALOHA system: Another alternative for computer communications", in *Proceedings of the November 17-19, 1970, fall joint computer conference*, ACM, 1970, pp. 281–285.

Part B
Coordination of Robot Swarms

This thesis part is based on material published in

[P.2] **Hamer, M.**, Widmer, L., D'Andrea, R., "Fast generation of collision-free trajectories for robot swarms using GPU acceleration", *IEEE Access*, vol. 7, pp. 6679–6690, 2019

1

Overview

Improvements in the capabilities of robots and their control systems have allowed robots to operate in increasingly cluttered environments and in close proximity to each other. In such scenarios, trajectories for individual robots can be computed quickly using existing methods; however, the requirement for collision avoidance introduces a non-convex coupling between robot trajectories making the trajectory generation problem difficult to solve in a time-efficient manner.

This thesis addresses the swarm state-transition problem, in which each robot in the swarm is tasked with transitioning from a given initial state to a given goal state without colliding and while satisfying other constraints placed on the trajectories. This thesis proposes a parallelizable formulation of such problems, as well as a method for solving such problems efficiently on modern tensor or graphics processing units (GPUs). Trajectories for each robot are initialized independently using existing methods and without considering inter-robot collisions. This initialization is then iteratively improved using momentum-based gradient descent of a given loss function until feasibility. Given the non-convexity of the problem and the usage of gradient descent, the proposed method yields solutions in the local neighborhood of the initialization, and while not guaranteed to find an optimal solution, the focus of this thesis is on generating feasible and objectively "good" trajectories in a time-efficient manner, which is itself not an easy task given the number of robots interacting.

This thesis part begins with a review of related literature in Chapter 2. A general formulation of the proposed method is then presented in Chapter 3. Two simulation-based case studies are presented in Chapter 4:

1. the "Sort 200" quadcopter maze-traversal benchmark problem [1] (Fig. 1.1) is addressed in Section 4.1; and
2. a ground robot transition problem using a fleet of heterogeneous ground robots with bicycle dynamics (Fig. 1.2) is presented in Section 4.2. Such robots are often used as benchmarks for non-holonomic systems.

These case studies are used to demonstrate the method's application to nonlinear systems; to discuss the implementation of various state and input constraints; and to address various performance caveats and methods of increasing the method's convergence speed. In both cases, the method generates feasible, collision-free trajectories for the entire swarm

in seconds. These trajectories are fully-defined state and input trajectories, which can then be provided as reference to a trajectory-tracking controller on each robot.

Chapter 5 concludes this thesis part with a discussion of the proposed method's limitations and possibilities for future research.

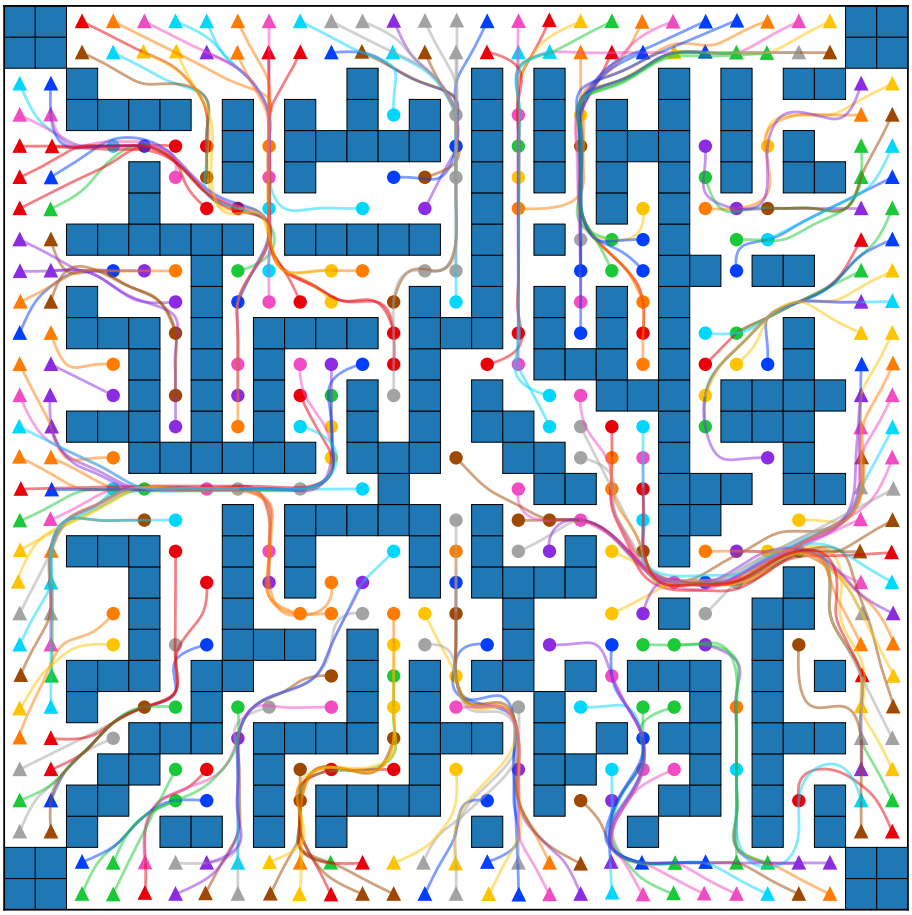


Figure 1.1: An example of the type of problem addressed by this method. In this benchmark example, 200 quadcopters are tasked with finding their way out of a maze without colliding with each other or with the maze. The proposed method takes approximately 2.3 seconds to generate feasible, collision-free trajectories for all 200 quadcopters. Further details are presented in Section 4.1.

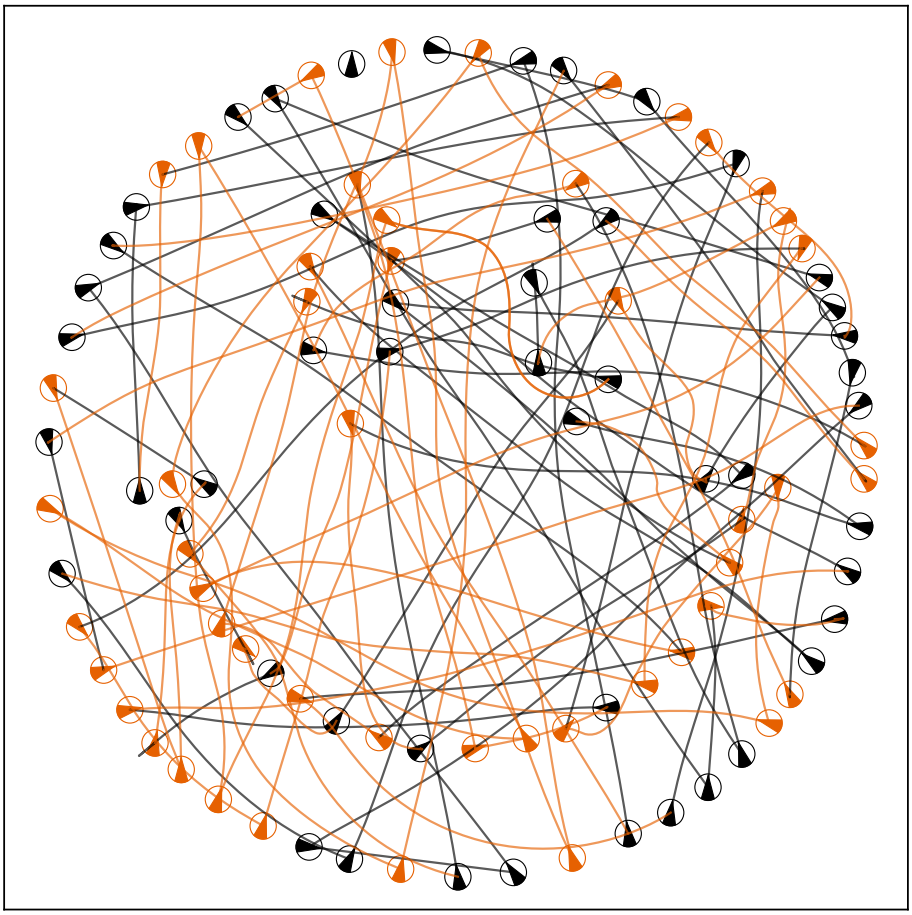


Figure 1.2: A further example of the type of problem addressed by this method. In this example, 100 robots with bicycle dynamics (e.g. cars, warehouse robots, etc.) are tasked with exchanging positions in a "smiley face" formation without colliding. To exemplify the application of the proposed method to heterogeneous fleets of robots, the steering angle of 50 robots (colored black) is constrained to 20°, and the steering angle of the other 50 robots (colored orange) to 70°. As elaborated upon in Section 4.2, the proposed method takes approximately 1.6 seconds to generate feasible, collision-free trajectories for the 100 robots.

2

Literature Review

2.1 Robot trajectory generation

Trajectory generation for individual robots is a well established field of research, with high-performance algorithms existing for most classes of robots. Many methods of trajectory generation express trajectories as the piecewise connection of basis functions. In [2], for example, trajectories are described using a piecewise constant jerk and generated for time-optimality using a bang-bang approach. In [3]–[6], trajectories are represented as polynomials generated to yield minimum jerk or minimum snap motions.

In cluttered environments, collisions with obstacles often prevent the direct application of the above methods. In such situations, graph-search methods can be used to plan trajectories in a discretized state- or action-space [7], [8]; or sampling based methods, for example Rapidly-exploring Random Trees (RRT) [9], RRT* [10] or Probabilistic Roadmaps (PRM) [11], can be used to find a feasible path through a continuous space.

Such search-based methods scale poorly with the dimensionality of the space and are thus often used to find feasible paths through a lower-dimensional space, before smoothing trajectories (e.g. piecewise polynomials or splines) are fit through the sequence of waypoints that define the path [12]. These search based approaches are very effective at generating feasible paths and trajectories for individual robots; however, as the number of robots in a swarm increases, so does the dimensionality of the search space and these methods again run into issues with scaling.

2.2 Swarm trajectory generation & collision avoidance

Generating collision-free trajectories for robot swarms has historically proven to be a highly combinatorial and high dimensional problem. Previous approaches have used convex approximations or reformulations of the problem to improve computational tractability. Examples of such approaches include [13], who solve the problem by iteratively solving mixed-integer linear programs; [14]–[18], who employ sequential convex programming

to iteratively refine the solution towards feasibility; [19], who formulate and solve the problem as a mixed-integer quadratic program; and [20], who solve the problem using a variation of the alternating direction method of multipliers.

The application of these aforementioned methods to large swarms is limited by their computational complexity. This is addressed in [21], who demonstrate that a significant increase in computation speed can be achieved by only including collision constraints for pairs of robots in each other's vicinity, and by formulating the problem such that robot trajectories can be optimized in parallel.

A further improvement in computational speed can be achieved by optimizing both trajectory and robot assignment in parallel, leveraging the fact that an optimal assignment of robots to goals will generally require shorter trajectories, and fewer collisions to be avoided, than a non-optimal assignment. This property is exploited by [22], who reformulate the optimization problem to be solved as a linear sum assignment problem and apply their method to swarms of hundreds of robots; and by [1], [23], [24], who iterate a search-based roadmap planner with a trajectory smoothing step until feasible robot trajectories are found, and who demonstrate their method by planning collision-free trajectories for hundreds of robots through densely cluttered environments. The recent tendency towards solving such problems in parallel, suggests that the application of GPUs to such optimization problems warrants investigation.

2.3 GPU-based trajectory generation

Accelerating computation using the parallel-processing architecture of GPUs is commonplace in many fields; however, the potential application of GPUs to robot trajectory generation problems remains largely unexplored.

The majority of papers to date use the GPU to parallelize the search for feasible trajectories for a single robot. Approaches include, for example, variations of genetic algorithms [25]–[29]; parallel implementations of PRM [30], [31]; and a parallel implementation of R* search [32], [33].

More in line with the approach of this thesis are [34], [35], within which the GPU is used to parallelize the evaluation of dynamics and collision constraints. Both papers, however, only deal with trajectories for a single robot. The non-convex trajectory generation problem is solved in [34] directly; however in this case the GPU is used only for parallel constraint evaluation, with results then transferred to a CPU-based solver.

3

Problem Formulation

The work presented in this part of the thesis advances the current research and literature in both the intended problem domain: swarms of hundreds to thousands of robots; and in the approach: standard methods are used to generate feasible initializations for individual robot trajectories, and the GPU is then used both to evaluate constraints in parallel and to solve the non-convex trajectory generation problem, allowing the proposed method to leverage the computational power of modern GPUs to solve the problem quickly.

The proposed method addresses the generation of feasible, collision-free trajectories for robots operating simultaneously and in close proximity. Each robot's trajectory is initialized independently without considering inter-robot collisions, and constraints on the trajectories (e.g. state and collision constraints) are modeled as soft constraints and included in an objective function to be minimized. Momentum-based gradient descent is then employed to iteratively improve robot trajectories until all constraints are satisfied. Given the non-convexity of the problem and the usage of gradient descent, the proposed method is not guaranteed to find an optimal solution, rather yielding feasible solutions in the local neighborhood of the initialization.

This section formalizes the above problem, introduces notation, outlines assumptions and requirements, and presents the method formally in Algorithm 1.

3.1 Robot states and inputs

Consider a swarm of R robots, in which robots are numbered sequentially from 1 to R and where indices i and j are used to refer to arbitrary robots within the swarm. For simplicity of the following explanation, all robots in the swarm are assumed to be identical; however, this method can be trivially extended for swarms of different robots, an example of which is shown in the case study presented in Section 4.2.

Let each of the R robots have an input space of dimension M, and a state space of dimension N. Inputs are issued to each robot at discrete times $k = 0, \ldots, K-1$, and are held constant between time instants. Robot i's input trajectory is denoted by $U_i \in \mathbb{R}^{K \times M}$, and its state trajectory by $S_i \in \mathbb{R}^{(K+1) \times N}$. Each robot's initial state $S_i[0]$ and desired goal state $G_i \in \mathbb{R}^N$ are assumed known and feasible. The proposed method generates a feasible

and collision-free trajectory for each robot, which transitions it from its initial state to within a defined threshold of its goal state (e.g. a 5 cm final position tolerance is used in the case studies in Chapter 4).

3.2 Robot dynamics

Robot i's input trajectory is mapped to its state trajectory through its known dynamics $f(\cdot, \cdot)$ as

$$S_i[k+1] = f(U_i[k],\ S_i[k]) \text{ for all } k \in [0,\ K-1]. \tag{3.1}$$

Iterating f across the entire input trajectory,

$$S_i = \mathcal{F}(U_i,\ S_i[0]) \tag{3.2}$$

expresses the relationship between robot i's input and state trajectory.

For notational simplicity, the input and state trajectories of all robots are stacked into the tensors $\boldsymbol{U} \in \mathbb{R}^{K \times M \times R}$ and $\boldsymbol{S} \in \mathbb{R}^{(K+1) \times N \times R}$ respectively. This is not always possible in the case of heterogenous swarms; however, in such cases, operations can be performed on trajectories individually, while incurring minimal computational overhead. This stacking is therefore without loss of generality. Overloading notation, the relationship between the input and state trajectories of the swarm is written as

$$\boldsymbol{S} = \mathcal{F}(\boldsymbol{U},\ \boldsymbol{S}[0]), \tag{3.3}$$

noting that $\boldsymbol{S}$ is a function of both the robots' initial states $\boldsymbol{S}[0]$ and the robots' input trajectories $\boldsymbol{U}$. Hidden behind this notation are a number of performance caveats, which are discussed in Section 3.6.

3.3 Constraints & optimization objective

The robots' input and state trajectories are subject to constraints, which can include inter-robot and environmental collision constraints; constraints on the final state (e.g. goal constraints); bounds on the state trajectory (e.g. minimum and maximum velocity); as well as actuator limits (e.g. minimum and maximum input). This general formulation also allows for coupled input and state constraints (e.g. state-dependent input bounds), or for constraints which are only active on specific robots.

These constraints are modeled as soft constraints and $\mathscr{C}$ is used to denote the set of all constraints. Each constraint $c \in \mathscr{C}$ is modeled to consist of two components:

1. a loss function $\mathcal{L}_c(\boldsymbol{U}, \boldsymbol{S})$, which is included in the optimization objective function (detailed below) and whose negative gradient provides a direction towards a feasible solution; and
2. a satisfaction function $\mathcal{S}_c(\boldsymbol{U}, \boldsymbol{S})$, whose boolean output indicates whether the constraint c is satisfied. If $\mathcal{S}_c(\boldsymbol{U}, \boldsymbol{S})$ is true for all $c \in \mathscr{C}$, trajectories are deemed feasible and the optimization terminates.

Weighting each loss by a parameter w_c, the optimization objective is to minimize

$$\mathcal{L}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq \sum_{c \in \mathscr{C}} w_c \mathcal{L}_c(\boldsymbol{U}, \boldsymbol{S}) \tag{3.4}$$

with respect to the input trajectories $\boldsymbol{U}$. This is achieved by descending the loss gradient

$$\nabla_{\boldsymbol{U}} \mathcal{L}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq \sum_{c \in \mathscr{C}} w_c \left(\frac{\partial \mathcal{L}_c}{\partial \boldsymbol{U}} + \frac{\partial \mathcal{L}_c}{\partial \boldsymbol{S}} \cdot \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right) (\boldsymbol{U}, \boldsymbol{S}), \tag{3.5}$$

recalling that $\boldsymbol{S}$ is a function of $\boldsymbol{U}$. This gradient calculation and back-propagation can be automated using the auto-differentiation functionality of tensor arithmetic libraries such as Tensorflow [36], PyTorch [37] or MXNet [38], or can be computed manually. Examples of a number of common constraints are presented in the case studies of Chapter 4.

3.4 Optimization

Since the proposed method uses gradient descent, the speed of the algorithm and quality of the solution are highly dependent on the quality of the initialization. Robot input trajectories should therefore be initialized to transition the robot from its initial state to its goal state, while satisfying individual robot constraints, and in a manner that is appropriate for the problem (e.g. time optimal, minimum jerk, etc.). This initialization is performed for each robot independently and without considering inter-robot collisions.

The requirement for collision-avoidance between robots introduces a coupling between the trajectories of individual robots and makes the problem non-trivial to solve. Using a gradient descent optimizer, for example Adam [39], the proposed method iteratively improves the input tensor $\boldsymbol{U}$ by descending the loss function's gradient (3.5). When trajectories are found to satisfy all constraints (i.e. $\mathcal{S}_c(\boldsymbol{U}, \boldsymbol{S})$ is true for all $c \in \mathscr{C}$) the algorithm terminates. This optimization procedure is formalized in Algorithm 1.

Algorithm 1 The proposed method uses momentum-based gradient descent to iteratively improve robot input trajectories $\boldsymbol{U}$ until the corresponding state trajectories $\boldsymbol{S}$ are collision-free and feasible with respect to the given constraints.

Require:

- Initialization of robot input trajectories $\boldsymbol{U}$, which are feasible for individual robots if robot collisions are ignored
- Initial state of each robot $\boldsymbol{S}[0]$
- Robot dynamics $\mathcal{F}(\cdot, \cdot)$ which maps the robots' inputs and initial states to their state trajectories as in (3.3)
- Set of constraints $\mathscr{C}$, where each constraint $c \in \mathscr{C}$ is defined by a loss function $\mathcal{L}_c(\cdot, \cdot)$ and satisfaction check $\mathcal{S}_c(\cdot, \cdot)$
- Weighting parameter w_c for each constraint $c \in \mathscr{C}$
- Gradient descent optimizer $\mathrm{OPT}(\cdot, \cdot \,|\, \eta)$, parameterized by a given learning rate η and which takes a gradient and momentum state, and returns a step and updated momentum.

Optimization Procedure:

Reset optimizer momentum $\boldsymbol{\rho}$
loop
  Calculate robot state trajectories as in (3.3)
  $\boldsymbol{S} \leftarrow \mathcal{F}(\boldsymbol{U}, \boldsymbol{S}[0])$
  Calculate loss gradient as in (3.5)
  $\nabla_{\boldsymbol{U}} \mathcal{L} \leftarrow \sum_c w_c \left(\frac{\partial \mathcal{L}_c}{\partial \boldsymbol{U}} + \frac{\partial \mathcal{L}_c}{\partial \boldsymbol{S}} \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{U}} \right) (\boldsymbol{U}, \boldsymbol{S})$
  Check constraint satisfaction
  if $\mathcal{S}_c(\boldsymbol{U}, \boldsymbol{S})$ is false for any $c \in \mathscr{C}$ **then**
    Run optimizer
    $\Delta \boldsymbol{U}, \boldsymbol{\rho} \leftarrow \mathrm{OPT}(\nabla_{\boldsymbol{U}} \mathcal{L}, \boldsymbol{\rho} \,|\, \eta)$
    Update robot input trajectories
    $\boldsymbol{U} \leftarrow \boldsymbol{U} + \Delta \boldsymbol{U}$
  else
    return $\boldsymbol{U}, \boldsymbol{S}$
  end if
end loop

Returns:

- Input trajectories $\boldsymbol{U}$ that are feasible with respect to all input constraints
- Collision-free state trajectories $\boldsymbol{S}$ that are feasible with respect to all state constraints

3.5 Hyperparameter tuning

With respect to (3.5), observe that the gradient and therefore the direction and size of the gradient descent update are dependent on the set of constraint weighting factors $\{w_c \,|\, c \in \mathscr{C}\}$. In addition, the size of each update step is proportional to the so-called "learning rate" of the gradient descent optimizer. The learning rate, together with the constraint weights are the hyperparameters of the proposed method.

The selection of appropriate hyperparameters is specific to the class of problem being addressed. Selecting appropriate hyperparameters is critical to achieving fast convergence to reasonable results. Hyperparameters should be chosen such that the number of collisions is quickly reduced, while ensuring that other constraints which may be violated during the optimization process are quickly reoptimized to feasibility. An automated hyperparameter search [40] was used to select hyperparameters for the problem classes presented in Chapter 4. This search involves running hundreds of thousands of simulations with different hyperparameter values, and selecting the hyperparameter set that minimizes the 90th-percentile convergence speed of the algorithm.

The effects of hyperparameter tuning are investigated in more detail in Section 4.2.5.

3.6 Performance considerations

The speed of the proposed method comes from the ability to parallelize many of the algorithm's steps and thus leverage the computational power of a modern GPU. With reference to Algorithm 1, the two major steps of the algorithm are the calculation of state trajectory (line 3), and the calculation of a loss gradient (line 4). It is important to ensure that these steps are implemented in such a way as to enable their effective parallelization.

3.6.1 Calculation of state trajectory

When computing the state trajectory of each robot from its input trajectory, it is important to note that the dynamics $\mathcal{F}(\boldsymbol{U}, \boldsymbol{S}[0])$ are defined to operate on the input trajectory as a whole, rather than iterating the dynamics across time (as in (3.1)).

Significant performance gains can be realized when the dynamics are implemented using cumulative sums or cumulative products, since these cumulative operations can be efficiently parallelized to run in logarithmic time [41]. The case studies presented in Chapter 4 show two examples of how robot dynamics expressed as standard difference equations can be implemented using cumulative sums.

3.6.2 Calculation of loss gradient

Losses are typically independent across time and across robots, and can be effectively implemented using tensor operations, for example as provided by libraries such as Tensorflow [36], PyTorch [37] or MXNet [38]. Tensor operations are inherently parallelizable

and thus enable the evaluation of losses in parallel. Likewise, the evaluation of the loss gradient follows through back-propagation, which is based on tensor multiplications and additions and is thus also inherently parallelizable.

A significant bottleneck when calculating loss gradient is the computation of inter-robot collisions. This operation requires naively $O(R^2)$ comparisons if all robots are compared with all other robots. Although each comparison is independent, the sheer number of comparisons for large swarms quickly exhausts the GPU's ability to process these in parallel. In Section 4.1.4 an alternative approach is introduced, which leverages parallel sorting to reduce the time complexity of this step to $O(\log\ R)$ [41], [42].

4

Example Case Studies

This chapter demonstrates the application of the proposed method to two different scenarios.

1. Section 4.1 implements the "Sort 200" quadcopter maze-traversal benchmark problem, first proposed in [1] (see Fig. 1.1)
2. Section 4.2 implements a ground robot transition problem using a fleet of heterogeneous ground robots with bicycle dynamics (see Fig. 1.2).

An example implementation of each scenario is available at [43].

Results presented in this section were generated using an Nvidia GeForce GTX 1080 Ti GPU, a widely-available consumer GPU costing approximately $700 at the time of writing. Implementations were programmed using Python 3.6, PyTorch version 0.4.1, CUDA version 9.2.148 and cuDNN version 7.1.4. Tensor arithmetic was performed using 32-bit floating point. GPU utilization was between 20% and 40% in all scenarios, implying that further performance may be achievable through a more optimized implementation. Using a more advanced GPU than was available at the time of writing, further performance improvements could be achieved by using 16-bit floating point operations, which ideally yield twice the throughput compared with 32-bit floating point operations.

4.1 "Sort200" quadcopter maze benchmark

In the "Sort 200" benchmark scenario, proposed in [1], 200 quadcopter robots begin at random (x, y) locations within a maze and must fly to a goal location outside the maze without colliding. Initial and goal positions all lie on the $z = 0$ plane. As in the original benchmark scenario, quadcopters are allowed to move in the z dimension if required to avoid collisions.

An example of the Sort 200 scenario is shown in Fig. 1.1, which shows the initial and final positions of 200 quadcopters connected by their collision-free trajectories.

4.1.1 Robot modeling

The dynamic model of a quadcopter is differentially flat [44], a property which is often exploited to allow trajectories to be planned in each of the inertial axes independently (see, e.g. [2]–[5]). A similar approach is taken in this case study, and jerk trajectories are planned for each quadcopter in each axis independently. These trajectories can at a later stage be converted to nominal thrust and body-rate inputs, or could be provided to a trajectory-tracking controller.

Denoting the three inertial axes with subscripts x, y and z, the position, velocity, acceleration and jerk of the quadcopter in the inertial frame are respectively denoted by

$$\mathrm{p} \coloneqq \left(\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_z \right), \tag{4.1}$$
$$\mathrm{v} \coloneqq \left(\mathrm{v}_x, \mathrm{v}_y, \mathrm{v}_z \right), \tag{4.2}$$
$$\mathrm{a} \coloneqq \left(\mathrm{a}_x, \mathrm{a}_y, \mathrm{a}_z \right), \text{ and} \tag{4.3}$$
$$\mathrm{j} \coloneqq \left(\mathrm{j}_x, \mathrm{j}_y, \mathrm{j}_z \right). \tag{4.4}$$

The $M = 3$ dimensional input space of a quadcopter is defined as

$$U \coloneqq \left(\mathrm{j}_x, \mathrm{j}_y, \mathrm{j}_z \right) \tag{4.5}$$

and the quadcopter's $N = 9$ dimensional state is defined as

$$S \coloneqq \left(\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_z, \mathrm{v}_x, \mathrm{v}_y, \mathrm{v}_z, \mathrm{a}_x, \mathrm{a}_y, \mathrm{a}_z \right). \tag{4.6}$$

Discretizing using the Euler forward method for a sampling period of $T = 50\,\mathrm{ms}$, the quadcopter's state in the inertial frame evolves as

$$\mathrm{a}[k+1] = \mathrm{a}[k] + T\,\mathrm{j}[k] \tag{4.7}$$
$$\mathrm{v}[k+1] = \mathrm{v}[k] + T\,\mathrm{a}[k] + \tfrac{1}{2}T^2\mathrm{j}[k] \tag{4.8}$$
$$\mathrm{p}[k+1] = \mathrm{p}[k] + T\,\mathrm{v}[k] + \tfrac{1}{2}T^2\mathrm{a}[k] + \tfrac{1}{6}T^3\mathrm{j}[k], \tag{4.9}$$

where it is assumed that the input $\mathrm{j}[k]$ is held constant during timestep k. These recursive difference equations can be rewritten to express the state at time $k + 1$ in terms of the

input and state history:

$$\mathrm{a}[k{+}1] = \mathrm{a}[0] + T\sum_{\kappa=0}^{k}\mathrm{j}[\kappa] \tag{4.10}$$

$$\mathrm{v}[k{+}1] = \mathrm{v}[0] + T\sum_{\kappa=0}^{k}\mathrm{a}[\kappa] + \tfrac{1}{2}T^2\sum_{\kappa=0}^{k}\mathrm{j}[\kappa] \tag{4.11}$$

$$\mathrm{p}[k{+}1] = \mathrm{p}[0] + T\sum_{\kappa=0}^{k}\mathrm{v}[\kappa] + \tfrac{1}{2}T^2\sum_{\kappa=0}^{k}\mathrm{a}[\kappa] + \tfrac{1}{6}T^3\sum_{\kappa=0}^{k}\mathrm{j}[\kappa]. \tag{4.12}$$

Lifting the above in time, observe that the state trajectory can be computed using cumulative summation (denoted CS):

$$\begin{aligned}\mathrm{a}[1{:}K] &= \mathrm{a}[0] + T{\cdot}\mathrm{CS}(\mathrm{j}) \\ \mathrm{v}[1{:}K] &= \mathrm{v}[0] + T{\cdot}\mathrm{CS}(\mathrm{a}[0{:}K{-}1]) + \tfrac{1}{2}T^2{\cdot}\mathrm{CS}(\mathrm{j}) \\ \mathrm{p}[1{:}K] &= \mathrm{p}[0] + T{\cdot}\mathrm{CS}(\mathrm{v}[0{:}K{-}1]) + \tfrac{1}{2}T^2{\cdot}\mathrm{CS}(\mathrm{a}[0{:}K{-}1]) + \tfrac{1}{6}T^3{\cdot}\mathrm{CS}(\mathrm{j}).\end{aligned} \tag{4.13}$$

This implementation can be effectively parallelized to run in logarithmic time, which in this case-study executes an order of magnitude faster than the linear-time, recursive implementation.

4.1.2 Assignment of robots to goals

Prior to trajectory generation, the 200 quadcopters must be assigned one of 200 goal positions located on the perimeter of the maze. To this end, the given 2D maze is transformed into a graph representation by discretizing the graph coordinates and connecting points with an unobstructed line of sight using an edge with weight equal to the points' Euclidean distance. The path length between all (discretized) positions within the maze can then be computed using Dijkstra's algorithm [45]. This step is only required if the maze changes.

Using these computed path lengths and the known initial quadcopter positions, the Hungarian method [46] is then used to compute an allocation of quadcopters to goal positions that minimizes the sum of squared distance travelled by the swarm.

4.1.3 Initialization of individual trajectories

Once the allocation of initial positions to goal positions is known, splines can be fitted to the shortest path connecting these positions (computed as a result of Dijkstra's algorithm in the previous step). As in the original benchmark problem, quadcopters are given 10 seconds to exit the maze. Splines are scaled in time to have this duration, and then sampled at the desired rate (in this case, 20 Hz) to yield the initial trajectories for each quadcopter.

The results of this stage are 200 jerk trajectories in x and y, which quickly and smoothly transition robots from their initial positions within the maze to their goal positions at its perimeter. The z component of each trajectory is initialized to zero. At this stage, quadcopters are treated independently and as such the initial trajectories result in collisions between quadcopters. This and the previous step are computed on the CPU due to the availability of fast centralized solvers for Dijkstra's algorithm and the Hungarian method.

4.1.4 Constraints

The constraint set consists of constraints based on the quadcopters' physical capabilities, constraints on quadcopter collisions with each other and with the maze walls, and constraints on the final state of each quadcopter. Based on these constraints, the loss function is defined as

$$\begin{aligned}\mathcal{L}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq\ & w_{\text{thrust}}\, \mathcal{L}_{\text{thrust}}(\boldsymbol{U}, \boldsymbol{S}) + \\ & w_{\text{body-rate}}\, \mathcal{L}_{\text{body-rate}}(\boldsymbol{U}, \boldsymbol{S}) + \\ & w_{\text{quad-collision}}\, \mathcal{L}_{\text{quad-collision}}(\boldsymbol{U}, \boldsymbol{S}) + \\ & w_{\text{maze-collision}}\, \mathcal{L}_{\text{maze-collision}}(\boldsymbol{U}, \boldsymbol{S}) + \\ & w_{\text{goal-pos}}\, \mathcal{L}_{\text{goal-pos}}(\boldsymbol{U}, \boldsymbol{S}) + \\ & w_{\text{goal-vel}}\, \mathcal{L}_{\text{goal-vel}}(\boldsymbol{U}, \boldsymbol{S}).\end{aligned} \tag{4.14}$$

The following defines and discusses the implementation of these constraints and their respective loss functions.

Dynamics constraints In line with [1], the quadcopter dynamic limits are set based on a Crazyflie 2.0 quadcopter [47]. These limits include the minimum and maximum mass-normalized thrusts $\mathrm{f}_{\min} = 5\,\mathrm{m\,s^{-2}}$ and $\mathrm{f}_{\max} = 15\,\mathrm{m\,s^{-2}}$, and the maximum tilting body-rate $\omega_{\max} = 30\,\mathrm{rad\,s^{-1}}$ [A.3].

Letting g denote the vector of gravitational acceleration, quadcopter i's thrust at time step k is

$$\mathrm{f}_i[k] = \|\mathrm{a}_i[k] - g\|_2, \tag{4.15}$$

which is constrained to the feasible range using the loss function

$$\begin{aligned}\mathcal{L}_{\text{thrust}}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq & \sum_{k=1}^{K} \sum_{i=1}^{R} \max\left\{0,\ \mathrm{f}_i[k] - \mathrm{f}_{\max}\right\} + \\ & \sum_{k=1}^{K} \sum_{i=1}^{R} \max\left\{0,\ \mathrm{f}_{\min} - \mathrm{f}_i[k]\right\}.\end{aligned} \tag{4.16}$$

As shown in [5], the magnitude of quadcopter i's tilting body-rates $\omega_{x,i}[k]$ and $\omega_{y,i}[k]$ at timestep k can be upper-bounded by a function of its jerk and thrust as:

$$\sqrt{\omega_{x,i}[k]^2 + \omega_{y,i}[k]^2} \leq \frac{\|\mathrm{j}_i[k]\|_2}{\mathrm{f}_i[k]}. \tag{4.17}$$

This upper bound is used to constrain the quadcopters' tilting body-rates using the loss function

$$\mathcal{L}_{\text{body-rate}}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq \sum_{k=0}^{K-1} \sum_{i=1}^{R} \max\left\{0,\ \frac{\|\mathrm{j}_i[k]\|_2}{\mathrm{f}_i[k]} - \omega_{\max}\right\}. \tag{4.18}$$

A quadcopter's thrust and body-rates are defined to satisfy their respective constraints if the associated loss function is zero.

Quadcopter collision constraints Two quadcopters are assumed to collide if their (x, y, z) centers are within a certain distance D, referred to as the collision distance and defined as $D \coloneqq 0.25\,\mathrm{m}$ in line with [1]. The collision loss function is

$$\mathcal{L}_{\text{collision}}(\boldsymbol{U}, \boldsymbol{S}) \coloneqq \sum_{k=1}^{K} \sum_{i=1}^{R} \sum_{j=i+1}^{R} 2 \cdot \max\left\{0,\ D - d_{ij}[k]\right\}, \tag{4.19}$$

where $d_{ij}[k] \coloneqq \|\mathrm{p}_i[k] - \mathrm{p}_j[k]\|_2$ is the center-to-center distance between quadcopters i and j at timestep k. Note that due to the pairwise nature of collisions, only quadcopters with a higher index must be checked, and the collision loss is therefore doubled.

Checking for robot-robot collisions is one of the most time consuming steps in the optimization pipeline. Directly computing the above loss requires that for each step in time, the positions of all robots are checked against the positions of all other robots to determine whether the robots collide, naively requiring $O(R^2)$ comparisons. As noted in [21], the time required for this step can be significantly reduced if pairs of quadcopters are only checked for a collision if they are closer than a given threshold.

A similar approach is taken in Algorithm 2, which uses a parallel argsort to sort robots according to their position $\mathrm{p}_{\alpha,i}$ in the axis $\alpha \in \{x, y, z\}$. It is then possible to compare each robot only with those robots having similar α coordinates. By choosing α to be an axis with a large position variance (e.g. in this case study x or y are good choices), the number of checks required can be reduced to a small, approximately constant value, and as such the time complexity of this approach is dominated by the $O(\log R)$ complexity of parallel sort [41], [42].

Maze collision constraints The maze collision loss function is implemented as a two-dimensional lookup. For every quadcopter and at every timestep the quadcopter's x and y position are checked for a collision with the maze boundaries. If a collision is found

Algorithm 2 This algorithm drastically improves the speed of collision checks by leveraging the efficiency of parallel sorting algorithms. Robot positions are sorted along an axis α with high position variance, thus enabling collision checks to be performed in the robot's local neighborhood rather than across all robots in the swarm. This algorithm implements the loss function given in (4.19).

Require:

- Robot positions $\mathrm{p}_i[k] \coloneqq \big(\mathrm{p}_{x,i}[k],\, \mathrm{p}_{y,i}[k],\, \mathrm{p}_{z,i}[k]\big)$
- Axis $\alpha \in \{x,\, y,\, z\}$ with large position variance
- Minimum allowed center-to-center distance D

Robot collision detection procedure:

for $k \in [1, K]$ **in parallel do**
  Order robots at time k by their position in axis α
  $\mathrm{Idx}[k, :] \leftarrow \mathrm{ParallelArgsort}(\{\mathrm{p}_{\alpha,i}[k] \,|\, i \in [1, R]\})$
end parallel for
In parallel, check collisions for all robots at all timesteps
for $i \in [1,\, R]$ **and** $k \in [1, K]$ **in parallel do**
  Sequentially (to allow early stopping), check collisions with the current robot i
  for $j \in [i+1,\, R]$ **sequentially do**
    Indexes of robot pair
    $r_i \leftarrow \mathrm{Idx}[k, i];\; r_j \leftarrow \mathrm{Idx}[k, j]$
    if $\mathrm{p}_{\alpha,r_j}[k] - \mathrm{p}_{\alpha,r_i}[k] > D$ **then**
      Since $p_{\alpha,r_{j+1}}[k] \geq p_{\alpha,r_j}[k]$, stop early
      break
    end if
    Compute pairwise distance
    $d \leftarrow \|\mathrm{p}_{r_j} - \mathrm{p}_{r_i}\|_2$
    Compute loss and loss gradients
    if $d \leq D$ **then**
      atomic $\left\{\; \mathcal{L}_{\text{collision}} \leftarrow \mathcal{L}_{\text{collision}} + 2 \cdot (D - d) \;\right\}$
      atomic $\left\{\; \nabla_{\mathrm{p}_{r_i}} \mathcal{L} \leftarrow \nabla_{\mathrm{p}_{r_i}} \mathcal{L} + (\mathrm{p}_{r_j} - \mathrm{p}_{r_i})/d \;\right\}$
      atomic $\left\{\; \nabla_{\mathrm{p}_{r_j}} \mathcal{L} \leftarrow \nabla_{\mathrm{p}_{r_j}} \mathcal{L} - (\mathrm{p}_{r_j} - \mathrm{p}_{r_i})/d \;\right\}$
    end if
  end sequential for
end parallel for
return $\mathcal{L}_{\text{collision}},\, \{\nabla_{\mathrm{p}_i} \mathcal{L} \,|\, i \in [1, R]\}$

Returns:

- Collision loss $\mathcal{L}_{\text{collision}}$
- Collision loss gradient $\nabla_{\mathrm{p}_i} \mathcal{L}$ for each robot i

to occur, that is if the quadcopter's position is closer to a maze wall than its collision distance allows, a loss with a gradient normal to the boundary is added to the global loss.

It is useful to distinguish between the case of a quadcopter's trajectory traveling too close to a wall, and the case of the trajectory moving into a wall. During the optimization process, quadcopter state trajectories are perturbed in order to avoid collisions and to satisfy other constraints. This is an iterative process, and satisfying one constraint may require temporarily violating another constraint during an intermediate iteration of the optimization. Such temporary constraint violations are resolved in later iterations. As an example, it is often unavoidable that in trying to avoid a collision with another quadcopter, a quadcopter's trajectory is temporarily perturbed to travel too close to a wall. This situation can be resolved in later iterations; however, if a quadcopter's trajectory is perturbed so significantly as to enter a wall, it is possible that the gradient descent optimization will force the trajectory through the wall, and will thus optimize towards an inescapable and infeasible local minimum. For this reason, being too close to a wall is penalized lightly, while significantly more penalty is applied to trajectory steps that enter a wall.

Goal constraints As previously mentioned, during individual iterations of the optimization, satisfying one constraint might require temporarily violating another. As a further example of this, perturbing trajectories to avoid collisions often results in trajectories that no longer end at the desired goal state. In order to ensure the final feasibility of the trajectories, the deviation of the final state from the goal state must be penalized. In this case study, the squared deviation of the final position and final velocity from the desired values ($G_{\mathrm{p},i}$ and 0 respectively) are penalized using the loss functions

$$\mathcal{L}_{\text{goal-pos}} \coloneqq \sum_{i=1}^{R} \|\mathrm{p}_i[K] - G_{\mathrm{p},i}\|_2^2 \tag{4.20}$$

$$\mathcal{L}_{\text{goal-vel}} \coloneqq \sum_{i=1}^{R} \|\mathrm{v}_i[K]\|_2^2; \tag{4.21}$$

while final acceleration is not penalized.

Robot i's final state constraints are satisfied if

$$\begin{gathered} \|\mathrm{p}_i[K] - G_{\mathrm{p},i}\|_2 \leq D_{\text{goal-pos}}, \text{ and} \\ \|\mathrm{v}_i[K]\|_2 \leq D_{\text{goal-vel}} \end{gathered} \tag{4.22}$$

where $D_{\text{goal-pos}}$ and $D_{\text{goal-vel}}$ define an acceptable deviation of the final position and velocity around the desired goal states. In this case study $D_{\text{goal-pos}} = 0.05\,\mathrm{m}$ and $D_{\text{goal-vel}} = 0.05\,\mathrm{m\,s^{-1}}$, deviations which are well within the ability of a hover controller to stabilize.

4.1.5 Optimization & Results

The Adam gradient descent method [39] was used to minimize the loss function (4.14). An automatic hyperparameter search [40] yielded the hyperparameter values shown in Table 4.1.

Fig. 4.1 shows a histogram of the time required to generate feasible, collision-free trajectories for 200 quadcopters exiting the maze. These results are based on 1000 random initializations of this case study. These results show that 50% of initializations were solved in under 2.28 seconds, and 90% of initializations in under 3.38 seconds.

Table 4.1: Hyperparameters for the "Sort 200" case study, as optimized by a hyperparameter search aiming to minimize the algorithm's 90th-percentile convergence time.

learning-rate	1.46×10^{-3}
$w_{\text{quad-collision}}$	1.16×10^{4}
$w_{\text{maze-collision}}$	2.71×10^{2}
$w_{\text{maze-wall-entry}}$	2.71×10^{5}
$w_{\text{goal-pos}}$	1.86×10^{4}
$w_{\text{goal-vel}}$	2.11×10^{4}
w_{thrust}	1×10^{5}
$w_{\text{body-rate}}$	1×10^{5}

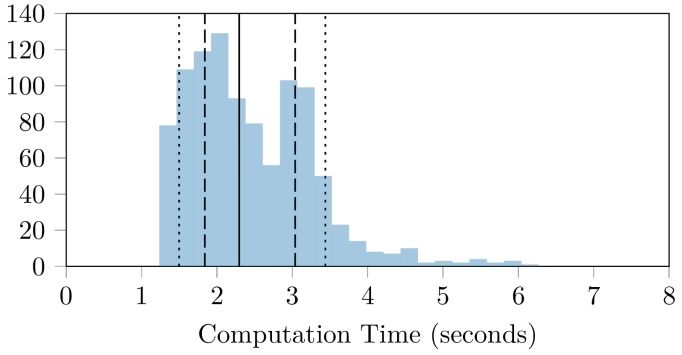


Figure 4.1: A histogram showing the time required for the generation of feasible, collision-free trajectories for a swarm of 200 quadcopters. This histogram summarizes the results of 1000 trials. The median calculation time of 2.28 seconds is shown as a solid line. The 10th and 90th percentiles (1.52 and 3.38 seconds respectively) are shown as dotted lines, and the 25th and 75th percentiles (1.83 and 3.03 seconds respectively) are shown as dashed lines.

4.2 Ground robot transitions

In this scenario, a fleet of 100 ground robots with front-steer bicycle dynamics is tasked with changing its formation. The front-steer bicycle model is a common choice for modeling front-steer vehicles, for example cars. Robots begin and end at rest and are required to arrive at their goal positions at the same time. As an example of the proposed method's applicability to heterogenous swarms, 50 of the 100 robots are chosen at random to have a maximum steering angle of 20°, and the other 50 robots to have a maximum steering angle of 70°. An example initialization of this problem is shown in Fig. 1.2, where each robot in the fleet changes its position within a "smiley face" formation.

4.2.1 Robot modeling

Each ground robot is modeled using the kinematic model of a bicycle [48]. Dynamics not modeled by the kinematic model (e.g. inertia) are assumed to be compensated for by a controller able to track the state and input trajectories generated by the proposed method.

With reference to Fig. 4.2, each robot is modeled as a bicycle with a fixed rear wheel located $l_r = 0.5\,\mathrm{m}$ from its center and a steerable front wheel located $l_f = 0.5\,\mathrm{m}$ from its center. The angle of its front wheel with respect to its longitudinal axis is denoted δ. The location of each robot's center in the two-dimensional inertial frame is denoted by p_x and p_y, the angle of its longitudinal axis relative to the inertial frame by φ, its forward velocity and acceleration by v_b and a_b, and the angle of its forward velocity relative to

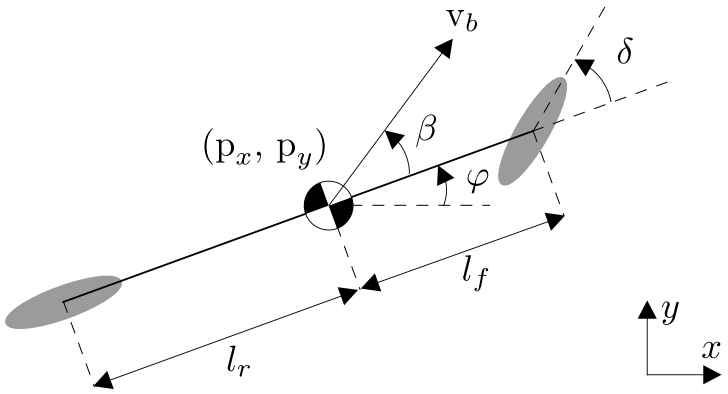


Figure 4.2: An illustration of a robot with bicycle dynamics. Such robots have a steerable front wheel and a fixed rear wheel located at distances l_f and l_r from the robot's center. The steering angle of the front wheel is denoted by δ, the angle of the robot with respect to the inertial frame by φ, and the angle of the robot's velocity v_b with respect to its longitudinal axis by β. The location of the robot in the inertial frame is denoted by p_x and p_y. Figured adapted from [48].

its longitudinal axis by β. The continuous-time kinematics of a bicycle robot are then

$$\dot{\mathrm{p}_x} = \mathrm{v}_b \cos(\varphi + \beta) \tag{4.23}$$
$$\dot{\mathrm{p}_y} = \mathrm{v}_b \sin(\varphi + \beta) \tag{4.24}$$
$$\dot{\varphi} = \frac{\mathrm{v}_b}{l_r} \sin(\beta) \tag{4.25}$$
$$\dot{\mathrm{v}_b} = \mathrm{a}_b, \tag{4.26}$$

where

$$\beta \coloneqq \tan^{-1}\left(\frac{l_r}{l_f + l_r} \tan(\delta)\right). \tag{4.27}$$

The above is discretized using the Euler forward method with a sampling period of $T = 50\,\mathrm{ms}$ to arrive at the discrete-time model

$$\mathrm{p}_x[k+1] = \mathrm{p}_x[k] + T\,\mathrm{v}_b[k] \cos(\varphi[k] + \beta[k]) \tag{4.28}$$
$$\mathrm{p}_y[k+1] = \mathrm{p}_y[k] + T\,\mathrm{v}_b[k] \sin(\varphi[k] + \beta[k]) \tag{4.29}$$
$$\varphi[k+1] = \varphi[k] + T\,\frac{\mathrm{v}_b[k]}{l_r} \sin(\beta[k]) \tag{4.30}$$
$$\mathrm{v}_b[k+1] = \mathrm{v}_b[k] + T\,\mathrm{a}_b[k] \tag{4.31}$$

where

$$\beta[k] = \tan^{-1}\left(\frac{l_r}{l_f + l_r} \tan(\delta[k])\right). \tag{4.32}$$

Based on the above model, the $N = 4$ dimensional state space of a robot is

$$S \coloneqq \big(\,\mathrm{p}_x,\, \mathrm{p}_y,\, \mathrm{v}_b,\, \varphi\,\big) \in \mathbb{R}^{K+1 \times N}. \tag{4.33}$$

As in the previous case study, these recursive difference equations can be implemented

using cumulative summation:

$$\begin{aligned}
\beta[0\!:\!K\!-\!1] &= \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta[0\!:\!K\!-\!1])\right) \\
\mathrm{v}_b[1\!:\!K] &= \mathrm{v}_b[0] + T\cdot\mathrm{CS}(\mathrm{a}_b[0\!:\!K\!-\!1]) \\
\varphi[1\!:\!K] &= \varphi[0] + T\cdot\mathrm{CS}\left(\frac{\mathrm{v}_b[0\!:\!K\!-\!1]}{l_r}\sin(\beta[0\!:\!K\!-\!1])\right) \\
\mathrm{v}_x[0\!:\!K\!-\!1] &= \mathrm{v}_b[0\!:\!K\!-\!1]\cos(\varphi[0\!:\!K\!-\!1]+\beta[0\!:\!K\!-\!1]) \\
\mathrm{v}_y[0\!:\!K\!-\!1] &= \mathrm{v}_b[0\!:\!K\!-\!1]\sin(\varphi[0\!:\!K\!-\!1]+\beta[0\!:\!K\!-\!1]) \\
\mathrm{p}_x[1\!:\!K] &= \mathrm{p}_x[0] + T\cdot\mathrm{CS}(\mathrm{v}_x[0\!:\!K\!-\!1]) \\
\mathrm{p}_y[1\!:\!K] &= \mathrm{p}_y[0] + T\cdot\mathrm{CS}(\mathrm{v}_y[0\!:\!K\!-\!1]),
\end{aligned} \tag{4.34}$$

where the ordering of equations indicates the necessary order of calculation.

4.2.2 Initial and final conditions

Robot i is assumed to begin at the known position

$$\mathrm{p}_i[0] \coloneqq \left(\mathrm{p}_{x,i}[0],\, \mathrm{p}_{y,i}[0]\right), \tag{4.35}$$

and at standstill, such that

$$\mathrm{v}_{b,i}[0] = 0. \tag{4.36}$$

Robot i is randomly assigned the goal location

$$G_{\mathrm{p},i} \coloneqq (\, G_{x,i},\, G_{y,i}\,), \tag{4.37}$$

and it is assumed that the robot begins facing this goal

$$\varphi_i[0] = \mathrm{arctan2}\left(\mathrm{p}_{y,i} - G_{y,i},\, \mathrm{p}_{x,i} - G_{x,i}\right). \tag{4.38}$$

Robot i is required to finish at its goal and at rest, that is

$$\begin{aligned}
\mathrm{p}_{x,i}[K] &\approx G_{x,i} \\
\mathrm{p}_{y,i}[K] &\approx G_{y,i} \\
\mathrm{v}_{b,i}[K] &\approx 0,
\end{aligned} \tag{4.39}$$

with tolerances around the goal state as defined in Section 4.2.4. The robot's final orientation $\varphi_i[K]$ is left unconstrained.

4.2.3 Initialization of individual trajectories

Since each robot begins facing its goal, the trajectory generation problem is reduced to a one-dimensional problem of generating a straight-line trajectory, which is feasible under the dynamic constraints (discussed below). The position trajectory is parameterized using a fifth-order polynomial optimized to minimize trajectory jerk [5]. Trajectory duration is calculated such that the robot with the furthest distance to travel will reach its goal as quickly as is allowed by its acceleration limits and by the trajectory parameterization. The polynomial corresponding to each robot's acceleration trajectory is sampled at the desired rate (in this case, 20 Hz) to yield the robot's initial acceleration trajectory. Each robot's steering trajectory is initialized to zero. This stage can be computed entirely on the GPU.

4.2.4 Constraints

The constraint set consists of constraints on the robots' accelerations and steering angles, constraints prohibiting robot collisions, and constraints on the final state of each robot.

Input constraints In this case-study, robot i's acceleration is constrained to

$$\mathrm{a}_{b,i} \in (-\mathrm{a}_{\max}, \mathrm{a}_{\max}), \tag{4.40}$$

where $\mathrm{a}_{\max} \coloneqq 2\,\mathrm{m\,s^{-2}}$, and its steering angle is constrained to

$$\delta_i \in (-\delta_{i,\max}, \delta_{i,\max}), \tag{4.41}$$

where $\delta_{i,\max} \coloneqq 20^\circ$ if $i \leq 50$ or otherwise 70°.

As demonstrated in the previous case-study, constraints can be implemented using loss functions added to the objective function. Although input constraints can also be modeled in this way, it is often easier to enforce simple input constraints directly by expressing a robot's constrained input as a function of an unconstrained optimization variable. In this case study, the $\tanh(\cdot)$ function is used to map an input on the domain of $(-\infty, \infty)$ to an output on the range $(-1, 1)$. The input constraints can then be directly expressed as

$$\mathrm{a}_{b,i} = \mathrm{a}_{\max} \tanh(\bar{\mathrm{a}}_{b,i}) \tag{4.42}$$

$$\delta_i = \delta_{i,\max} \tanh(\bar{\delta}_i), \tag{4.43}$$

where $\bar{\mathrm{a}}_{b,i}$ and $\bar{\delta}_i$ are the unconstrained targets of the optimization routine. The robot state update equations (4.34) are amended with the above transformations, and the $M = 2$ di-

mensional input space of each robot then defined as

$$U_i \coloneqq \left(\bar{\mathrm{a}}_{b,i},\, \bar{\delta}_i \right), \tag{4.44}$$

from which the actual, constrained robot inputs $\mathrm{a}_{b,i}$ and δ_i can later be recovered.

Robot collision constraints In this case study robots are modeled as circles and require a minimum center-to-center distance of $D \coloneqq 1.0\,\mathrm{m}$. Algorithm 2 is again used to compute the corresponding collision loss (4.19) and associated gradients.

Goal constraints The deviation from the desired goal position and velocity is penalized as in (4.20), with $D_{\text{goal-pos}} \coloneqq 0.05\,\mathrm{m}$ and $D_{\text{goal-vel}} \coloneqq 0.05\,\mathrm{m\,s^{-1}}$.

Table 4.2: Hyperparameter values for the ground robot transition case study, as determined by a hyperparameter search [40] aiming to minimize the algorithm's 90th percentile convergence time.

learning-rate	6.94×10^{-3}
$w_{\text{collision}}$	2.14×10^{3}
$w_{\text{final-pos}}$	2.07×10^{4}
$w_{\text{final-vel}}$	5.95×10^{3}

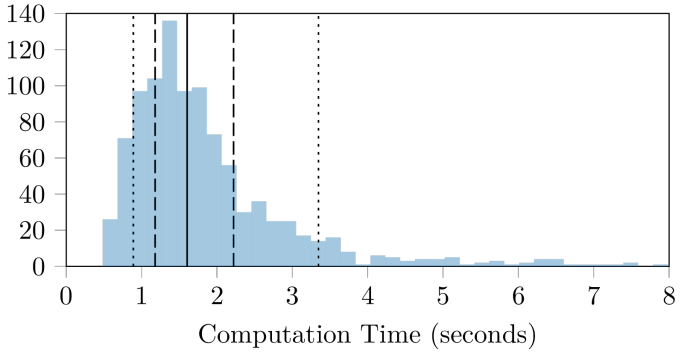


Figure 4.3: A histogram showing the time required for the generation of feasible, collision-free trajectories for a fleet of 100 ground robots. This histogram summarizes the results of 1000 trials. The median calculation time of 1.6 seconds is shown as a solid line. The 10th and 90th percentiles (0.88 and 3.34 seconds respectively) are shown as dotted lines, and the 25th and 75th percentiles (1.18 and 2.22 seconds respectively) are shown as dashed lines.

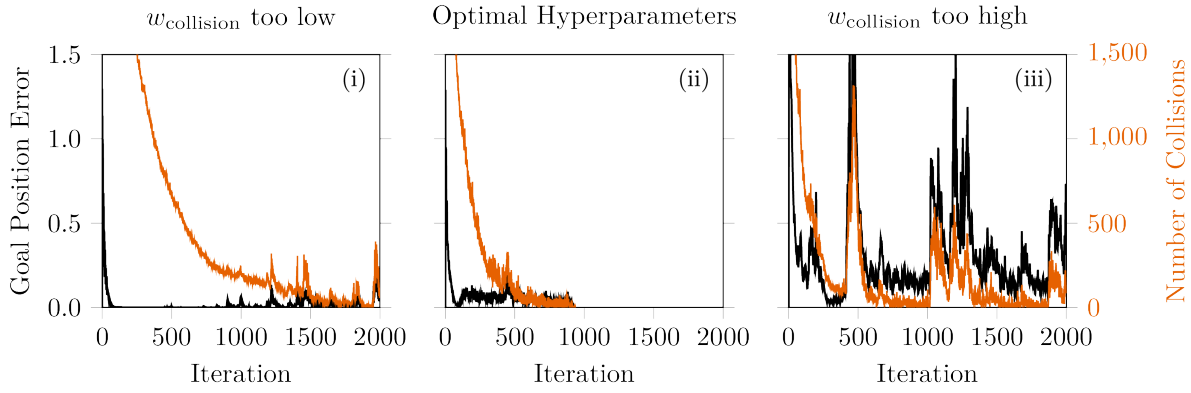
4.2.5 Optimization & Results

The Adam gradient descent method [39] was used to minimize the global loss. An automatic hyperparameter search [40] yielded the hyperparameter values shown in Table 4.2.

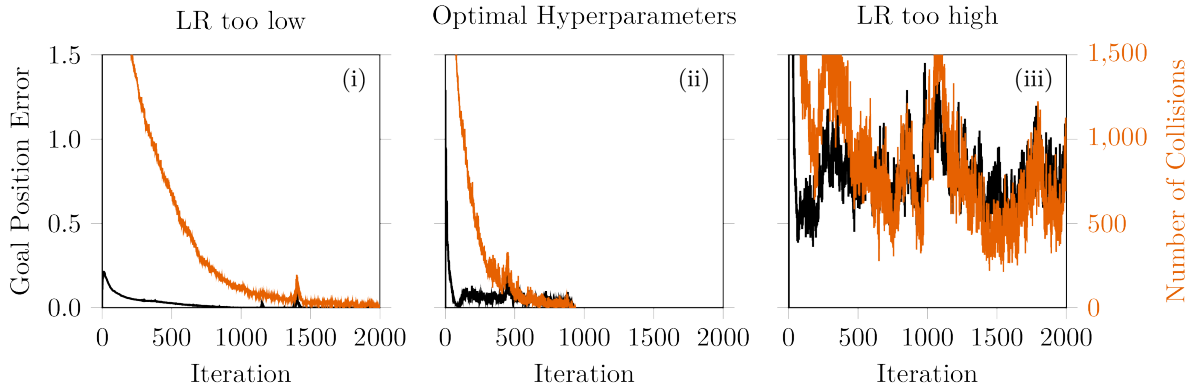
Fig. 4.3 presents a histogram of the time required to generate feasible, collision-free trajectories for 100 ground robots based on 1000 random initializations of this case study. These results show that 50% of initializations were completed in under 1.6 seconds, and 90% of initializations in under 3.34 seconds.

As discussed in Section 3.5, the hyperparameter values (loss weights, as well as the learning rate of the optimizer) play an important role in determining the speed of convergence. The effects of hyperparameter tuning are investigated in Fig. 4.4 by plotting the number of collisions (orange) and the root-mean-squared distance violation of the final position constraint (black), against the optimization iteration. Each plot begins from an identical initial state, thus allowing the effects of hyperparameter variation on convergence speed to be directly compared.

The center plots of Fig. 4.4 show the convergence toward feasibility when using the optimal hyperparameters (Table 4.2), which strike a balance between quickly decreasing the number of collisions, while keeping the final positions close to their goals. Observe in Fig. 4.4(a)(i) that by reducing $w_{\text{collision}}$, the number of collisions is not reduced as quickly, however the final positions remain closer to their goals. By increasing $w_{\text{collision}}$ in Fig. 4.4(a)(iii) the opposite can be observed: the number of collisions is quickly reduced (by drastically perturbing robot trajectories), thus causing an increase in the violation of the final position constraint. In Fig. 4.4(b)(i) the effect of a low learning rate, which causes a slow convergence to feasibility, is contrasted against the optimal learning rate in Fig. 4.4(b)(ii), and against a high learning rate in Fig. 4.4(b)(iii), which fails to converge.



(a) The effect of varying $w_{\text{collision}}$, the weight given to reducing quadcopter collisions



(b) The effect of varying the "learning rate" of the gradient descent optimizer

Figure 4.4: These figures exemplify the effects of hyperparameter tuning by plotting the root-mean-squared distance violation of the final position constraint (black) and the number of collisions (orange) against the optimization iteration. Each plot begins from an identical initial state, thus demonstrating how variations in hyperparameter values affect convergence. These plots exemplify the importance of hyperparameter selection and demonstrate how the optimal selection of hyperparameters strikes a careful balance between quickly decreasing the number of collisions and ensuring that other constraints are not too drastically violated.

5

Future Work

This thesis introduces a method for quickly generating feasible, collision-free trajectories for robot swarms. The proposed method leverages the computational power of modern GPUs to directly solve the non-convex optimization problem using gradient descent. As experimentally shown in the case-studies of Chapter 4, the proposed method is capable of generating feasible, collision-free trajectories for swarms of hundreds of robots in seconds, and can easily be extended to heterogenous swarms as shown in the case study of Section 4.2. Given the parallel nature of the problem, it is expected that the proposed method's performance will increase with progressive advances in GPU processing power.

5.1 Real-time, model predictive control

Although not touched upon in this thesis, the speed with which the proposed method can generate trajectories for large swarms of robots suggests a possible application to real-time reference generation. Warm starting the algorithm should reduce the number of iterations required for feasibility and allow for trajectories to be quickly replanned to account for situational changes and compensate for unmodeled effects.

5.2 Factorization into local policies

The centralized method presented in this thesis is efficient at computing trajectories, but communication bottlenecks may limit its application to large swarms if trajectories need to be communicated or updated in real time. Despite the complexity of the swarm trajectory generation problem, once a feasible solution is found, the collision avoidance behaviors demonstrated by individual robots are largely predictable (e.g if a robot is blocking the direction of travel, slow down). This suggests that such behavior could be encoded in a local policy and run on each robot independently.

The student project [SP.4] shows promising results on a small fleet of holonomic ground robots. In this project, robots transition from a start to goal state while avoiding collisions by using a neural-network-based policy, which was trained based on the solutions of the centralized planner presented in this thesis. Similar research is the topic of [49]–[52],

who show the effectiveness of training distributed collision avoidance and control policies based on the results of centralized solvers.

5.3 Improved convergence speed using hyperparameter scheduling

The convergence speeds presented in this thesis are highly dependent on the selection of hyperparameters (see Section 3.5). In the presented case-studies, a single set of hyperparameters was chosen for each case-study, which struck a balance between quickly reducing collisions and remaining close to a feasible solution with respect to other constraints. While this set of hyperparameters worked well for many initializations of the same problem, outliers suggest that it is may not be the optimal choice for all initializations; furthermore, as iterations progress closer to feasibility, a different selection of hyperparameters may result in improved convergence. Convergence speed is a very current topic of research in the field of deep learning, with recent results in, for example [53]–[56], showing that convergence speed can be improved by varying hyperparameters (such as learning rate) during training.

5.4 Alternative trajectory parameterizations

In the presented case studies, trajectories were sampled at discrete points in time, and each trajectory sample was treated as an optimization variable. The methods presented in this thesis can be extended to alternative trajectory parameterizations, such as polynomial or spline-based parameterizations, as long as a gradient can be calculated between the various constraint losses and the trajectory parameters. Alternative parameterizations can be used to reduce the computation time of trajectory integration; may allow for the simplification of constraint functions (e.g. as demonstrated in the trajectory feasibility check of [5]); and, due to having significantly fewer parameters than a sampling-based parameterization, might result in faster optimization. The downside of such parameterizations is a reduction in the size of the solution space, and the optimization problem may therefore be more difficult to solve.

Bibliography: Part B

[1] Preiss, J. A., Hönig, W., Ayanian, N., Sukhatme, G. S., "Downwash-aware trajectory planning for large quadrotor teams", in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017, pp. 250–257.

[2] Hehn, M., D'Andrea, R., "Quadrocopter trajectory generation and control", *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[3] Mellinger, D., Kumar, V., "Minimum snap trajectory generation and control for quadrotors", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2520–2525.

[4] Mellinger, D., Michael, N., Kumar, V., "Trajectory generation and control for precise aggressive maneuvers with quadrotors", *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[5] Mueller, M. W., Hehn, M., D'Andrea, R., "A computationally efficient motion primitive for quadrocopter trajectory generation", *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[6] Richter, C., Bry, A., Roy, N., "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments", in *Robotics Research*, Springer, 2016, pp. 649–666.

[7] Liu, S., Atanasov, N., Mohta, K., Kumar, V., "Search-based motion planning for quadrotors using linear quadratic minimum time control", in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017, pp. 2872–2879.

[8] Liu, S., Mohta, K., Atanasov, N., Kumar, V., "Search-based motion planning for aggressive flight in SE(3)", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[9] LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning", Tech. Rep., 1998.

[10] Karaman, S., Frazzoli, E., "Incremental sampling-based algorithms for optimal motion planning", *Robotics Science and Systems VI*, vol. 104, p. 2, 2010.

[11] Svestka, P., Latombe, J., Overmars Kavraki, L., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] Flores, M. E., "Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational b-spline basis functions", PhD thesis, California Institute of Technology, 2008.

[13] Earl, M. G., D'Andrea, R., "Iterative MILP methods for vehicle-control problems", *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[14] Augugliaro, F., Schoellig, A. P., D'Andrea, R., "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1917–1922.

[15] Chen, Y., Cutler, M., How, J. P., "Decoupled multiagent path planning via incremental sequential convex programming", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 5954–5961.

[16] Alonso-Mora, J., Baker, S., Rus, D., "Multi-robot navigation in formation via sequential convex programming", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 4634–4641.

[17] Morgan, D., Subramanian, G. P., Chung, S.-J., Hadaegh, F. Y., "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming", *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[18] Tang, S., Kumar, V., "Safe and complete trajectory generation for robot teams with higher-order dynamics", in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 1894–1901.

[19] Mellinger, D., Kushleyev, A., Kumar, V., "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 477–483.

[20] Bento, J., Derbinsky, N., Alonso-Mora, J., Yedidia, J. S., "A message-passing algorithm for multi-agent trajectory planning", in *Advances in neural information processing systems*, 2013, pp. 521–529.

[21] Luis, C. E., Schoellig, A. P., "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control", *arXiv preprint arXiv:1809.04230*, 2018.

[22] Agarwal, S., Akella, S., "Simultaneous optimization of assignments and goal formations for multiple robots", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6708–6715.

[23] Debord, M., Hönig, W., Ayanian, N., "Trajectory planning for heterogeneous robot teams", in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*, IEEE, 2018.

[24] Hönig, W., Preiss, J. A., Kumar, T. S., Sukhatme, G. S., Ayanian, N., "Trajectory planning for quadrotor swarms", *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[25] Lucas, D., Crane, C., "Development of a multi-resolution parallel genetic algorithm for autonomous robotic path planning", in *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, IEEE, 2012, pp. 1002–1006.

[26] Cekmez, U., Ozsiginan, M., Sahingoz, O. K., "A UAV path planning with parallel ACO algorithm on CUDA platform", in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, IEEE, 2014, pp. 347–354.

[27] Cai, P., Cai, Y., Chandrasekaran, I., Zheng, J., "A GPU-enabled parallel genetic algorithm for path planning of robotic operators", in *GPU Computing and Applications*, Springer, 2015, pp. 1–13.

[28] Cai, P., Cai, Y., Chandrasekaran, I., Zheng, J., "Parallel genetic algorithm based automatic path planning for crane lifting in complex environments", *Automation in Construction*, vol. 62, pp. 133–147, 2016.

[29] Roberge, V., Tarbouchi, M., Labonté, G., "Fast genetic algorithm path planner for fixed-wing military UAV using GPU", *IEEE Transactions on Aerospace and Electronic Systems*, 2018.

[30] Pan, J., Lauterbach, C., Manocha, D., "g-Planner: Real-time motion planning and global navigation using GPUs", in *AAAI*, 2010.

[31] Pan, J., Lauterbach, C., Manocha, D., "Efficient nearest-neighbor computation for GPU-based motion planning", in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 2243–2248.

[32] Kider, J. T., Henderson, M., Likhachev, M., Safonova, A., "High-dimensional planning on the GPU", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2515–2522.

[33] Likhachev, M., Stentz, A., "R* search", in *In Proceedings of the National Conference on Artificial Intelligence (AAAI*, Citeseer, 2008.

[34] Chretien, B., Escande, A., Kheddar, A., "GPU robot motion planning using semi-infinite nonlinear programming", *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[35] Pan, J., Manocha, D., "GPU-based parallel collision detection for fast motion planning", *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 187–200, 2012.

[36] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., "Tensorflow: A system for large-scale machine learning.", in *OSDI*, vol. 16, 2016, pp. 265–283.

[37] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., "Automatic differentiation in pytorch", 2017.

[38] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z., "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems", *arXiv preprint arXiv:1512.01274*, 2015.

[39] Kingma, D. P., Ba, J., "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[40] Snoek, J., Larochelle, H., Adams, R. P., "Practical bayesian optimization of machine learning algorithms", in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[41] Hillis, W. D., Steele Jr, G. L., "Data parallel algorithms", *Communications of the ACM*, vol. 29, no. 12, pp. 1170–1183, 1986.

[42] Cole, R., "Parallel merge sort", *SIAM Journal on Computing*, vol. 17, no. 4, pp. 770–785, 1988.

[43] **Hamer, M.** (2019). Fast generation of collision-free trajectories for robot swarms using GPU acceleration: Software release. `http://mikehamer.info/swarm-trajectories` and `https://github.com/mikehamer/swarm-trajectories`.

[44] Mahony, R., Kumar, V., Corke, P., "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor", *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[45] Dijkstra, E. W., "A note on two problems in connexion with graphs", *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[46] Kuhn, H. W., "The Hungarian method for the assignment problem", *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[47] Bitcraze AB, (2018). Crazyflie 2.0, [Online]. Available: `https://www.bitcraze.io/crazyflie-2/`.

[48] Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F., "Kinematic and dynamic vehicle models for autonomous driving control design.".

[49] Long, P., Liu, W., Pan, J., "Deep-learned collision avoidance policy for distributed multiagent navigation", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[50] Fan, T., Long, P., Liu, W., Pan, J., "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios", *arXiv preprint arXiv:1808.03841*, 2018.

[51] Chen, Y. F., Liu, M., Everett, M., How, J. P., "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning", in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 285–292.

[52] Chen, Y. F., Everett, M., Liu, M., How, J. P., "Socially aware motion planning with deep reinforcement learning", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 1343–1350.

[53] Loshchilov, I., Hutter, F., "Sgdr: Stochastic gradient descent with warm restarts", *arXiv preprint arXiv:1608.03983*, 2016.

[54] Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., Storkey, A., "Three factors influencing minima in sgd", *arXiv preprint arXiv:1711.04623*, 2017.

[55] Smith, L. N., "Cyclical learning rates for training neural networks", in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2017, pp. 464–472.

[56] Smith, L. N., Topin, N., "Super-convergence: Very fast training of neural networks using large learning rates", *arXiv preprint arXiv:1708.07120v3*, 2017.

III

Outlook

This thesis presented scalable methods for the localization and coordination of robot swarms, as developed during the doctoral research and published in the peer-reviewed papers [P.1], [P.2], [P.3] and [P.4].

A localization system based on UWB radio was presented in Part A and a method of quickly generating collision-free trajectories for robot swarms was presented in Part B. Each of these thesis parts concluded with a technical outlook with suggested directions for future research (in Part A, Chapter 9 and Part B, Chapter 5). In the following, the research contributions are summarized and placed in the broader context.

As shown in Part A, UWB radio is a promising technology for localization in indoor environments. Owing to their impulse-like nature, the transmission and reception time of UWB signals can be accurately measured (Part A, Section 2.1). The communication of these timestamps to other modules enables distance measurement (Part A, Section 2.2); clock synchronization (Part A, Chapter 4); and localization, both of the anchors themselves (Part A, Chapter 6) and of robots operating in the space (Part A, Chapter 7).

The topology and purpose of the system in Part A is best described as an "indoor GPS" system. Radio transceivers are placed at fixed positions within the environment and routinely transmit UWB signals, fulfilling a similar function to GPS satellites; while robots move within the environment, and receive transmissions from the stationary transceivers. Robots can localize themselves based only on received and local information, and much like GPS, the system therefore scales to support an unlimited number of robots. However, the use of UWB technology does present some inherent challenges.

As experimentally demonstrated in Part A, Section 2.4, measurements of a UWB signal's reception time are affected by systematic biases that depend on the relative orientation and distance of the transmitter and receiver. As a robot moves within the space, its relative position and orientation to each anchor changes, resulting in a change in its bias to each anchor. This results in an ever-changing offset between the robot's estimated and actual position. Results in Part A, Chapter 7 show that when moving within a space of size 6m $\times$ 7m $\times$ 3.5m, a robot's estimated position is affected by a position-dependent systematic bias within the range of $\pm 100\,\mathrm{mm}$. Preliminary research supported by the results of this thesis suggests that the magnitude of systematic measurement biases can be reduced by using machine-learning to learn a compensation model [R.2], [R.4]. Despite

these systematic biases, the system allows robots to localize themselves with sufficient accuracy to operate autonomously. However, as is also the case for outdoor GPS localization, the ability to localize within an environment does not imply awareness of the environment. It is therefore suggested that robots employ additional sensors, for example LIDAR or vision sensors, to provide awareness of and allow navigation relative to the robot's surroundings. Although this is not a focus of the thesis, preliminary results supported by the UWB developments in Part A highlight the potential of such an approach [R.3].

Having developed a localization system capable of supporting the simultaneous operation of a swarm of robots, Part B of the thesis addressed one possible method of coordinating such a swarm. This work was published in [P.2] and focuses on the problem of quickly generating collision-free trajectories for large robot swarms. By reformulating the trajectory generation problem as described in Part B, Chapter 3, the computational power of a modern GPU was leveraged to optimize robot trajectories using gradient descent. The method was applied to a swarm of 200 quadcopter robots operating in a cluttered maze environment (Part B, Section 4.1); and to a heterogenous swarm of 100 bicycle robots, a common benchmark used for non-holonomic planning problems (Part B, Section 4.2). In both cases, the method handled nonlinear dynamics and various state and input constraints with ease, and generated feasible, collision-free trajectories for the swarm within seconds.

The method presented in Part B proved to be effective at quickly generating trajectories for large robot swarms. These trajectories are generated in a central location and are assumed to be tracked by a controller on each robot. In many applications, centralized coordination of robots is required; for example, to manage task or resource allocation; to optimize a global performance metric; or to enable long-term, multi-robot planning. In such situations, the proposed method of centralized trajectory generation has many advantages. In other situations, a more distributed approach may be desired, in which robots are allocated tasks centrally, but plan and navigate independently to their goals. In these situations, centralized planning may not be the best solution. As suggested in Part B, Chapter 5, the proposed method of centralized trajectory generation could be used to train a distributed control strategy using a supervised learning approach.

The suggested research directions presented in this outlook extend the contributions of Part A and Part B, and provide a pathway towards the application of these technologies to the localization and coordination of large robot swarms in indoor environments.

List of Publications

Journal publications

[1] **Hamer, M.**, D'Andrea, R., "Self-calibrating ultra-wideband network supporting multi-robot localization", *IEEE Access*, vol. 6, pp. 22 292–22 304, 2018.

[2] **Hamer, M.**, Widmer, L., D'Andrea, R., "Fast generation of collision-free trajectories for robot swarms using GPU acceleration", *IEEE Access*, vol. 7, pp. 6679–6690, 2019.

[3] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Angle of arrival estimation based on channel impulse response measurements", *(in preparation)*, 2019.

[4] Augugliaro, F., Lupashin, S., **Hamer, M.**, Male, C., Hehn, M., Mueller, M. W., Willmann, J. S., Gramazio, F., Kohler, M., D'Andrea, R., "The flight assembled architecture installation: Cooperative construction with flying machines", *IEEE Control Systems*, vol. 34, no. 4, pp. 46–64, 2014.

Conference publications (peer reviewed)

[5] Ledergerber, A., **Hamer, M.**, D'Andrea, R., "A robot self-localization system using one-way ultra-wideband communication", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 3131–3137.

[6] Mueller, M. W., **Hamer, M.**, D'Andrea, R., "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1730–1736.

[7] Hoeller, D., Ledergerber, A., **Hamer, M.**, D'Andrea, R., "Augmenting ultra-wideband localization with computer vision for accurate flight", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 734–12 740, 2017.

[8] **Hamer, M.**, Waibel, M., D'Andrea, R., "Knowledge transfer for high-performance quadrocopter maneuvers", in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 1714–1719.

Other publications (not peer reviewed)

[9] Förster, J., **Hamer, M.**, D'Andrea, R., "System identification of the Crazyflie 2.0 nano quadrocopter", Bachelor's thesis, ETH Zurich, 2015.

- [5] resulted from the master's thesis [MT.1], which was directly supervised by the author

- Equal contributions were made to [6], which combined two distinct research fields

Student supervision

Masters thesis

The master's thesis is a six-month, full-time project.

[MT.1] Ledergerber, A., "A robot self-localization system using one-way ultra-wideband communication", Master's thesis, ETH Zurich, 2014.

[MT.2] Jia, Z., "Distributed control and coordination of quadcopter swarms", Master's thesis, ETH Zurich, 2017.

Semester project

A semester-long, part-time project conducted during the master's studies

[SP.1] Zahner, M., "Application of machine learning to quadrocopter slalom flying", Semester project, ETH Zurich, 2013.

[SP.2] Grob, M., Stadler, M., "Crazyflie onboard estimation & control with single point localization", Semester project, ETH Zurich, 2016.

[SP.3] Stadler, M., Grob, M., "Crazyflie slalom flight using single point localization", Semester project, ETH Zurich, 2016.

[SP.4] Ma, Y., "Learning local policies from global solutions", Semester project, ETH Zurich, 2018.

[SP.5] Widmer, L., "Fast computation of swarm transition trajectories", Semester project, ETH Zurich, 2018.

Bachelors thesis

The bachelor's thesis is a three-month, full-time project.

[BT.1] Estandia, A., "Mechanical design of a quadrocopter D.I.Y. kit", Bachelor's thesis, ETH Zurich, 2015.

[BT.2] Förster, J., "System identification of the crazyflie nano-quadcopter", Bachelor's thesis, ETH Zurich, 2015.

Internship

Internships are practical projects lasting four to six months

[IN.1] Sun, J., "Ultra-wideband API development", Internship, ETH Zurich, 2014.

[IN.2] Alawieh, A., "Modelling systematic ultra-wideband ranging errors", Internship, ETH Zurich, 2014.

[IN.3] Lee, S. K., "Position control of a crazyflie nano-quadcopter", Internship, ETH Zurich, 2015.

[IN.4] Buono, A., "Improving nano-quadcopter motor control using IR reflectance sensors", Internship, ETH Zurich, 2017.